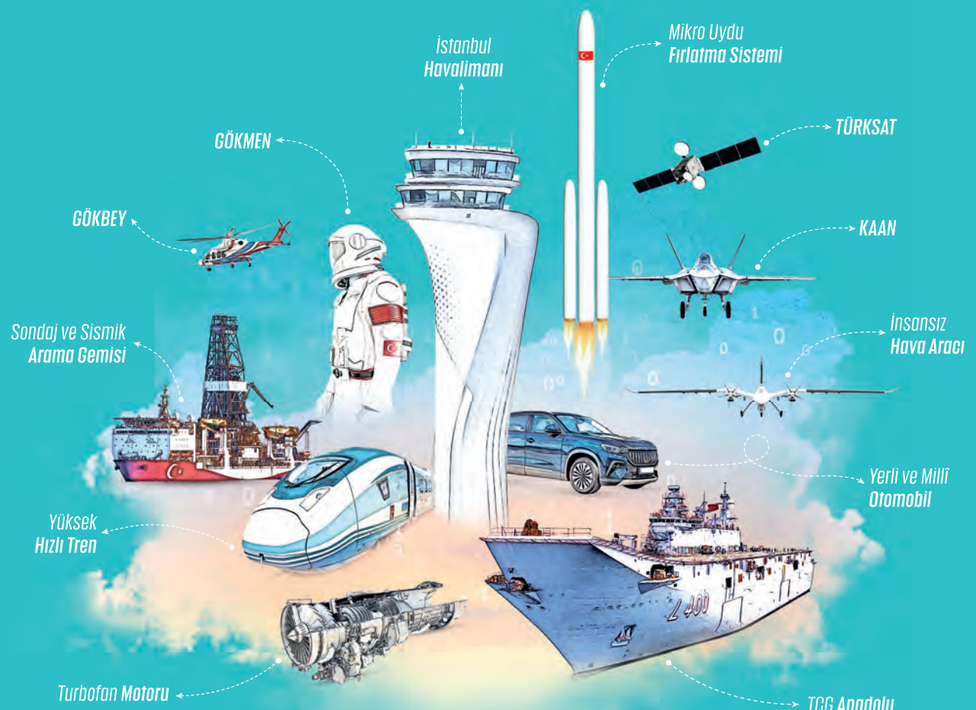
211460_Mobil Uygulamalari-11 Kapak.pdf

Bu kitaba sığmayan daha neler var!

eba
www.eba.gov.tr

Karekodu okut, bu kitapla ilgili EBA içeriklerine ulaş!

Teknoloji Yolculuğumuz



BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA
ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

ISBN: 978-975-11-7120-7

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmeliğin Beşinci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

BİLİŞİM TEKNOLOJİLERİ ALANI

MOBİL UYGULAMALAR 11

DERS MATERYALİ

MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

11 DERS MATERYALİ

MOBİL UYGULAMALAR

MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

MOBİL UYGULAMALAR

11

DERS MATERYALİ

Yazarlar

Atılım ÇİFTÇİ
İbrahim BİLGİN
İsmail GÖVERCİN
Mustafa CAYMAZ

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI 8341
YARDIMCI VE KAYNAK KİTAPLAR DİZ..... 2233

Her hakkı saklıdır ve Millî Eğitim Bakanlığına aittir. Ders materyalinin metin, soru ve şekilleri kısmen de olsa hiçbir surette alınıp yayımlanamaz.

HAZIRLAYANLAR

Dil Uzmanı
Melek DEMİR

Program Geliştirme Uzmanı
Esra YAVUZ

Ölçme ve Değerlendirme Uzmanı
Filiz İSNAÇ

Rehberlik Uzmanı
Gülşen YALIN

Görsel Tasarım Uzmanı
Ayşe KATIRCI KARBUKAN

BASKI
Ayrıntı Basım Yayım ve Mat. Hiz. Sanayi Ticaret A.Ş.
Saray Mahallesi 126. Cadde No: 20
Kahramankazan/Ankara
Sertifika No: 49599

ISBN: 978-975-11-7120-7

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Meslekî ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

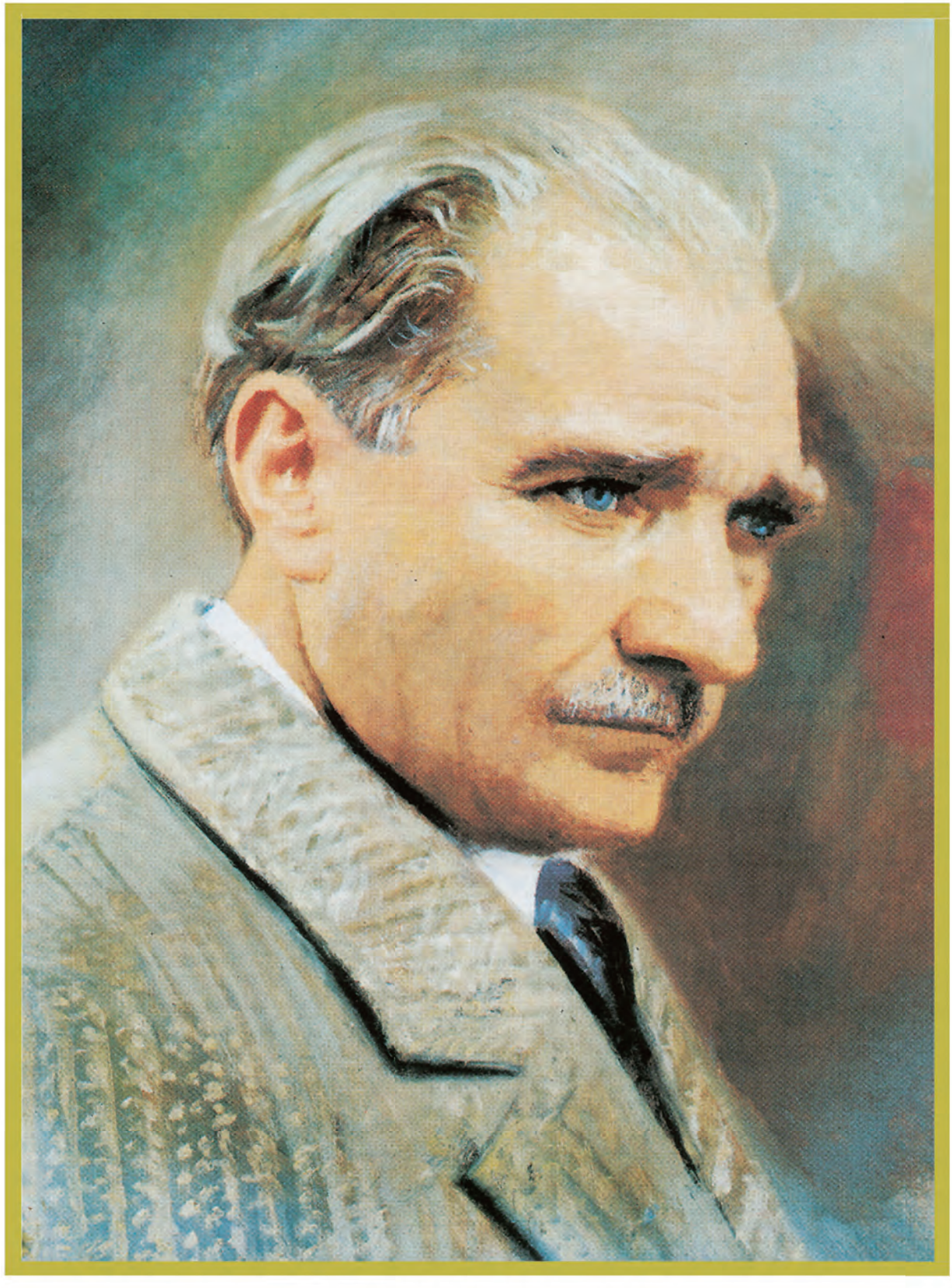
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

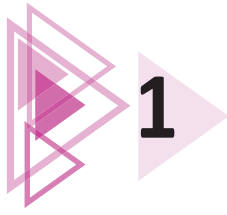
Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk



MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER



DERS MATERYALİNİN TANITIMI 14
1. MOBİL UYGULAMA GELİŞTİRMEYE HAZIRLIK 18
1.1. TEMEL BİLEŞENLER 20
1.1.1. Java Software Development Kit (JDK) Kurulumu 20
1.1.2. Android Studio ve Software Development Kit (SDK) Kurulumu 23
1.2. EMÜLATÖR KURULUMU 28
1.3. TASARIM YAPILARI (ACTIVITY TEMPLATES) 35
1.3.1. Activity 36
1.3.2. Activity Çeşitleri 36
1.3.3. Activity Yaşam Döngüsü 37
1.3.4. Activity Yaşam Döngüsü Metotları 38
1.4. PROJE OLUŞTURMA 39
1.4.1. Projenin Emülatörde Çalıştırılması 41
1.4.2. Android Studio Tasarım Ekranı 42
1.4.3. Ön İzleme Görünümünü Değiştirmek 43
1.5. DOSYA VE DİZİN YAPILARI 45
ÖLÇME VE DEĞERLENDİRME 47

2. EKRAN TASARIMI 48
2.1. MOBİL UYGULAMA EKRAN TASARIMINA GİRİŞ 50
2.2. MOBİL UYGULAMA EKRAN YAPISI 51
2.3. VIEW 52
2.4. GÖRÜNÜM YERLEŞTİRME YÖNTEMLERİ 54
2.4.1. Görünüm Yerleştirmek İçin XML Kullanımı 55
2.4.2. Görünüm Yerleştirmek İçin Palette Kullanımı 56
2.4.3. Görünüm Yerleştirmek İçin Java Kullanımı 59
2.5. TEMEL GÖRÜNÜM SINIFLARI 59
2.5.1. TextView 59
2.5.2. EditText 62
2.5.3. Button 64
2.5.4. ImageView 65
2.5.5. CheckBox 68
2.5.6. ProgressBar 69
2.6. LAYOUT ÇEŞİTLERİ 70
2.6.1. ConstraintLayout 71
2.6.2. LinearLayout 74
2.6.3. RelativeLayout 75
2.6.4. FrameLayout 77
ÖLÇME VE DEĞERLENDİRME 79

3

3. TEMEL KOMUTLAR 80
3.1. VERİ 82
3.2. DEĞİŞKENLER 82
3.2.1. Değişken Yapısı 85
3.2.2. Değişken Tanımlama ve Atama 85
3.3. VERİ TİPLERİ 86
3.3.1. İlkel (Temel) Veri Tipleri 87
3.3.2. Referans Veri Tipleri 97
3.4. SABİTLER 98
3.5. İSİMLENDİRME KURALLARI 100
3.6. OPERATÖRLER 102
3.6.1. Aritmetik Operatörler 102
3.6.2. Atama Operatörleri 104
3.6.3. Karşılaştırma Operatörleri 106
3.6.4. Mantıksal Operatörler 107
3.7. HATA AYIKLAMA 109
3.7.1. Logcat 110
3.7.2. Durak Noktası 113
3.7.3. Değişken İzleme (Watch) 114
3.8. HATA DÜZELTME 114
ÖLÇME VE DEĞERLENDİRME 117

4

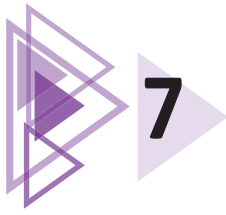
4. KARAR VE DÖNGÜ YAPILARI 118
4.1. KARAR YAPILARI 120
4.1.1. if Karar Yapısı 120
4.1.2. if-else Karar Yapısı 125
4.1.3. if-else-if Karar Yapısı 134
4.1.4. Switch-Case Yapısı 136
4.2. DÖNGÜ YAPILARI 139
4.2.1. Sayaçlar 140
4.2.2. for Döngüsü 140
4.2.3. while Döngüsü 150
4.2.4. do-while Döngüsü 154
ÖLÇME VE DEĞERLENDİRME 155

5

5. GELİŞMİŞ KOMUTLAR 160
5.1. METOT 162
5.1.1. Metot Yapısı 163
5.1.2. Değer Döndürmeyen Metotlar 163
5.1.3. Değer Döndüren Metotlar 166
5.1.4. Parametre Alan Metotlar 167
5.1.5. Metotlarda Aşırı Yükleme 170
5.2. SINIF VE NESNE KAVRAMLARI 174
5.2.1. Sınıf (Class) 177
5.2.2. Nesne (Object) Oluşturma 177
5.2.3. Sınıf Yapısı 178
5.2.4. Erişim Belirleyiciler (Erişim Düzeyleri) 182
5.2.5. Kurucu veya Yapıcı Metotlar (Constructors) 184
5.3. KAPSÜLLEME (ENCAPSULATION) 188
5.3.1. Kapsülleme Yapısı 189
5.3.2. Getter ve Setter Metotları 190
5.4. KALITIM (INHERITANCE) 194
5.5. ÇOKBİÇİMLİLİK (POLYMORPHISM) 202
5.5.1. Çokbiçimlilik Yapısı 202
5.6. DİZİLER 206
5.6.1. Diziye Değer Atama 207
5.6.2. ArrayList 210
ÖLÇME VE DEĞERLENDİRME 222

6

6. UYGULAMA TASARIMI 224
6.1. PROJE YAPILANDIRMA AYARLARI 226
6.1.1. AndroidManifest.xml Dosya Yapısı 226
6.1.2. MainActivity 228
6.1.3. res 230
6.1.4. Gradle Scripts 235
6.2. VIEW BINDING 238
6.3. ACTIVITY YAPISI 247
6.3.1. Activity Yaşam Döngüleri (Activity Life Cycle) 248
6.3.2. Çoklu Aktiviteler 253
6.4. ARAYÜZ TASARIMI 267
6.5. FRAGMENT YAPISI 278
6.6. MOBİL UYGULAMADA İZİNLER VE İZİN YAPISI 297
6.6.1. İzin Tanımlamaları Yapısı 298
6.6.2. İzin Onayları Almak 299
ÖLÇME VE DEĞERLENDİRME 308



7. VERİ TABANI İŞLEMLERİ..... 312

7.1. sharedpreferences KULLANIMI 314

7.2. YEREL VERİ TABANIYLA ÇALIŞMAK 320

7.2.1. Sorgulama Komutları 320

7.2.3. Mobil Uygulama Geliştirme Ortamında Yerel Veri Tabanı Kullanmak..... 323

7.2.4. Mobil Uygulama Geliştirme Ortamında Kayıt Eklemek..... 324

7.2.5. Mobil Uygulama Geliştirme Ortamında Kayıt Silmek 325

7.2.6. Mobil Uygulama Geliştirme Ortamında Kayıt Güncellemek 326

7.2.7. Mobil Uygulama Geliştirme Ortamında Tüm Kayıtları Listelemek 327

7.2.8. Özel Adaptör Kullanmak 328

7.2.9. Model Oluşturmak..... 328

7.2.10. ArrayAdapter Sınıfıyla Özel Adaptör Oluşturmak 343

7.3. UZAK VERİ TABANIYLA ÇALIŞMAK 349

7.3.1. JSON Veri Düzeni 349

7.3.2. Mobil Uygulama Geliştirme Ortamında Uzak Sunucu Yapılandırması 351

7.3.3. Uzak Veri Tabanı Yönetimi..... 356

7.3.4. Firestore Veri Tabanı Oluşturmak 358

7.3.5. Uzak Veri Tabanında Yetkilendirme İşlemleri..... 362

7.3.6. Mobil Uygulama Geliştirme Ortamında Uzak Veri Tabanıyla Çalışmak..... 366

7.3.7. Mobil Uygulama Geliştirme Ortamında RecyclerView Nesnesiyle Çalışmak 376

7.3.8. Uzak Veri Tabanından Veri Getirmek 379

7.3.9. Mobil Uygulama Geliştirme Ortamında Uzak Veri Tabanına Veri Eklemek 393

7.3.10. Uzak Veri Tabanından Veri Silmek..... 397

7.3.11. Uzak Veri Tabanında Veri Güncellemek 398

7.3.12. Firestore Veri Tabanında Kurallarla Çalışmak..... 403

ÖLÇME VE DEĞERLENDİRME.....408

8

8. GELİŞMİŞ UYGULAMA TASARLAMA 412
8.1. YAYIN ALGILAYICILARLA ÇALIŞMAK 414
8.1.1. BroadcastReceiver Sınıfı Oluşturmak 414
8.1.2. Kodla Yayın Algılayıcıları Tetiklemek 417
8.2. SMS VE İLETİ GÖNDERMEK 420
8.2.1. SMS Okumak 422
8.2.2. İleti Göndermek 426
8.3. BİLDİRİMLERLE ÇALIŞMAK 428
8.3.1. Bildirimlerden Veri Almak 432
8.4. BAŞKA UYGULAMALARLA ETKİLEŞİM KURMAK 438
8.4.1. Paylaş Butonuyla Diğer Uygulamalardan Veri Almak 439
8.4.2. BroadcastReceiver Kullanarak Uygulamaların Arasında Haberleşmek 442
8.5. SERVİSLERLE ÇALIŞMAK 446
8.5.1. Arka Plan Servislerle Çalışmak 450
8.5.2. Ön Plan Servislerle Çalışmak 454
8.5.3. Sticky Servislerle Çalışmak 456
8.5.4. Bound Servislerle Çalışmak 457
8.5.5. IntentServislerle Çalışmak 461
8.6. WORKMANAGERLE ÇALIŞMAK 464
8.6.1. Kısıtlamaları Tanımlamak 466
8.6.2. Görevi Gecikmeli Başlatmak 466
8.6.3. Zincirleme Çalışmak 467
8.6.4. Görevleri İzlemek 467
8.6.5. Görevlere Dışarıdan Veri Göndermek 468
8.7. SENSÖRLERLE ÇALIŞMAK 471
8.7.1. Tüm Sensörlere Erişmek 472
8.7.2. Sensör Olaylarıyla Çalışmak 473
ÖLÇME VE DEĞERLENDİRME 477

9

9. UYGULAMA YAYIMLAMA 480
9.1. UYGULAMA MARKET GELİŞTİRİCİ AYARLARI 482
9.2. UYGULAMA YAYIMLAMA 486
ÖLÇME VE DEĞERLENDİRME 502

CEVAP ANAHTARLARI 504
KAYNAKÇA 509

DERS MATERYALİNİN TANITIMI

Öğrenme biriminde neler öğrenileceğini gösterir.

Öğrenme biriminin numarasını gösterir.

Öğrenme biriminin konularını gösterir.

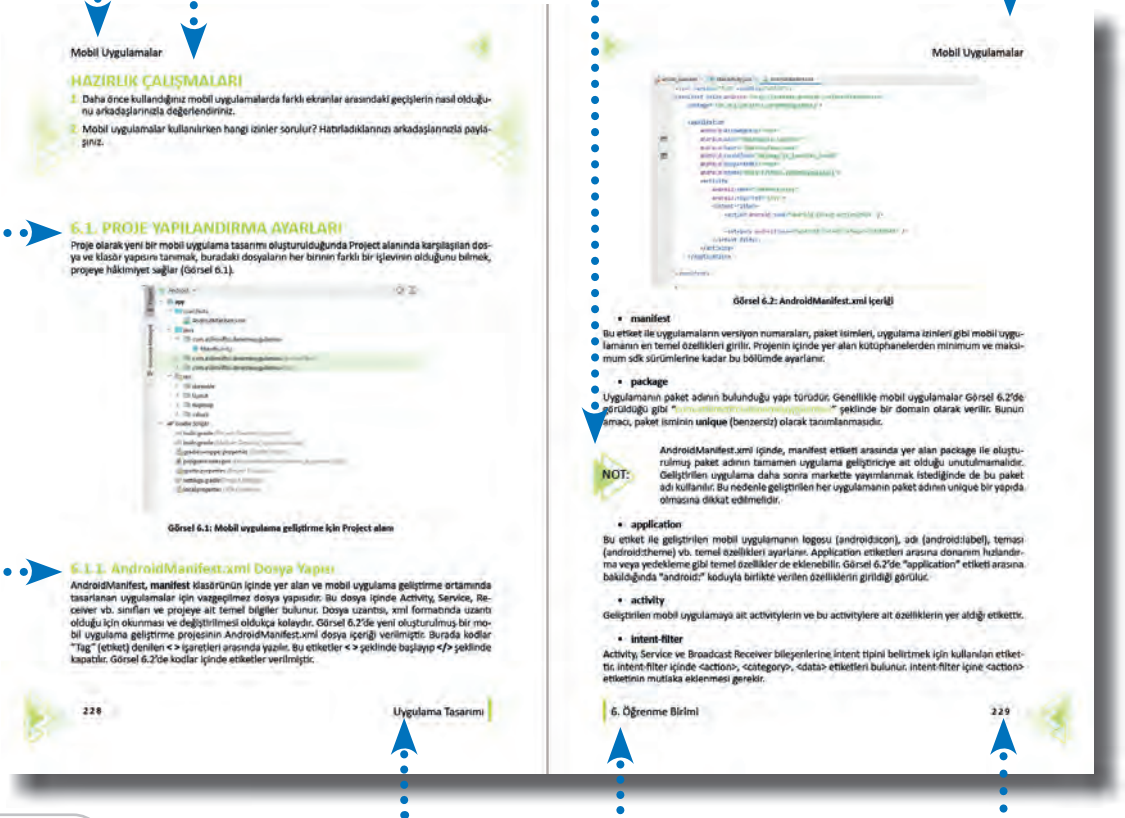
Öğrenme biriminin adını gösterir.

Öğrenme biriminin temel kavramlarını gösterir.

Dersin adını gösterir.

Öğrencilerin öğrenme birimiyle ilgili ön bilgilerini harekete geçirmek ve konuya ilgilerini çekmek için yapılacak çalışmaları gösterir.

Konu ile ilgili notları gösterir.



Mobil Uygulamalar

HAZIRLIK ÇALIŞMALARI

1. Daha önce kullandığınız mobil uygulamalarda farklı ekranlar arasındaki geçişlerin nasıl olduğunu arkadaşlarınızla değerlendiriniz.
2. Mobil uygulamalar kullanılırken hangi izinler sorulur? Hatırladıklarınızı arkadaşlarınızla paylaşınız.

6.1. PROJE YAPILANDIRMA AYARLARI

Proje olarak yeni bir mobil uygulama tasarımı oluşturulduğunda Project alanında karşılaşılan dosya ve klasör yapısını tanımak, buradaki dosyaların her birinin farklı bir işlevinin olduğunu bilmek, projeye hâkimiyet sağlar (Görsel 6.1).

Görsel 6.1: Mobil uygulama geliştirme için Project alanı

6.1.1. AndroidManifest.xml Dosya Yapısı

AndroidManifest, **manifest** klasörünün içinde yer alan ve mobil uygulama geliştirme ortamında tasarlanan uygulamalar için vazgeçilmez dosya yapısıdır. Bu dosya içinde Activity, Service, Receiver vb. sınıfları ve projeye ait temel bilgiler bulunur. Dosya uzantısı, xml formatında uzantı olduğu için okunması ve değiştirilmesi oldukça kolaydır. Görsel 6.2'de yeni oluşturulmuş bir mobil uygulama geliştirme projesinin AndroidManifest.xml dosya içeriği verilmiştir. Burada kodlar "Tag" (etiket) denilen < > işaretleri arasında yazılır. Bu etiketler < > şeklinde başlayıp </> şeklinde kapatılır. Görsel 6.2'de kodlar içinde etiketler verilmiştir.

228 Uygulama Tasarımı

Mobil Uygulamalar

Görsel 6.2: AndroidManifest.xml içeriği

- **manifest**

Bu etiket ile uygulamaların versiyon numaraları, paket isimleri, uygulama izinleri gibi mobil uygulamanın en temel özellikleri girilir. Projenin içinde yer alan kütüphanelerden minimum ve maksimum sdk sürümlerine kadar bu bölümde ayarlanır.

- **package**

Uygulamanın paket adının bulunduğu yapı türüdür. Genellikle mobil uygulamalar Görsel 6.2'de görüldüğü gibi " " şeklinde bir domain olarak verilir. Bunun amacı, paket isminin **unique** (benzersiz) olarak tanımlanmasıdır.

NOT: AndroidManifest.xml içinde, manifest etiketi arasında yer alan package ile oluşturulmuş paket adının tamamen uygulama geliştiriciye ait olduğu unutulmamalıdır. Geliştirilen uygulama daha sonra markette yayımlanmak istediğinde de bu paket adı kullanılır. Bu nedenle geliştirilen her uygulamanın paket adının unique bir yapıda olmasına dikkat edilmelidir.

- **application**

Bu etiket ile geliştirilen mobil uygulamanın logosu (android:icon), adı (android:label), teması (android:theme) vb. temel özellikleri ayarlanır. Application etiketleri arasına donanım hızlandırma veya yedekleme gibi temel özellikler de eklenebilir. Görsel 6.2'de "application" etiketi arasına bakıldığında "android:" koduyla birlikte verilen özelliklerin girildiği görülür.

- **activity**

Geliştirilen mobil uygulamaya ait activitylerin ve bu activitylere ait özelliklerin yer aldığı etikettir.

- **intent-filter**

Activity, Service ve Broadcast Receiver bileşenlerine intent tipini belirtmek için kullanılan etiket tir. intent-filter içinde <action>, <category>, <data> etiketleri bulunur. intent-filter içine <action> etiketinin mutlaka eklenmesi gerekir.

6. Öğrenme Birimi 229

Konu başlıklarını gösterir.

Öğrenme biriminin adını gösterir.

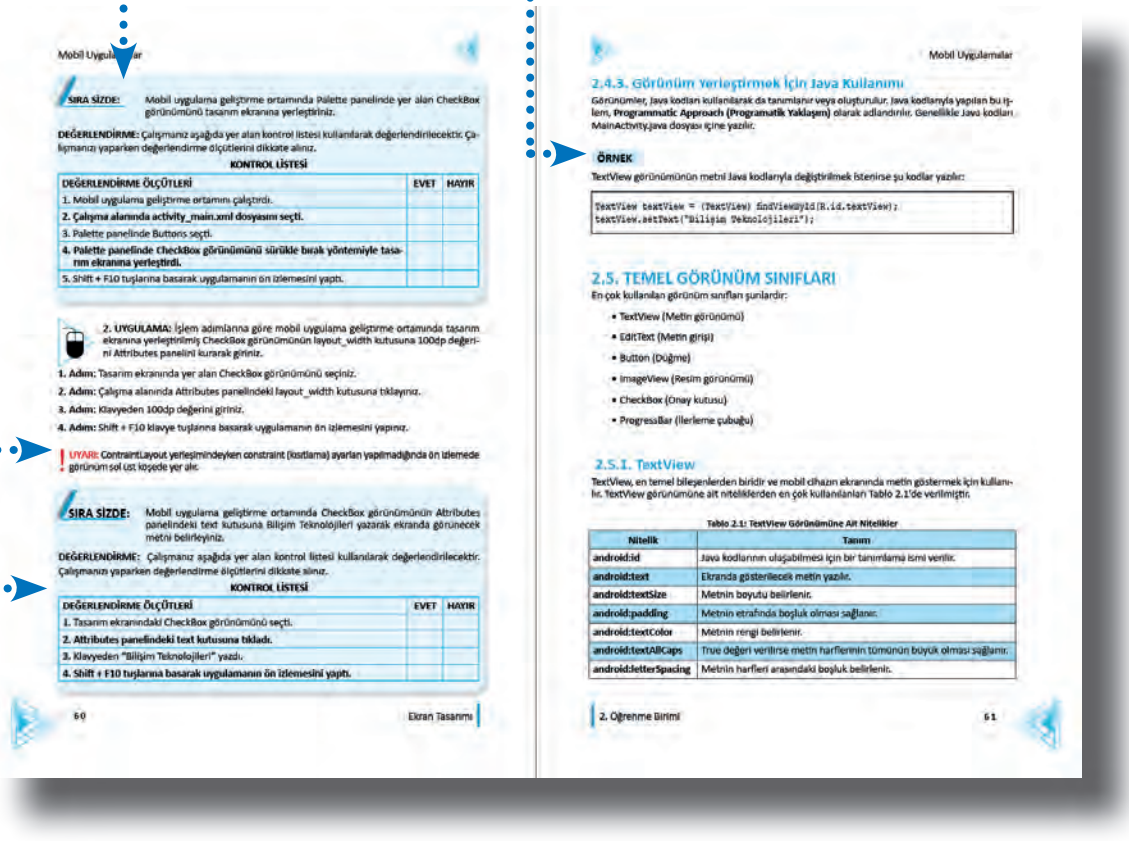
Öğrenme biriminin numarasını gösterir.

Sayfa numarasını gösterir.

Konu ile ilgili uyarıları gösterir.

Konu içindeki öğrenci çalışmalarını gösterir.

Konu ile ilgili örnekleri gösterir.



Mobil Uygulamalar

SIRA SİZDE: Mobil uygulama geliştirme ortamında Palette panelinde yer alan CheckBox görünümünü tasarım ekranına yerleştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme ortamını çalıştırdı.		
2. Çalışma alanında activity_main.xml dosyasını seçti.		
3. Palette panelinde Buttons seçti.		
4. Palette panelinde CheckBox görünümünü sürükle bırak yöntemiyle tasarım ekranına yerleştirdi.		
5. Shift + F10 tuşlarına basarak uygulamanın ön izlemesini yaptı.		

2. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında tasarım ekranına yerleştirilmiş CheckBox görünümünün layout_width kutusuna 100dp değerini Attributes panelini kullanarak giriniz.

1. Adım: Tasarım ekranında yer alan CheckBox görünümünü seçiniz.
2. Adım: Çalışma alanında Attributes panelindeki layout_width kutusuna tıklayınız.
3. Adım: Klavyeden 100dp değerini giriniz.
4. Adım: Shift + F10 klavye tuşlarına basarak uygulamanın ön izlemesini yapınız.

UYARI: ContraintLayout yerleşimindeyken constraint (kısıtlama) ayarları yapılmadığında ön izlemede görünüm sol üst köşede yer alır.

SIRA SİZDE: Mobil uygulama geliştirme ortamında CheckBox görünümünün Attributes panelindeki text kutusuna Bilişim Teknolojileri yazarak ekranda görünecek metni belirleyiniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Tasarım ekranındaki CheckBox görünümünü seçti.		
2. Attributes panelindeki text kutusuna tıkladı.		
3. Klavyeden "Bilişim Teknolojileri" yazdı.		
4. Shift + F10 tuşlarına basarak uygulamanın ön izlemesini yaptı.		

60 Ekran Tasarımı

Mobil Uygulamalar

2.4.3. Görünüm Yerleştirmek İçin Java Kullanımı

Görünümler, Java kodları kullanılarak da tanımlanır veya oluşturulur. Java kodlarıyla yapılan bu işlem, **Programmatic Approach (Programatik Yaklaşım)** olarak adlandırılır. Genellikle Java kodları MainActivity.java dosyası içine yazılır.

ÖRNEK

TextView görünümünün metni Java kodlarıyla değiştirilmek istenirse şu kodlar yazılır:

```
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText("Bilişim Teknolojileri");
```

2.5. TEMEL GÖRÜNÜM SINIFLARI

En çok kullanılan görünüm sınıfları şunlardır:

- TextView (Metin görünümü)
- EditText (Metin girişi)
- Button (Düğme)
- ImageView (Resim görünümü)
- CheckBox (Onay kutusu)
- ProgressBar (İlerleme çubuğu)

2.5.1. TextView

TextView, en temel bileşenlerden biridir ve mobil cihazın ekranında metin göstermek için kullanılır. TextView görünümüne ait niteliklerden en çok kullanılanları Tablo 2.1'de verilmiştir.

Tablo 2.1: TextView Görünümüne Ait Nitelikler

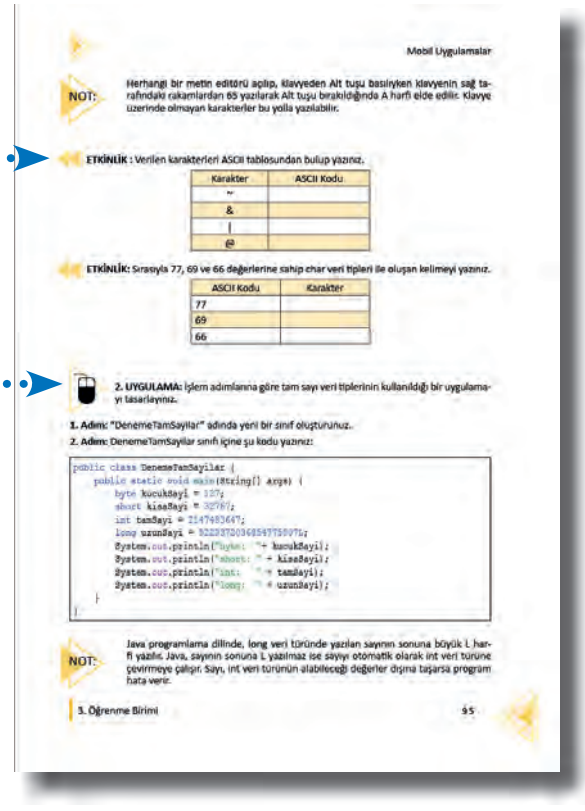
Nitelik	Tanım
android:id	Java kodlarının ulaşabilmesi için bir tanımlama ismi verilir.
android:text	Ekranda gösterilecek metin yazılır.
android:textSize	Metnin boyutu belirlenir.
android:padding	Metnin etrafında boşluk olması sağlanır.
android:textColor	Metnin rengi belirlenir.
android:textAllCaps	True değeri verilirse metin harflerinin tümünün büyük olması sağlanır.
android:letterSpacing	Metnin harfleri arasındaki boşluk belirlenir.

2. Öğrenme Birimi 61

Öğrencilerin yapacağı çalışmaların kontrolünde ve değerlendirilmesinde kullanılan ölçütleri gösterir.

Konu içindeki öğrenci etkinliklerini gösterir.

Konu içindeki uygulamaları gösterir.



Mobil Uygulamalar

NOT: Herhangi bir metin editörü açılıp, klavyeden Alt tuşu basılıyken klavyenin sağ tarafındaki rakamlardan 65 yazılarak Alt tuşu bırakıldığında A harfi elde edilir. Klavye üzerinde olmayan karakterler bu yolla yazılabilir.

ETKİNLİK : Verilen karakterleri ASCII tablosundan bulup yazınız.

Karakter	ASCII Kodu
~	
&	
|	
@	

ETKİNLİK: Sırasıyla 77, 69 ve 66 değerlerine sahip char veri tipleri ile oluşan kelimeyi yazınız.

ASCII Kodu	Karakter
77	
69	
66	

2. UYGULAMA: İşlem adımlarına göre tam sayı veri tiplerinin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "DenemeTamSayilar" adında yeni bir sınıf oluşturunuz.

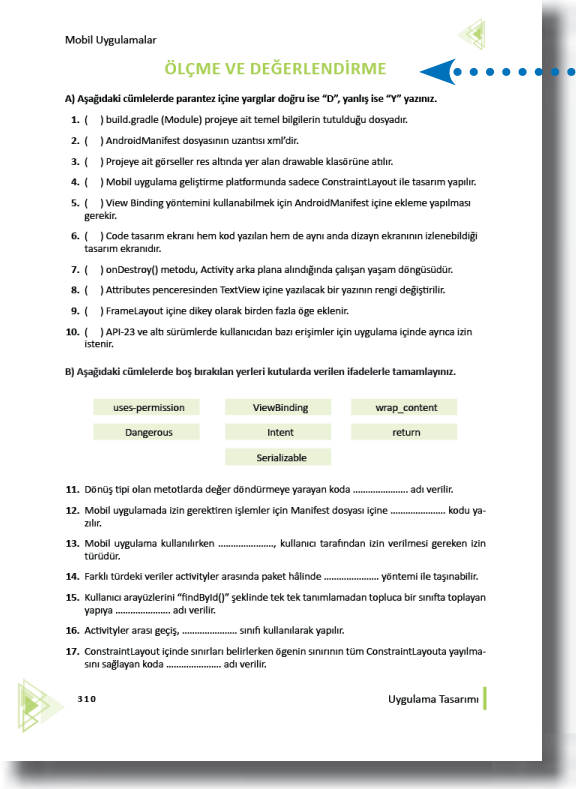
2. Adım: DenemeTamSayilar sınıfı içine şu kodu yazınız:

```
public class DenemeTamSayilar {
    public static void main(String[] args) {
        byte kucukSayi = 125;
        short kisaSayi = 32767;
        int tamSayi = 2147483647;
        long uzunSayi = 9223372036854775975L;
        System.out.println("byte:  " + kucukSayi);
        System.out.println("short: " + kisaSayi);
        System.out.println("int:   " + tamSayi);
        System.out.println("long:  " + uzunSayi);
    }
}
```

NOT: Java programlama dilinde, long veri türünde yazılan sayının sonuna büyük L harfi yazılır. Java, sayının sonuna L yazılmaz ise sayıyı otomatik olarak int veri türüne çevirmeye çalışır. Sayı, int veri türünün alabileceği değerler dışına taşarsa program hata verir.

3. Öğrenme Birimi 95

Ölçme ve değerlendirme çalışmalarını gösterir.



Mobil Uygulamalar

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) build.gradle (Module) projeye ait temel bilgilerin tutulduğu dosyadır.
2. ( ) AndroidManifest dosyasının uzantısı xml'dir.
3. ( ) Projeye ait görseller res altında yer alan drawable klasörüne atılır.
4. ( ) Mobil uygulama geliştirme platformunda sadece ConstraintLayout ile tasarım yapılır.
5. ( ) View Binding yöntemini kullanabilmek için AndroidManifest içine ekleme yapılması gerekir.
6. ( ) Code tasarım ekranı hem kod yazılan hem de aynı anda dizayn ekranının izlenebildiği tasarım ekranıdır.
7. ( ) onDestroy() metodu, Activity arka plana alındığında çalışan yaşam döngüsüdür.
8. ( ) Attributes penceresinden TextView içine yazılacak bir yazının rengi değiştirilir.
9. ( ) FrameLayout içine dikey olarak birden fazla öge eklenir.
10. ( ) API-23 ve altı sürümlerde kullanıcıdan bazı erişimler için uygulama içinde ayrıca izin istenir.

B) Aşağıdaki cümlelerde boş bırakılan yerleri kutularda verilen ifadelerle tamamlayınız.

uses-permission	ViewBinding	wrap_content
Dangerous	Intent	return
	Serializable	

11. Dönüş tipi olan metotlarda değer döndürmeye yarayan koda ..... adı verilir.
12. Mobil uygulamada izin gerektiren işlemler için Manifest dosyası içine ..... kodu yazılır.
13. Mobil uygulama kullanılırken ....., kullanıcı tarafından izin verilmesi gereken izin türüdür.
14. Farklı türdeki veriler activityler arasında paket hâlinde ..... yöntemi ile taşınabilir.
15. Kullanıcı arayüzlerini "findById()" şeklinde tek tek tanımlamadan topluca bir sınıfta toplayan yapıya ..... adı verilir.
16. Activityler arası geçiş, ... sınıfı kullanılarak yapılır.
17. ConstraintLayout içinde sınırları belirlerken ögenin sınırının tüm ConstraintLayouta yayılmasını sağlayan koda ..... adı verilir.

310 Uygulama Tasarımı

1. ÖĞRENME BİRİMİ

MOBİL UYGULAMA GELİŞTİRMEYE HAZIRLIK

KONULAR

1.1. TEMEL BİLEŞENLER

1.2. EMÜLATÖR KURULUMU

1.3. TASARIM YAPILARI (ACTIVITY TEMPLATES)

1.4. PROJE OLUŞTURMA

1.5. DOSYA VE DİZİN YAPILARI

NELER ÖĞRENECEKSİNİZ?

- Java Development Kit kurulumu
- Android Studio kurulumu
- Emülatör kurulumu
- Temel tasarım yapıları oluşturma
- Proje oluşturma
- Projelerdeki dosya ve dizin yapılarını kullanma

ANAHTAR KELİMELER

- Activity
- Android Studio
- Emülatör
- Java
- JDK
- XML
- Proje

HAZIRLIK ÇALIŞMALARI

1. Cep telefonunda en çok kullandığınız uygulama hangisidir? Sizce bu uygulamaya eklenebilecek farklı özellikler neler olabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Mobil uygulama geliştirme ortamında nasıl bir uygulama tasarlamak istediğinizi arkadaşlarınızla paylaşınız.

1.1. TEMEL BİLEŞENLER

Mobil uygulama kavramı, taşınabilir cihazlar için kodlanmış yazılımlardır. Taşınabilir cihazlar veya mobil işletim sistemi kullanılan cihazlar denildiğinde ilk akla gelenler şunlardır:

- Akıllı telefonlar
- Tabletler
- E-kitap okuyucu
- Akıllı televizyonlar
- Otomobiller
- Akıllı saatler

Taşınabilir cihazlarda yaygın olarak kullanılan işletim sistemleri Android ve IOS'tur. Mobil uygulama geliştirilirken tercih edilebilecek yöntemler şunlardır:

- **Native (Yerel) Uygulama:** Belirli bir işletim sistemi veya cihaz için geliştirilen uygulamalardır. Bu uygulamalar, yazılımın tasarlandığı işletim sistemi ve cihazda çalışır, farklı işletim sistemi ve cihazda çalışmaz.
- **Cross Platform (Çapraz Platform) Uygulama:** Birden fazla işletim sistemi veya cihazda çalışabilecek şekilde geliştirilen uygulamalardır.

Android işletim sistemi, Linux çekirdeğini kullanır ve açık kaynak kodludur. Bu işletim sistemi; taşınabilir cihazlarda, akıllı televizyonlarda ve oyun konsollarında yaygın kullanımı nedeniyle Java programlama diliyle birlikte tercih edilir.

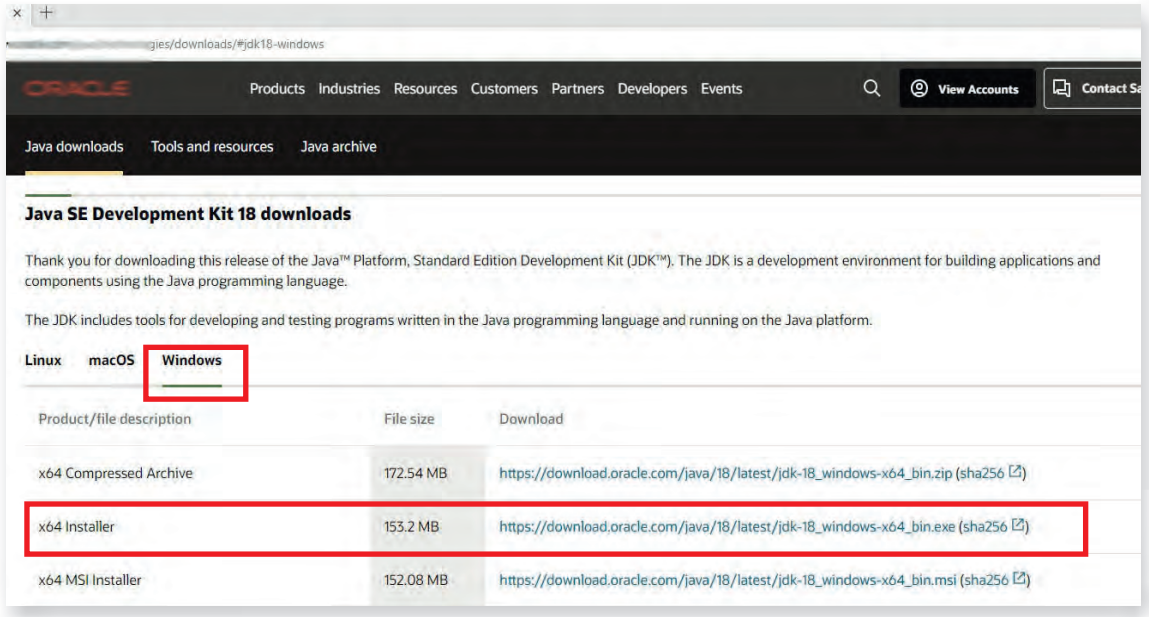
Android işletim sisteminde Java programlama diliyle mobil uygulama geliştirmek için **Java Software Development Kit** ve **Android Studio + SDK** olmak üzere iki temel programın bilgisayara kurulması gerekir.

1.1.1. Java Software Development Kit (JDK) Kurulumu

Java Runtime Environment (JRE) ve Java Software Development Kit (JDK) olmak üzere iki farklı Java paketi vardır. JRE, Java programlama dilinde yazılmış programları çalıştırmak için kullanılırken JDK ise yazılımcıların Java programları geliştirmesinde kullanılır.

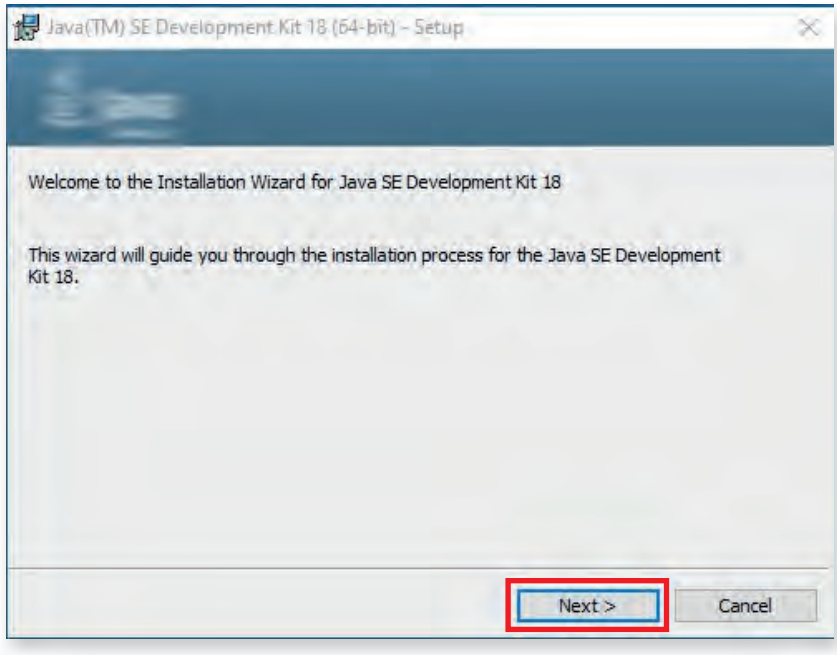
Herhangi bir web tarayıcının adres çubuğunda https://www.oracle.com/java/technologies/downloads/ adresine gidildiğinde en güncel JDK sürümüne ulaşılır ve kurulum programı bilgisayara indirilir.

İşletim sistemine uygun JDK kurulum dosyası seçildikten sonra indirme bağlantısı tıklanır (Görsel 1.1).



Görsel 1.1: JDK web sayfası

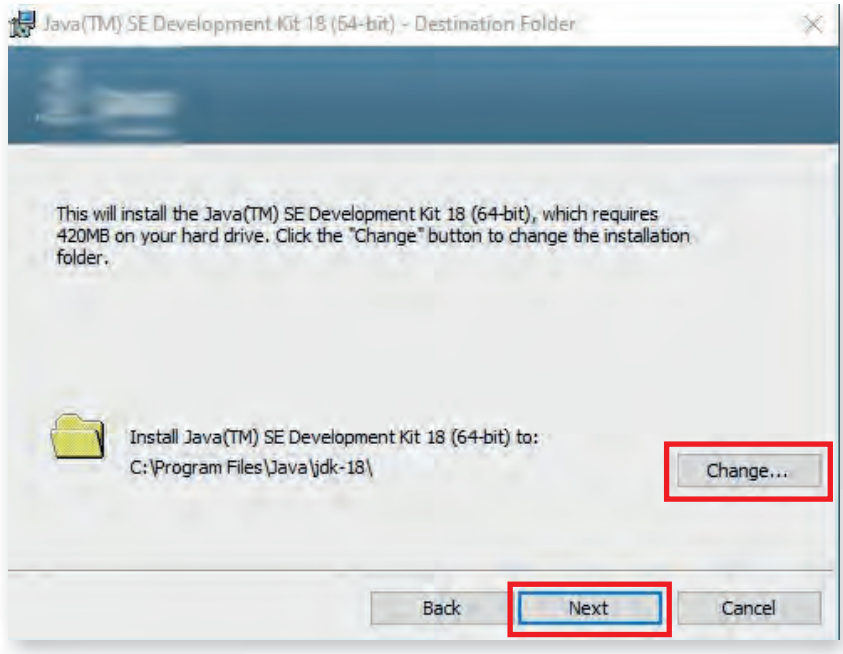
Dosya indirildikten sonra dosyanın üzerine tıklanarak çalıştırılır. JDK kurulum dosyası çalıştırılınca ilk olarak bir karşılama penceresi gelir. Bu pencerede kullanıcıya kurulum boyunca rehberlik yapılacağına dair bir bilgi verilir. Bu pencere **Next (İleri)** düğmesine tıklanarak geçilir (Görsel 1.2).



Görsel 1.2: JDK kurulum karşılama penceresi

Karşılama penceresinden sonra kullanıcıyı JDK programının hangi klasörlere kurulacağını ifade eden ve **Change (Değiştir)** düğmesi ile bu klasörlerin değiştirilebileceği ikinci bir pencere gelir.

Öngörülen kurulum klasörleri zorunlu değilse değiştirilmemelidir. Next düğmesiyle bu pencere de geçilir (Görsel 1.3).



Görsel 1.3: JDK kurulum klasörleri

Son olarak kurulumun başarıyla tamamlandığını ifade eden bir pencere ile kullanıcıya bilgi verilir. Bu pencerede yer alan **Close (Kapat)** düğmesine tıklanarak kurulum tamamlanır (Görsel 1.4).

Görsel 1.4: JDK kurulumunun tamamlanması

SIRA SİZDE: Java Software Development Kit kurulumunu yapınız.

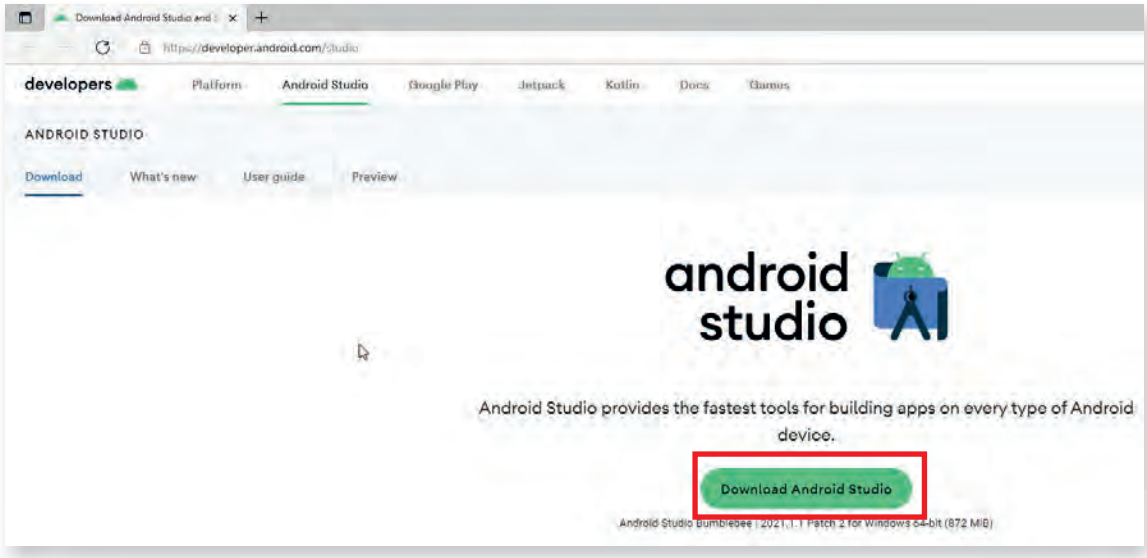
DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Java Software Development Kit programının web sayfasına gitti.		
2. Java Software Development Kit programını indirdi.		
3. Java Software Development Kit kurulum programını çalıştırdı.		
4. Java Software Development Kit programının kurulum aşamalarını yaptı.		

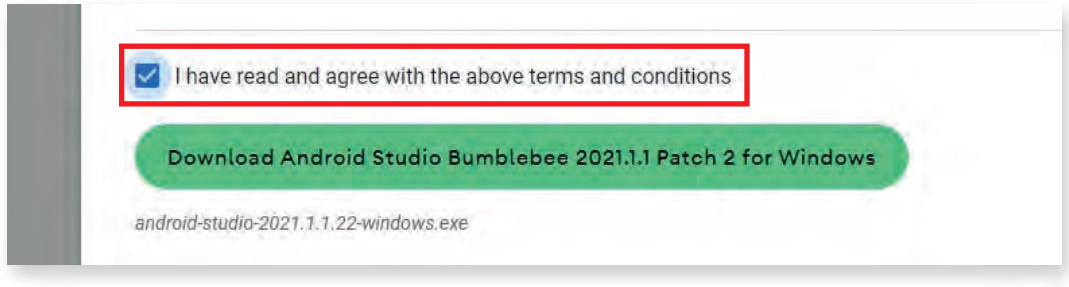
1.1.2. Android Studio ve Software Development Kit (SDK) Kurulumu

JDK kurulumu tamamlandıktan sonra Java diliyle mobil uygulama geliştirme ortamı olarak Android Studio programının kurulması gerekir. Android Studio programının güncel sürümünü https://developer.android.com/studio web adresinden indirmek mümkündür (Görsel 1.5).



Görsel 1.5: Android Studio indirme sayfası

Android Studio indirme sayfasındaki **Download Android Studio** düğmesine tıklanır. Yeni gelen pencerede kullanıcı sözleşmesinin kabul edildiğini ifade eden onay kutusu işaretlendikten sonra **Download Android Studio…** düğmesi tıklanarak indirme başlatılır (Görsel 1.6).

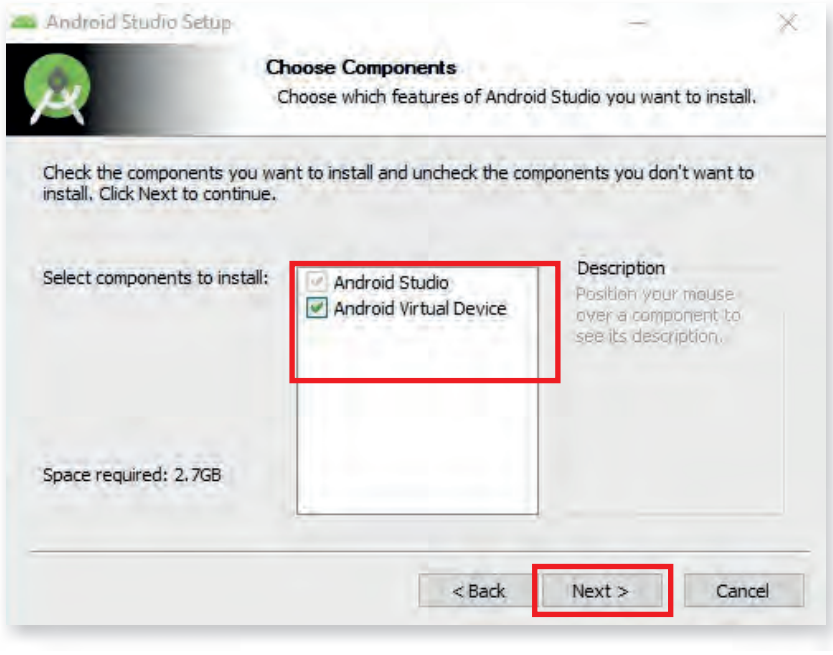


Görsel 1.6: Sözleşme onay kutusu

Android Studio kurulum dosyasının indirme işlemi bittikten sonra dosya çalıştırılır. Kurulum dosyalarının doğruluğu kontrol edilir ve gelen karşılama penceresinde Next düğmesine tıklanarak kurulum devam ettirilir (Görsel 1.7).

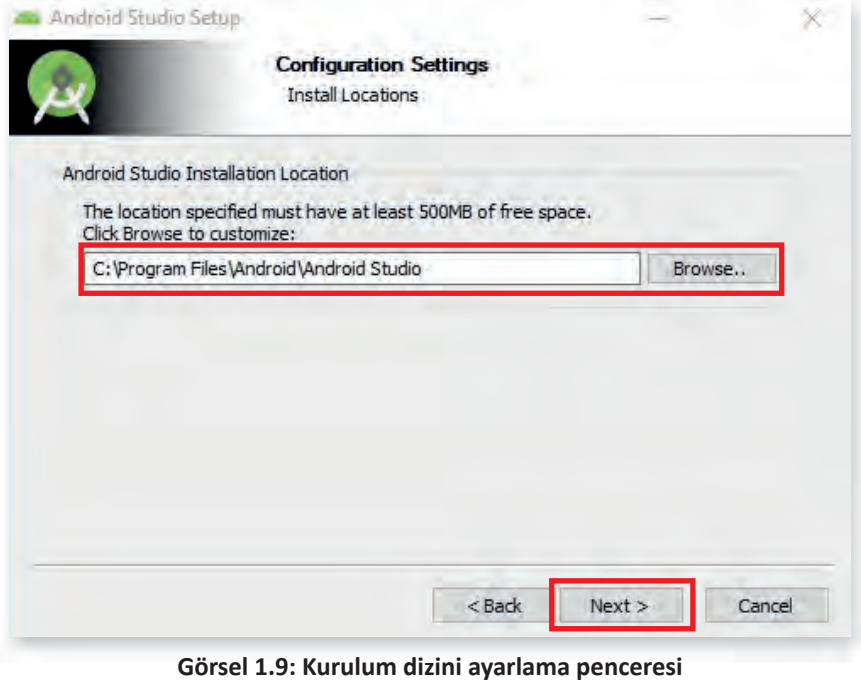
Görsel 1.7: Android Studio kurulum karşılama penceresi

Kurulumun bir sonraki aşamasında hangi bileşenlerin kurulacağının seçilebileceği onay kutularının bulunduğu pencere gelir. Bu pencerede Android Studio onay kutusu zorunlu olarak seçilidir ve işaret değiştirilemez ancak Android Virtual Device (Android Sanal Cihaz) isimli ikinci bileşenin kurulumu isteğe bağlıdır. Bu aşamada her iki onay kutusu da işaretliyken Next düğmesine tıklanır (Görsel 1.8).



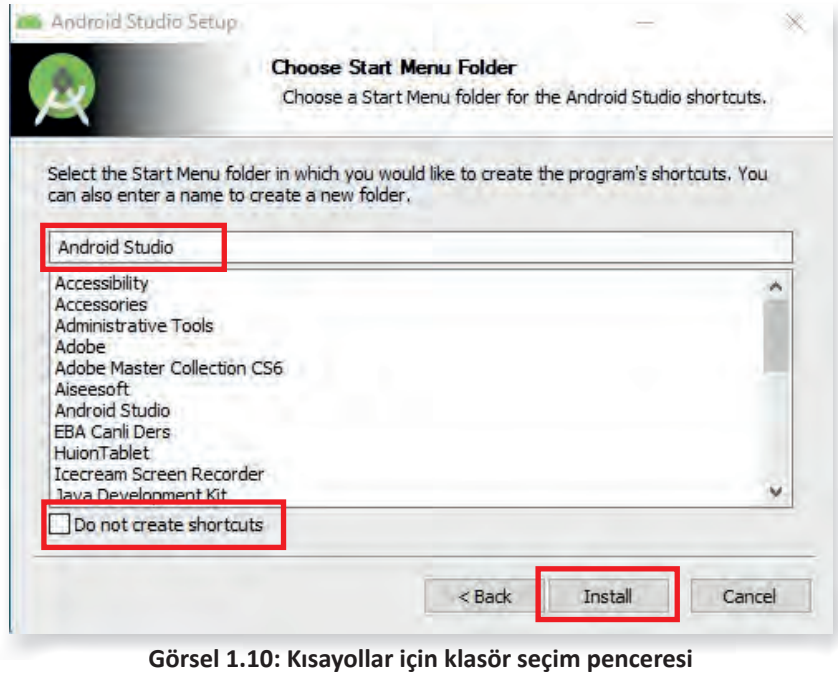
Görsel 1.8: Kurulacak bileşenlerin seçim penceresi

Bir sonraki aşamada ise Android Studio programının kurulacağı dizin ile ilgili bilgilerin girilebileceği bir pencere gelir. Bu penceredeki dizin bilgileri zorunlu hâller dışında değiştirilmemelidir. Next düğmesine tıklanarak bir sonraki aşamaya geçilir (Görsel 1.9).



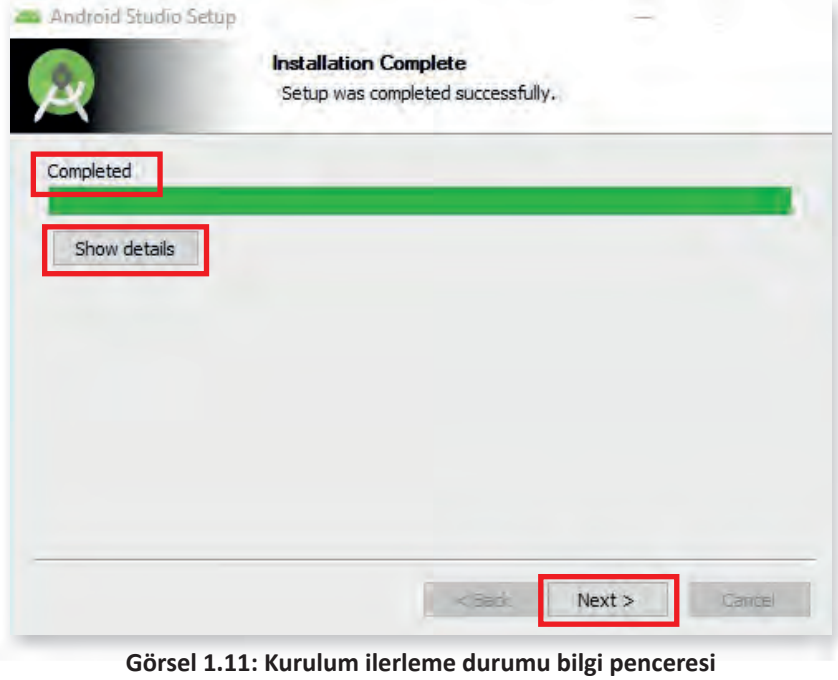
Görsel 1.9: Kurulum dizini ayarlama penceresi

Kullanıcı, bu aşamada Android Studio kısayollarının hangi klasörde oluşturulacağını veya yeni bir isim belirleyerek oluşan klasörde kısayolların yer almasını sağlayabilir. Pencerenin altındaki **Do not create shortcuts (Kısayol oluşturma)** onay kutusu işaretlenerek kısayol oluşturulmaması da seçilebilir (Görsel 1.10). Bu pencerede **Install (Yükle)** düğmesi tıklanarak kurulum başlatılır.



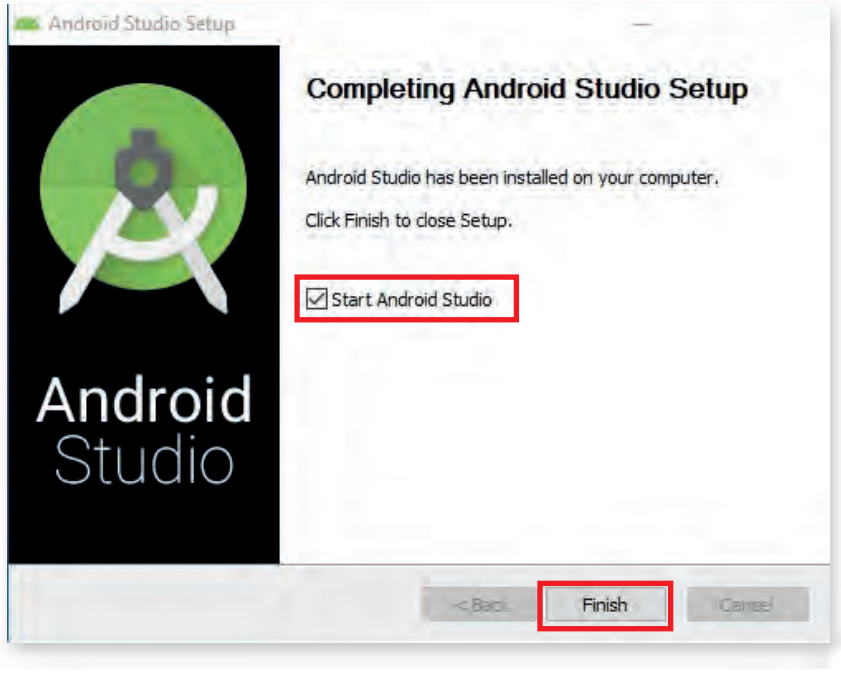
Görsel 1.10: Kısayollar için klasör seçim penceresi

Kurulum başlatıldıktan sonra gelen pencerede kurulumun ne kadarının tamamlandığını ifade eden yeşil renkli yükleme çubuğu ve ayrıntıların gözlemlenebileceği **Show details (Ayrıntıları göster)** düğmesi yer alır. Kurulum işlemi bitince yeşil renkli yükleme çubuğunun üzerinde **Completed (Tamamlandı)** yazısı belirir. Next düğmesine tıklanır (Görsel 1.11).



Görsel 1.11: Kurulum ilerleme durumu bilgi penceresi

Gelen son pencerede Android Studio programının kurulumunun bittiğini ifade eden bilgi ve **Finish (Bitir)** düğmesine tıklandıktan sonra programın başlatılmasını sağlayan **Start Android Studio (Android Studio Başlat)** onay kutusu yer alır. Bu onay kutusu işaretli olarak gelir. Finish düğmesine tıklandıktan sonra Android Studio başlatılmayacaksa bu onay kutusundaki işaret kaldırılır. Android Studio başlatılacaksa onay kutusundaki işaret kaldırılmadan Finish düğmesine tıklanarak kurulum sonlandırılır (Görsel 1.12).



Görsel 1.12: Kurulumu tamamlama penceresi

SIRA SİZDE: Android Studio kurulumunu yapınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

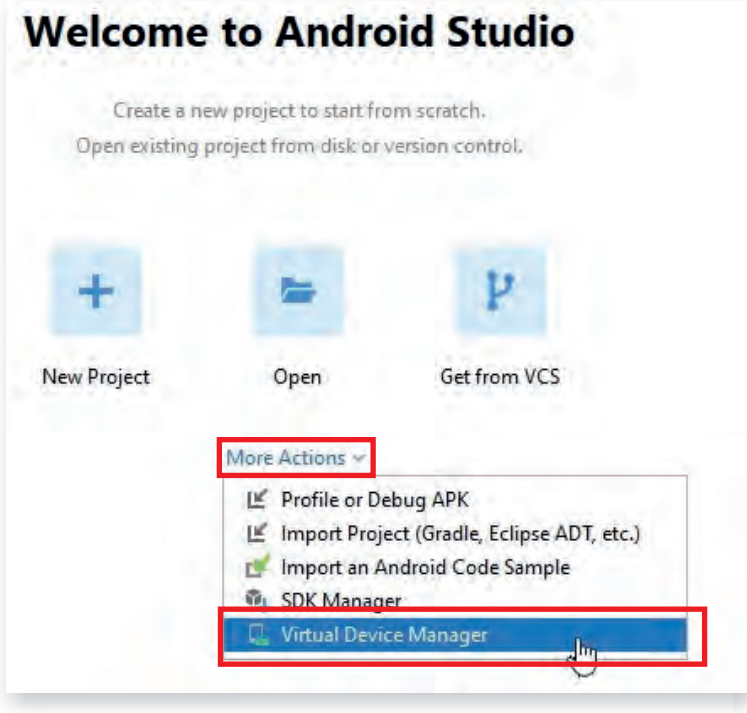
KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Android Studio programının web sayfasına gitti.		
2. Android Studio programını indirdi.		
3. Android Studio kurulum programını çalıştırdı.		
4. Android Studio programının kurulum aşamalarını yaptı.		

1.2. EMÜLATÖR KURULUMU

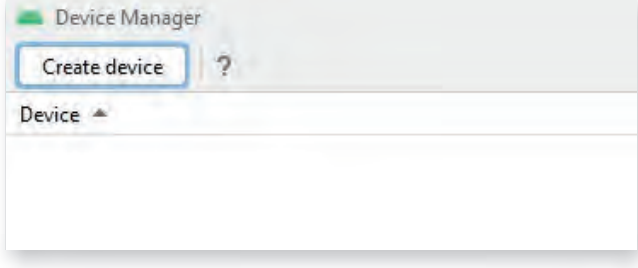
Farklı işletim sistemine ait programları yerel işletim sisteminde çalıştırmak için aracı yazılımlara ihtiyaç duyulur. Bu aracı yazılımlara emülatör ismi verilir. Android Studio içinde mobil işletim sistemine uygun emülatör programı kurulabilir. Geliştirilen uygulamalar bu emülatörde test edilebilir.

Kurulumdan sonra Android Studio çalıştırıldığında karşılama ekranında **More Actions (Ek Eylemler)** listesindeki **Virtual Device Manager (Sanal Cihaz Yöneticisi)** seçeneği tıklanır (Görsel 1.13).



Görsel 1.13: Virtual Device Manager seçimi

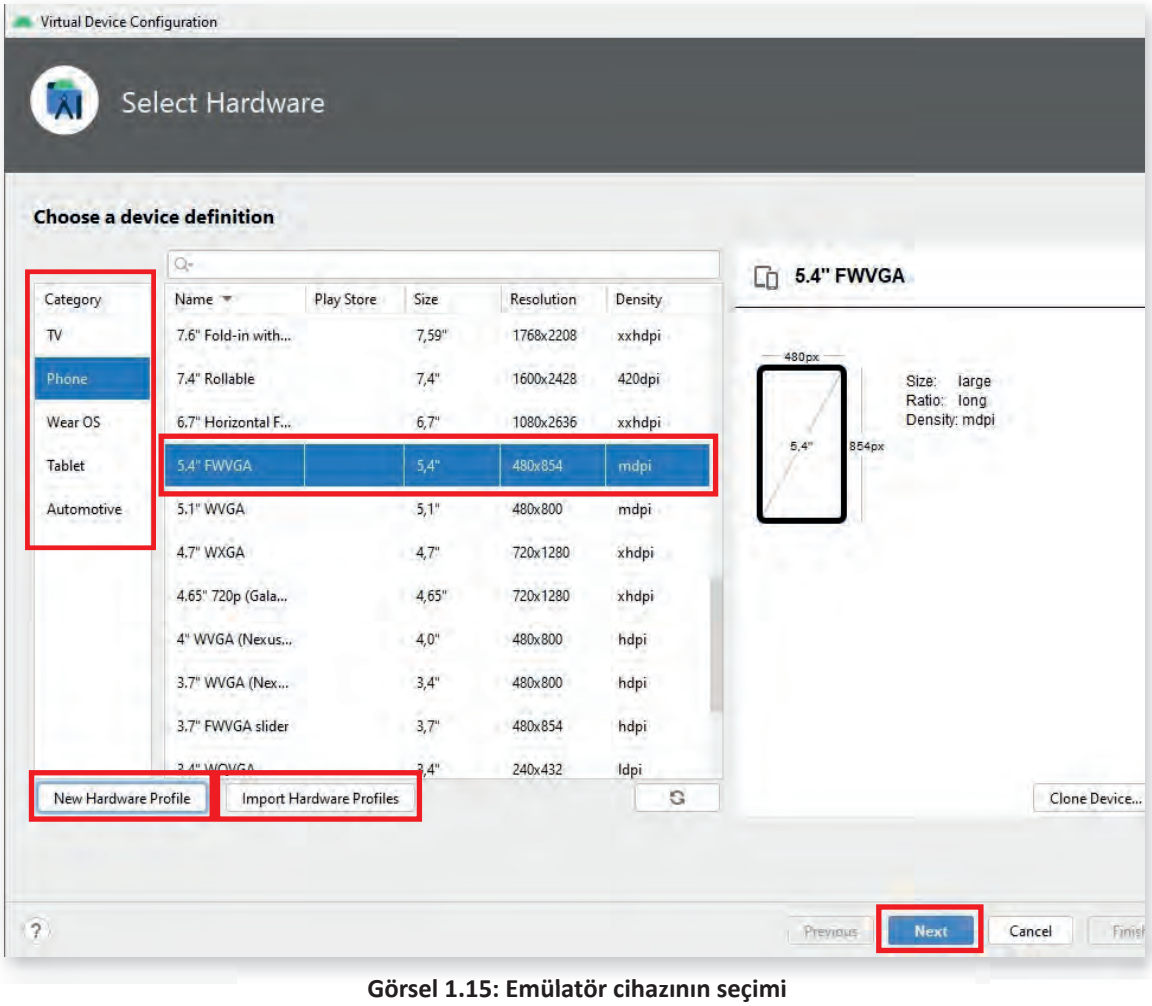
Açılan **Device Manager (Cihaz Yöneticisi)** penceresindeki **Create device (Cihaz oluştur)** düğmesine tıklanır (Görsel 1.14).



Görsel 1.14: Create device düğmesi

Select Hardware (Donanım Seç) penceresi açılır. Bu pencerede **Category (Kategori)** kısmında TV, Phone (Cep telefonu), Wear OS (Giyilebilir cihazlar), Tablet ve Automotive (Otomotiv tabletleri) yer alır. Uygulamanın çalıştırılacağı cihaz hangi kategorideyse o kategori seçilerek uygun ekran büyüklüğü ve çözünürlük belirlenir. Görsel 1.15'te 480 piksel genişliğinde, 854 piksel uzunluğunda ve 5.4 inch ekran büyüklüğüne sahip bir cep telefonu cihazı seçilmiştir. İhtiyaç duyulması hâ-

linde **New Hardware Profile (Yeni Donanım Profili)** düğmesiyle boyut ve çözünürlüğünü kullanıcının ayarlayabileceği bir cihaz oluşturulabilir. **Import Hardware Profiles (Donanım Profili İçe Aktar)** düğmesiyle daha önceden dosyaya kaydedilmiş bir donanım profili yüklenebilir. **Clone Device (Cihazı Kopyala)** düğmesi kullanılarak da belirli bir cihazın özellikleri kopyalanır. Kopyalanan cihazın birkaç özelliğinde değişiklik yapılabilir. Next düğmesi tıklanarak bir sonraki pencereye geçilir.



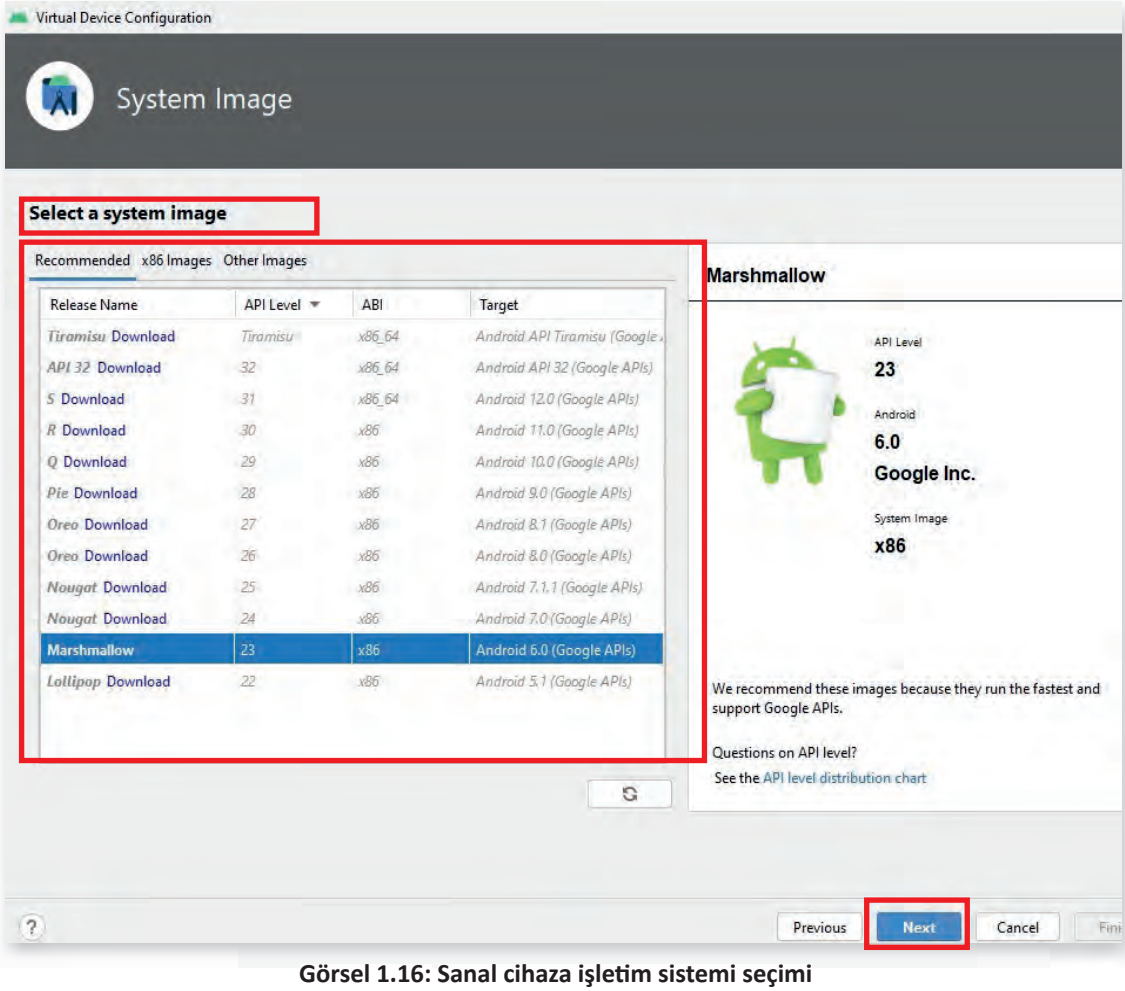
Görsel 1.15: Emülatör cihazının seçimi

Select a system image (Sistem imajı seçimi) penceresinde emülatör cihazına bir işletim sistemi seçilir. Pencerenin üstünde **Recommended (Tavsiye edilen)**, **x86 Images (x86 İşlemciler için imajlar)** ve **Other Images (Diğer imajlar)** olmak üzere üç adet sekme yer alır.

- **Recommended:** Bu sekme seçiliyken en hızlı çalışacak işletim sistemi imajı kullanıcıya görüntülenir.
- **x86 Images:** x86 işlemcilerde çalışacak işletim sistemi imajları görüntülenir. 32 bitlik işlemciler için listede x86 olarak belirtilmişken 64 bitlik işlemciler için x86_64 olarak işletim sistemleri belirtilmiştir.
- **Other Images:** Diğer işlemcilerde çalışacak işletim sistemi imajları görüntülenir.

UYARI: Her Android işletim sistemi sürümü için tek bir API düzeyi belirlenmiştir. Application Programming Interface (Uygulama Programlama Arayüzü) kelimelerinin baş harfleriyle API isimlendirilmesi yapılır. API, yazılımların kendi aralarında iletişim kurmasını sağlayan bir yapıdır. API düzeyi düştükçe yazılan uygulamanın çalışacağı cihaz sayısı artar ancak uygulamada kullanılacak özellikler azalır. API düzeyi yükseldikçe uygulamada kullanılacak özellikler artar ancak kapsadığı cihaz sayısı azalır.

Recommended sekmesindeki Android işletim sistemlerinin isimlerinin hemen yanında Download düğmesi yer alır. Bu düğmeyle o işletim sisteminin imajı indirilir ve işletim sistemi seçime hazır hâle getirilir. Görsel 1.16 penceresinde Marshmallow (Android 6.0) işletim sistemi indirilmiş ve seçime hazır hâle getirilmiştir. Bu sürüm seçilir ve Next düğmesine tıklanır.

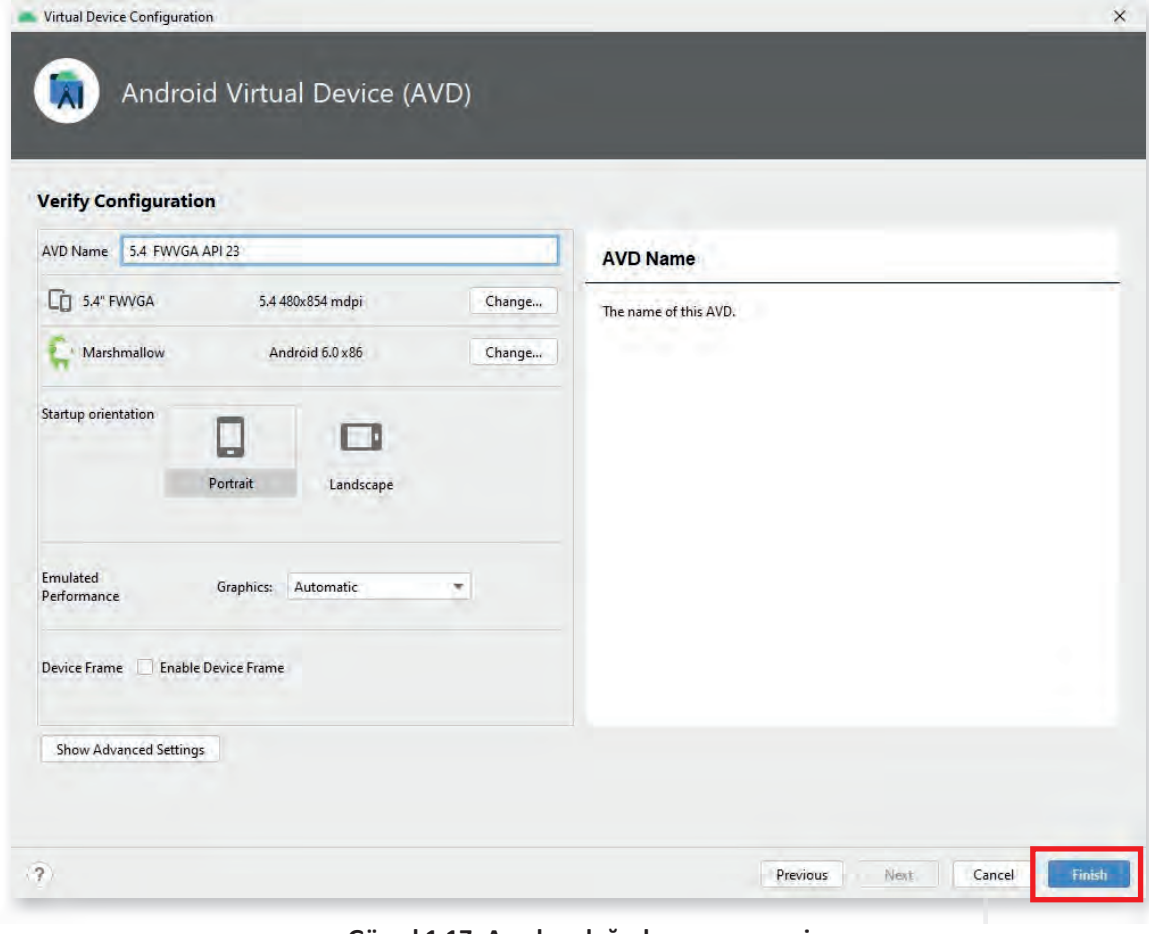


Görsel 1.16: Sanal cihaza işletim sistemi seçimi

Kurulumun son penceresinde oluşturulan ayarların doğruluğu kontrol edilebilir ve bu pencereden ayarlar değiştirilebilir (Görsel 1.17).

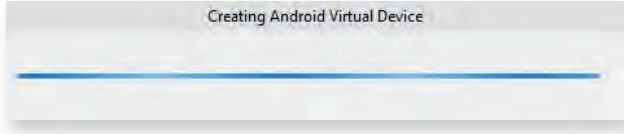
- **AVD Name (AVD İsmi):** Ayarlanan cihazın ismi bu kutuya yeni değer girilerek değiştirilebilir.
- **5.4" FWVGA 5.4 480x854 mdpi:** Bu yazının sağ tarafında yer alan **Change... (Değiştir)** düğmesi ile ekran boyutları farklı bir cihaz seçilebilir.
- **Marshmallow Android 6.0 x86:** Bu yazının sağ tarafında yer alan **Change...** düğmesi ile seçilen işletim sistemi değiştirilebilir.

- **Startup Orientation (Başlangıç Yönlendirmesi):** Bu bölümde yar alan iki farklı **Portrait (Portre) ve Landscape (Manzara)** simgelerinden biri seçilerek cihazın dikey ve yatay kullanılacağı belirlenebilir. Dikey kullanım için Portrait simgesi seçilir, yatay kullanım içinse Landscape simgesi seçilir.
- **Emulated Performance (Emülasyon Performansı):** Bu yazının sağ tarafındaki **Graphics: (Grafikler)** açılır kutusunda üç seçenek yer alır. **Automatic (Otomatik)**, **Hardware (Donanım)** ve **Software (Yazılım)** isimli seçeneklerden bilgisayarın grafik kartı kullanılacaksa emülasyon için Hardware seçilir. Grafik kartıyla problemler yaşanırsa ve grafik kartı yerine yazılım kullanılacaksa emülasyon için Software seçilir. Emülasyonun duruma göre otomatik olması istenirse Automatic seçilebilir.
- **Device Frame (Cihaz Çerçevesi):** Bu onay kutusu işaretlenerek sanal cihazın kenarında çeşitli işlev düğmeleri kullanılabilir hâle getirilir. Bu düğmeler istenmezse bu onay kutusu işaretlenmez.
- **Show Advanced Settings (Gelişmiş Ayarları Göster):** Sanal cihazla ilgili ayrıntılı ayarlar yapılması gerektiğinde bu düğme tıklanır. Bu düğme tıklandıktan sonra cihaz ön ve arka kamera ayarları, ana bellek miktarı, yardımcı depolama bellek miktarı, SD kart bellek miktarı, cihazın işlemci sayısı gibi ayrıntılı ayarların belirlenebileceği kutular pencerede belirir.



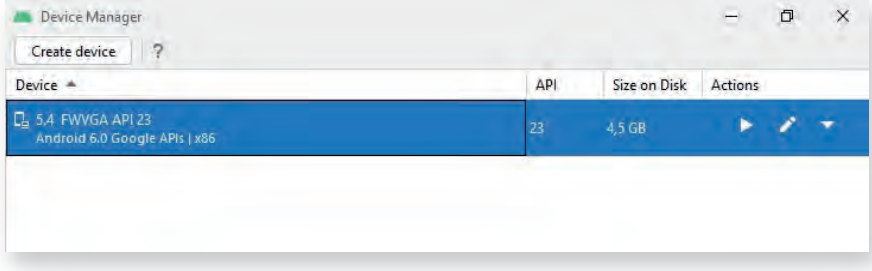
Görsel 1.17: Ayarları doğrulama penceresi

Finish düğmesine tıklandıktan sonra pencerenin ortasında **Creating Android Virtual Device (Android Sanal Cihaz Oluşturuluyor)** ifadesi görüntülenir (Görsel 1.18).



Görsel 1.18: Sanal cihaz oluşturuluyor bilgisi

Sanal cihaz oluşturulduktan sonra pencerenin en üstünde Görsel 1.19'daki bilgiler ve simgeler görünür.



Görsel 1.19: Sanal cihaz bilgileri

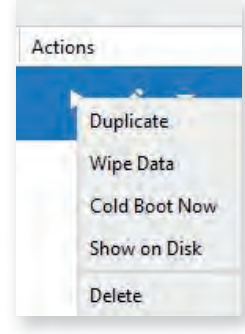
Bu bilgilerde sanal cihazın ismi, sanal cihazda yüklü olan işletim sistemi, API düzeyi, yardımcı depolama biriminde kapladığı alan ve Actions (Eylemler) simgeleri yer alır. Bu simgeler şunlardır:

Launch this AVD in the emulator (Bu Android Sanal Cihazı emülatörde çalıştır): Bu simge tıklanarak sanal cihaz çalıştırılır.

Edit this AVD (Bu Android Sanal Cihazı düzenle): Sanal cihazla ilgili değişiklik yapılması gerekirse bu simge tıklanır.

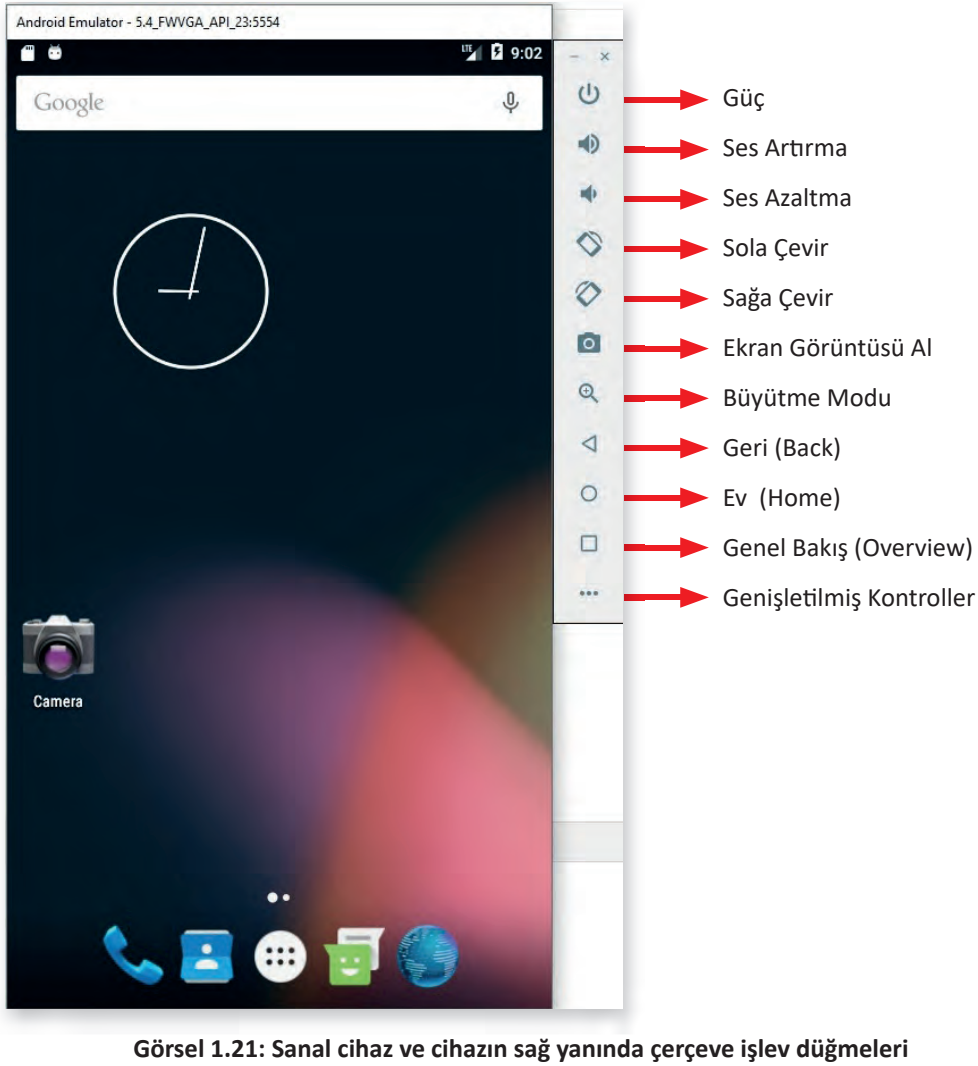
Bu simge tıklandığında açılır bir menü ile karşılaşılır (Görsel 1.20). Bu menüdeki seçenekler şunlardır:

- **Duplicate (Çoğaltma):** Sanal cihazın özelliklerine sahip ikinci bir kopya oluşturmak için kullanılır.
- **Wipe Data (Verileri Sil):** Sanal cihazı fabrika ayarlarına döndürmek için kullanılır. Sonradan düzenlenen verileri siler ve cihazı ilk açılış durumuna getirir.
- **Cold Boot Now (Soğuk Önyükleme Şimdi):** Sanal cihazın herhangi bir kaydedilmiş durumdan değil, en baştan açılmasını sağlar.
- **Show on Disk (Disk Üzerinde Göster):** Sanal cihazın dosyalarını bulunduğu klasörde gösterir.
- **Delete (Sil):** Sanal cihazın silinmesini sağlar.



Görsel 1.20: Açılır menü seçenekleri

Emülatörde sanal cihazı başlatmak için simgesine tıklanır. Her şey doğru ayarlandıysa Android işletim sistemi yüklü bir cep telefonu cihaz penceresi ekranda belirir (Görsel 1.21).



Görsel 1.21: Sanal cihaz ve cihazın sağ yanında çerçeve işlev düğmeleri

1. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında 5.1 inch büyüklüğünde Android 6.0 işletim sistemi kullanan bir emülatör kurulumunu yapınız.

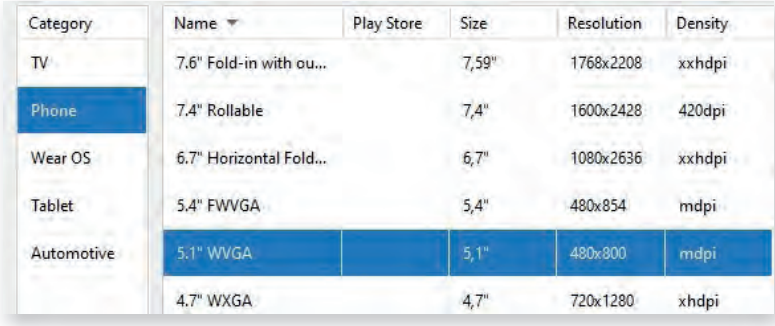
1. Adım: Mobil uygulama geliştirme programını çalıştırınız.

2. Adım: Karşılama ekranında More Actions seçeneğine tıklayınız.

3. Adım: Açılan listeden Virtual Device Manager seçeneğine tıklayınız.

4. Adım: Yeni gelen pencerede Create device düğmesine tıklayınız.

5. Adım: Select Hardware penceresinde Category kısmından Phone ve 5.1”WVGA seçiniz (Görsel 1.22).



Görsel 1.22: Emülatör seçimi

6. Adım: Next düğmesini tıklayınız.

7. Adım: Select a system image penceresinde Marshmallow seçiniz.

8. Adım: Next düğmesini tıklayınız.

9. Adım: Verify Configuration penceresinde Finish düğmesine tıklayınız.

SIRA SİZDE: Mobil uygulama geliştirme ortamında 6.7 inch büyüklüğünde Android 6.0 işletim sistemi kullanan bir emülatör kurulumunu yapınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme ortamını çalıştırdı.		
2. Karşılama ekranında More Actions seçeneğine tıkladı.		
3. Açılan listeden Virtual Device Manager seçeneğine tıkladı.		
4. Yeni gelen pencerede Create device düğmesine tıkladı.		
5. Select Hardware penceresinde Category kısmından Phone ve 6.7” Horizantal… seçti.		
6. Next düğmesine tıkladı.		
7. Select a system image penceresinde Marshmallow seçti.		
8. Next düğmesine tıkladı.		
9. Verify Configuration penceresinde Finish düğmesine tıkladı.		

1.3. TASARIM YAPILARI (ACTIVITY TEMPLATES)

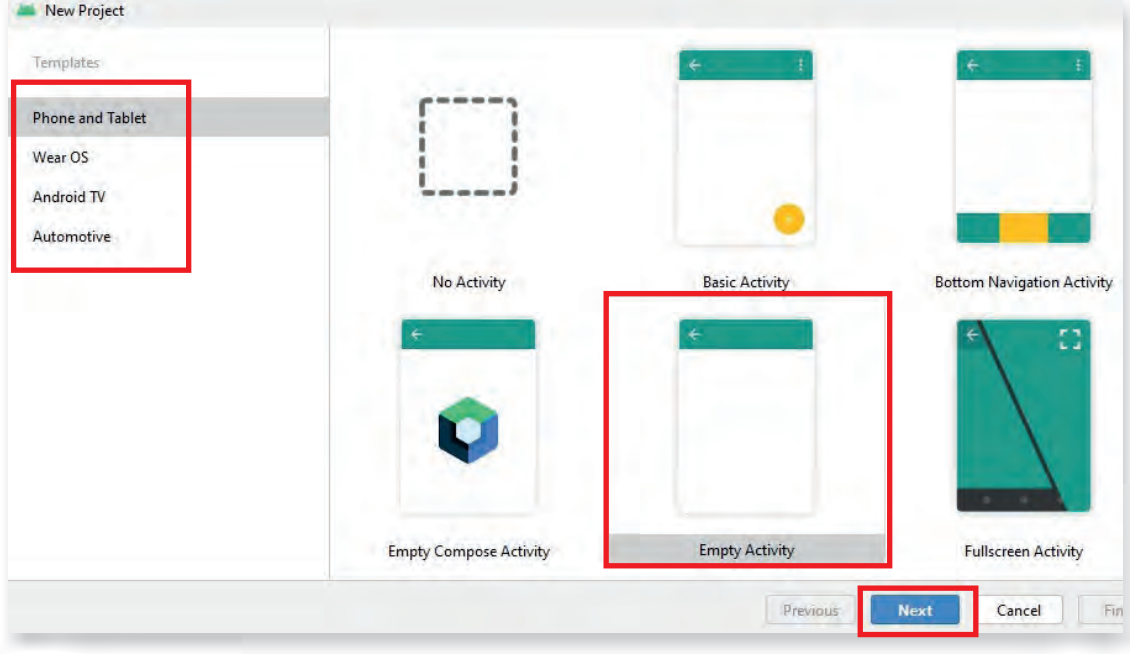
Emülatör kurulumu tamamlanıp pencereler kapatıldıktan sonra Android Studio karşılama ekranına tekrar dönülür (Görsel 1.23).

Görsel 1.23: Yeni Proje simgesi

Bu ekrandaki simgeler şunlardır:

- **New Project (Yeni Proje):** En baştan yeni bir proje oluşturulacaksa bu simge tıklanır.
- **Open (Aç):** Daha önce yerel sürücüye proje kaydedildiyse projeyi açmak için kullanılır.
- **Get from VCS (Sürüm Kontrol Sistemlerinden Yükle):** Git, GitHub gibi sitelerde proje geliştirilirse buradan proje yüklenebilir.

New Project simgesi tıklandıktan sonra Templates (Şablonlar) bölümünden Phone and Tablet seçilir. Uygulamaya Activity Templates (Tasarım Yapıları) eklenebilen seçim penceresi gelir (Görsel 1.24).



Görsel 1.24: Activity Templates

Bu pencerede uygulama geliştirilecek cihaza göre **Phone and Tablet (Telefon ve Tablet)**, **Wear Os (Giyilebilir Cihaz)**, **Android TV (Akıllı Televizyon)** ve **Automotive (Otomotiv)** olarak kategorilere ayrılmış Activity Templates yer alır.

Uygulama geliştirilmek istenen cihaz olarak Phone and Tablet seçilir ve hazır activitylerden **Empty Activity (Boş Activity)** şablonu seçilir. Next düğmesine basılarak işleme devam edilir.

SIRA SİZDE: Mobil uygulama geliştirme ortamında "Fullscreen Activity" tasarım yapısını seçerek uygulama geliştirmeye başlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme ortamını çalıştırdı.		
2. Karşılama ekranında New Project seçeneğine tıkladı.		
3. Açılan pencerede Phone and Tablet seçeneğine tıkladı.		
4. Pencerenin sağ tarafındaki bölümden Fullscreen Activity seçti.		
5. Next düğmesine tıkladı.		

1.3.1. Activity

Activity, bir Android uygulamasında kullanıcıya gösterilen ve üzerinde kullanıcı arayüz bileşenleri (düğmeler, onay kutuları, radyo düğmeleri vb.) yer alan ekran olarak tanımlanabilir. Activity, işletim sistemlerinde bulunan pencerelerle benzerlikler taşır. Bir Android uygulaması bir veya daha fazla activity içerebilir. Bu durum, uygulamanın bir veya daha fazla ekrana sahip olması anlamına gelir. Android uygulaması geliştirilirken bir activity seçilip projeye dâhil edilir çünkü Android uygulamaları kullanıcıya göstermek için bir veya daha fazla kullanıcı arayüz bileşeni içerir. Kullanıcı, bu bileşenler sayesinde uygulama ile etkileşime girer.

1.3.2. Activity Çeşitleri

Yeni bir proje oluşturulurken hazır activitylerden biri seçilmelidir. Hazır activitylerden bazıları şunlardır:

- **No Activity (Activity Yok):** Yeni bir boş proje oluşturmak anlamına gelir. Bu activity seçildiğinde ne bir XML dosyası ne de bir Java dosyası oluşturulur. Hiçbir dosya otomatik oluşturulmaz.
- **Basic Activity (Temel Activity):** Temel activity seçildiğinde mobil uygulamada bir menü

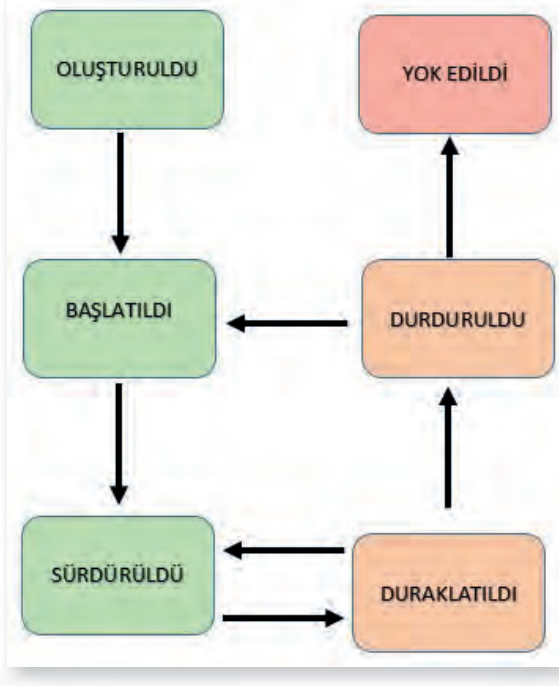
düğmesi ve kayan bir eylem düğmesi bulunur. Temel activity dosyaları otomatik olarak oluşturulur.

- **Bottom Navigation Activity (Alt Gezinme Activity):** Alt gezinme activity seçildiğinde mobil uygulamanın alt tarafında üç adet düğme oluşur. Bu düğmelere çoğu sosyal medya uygulamasında yer alan işlevler verilebilir. Bu activity içinde dosyalar otomatik oluşturulur.
- **Empty Activity (Boş Activity):** En çok kullanılan activitydir. Mobil uygulama geliştirilirken bu activity sık sık seçilir ve yeni bir boş activity oluşturulur.
- **Fullscreen Activity (Tam Ekran Activity):** Genellikle tam ekran çalışması gereken mobil uygulamalarda kullanılır. Sistem kullanıcı arabiriminin ve eylem çubuğunun görünürlüğünü değiştirir. Birçok uygulama slayt göstermek, video göstermek vb. için ekranın tamamını kullanır. Bu activity için uygun dosyalar otomatik olarak oluşturulur.
- **Login Activity (Giriş Activity):** Oturum açma mobil uygulaması geliştirmek için kullanılan activitydir. E-posta ve şifre girmek için alanlar ile bu alanları onaylamak için düğme bileşenleri ekranda yer alır. Bu activity için uygun dosyalar otomatik olarak oluşturulur.

1.3.3. Activity Yaşam Döngüsü

Android uygulaması içinde activitylerin bir yaşam döngüsü vardır. Program yazılırken yaşam döngüsünün iyi bilinmesi işleri kolaylaştırır.

Bir Android uygulaması ilk başlatıldığında main (ana) activity oluşturulur. Activity kullanıcıya hizmet vermeden önce **OLUŞTURULDU (CREATED), BAŞLATILDI (STARTED) ve SÜRDÜRÜLDÜ (RESUMED)** olmak üzere üç durumdan geçer (Görsel 1.25).



Görsel 1.25: Activity yaşam döngüsü

- Main activity, başka herhangi bir activity açarsa yeni açılan activityler de OLUŞTURULDU, BAŞLATILDI ve SÜRDÜRÜLDÜ durumlarından geçer.
- Bir A activity başka bir B activity açarsa A activity durumu **DURAKLATILDI (PAUSED)** hâline geçer. Kullanıcı, telefonun Geri (Back) tuşuna basarsa A activitye döner ve durumu SÜRDÜRÜLDÜ olur.
- Kullanıcı, Android cihazın ana ekranına dönerse tüm activityler önce DURAKLATILDI durumuna daha sonra da **DURDURULDU (STOPPED)** durumuna geçer. Kullanıcı, uygulamaya geri dönerse activity BAŞLATILDI ve SÜRDÜRÜLDÜ durumlarından geçer. Android cihazının hafizaya ihtiyacı olursa activitylerin durumu **YOK EDİLDİ (DESTROYED)** durumuna geçer. Bu sayede hafiza boşaltılır.

UYARI: Durumlar, Görsel 1.25'te yer alan okları takip eder. Bir activity OLUŞTURULDU durumundan SÜRDÜRÜLDÜ durumuna zıplayamaz. Sırasıyla OLUŞTURULDU, BAŞLATILDI ve SÜRDÜRÜLDÜ durumlarını geçer. Activityler SÜRDÜRÜLDÜ durumundan DURAKLATILDI durumuna geçebilir veya DURAKLATILDI durumundan SÜRDÜRÜLDÜ durumuna geçebilir. DURAKLATILDI durumundan DURDURULDU durumuna geçebilir ve oradan da BAŞLATILDI durumuna geçebilir fakat asla DURDURULDU durumundan SÜRDÜRÜLDÜ durumuna zıplayamaz.

1.3.4. Activity Yaşam Döngüsü Metotları

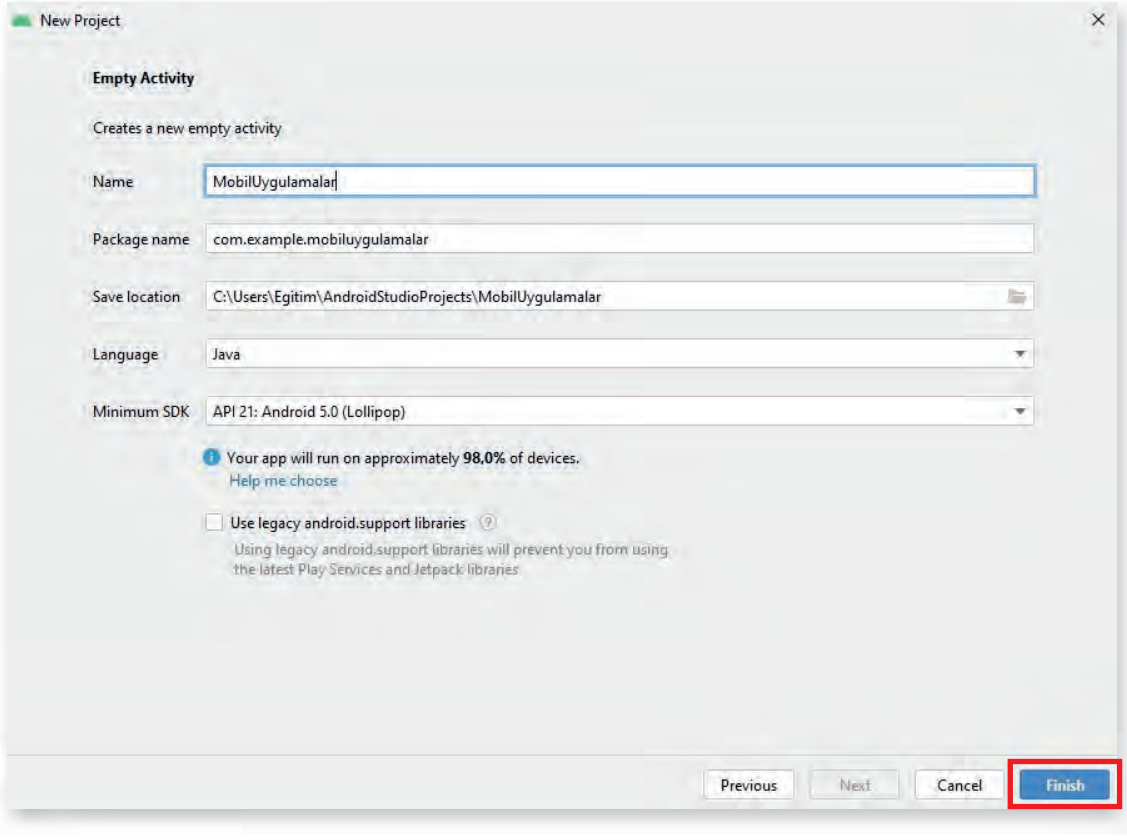
Java programlama dilinde programlar main() metoduyla başlar. Android uygulamalarda ise çok benzer şekilde çalıştırılan activitynin onCreate() metodunu çağırmasıyla OLUŞTURULDU durumuna geçilir. Tablo 1.1'de durumlar ve karşılığında çağrılacak metotlar verilmiştir.

Tablo 1.1: Metotlar

Çağrılan Metot	Tanımı	Durum
onCreate()	Activity ilk oluştuğunda çağrılan metottur.	OLUŞTURULDU
onStart()	Activity kullanıcı tarafından görülebilir hâle geldiğinde çağrılan metottur.	BAŞLATILDI
onResume()	Kullanıcı activity ile etkileşime girdiğinde çağrılan metottur.	SÜRDÜRÜLDÜ
onPause()	Geçerli activity duraklatılırken ve önceki activity devam ettirilirken çağrılan metottur.	DURAKLATILDI
onStop()	Activity artık görünür olmadığında çağrılan metottur.	DURDURULDU
onDestroy()	Activity sistem tarafından yok edilmeden çağrılan metottur.	YOK EDİLDİ
onRestart()	Activity durdurulduktan sonra yeniden başlatıldığında çağrılan metottur.	DURDURULDU-BAŞLATILDI

1.4. PROJE OLUŞTURMA

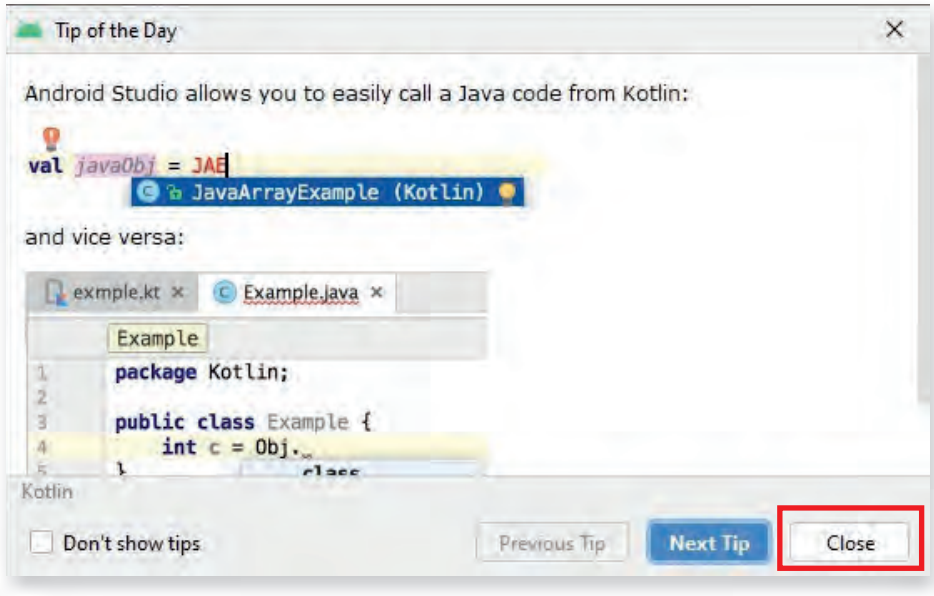
Empty Activity seçildikten sonra Next düğmesine basılınca proje ayarlarının yapılabileceği bir pencere belirir (Görsel 1.26).



Görsel 1.26: Yeni proje ayarları

- **Name (İsim):** Projenin ismi bu kutuda belirlenir. Uygulamanın ismi de projeye verilen bu isim olur.
- **Package name (Paket İsmi):** Paket ismi otomatik olarak verilir. İstenirse sonradan değiştirilebilir.
- **Save location (Kaydetme Dizini):** Projenin yerel kaydetme dizini bu kutuda belirlenir.
- **Language (Dil):** Projenin hangi programlama dilinde kodlanacağı belirlenir. Java programlama dili seçilir.
- **Minimum SDK (En Alt Yazılım Geliştirme Kiti):** Bu kutuda seçilecek API düzeyiyle uygulamanın hangi cihazlarda çalışacağı, hangi cihazlarda destekleneceği belirlenir. API düzeyi düşük olursa piyasadaki çoğu cihazda uygulama çalışır. Örneğin API 21 düzeyi seçilirse piyasadaki cihazların %98'inde uygulama çalışır. "Help me choose" (Seçim için yardım et) düğmesine tıklanarak, hangi API düzeyinin hangi yüzde ile kullanıldığı güncel olarak listelenebilir.
- **Use legacy android.support.libraries (Eski Android Kütüphaneleri)** onay kutusu seçilirse son güncel servisler ve kütüphanelerin kullanılması engellenir.

Finish düğmesine tıklanır. Açılan Tip of the Day (Günün ipucu) penceresi, Close (Kapat) düğmesi ile kapatılır (Görsel 1.27).



Görsel 1.27: Tip of the Day penceresi

2. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında tasarım yapılarından Empty Activity seçerek MobilUygulamam isimli uygulamayı geliştiriniz.

1. Adım: Mobil uygulama geliştirme programını çalıştırınız.

2. Adım: Karşılama ekranında New Project seçeneğine tıklayınız.

2. Adım: Templates bölümünden Phone and Tablet seçiniz.

3. Adım: Sağ taraftaki pencereden Empty Activity tasarım yapısını seçiniz.

4. Adım: Yeni gelen pencerede Name kutusuna MobilUygulamam yazınız.

5. Adım: Pencerenin sağ alt köşesindeki Finish düğmesine tıklayınız.

SIRA SİZDE: Mobil uygulama geliştirme ortamında Fullscreen Activity seçerek TamEkranUygulamam isimli uygulamayı geliştiriniz.

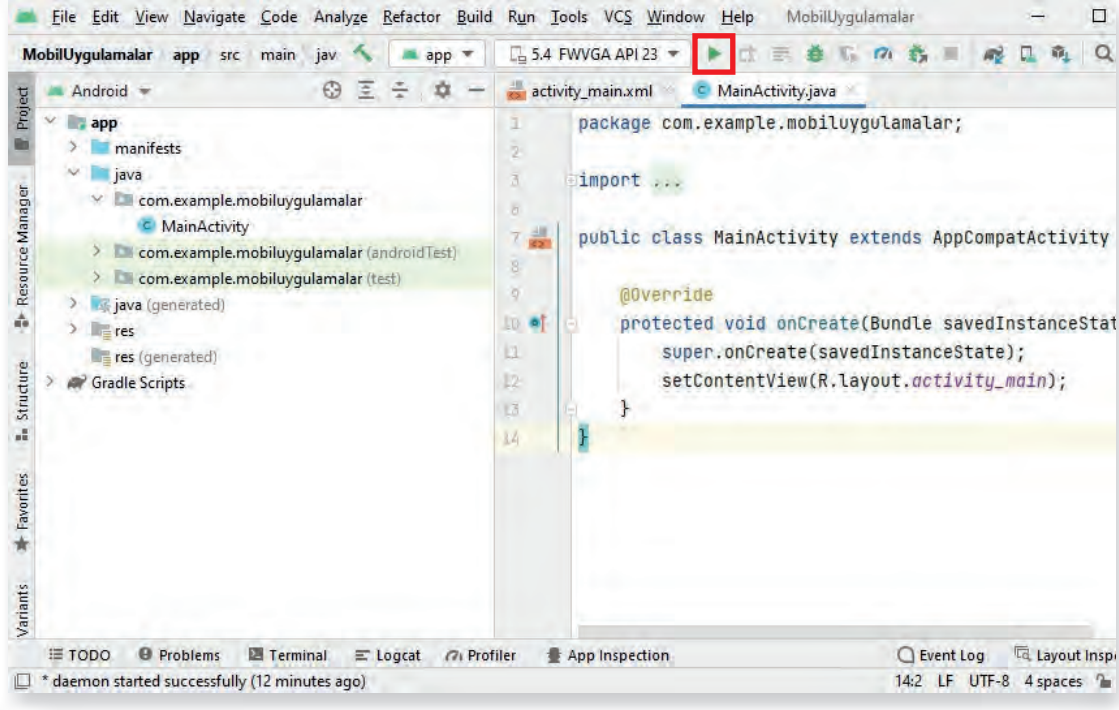
DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme ortamını çalıştırdı.		
2. Karşılama ekranında New Project seçeneğine tıkladı.		
3. Templates bölümünden Phone and Tablet seçti.		
4. Sağ taraftaki pencereden Fullscreen Activity tasarım yapısını seçti.		
5. Yeni gelen pencerede Name kutusuna TamEkranUygulamam yazdı.		
6. Pencerenin sağ alt köşesindeki Finish düğmesine tıkladı.		

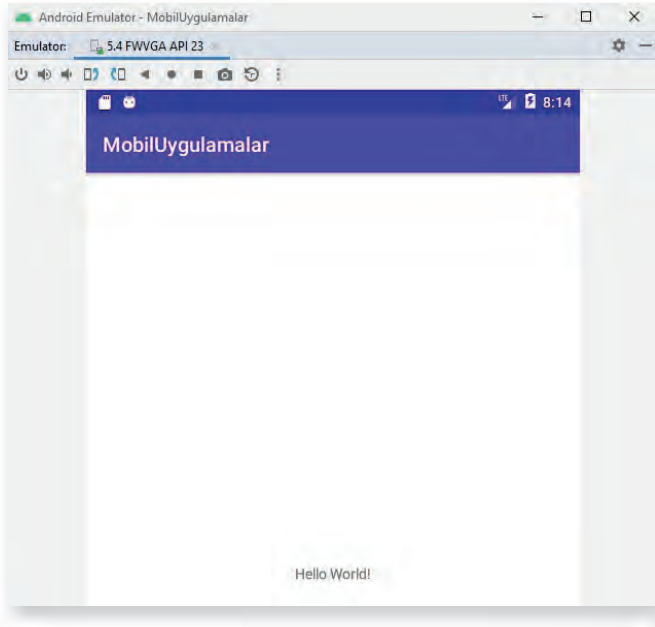
1.4.1. Projenin Emülatörde Çalıştırılması

"MobilUygulamalar" projesi oluşturulduktan sonra uygulama geliştirme ortamında Run 'app' simgesine veya Shift+F10 kısayol tuşlarına basılır (Görsel 1.28).



Görsel 1.28: Android Studio uygulama geliştirme ortamı

Daha önce oluşturulan sanal cihazda **Hello World!** yazısı görüntülenir ve oluşturulan projenin ilk denemesi yapılır (Görsel 1.29).



Görsel 1.29: Sanal cihazda çalıştırılan "MobilUygulamalar" projesi

1.4.2. Android Studio Tasarım Ekranı

Yeni bir mobil uygulama geliştirme ortam projesi açıldığında "MainActivity.java" dosyası varsayılan olarak seçili gelir. Bu dosyada tasarlanan mobil uygulamanın işlevleri kodlanır. Mobil uygulamanın ekran tasarımı, bir başka deyişle kullanıcı arayüzü tasarlanacaksa sol tarafındaki "activity_main.xml" dosyası seçilir (Görsel 1.30).

Palette (1): Çeşitli View ve ViewGroup bileşenlerini içeren paneldir. Sürükle bırak yöntemiyle bu bileşenler tasarım ekranına yerleştirilir.

Component Tree (2): Tasarım ekranındaki bileşenin hiyerarşisini gösterir.

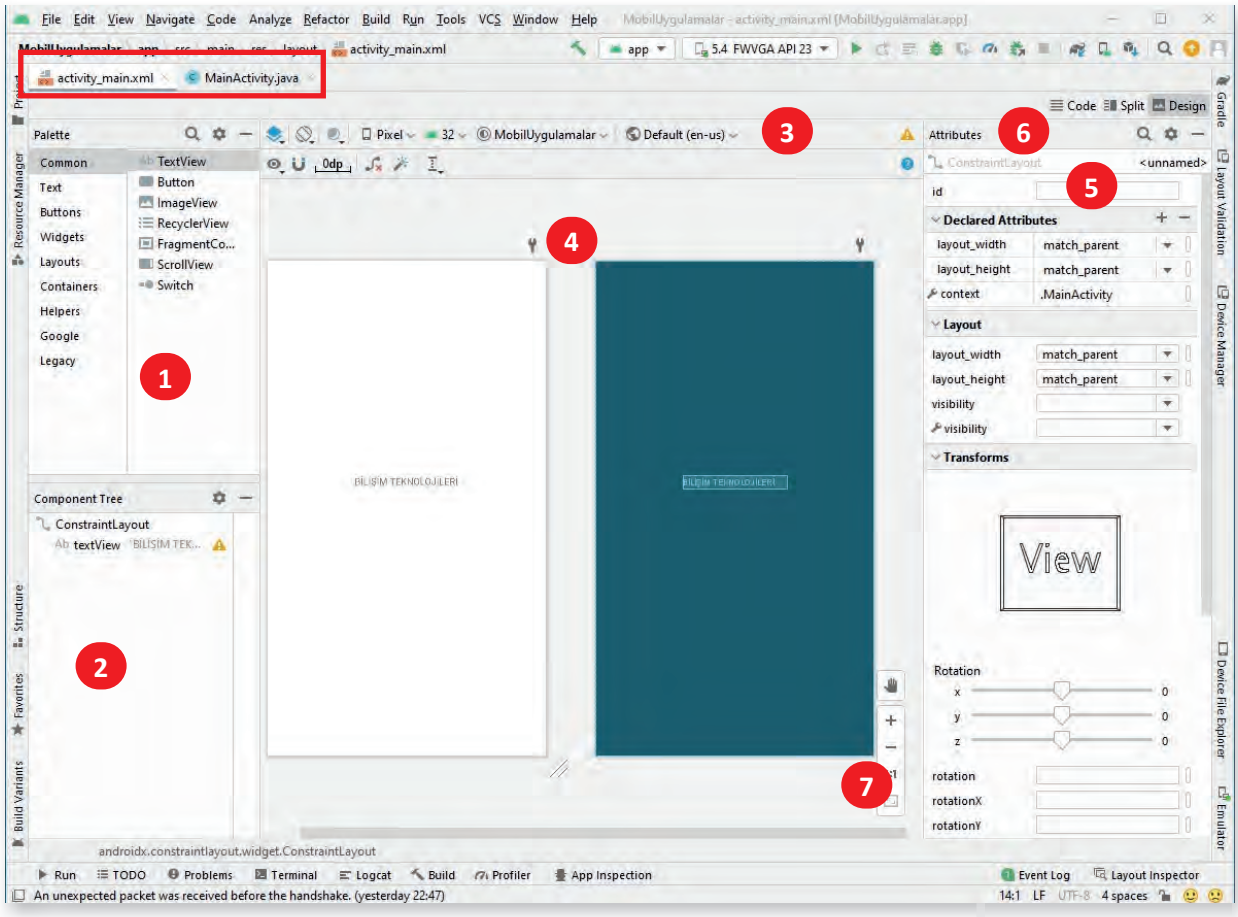
Toolbar (3): Buradaki düğmeler kullanılarak tasarım ekranının görünümü ve özellikleri değiştirilebilir.

Design Editor (4): Tasarım ekranının görünümünü sadece Tasarım (Design), sadece Taslak (Blueprint) veya her ikisi aynı anda şeklinde ayarlanabilir.

Attributes (5): Seçili view nesnesinin niteliklerinin düzenlendiği paneldir.

View mode (6): Tasarım ekranının Code (Kod), Design (Tasarım) ve Split (Böl) modunda gösterilmesini sağlar. Split modu hem Code hem de Design görünümünü aynı anda tek pencerede gösterir.

Zoom and pan controls (7): Buradaki düğmeler yardımıyla ön izlemenin boyutu ve konumu editör içinde ayarlanabilir.

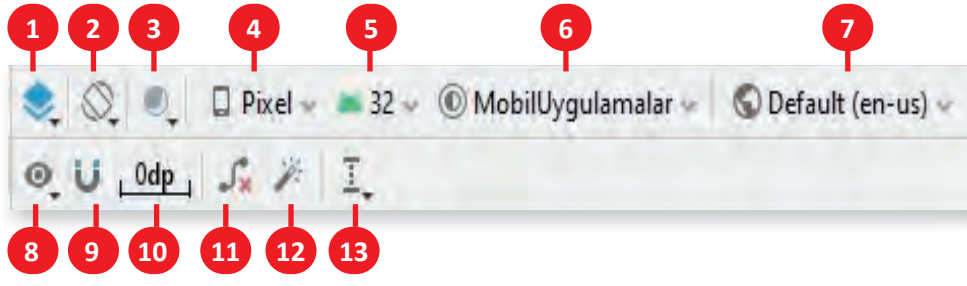


Görsel 1.30: Android Studio Tasarım Ekranı

UYARI: Code, Split ve Design görünümleri arasında geçiş yapmak için klavyeden Alt+Shift+Sağ/Sol Ok tuşları kullanılır.

1.4.3. Ön İzleme Görünümünü Değiştirmek

Tasarım ekranının en üstündeki düğmeler editör içindeki ön izleme görünümlerini değiştirmek için kullanılır (Görsel 1.31).



Görsel 1.31: Toolbar simgeleri

Bu simgelerle yapılan değişiklikler şunlardır:

1: Select Design Surface (Tasarım Yüzeyini Seçin) simgesi kullanılarak yerleşimin editörde nasıl olacağı seçilir. Design modu seçilirse yerleşimin ön izlemesi görülür. Blueprint seçilirse birleşenlerin sadece dış çizgilerinin görüntülendiği bir yerleşim görülür. Design + Blueprint birlikte olduğu seçenek seçilirse hem tasarım hem de taslak yan yana görülür. Klavyeden **B** tuşuna basılarak bu görünümler arasında geçiş sağlanır.

2: Orientation for Preview (Ön İzleme İçin Oryantasyon) simgesi kullanılarak ekran yerleşimi Landscape (yatay) veya Portrait (dikey) olarak ayarlanır. Klavyeden **O** tuşuna basılarak bu görünümler arasında geçiş sağlanır.

3: Changes the current Night Mode value (Geçerli Gece Modu Değerini Değiştir) simgesiyle ön izlemenin gece veya gündüz modu arasında geçiş yapması sağlanır. Klavyeden **N** tuşuna basılarak bu görünümler arasında geçiş sağlanır.

4: Device for Preview (Ön İzleme İçin Cihaz) simgesi ile hem cihaz tipi (cep telefonu, tablet, akıllı TV veya akıllı saat) hem de ekran büyüklüğü (boyut ve yoğunluk) değiştirilebilir. Daha önceden kullanıcı tarafından tanımlanmış bazı cihazlar da buradan seçilebilir. Klavyeden **D** tuşuna basılarak cihazlar arasında geçiş sağlanır.

5: API Version for Preview (Ön İzleme İçin API Sürümü) simgesi ile yerleşimin ön izlemesinin yapılacağı Android sürümünün seçilmesi sağlanır.

6: Theme for Preview (Ön İzleme Teması) simgesi ile ön izlemede gösterilecek kullanıcı arayüzünün seçilmesi sağlanır.

7: Locale for Preview (Ön İzleme İçin Yerel Dil) simgesi ile yerel dil seçimi ve seçilen dile uygun çeviri yapılabilir.

8: View Options (Görünüm Seçenekleri) simgesi ile ön izlemede **Show All Constraints (Tüm Kısıtlamaları Göster)**, **Show Margins (Kenar Boşluklarını Göster)**, **Fade Unselected Views (Seçilmemiş Viewleri Soldur)**, **Live Rendering (Canlı Görüntü)**, **Show System UI (Sistem Arayüzünü Göster)**, **Show Tooltips (Araç İpuçlarını Göster)** vb. seçeneklerin açılıp kapatılması sağlanır.

9: Disable Autoconnection To Parent (Otomatik Bağlanmayı Kaldır/Etkinleştir) simgesi ile otomatik bağlantı modu etkinleştirildiğinde, viewler ön izleme alanına sürüklendiğinde kısıtlamalar otomatik olarak yapılandırılır. Bileşenin hangi sınırlamalara sahip olması gerektiği tahmin edilir ve bunlar gerektiği gibi oluşturulmaya çalışılır.

10: Default Margins (Varsayılan Kenar Boşlukları) simgesi ile her bileşen için ayrı ayrı kenar boşlukları ayarlanabilir.

11: Clear All Constraints (Tüm Kısıtlamaları Temizle) simgesi ile yerleşimdeki tüm constraint ayarları kaldırılır.

12: Infer Constraint (Kısıtlamaları Otomatik Ayarla) simgesi ile otomatik olarak constraintler ayarlanır. Sadece bir düğmeye basılarak bu constraint ayarları yapılır.

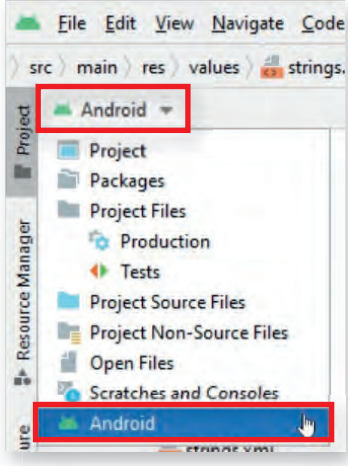
13: Guidelines (Kılavuz Çizgileri) simgesi kullanılarak yatay ve dikey kılavuz çizgileri tasarım ekranına yerleştirilir.

1.5. DOSYA VE DİZİN YAPILARI

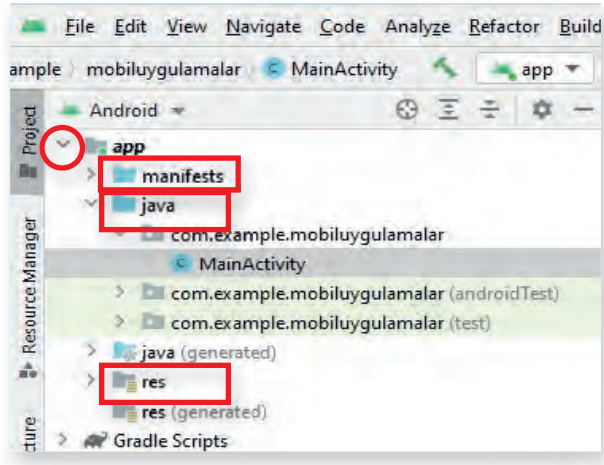
Android uygulama geliştirilmeye başlanmadan önce temel dosya ve dizin yapıları iyi bilinmelidir. Bir Android projenin dosya yapısı farklı biçimlerde görüntülenebilir.

Mobil uygulama geliştirme ortamı ekranının sol üst köşesindeki düğmeye tıklandığında açılan seçim kutusunda yer alan dosya hiyerarşisi görüntüleme biçimlerinden biri tercih edilebilir. Varsayılan olarak gelen ve Android uygulama geliştirme için kullanımı en kolay olan "Android" modudur (Görsel 1.32).

"Android" modu seçildikten sonra Görsel 1.33'teki dosya ve dizin yapısı pencerenin sol üst kenarında görüntülenir.



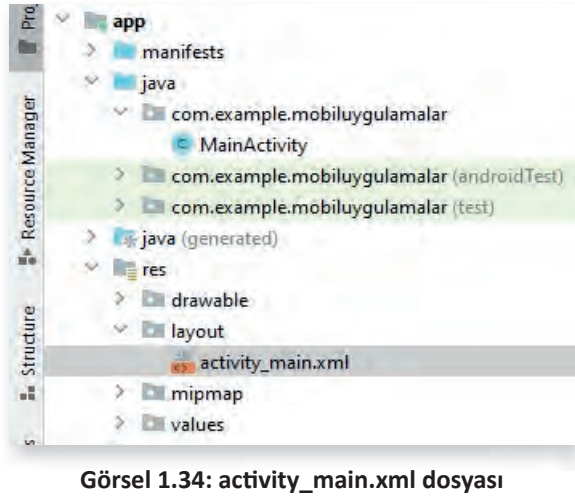
Görsel 1.32: Dosya yapısı görüntüleme seçenekleri



Görsel 1.33: Android modu dosya ve dizin yapısı

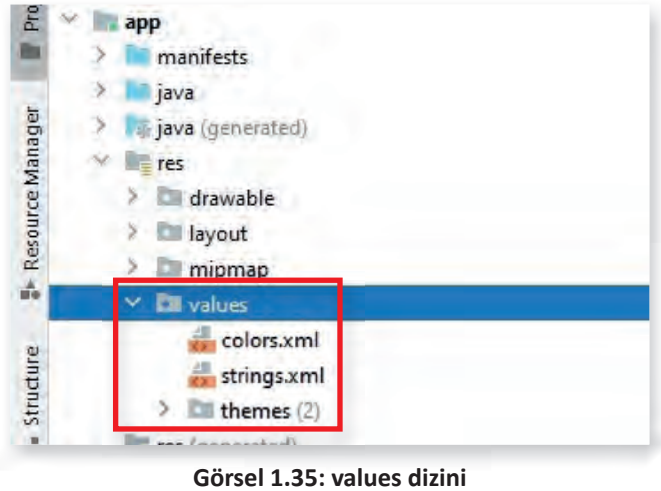
Görsel 1.33'te yer alan daire içindeki simge tıklandığında dizinin içeriği görüntülenebilir. Aynı simge tıklandığında görüntülenen içerik kapatılır. Bir Android uygulamasındaki üç temel dizin şunlardır:

1. **manifests:** Bu dizin içinde AndroidManifest.xml dosyası yer alır. Bu dosyada proje ayar parametreleri, izinler, servisler ve ek kütüphaneler bulunur.
2. **java:** Java programlama dilinde yazılmış kaynak kodlar bu dizinde yer alır. "MainActivity.java" isimli dosya otomatik olarak oluşturulur. Activityler ana Java sınıflarıdır, içeriğinde geliştirilecek uygulamanın ne yapacağını belirten Android kodlar bulunur.
3. **res:** Kaynak dosyaları bu dizinde yer alır. Kaynak kod dosyaları dışında projenin ihtiyaç duyduğu resim, müzik vb. dosyalar bu dizinde bulunur. Bu dizin içinde uygulamada kullanılacak resimlerin bulunduğu drawable dizini, uygulamanın nasıl görüneceğini belirleyen layout dizini ve temel değerleri tutan values dizini yer alır.
    - **drawable (Çizilebilir):** Uygulamada kullanılacak resimleri içeren dizindir.
    - **layout (Yerleşim):** Uygulamanın nasıl görüneceğini belirleyen xml dosyası bu dizinde yer alır (Görsel 1.34).



Görsel 1.34: activity_main.xml dosyası

- **values (Değerler):** Basit değerleri (metinler, tam sayılar, renkler vb.) içeren xml dosyalarıdır (Görsel 1.35). res/ alt dizininde yer alan XML kaynak dosyaları XML adına göre tek bir kaynak tanımlarken values/ dizinindeki dosyalar birden fazla kaynağı tanımlar. Bu dizinde oluşturulabilecek kaynaklar için bazı dosya adı kuralları şunlardır:
  - ► arrays.xml diziler için
  - ► colors.xml renk değerleri için
  - ► dimens.xml boyut değerleri için
  - ► strings.xml metin değerler için
  - ► styles.xml stiller için



Görsel 1.35: values dizini

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Native uygulama, belirli bir işletim sistemi veya belirli bir cihaz için geliştirilen uygulamalardır.

2. ( ) Farklı işletim sistemine ait programları yerel işletim sisteminde çalıştırmak için kullanılan aracı yazılımlara emülatör ismi verilir.

3. ( ) Portrait seçilerek mobil cihaz ekranının yatay olarak ayarlanması sağlanır.

4. ( ) Activity ilk oluşturulduğunda çağrılan onCreate() metodudur.

5. ( ) Activity artık görünür olmadığında çağrılan onRestart() metodudur.

6. ( ) Palette, View ve ViewGroup bileşenlerini içeren paneldir.

7. ( ) drawable, uygulamada kullanılacak resimleri içeren dizindir.

B) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

8. Aşağıdakilerden hangisi taşınabilir cihazlara örnek değildir?
A) Akıllı telefon B) Tablet C) Akıllı saat
D)Masaüstü bilgisayar E) E-kitap okuyucu

9. Aşağıdakilerden hangisi birden fazla işletim sistemi veya cihazda çalışabilecek uygulamalara verilen genel isimdir?
A) Native B) Cross Platform C) Emülatör
D) Sanal Cihaz E) API

10. Aşağıdakilerden hangisi bir mobil uygulamada kullanıcıya gösterilen ve üzerinde kullanıcı arayüz bileşenleri olan ekrana verilen isimdir?
A) Activity B) Sanal Cihaz C) Emülatör
D) Proje E) Metot

11. Aşağıdakilerden hangisi mobil uygulama tasarım ekranı panellerinde seçili bileşenin niteliklerini düzenler?
A) Palette B) Toolbar C) Attributes
D) View mode E) Component Tree

12. Aşağıdakilerden hangisi mobil uygulamanın nasıl görüneceğini belirleyen XML dosyalarının yer aldığı dizindir?
A) drawable B) layout C) mipmap
D) res E) values

13. Aşağıdaki metotlardan hangisi activity sistem tarafından yok edilmeden çağrılan metottur?
A) onStart() B) onPause() C) onStop()
D) onDestroy() E) onRestart()

14. Aşağıdakilerden hangisi activity kullanıcı tarafından görülebilir hâle geldiğinde çağrılan metottur?
A) onCreate() B) onStart() C) onResume()
D) onStop() E) onRestart()

2. ÖĞRENME BİRİMİ

EKRAN TASARIMI

KONULAR

2.1. MOBİL UYGULAMA EKRAN TASARIMINA GİRİŞ

2.2. MOBİL UYGULAMA EKRAN YAPISI

2.3. VIEW

2.4. GÖRÜNÜM YERLEŞTİRME YÖNTEMLERİ

2.5. TEMEL GÖRÜNÜM SINIFLARI

2.6. LAYOUT ÇEŞİTLERİ

NELER ÖĞRENECEKSİNİZ?

- Mobil cihaz ekran türleri
- Mobil cihaz ekran yapısı
- View sınıfı
- ViewGroup sınıfı
- Görünüm yerleştirme yöntemleri
- XML kodlama ile görünüm yerleştirme
- Palette paneli ile görünüm yerleştirme
- Java kodlama ile görünüm yerleştirme
- Temel görünüm sınıfları
- Layout çeşitleri

ANAHTAR KELİMELER

- Action Bar
- APK
- APPS
- Button
- CheckBox
- ConstraintLayout
- EditView
- Layout
- MainActivity
- Navigation bar
- ProgressBar
- RelativeLayout
- Status bar
- TextView
- View
- ViewGroup

HAZIRLIK ÇALIŞMALARI

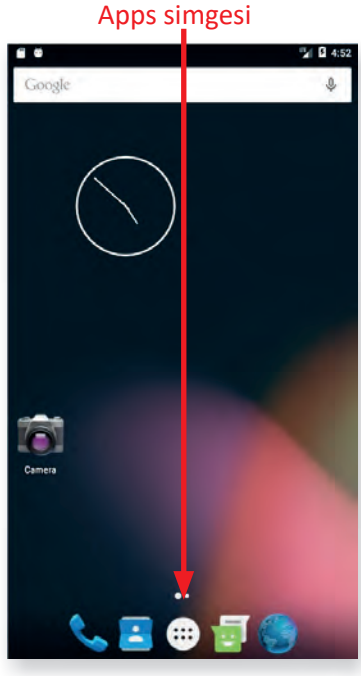
1. Günlük hayatta en çok tercih ettiğiniz mobil uygulamanın arayüzünü kullanırken zorlandığınız durumları anlatınız.
2. Bir mobil uygulamanın ekran tasarımını siz nasıl yapardınız? Düşüncelerinizi arkadaşlarınızla paylaşınız.

2.1. MOBİL UYGULAMA EKRAN TASARIMINA GİRİŞ

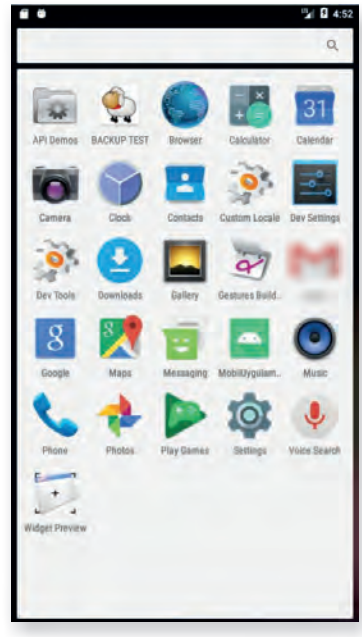
Android, taşınabilir cihazlarda yaygın olarak kullanılan bir işletim sistemidir. İngilizce Applications (Uygulamalar) kelimesinin kısaltılmış hâli olan **Apps**, bu işletim sisteminde çalışan mobil uygulamaları ifade etmek için kullanılır. Kurulum ve çalıştırılabilir dosyalara **Android Package Kit (APK)** adı verilir. Bu Android işletim sistemi dosyalarının uzantısı da **.apk** olarak belirlenir.

Bir mobil işletim sistemi arayüzünü temel olarak üçe ayırmak mümkündür.

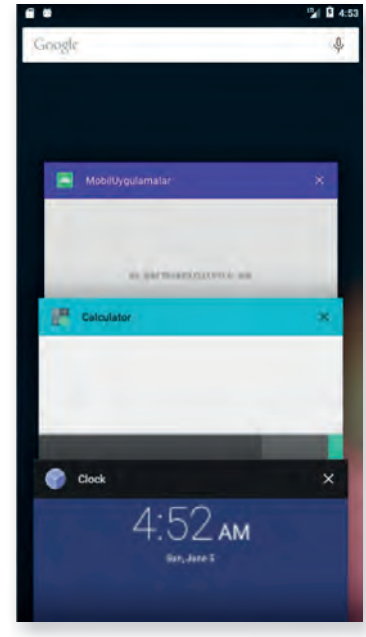
- **Home screen (Ana Ekran):** Taşınabilir cihaz açıldığında kullanıcıyı karşılayan ekrandır. Bu arayüz özelleştirilebilir veya arayüzün teması değiştirilebilir (Görsel 2.1a).
- **All apps (Tüm Uygulamalar):** Taşınabilir cihaza kurulu tüm uygulamaların gösterildiği arayüz ekranıdır (Görsel 2.1b). Uygulamalar ekranına geçmek için ana ekranda bulunan **Apps** simgesine dokunulur veya tıklanır.
- **Recent screen (Son Kullanılan Uygulamalar Ekranı):** Taşınabilir cihazda son kullanılan uygulamaların listesinin görüntülendiği arayüz ekranıdır (Görsel 2.1c).



Görsel 2.1a: Home screen



Görsel 2.1b: All apps



Görsel 2.1c: Recent screen

UYARI: Bu üç temel mobil cihaz ekranının dışında saatin, bildirimlerin, mesajların gösterilebildiği **Lock screen (Kilit ekranı)** de vardır.

SIRA SİZDE: Mobil uygulama geliştirme ortamında emülatörü çalıştırdıktan sonra Home screen, All apps ve Recent screen ekranlarına geçiş yapınız.

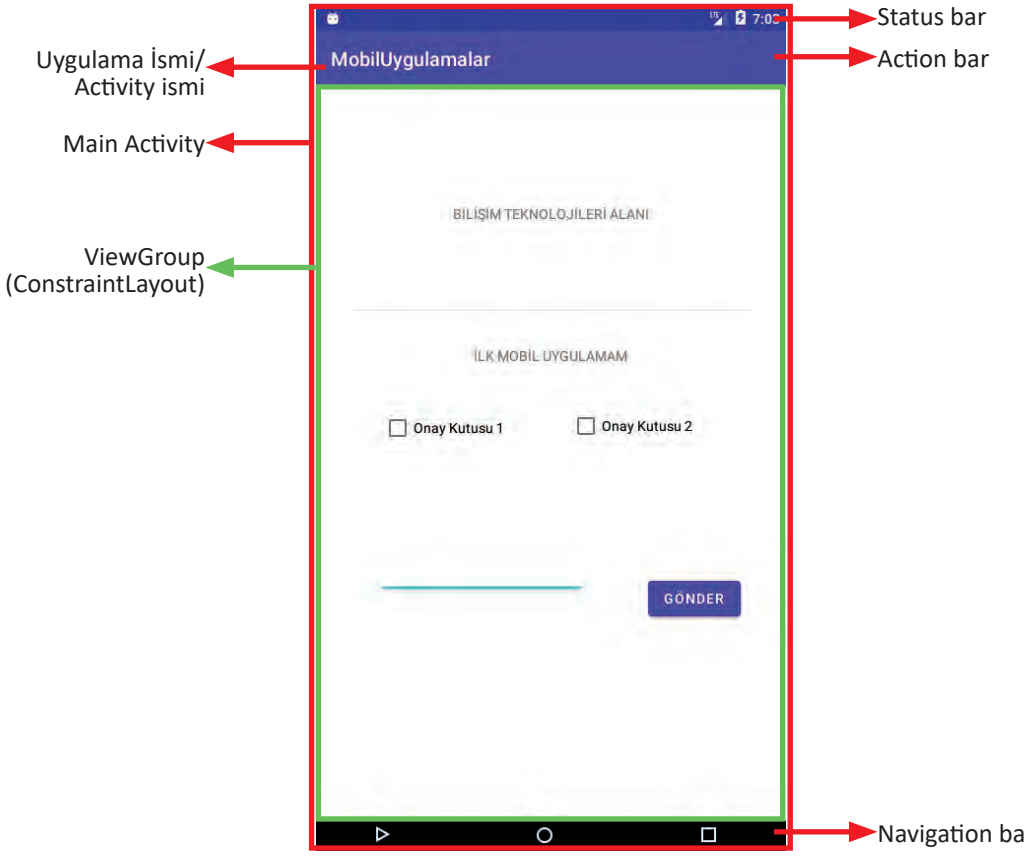
DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme ortamını çalıştırdı.		
2. Ön izleme yapmak için klavyeden Shift+F10 kısayol tuşlarına bastı.		
3. Açılan cihaza ait çerçeve işlev düğmelerinden Ev (Home) düğmesine tıkladı.		
4. Açılan cihaza ait çerçeve işlev düğmelerinden Genel Bakış (Overview) düğmesine tıkladı.		
5. Açılan cihazın işletim sisteminden Apps simgesine tıkladı.		

2.2. MOBİL UYGULAMA EKRAN YAPISI

Mobil uygulama geliştirilirken uygulama çalıştığında ekran yapısında bazı bileşenler yer alır. En üstte **status bar (durum çubuğu),** onun hemen altında **action bar (işlem çubuğu)** ve ekranın en altında ise **navigation bar (gezinti çubuğu)** bulunur. Uygulama çalıştığında ekranda görünmeyen activity (MainActivity) ve activity içinde kullanıcı ile etkileşime girecek bileşenlerin bulunduğu ViewGroup (Görünüm grubu) Görsel 2.2'de farklı renklerle gösterilmiştir.



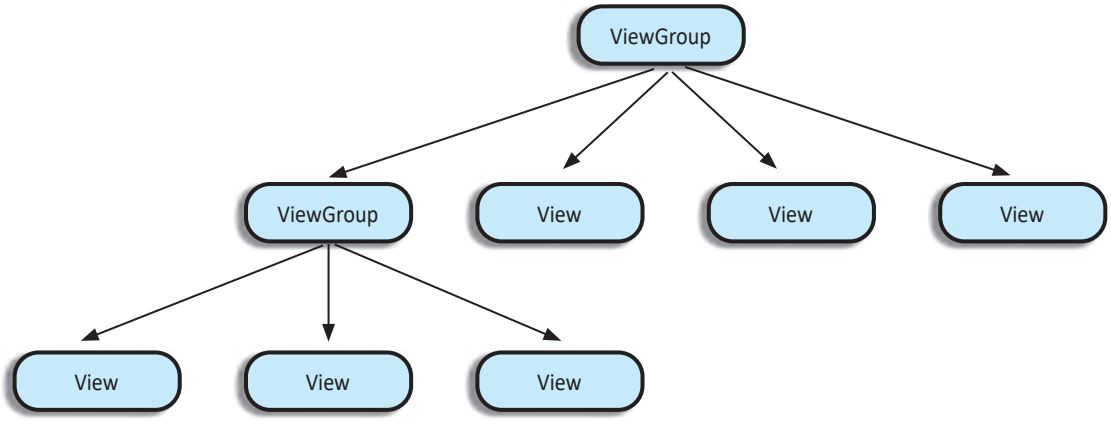
Görsel 2.2: Mobil uygulama ekran yapısı

- **Status Bar:** Mobil cihaz ekranının en üstünde yer alan bu çubukta pil durumu, saat, ağ bağlantı simgesi gibi bildirimler bulunur.
- **Action Bar:** Bu çubuk activitynin üst tarafında yer alır. Uygulama ismi, activity ismi veya simgeleri, ek görünümler ve etkileşimli nesneler bu çubukta bulunabilir. Ayrıca gezinti yapmak için işlem çubuğuna simgeler yerleştirilebilir.
- **Navigation Bar:** Alt gezinti çubuğu olarak da isimlendirilen, uygulama ekranının altında görünen çubuktur. Ekranlar ve uygulamalar arasında temel gezinme işlemleri buradaki simgelerden yapılır.

2.3. VIEW

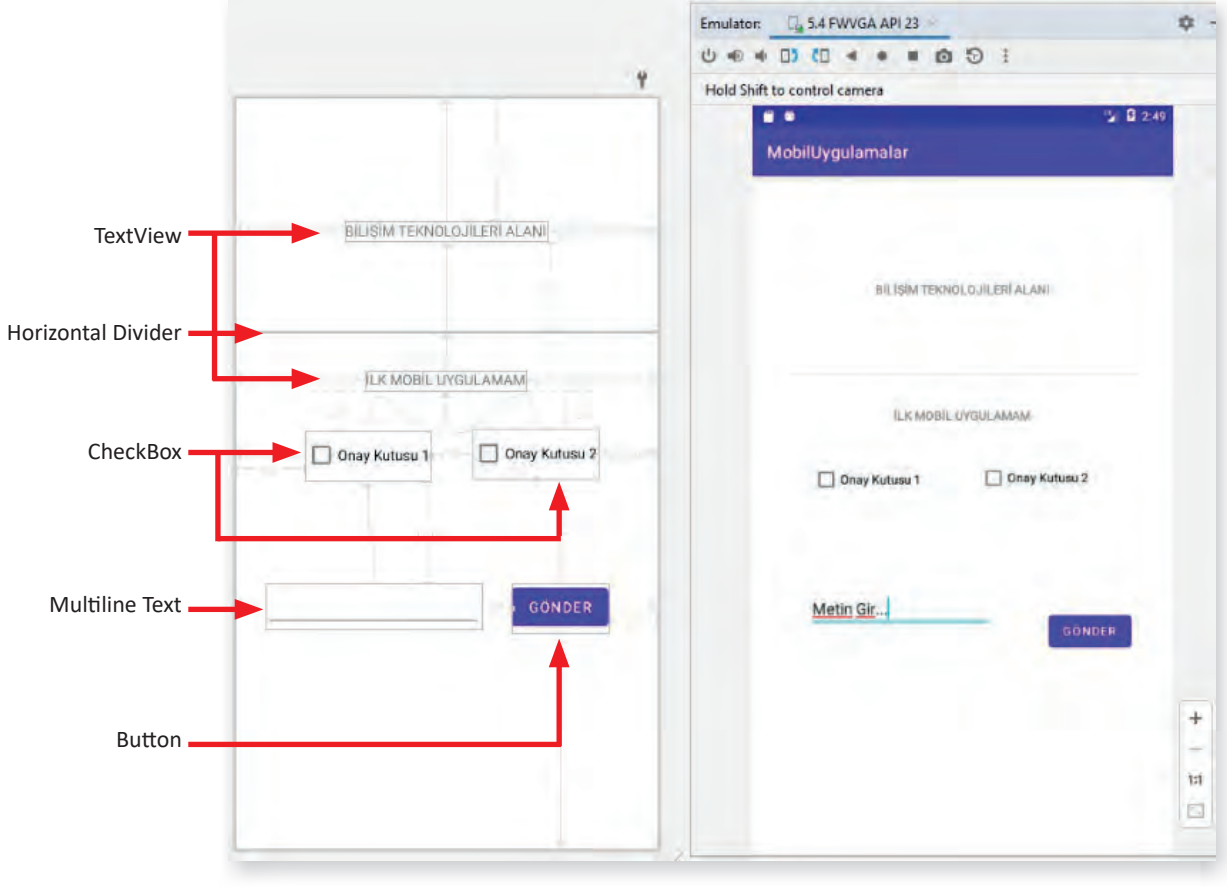
Bir mobil uygulamada kullanıcıyla tüm etkileşim, kullanıcı arabirimi (user interface-UI) üzerinden olur. Kullanıcı arabirimi tasarlamak için Android SDK bir dizi hazır View (Görünüm) sağlar. Bu hazır görünümlerden bazıları; Button (Düğme), CheckBox (Onay kutusu), ProgressBar (İlerleme çubuğu) ve TextView (Metin görünümü) sınıflarıdır. Görünümler, aletler (widgets) veya bileşenler (components) isimleriyle de ifade edilir.

Görünümler, görünmez kalıp niteliğindeki kapsayıcıların içine yerleştirilir. Bu görünmez kalıba ViewGroup ismi verilir. Bir veya daha fazla görünüm, ViewGroup içinde yer alabilir. ViewGroup, görünümlerin yerleştirilebildiği ve konumlarının belirlenebildiği özel bir görünümdür. ViewGroup içinde başka ViewGrouplar da bulunur (Görsel 2.3).



Görsel 2.3: ViewGroup yapısı

Bir mobil uygulamada her ne kadar kök bileşen activity olsa da bir activityde birden fazla görünüm ve ViewGroup nesnesi olabilir. Görünüm, ekranda görülen tüm bileşenlerin üst sınıfıdır. Buna ViewGroup da dâhildir. Görsel 2.4'te sol kısımdaki tasarım ekranında görünmeyen ViewGroup içine yerleştirilmiş farklı viewler yer alır. Görselin sağ tarafında ise uygulamanın emülatörde çalışan hâli görülür.

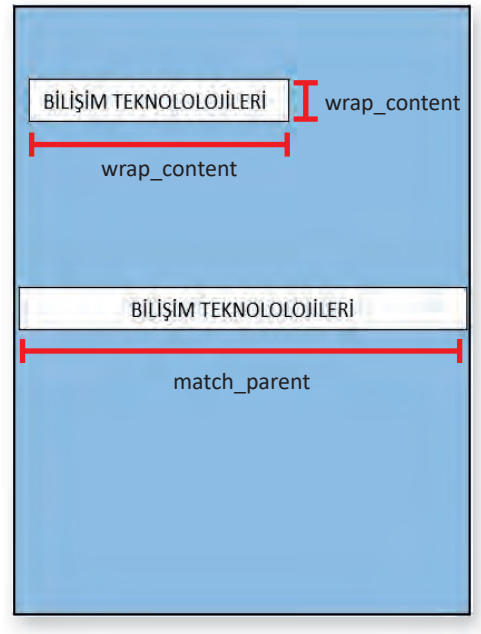


Görsel 2.4: Solda ViewGroup içinde görünümler ve sağda uygulama ön izlemesi

2.4. GÖRÜNÜM YERLEŞTİRME YÖNTEMLERİ

Görünüm, mobil uygulamalarda kullanıcı arabiriminin temel yapı taşıdır ve ekranda içerik gösteren dikdörtgendir. Görünüm; bir resim, bir metin parçası, bir düğme veya bir mobil uygulamanın görüntüleyebileceği herhangi bir şey olabilir. Tüm görünümler dikdörtgen şeklindedir ve bu dikdörtgenler ekranda görünmez. Bu dikdörtgenin boyutu, sayılar girilerek veya daha önceden tanımlanmış bazı değerler kullanılarak ayarlanabilir. Önceden tanımlanmış bu değerler şunlardır:

- **wrap_content:** İçeriğin görüntülenmesi için gereken kadar alanı kaplayacağı anlamına gelir.
- **match_parent:** Cihazın ekranında bulunan tüm alanı kaplayacağı anlamına gelir (Görsel 2.5).



Görsel 2.5: match_parent ile wrap_content farkı

Genişlik ve yükseklik niteliklerine wrap_content ve match_parent önceden tanımlı değerler girilebileceği gibi sayılar da girilebilir ancak bu sayıların yanında birim olması gerekir. Bu birimler şunlardır:

- **px (Pixels):** Girilen değerin ekranda gerçek piksel olarak karşılığıdır.
- **in (Inches):** Girilen değerin ekranda gerçek inç karşılığıdır.
- **mm (Millimeters):** Girilen değerin ekranda gerçek milimetre karşılığıdır.
- **pt (Points):** Girilen değer 1/72 inç oranındadır.
- **dp (Density-independent Pixels):** Ekranın fiziksel yoğunluğuna dayanan soyut bir birimdir. Bu birimler 160 dpi ekrana oranlanır. Bu nedenle 1 dp, 160 dpi ekranda 1 piksele karşılık gelir.
- **sp (Scaleable-independent Pixels):** Bu birim, dp birimine benzer ancak kullanıcının yazı tipi boyutu tercihine göre ölçeklendirilir.

Bir görünümün genişliği, XML kod olarak içeriği gösterecek kadar olsun istenirse **android:layout_width="wrap_content"** şeklinde yazılır veya görünümün genişliği 200 piksel olsun istenirse **android:layout_width="200px"** şeklinde yazılır.

ÖRNEK

- **android:layout_width="20in"** XML kodu, görünüm genişliğinin 20 inç olmasını sağlar.
- **android:layout_width="40mm"** XML kodu, görünüm genişliğinin 40 milimetre olmasını sağlar.

Görünümler, mobil uygulama ekranına yerleştirilirken üç farklı yöntem kullanılır. Bu yöntemler şunlardır:

1. XML kodlarını kullanarak görünüm yerleştirmek
2. Palette panelini kullanarak görünüm yerleştirmek
3. Java kodlarını kullanarak görünüm yerleştirmek

2.4.1. Görünüm Yerleştirmek İçin XML Kullanımı

Mobil uygulamada herhangi bir görünümü ekrana yerleştirmek için XML kodları kullanılır. XML kodları kullanılarak yapılan bu işlem, **Declarative Approach (Bildirimsel Yaklaşım)** olarak adlandırılır. Genellikle XML kodları **activity_main.xml** dosyasının içine yazılır.

XML kodları kullanılarak yapılan görünüm ayarları kod yapısı şu şekildedir:

```
<GörünümİsmiNitelik1=Değer1
      Nitelik2=Değer2
      Nitelik3=Değer3
      .
      .
      NitelikN=DeğerN
/>
```

XML kodlanırken dikkat edilecek özellikler şunlardır:

- Her zaman küçüktür işareti “<” ile başlar ve ardından görünüm ismi yazılır.
- Görünüm, ekranda nasıl görünecekse bir nitelik ve değer kod olarak yazılır. Her görünümün kendine has nitelikleri vardır.
- “/>” işaretleri kullanılarak görünüm ayarları kapatılır.

ÖRNEK

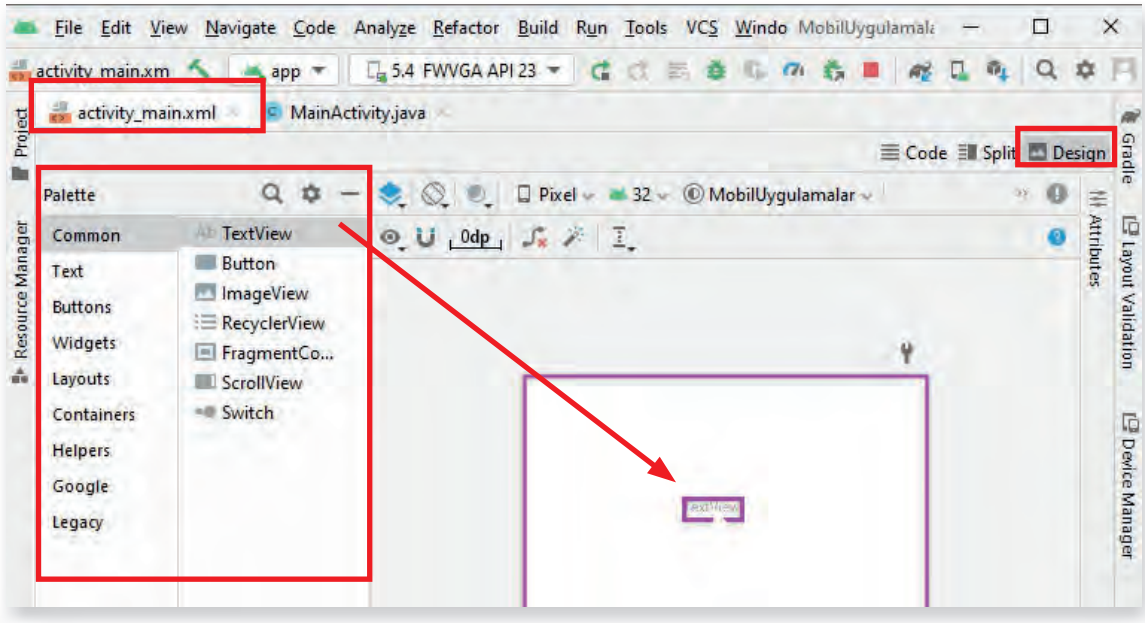
Ekranda “Programlamayı seviyorum.” metnini gösteren bir TextView görünümünün XML’de yazımı şu şekildedir:

```
<TextView
            android:id=”@+id/MU_TextView_id”
            android:layout_width=”wrap_content”
              android:layout_height=”wrap_content”
            android:text=”Programlamayı seviyorum.”
 />
```

Her görünüm, **android:layout_width (görünümün genişliği)** ve **android:layout_height (görünümün yüksekliği)** gibi en az iki niteliğe değer verilirse uygulama ekranında görüntülenir. Görünümün boyutları, bu niteliklere değer atanarak ayarlanır. Bu iki nitelik kullanılarak mobil uygulamadaki tüm görünüm çeşitlerinin boyutları ayarlanabilir.

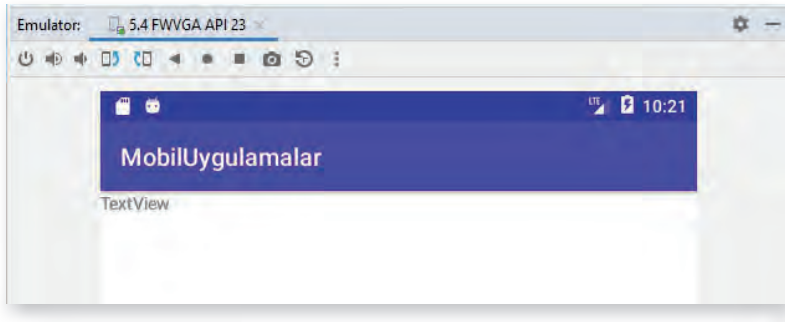
2.4.2. Görünüm Yerleştirmek İçin Palette Kullanımı

Bir mobil uygulamada herhangi bir görünümü ekrana yerleştirmek için mobil uygulama geliştirme ortamında bulunan Palette panelindeki bileşenlerden biri sürükle bırak yardımıyla Design (Tasarım) bölümüne bırakılır. Projedeki activity_main.xml dosyası seçiliyken görünümlerden Design düğmesi tıklanır ve çalışma alanında Palette panelindeki görünümlerden istenen herhangi biri sürükle bırak yöntemiyle tasarım alanına bırakılır (Görsel 2.6).



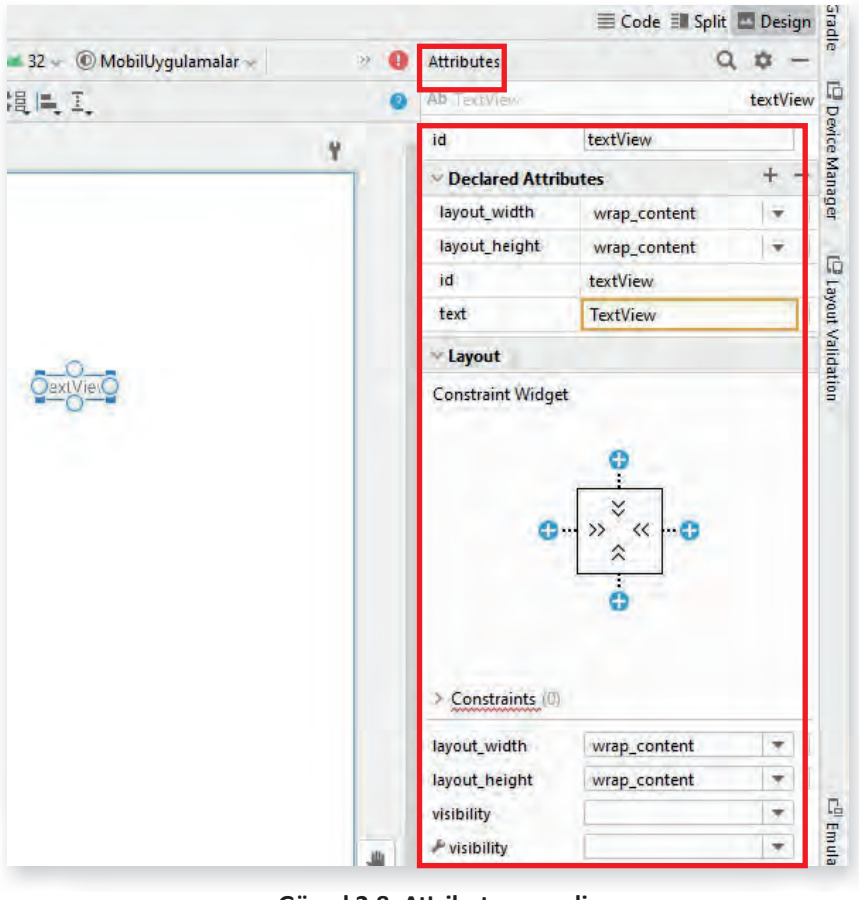
Görsel 2.6: Görünüm ekleme

Görünümlerden ekranda metin göstermek için kullanılan TextView, tasarım ekranına yerleştirilerek proje ön izleme simgesine tıklanır veya Shift+F10 klavye tuşlarına basılarak mobil uygulama çalıştırılır. Mobil uygulamada ConstraintLayout yerleşimi kullanıldığı ve TextView görünümünün nitelikleri ayarlanmadığı için TextView otomatik olarak ekranın sol üst bölümüne yerleşir (Görsel 2.7).



Görsel 2.7: TextView görünümünün ön izlemesi

Tasarım ekranında seçili görünümün nitelikleri, mobil uygulama geliştirme ortamında **Attributes** paneli kullanılarak değiştirilebilir. Bu paneldeki kutulara sayılar veya ön tanımlı değerler girilerek görünümün nitelikleri değiştirilir (Görsel 2.8).



Görsel 2.8: Attributes paneli

1. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında tasarım ekranına Palette panelini kullanarak Button görünümünü yerleştiriniz.

1. Adım: Mobil uygulama geliştirme programını çalıştırınız.

2. Adım: Çalışma alanında activity_main.xml dosyasını seçiniz.

3. Adım: Palette paneline ve Design ekranına geçmek için Code, Split ve Design sekmelerinden Design sekmesini tıklayınız.

4. Adım: Palette panelindeki Button görünümünü sürükle bırak yöntemiyle Design ekranına bırakınız.

5. Adım: Shift+F10 klavye tuşlarına basarak uygulamanın ön izlemesini yapınız.

SIRA SİZDE: Mobil uygulama geliştirme ortamında Palette panelinde yer alan CheckBox görünümünü tasarım ekranına yerleştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme ortamını çalıştırdı.		
2. Çalışma alanında activity_main.xml dosyasını seçti.		
3. Palette panelinde Buttons seçti.		
4. Palette panelinde CheckBox görünümünü sürükle bırak yöntemiyle tasarım ekranına yerleştirdi.		
5. Shift+F10 tuşlarına basarak uygulamanın ön izlemesini yaptı.		

2. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında tasarım ekranına yerleştirilmiş CheckBox görünümünün layout_width kutusuna 100 dp değerini Attributes panelini kurarak giriniz.

1. Adım: Tasarım ekranında yer alan CheckBox görünümünü seçiniz.

2. Adım: Çalışma alanında Attributes panelindeki layout_width kutusuna tıklayınız.

3. Adım: Klavyeden 100 dp değerini giriniz.

4. Adım: Shift+F10 klavye tuşlarına basarak uygulamanın ön izlemesini yapınız.

UYARI: ContraintLayout yerleşimindeyken constraint (kısıtlama) ayarları yapılmadığında ön izlemede görünüm sol üst köşede yer alır.

SIRA SİZDE: Mobil uygulama geliştirme ortamında CheckBox görünümünün Attributes panelindeki text kutusuna Bilişim Teknolojileri yazarak ekranda görünecek metni belirleyiniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Tasarım ekranındaki CheckBox görünümünü seçti.		
2. Attributes panelindeki text kutusuna tıkladı.		
3. Klavyeden “Bilişim Teknolojileri” yazdı.		
4. Shift+F10 tuşlarına basarak uygulamanın ön izlemesini yaptı.		

2.4.3. Görünüm Yerleştirmek İçin Java Kullanımı

Görünümler, Java kodları kullanılarak da tanımlanır veya oluşturulur. Java kodlarıyla yapılan bu işlem, **Programmatic Approach (Programatik Yaklaşım)** olarak adlandırılır. Genellikle Java kodları MainActivity.java dosyası içine yazılır.

ÖRNEK

TextView görünümünün metni Java kodlarıyla değiştirilmek istenirse şu kodlar yazılır:

```
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText("Bilişim Teknolojileri");
```

2.5. TEMEL GÖRÜNÜM SINIFLARI

En çok kullanılan görünüm sınıfları şunlardır:

- TextView (Metin görünümü)
- EditText (Metin girişi)
- Button (Düğme)
- ImageView (Resim görünümü)
- CheckBox (Onay kutusu)
- ProgressBar (İlerleme çubuğu)

2.5.1. TextView

TextView, en temel bileşenlerden biridir ve mobil cihazın ekranında metin göstermek için kullanılır. TextView görünümüne ait niteliklerden en çok kullanılanları Tablo 2.1'de verilmiştir.

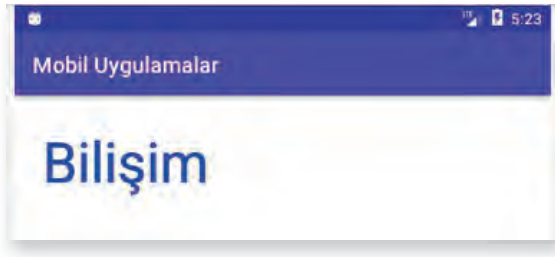
Tablo 2.1: TextView Görünümüne Ait Nitelikler

Nitelik	Tanım
android:id	Java kodlarının ulaşabilmesi için bir tanımlama ismi verilir.
android:text	Ekranda gösterilecek metin yazılır.
android:textSize	Metnin boyutu belirlenir.
android:padding	Metnin etrafında boşluk olması sağlanır.
android:textColor	Metnin rengi belirlenir.
android:textAllCaps	True değeri verilirse metin harflerinin tümünün büyük olması sağlanır.
android:letterSpacing	Metnin harfleri arasındaki boşluk belirlenir.

TextView görünümünün XML kodu şu şekildedir:

```
<TextView
    android:id="@+id/MU_TextView_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bilişim"
    android:textSize="50sp"
    android:padding="25dp"
    android:textColor="#0000FF"/>
```

XML kodu bir yerleşim içine yazılıp emülatörde ön izleme yapılırsa mavi renkli, metin boyutu **50 sp**, metin etrafında boşluğu **25 dp** olan "Bilişim" yazısı görüntülenir (Görsel 2.9).



Görsel 2.9: TextView ön izleme

3. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında tasarım ekranına XML kodlarıyla kırmızı renkli, 55 sp boyutunda, kenar boşlukları 30 dp olan "Teknolojileri" yazısını yazdırınız.

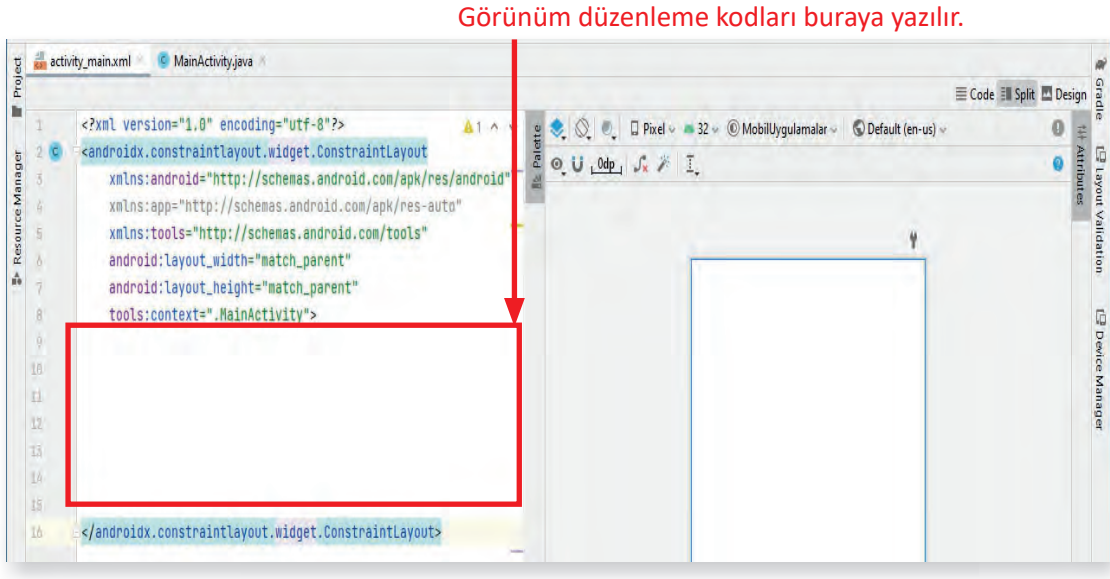
1. Adım: Mobil uygulama geliştirme programını çalıştırınız.

2. Adım: Çalışma alanında activity_main.xml dosyasını seçiniz.

3. Adım: Hem kod ekranını hem de Design ekranını görmek için Code, Split ve Design sekmelerinden Split sekmesini tıklayınız.

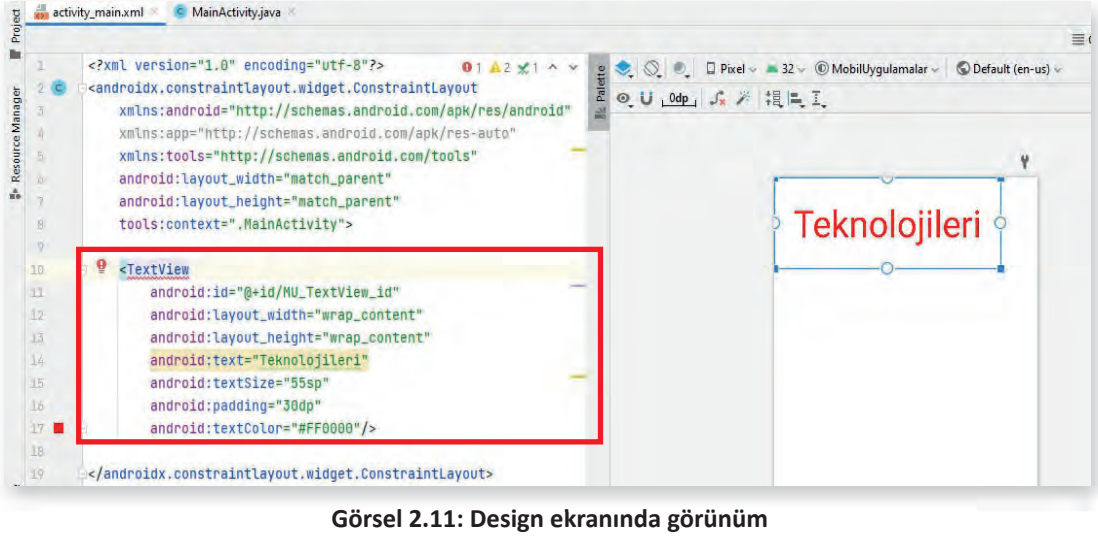
4. Adım: Kod ekranında TextView görünümünü düzenleyen şu XML kodlarını Görsel 2.10'da belirtilen yere yazınız:

```
<TextView
       android:id="@+id/MU_TextView_id"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Teknolojileri"
       android:textSize="55sp"
       android:padding="30dp"
       android:textColor="#FF0000"/>
```



Görsel 2.10: XML görünüm kodlarının yazılacağı alan

5. Adım: XML görünüm kodları yazıldıktan sonra Design ekranında görünüm belirir (Görsel 2.11). Klavyeden Shift+F10 tuşlarıyla uygulamanın ön izlemesini yapınız.



Görsel 2.11: Design ekranında görünüm

SIRA SİZDE: Mobil uygulama geliştirme ortamında tasarım ekranına XML kodlarıyla yeşil renkli, 60 sp boyutunda, kenar boşlukları 25 dp olan "Yazılım Geliştirme" yazısını yazdırınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme programını çalıştırdı.		
2. Çalışma alanında activity_main.xml dosyasını seçti.		
3. Hem kod ekranını hem de Design ekranını görmek için Code, Split ve Design sekmelerinden Split sekmesini tıkladı.		
4. Kod ekranında TextView görünümünü düzenleyen XML kodlarını layout kodlarının arasına yazdı.		
5. Shift+F10 tuşlarıyla uygulamanın ön izlemesini çalıştırdı.		

2.5.2. EditText

EditText, bir çeşit TextView görünümü olmasına rağmen düzenlenebilir. EditText, TextView ile hemen hemen aynı niteliklere sahiptir. Kullanıcının mobil uygulamaya metin girmesi gereken durumlarda EditText kullanılır. EditText görünümüne hem tek satırlı hem de çok satırlı metin girişi yapılabilir. Mobil uygulamada bir EditText görünümüne dokunulduğunda klavye otomatik olarak aktif hâle gelir. EditText görünümüne ait niteliklerden en çok kullanılanları Tablo 2.2'de verilmiştir.

Tablo 2.2: EditText Görünümüne Ait Nitelikler

Nitelik	Tanım
android:inputType	Kullanıcı tarafından girilen metnin nasıl olması gerektiği ve hangi amaçla kullanılacağı belirlenir. • **Text** • **textAutoComplete:** Kullanıcıya öneri metni sunulur. • **textAutoCorrect:** Kullanıcının girdiği metinde otomatik düzeltme etkinleştirilir. • **textPassword:** Kullanıcının girdiği metin ekranda gösterilmez. • **textUri:** Klavyede web sayfa uzantıları görülür. • **textEmailAddress:** Sadece e-posta girişi yapılmasına izin verilir. • **phone:** Numerik klavyenin açılması sağlanır.
android:minLines	Ekranda gösterilecek en az satır sayısı belirlenir.
android:maxLines	Ekranda gösterilecek en fazla satır sayısı belirlenir.
android:hint	EditText görünümüne metin girişi yapılmadan önce ipucu mesajı gösterilir.
android:maxLength	Kullanıcının metne girebileceği en fazla karakter sayısı belirlenir.

EditText görünümünün bir yerleşim içinde XML kodu şu şekildedir:

```
<EditText
    android:id="@+id/MU_EditText_id"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="E-postanızı buraya yazınız."
    android:textSize="30sp"
    android:inputType="textWebEmailAddress"
    android:maxLines="3"/>
```

EditText XML kodu bir yerleşim içine yazılıp emülatörde ön izleme yapılırsa e-posta metni girilebilecek bir ekran bileşeni görülür (Görsel 2.12).

Görsel 2.12: EditText ön izleme

4. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında tasarım ekranına XML kodlarıyla kullanıcının girdiği metni göstermeyen "Şifrenizi yazınız." metin görünümünü yerleştiriniz.

1. Adım: Mobil uygulama geliştirme programını çalıştırınız.

2. Adım: Çalışma alanında activity_main.xml dosyasını seçiniz.

3. Adım: Hem kod ekranını hem de Design ekranını görmek için Code, Split ve Design sekmelerinden Split sekmesini tıklayınız.

4. Adım: Kod ekranında EditText görünümünü düzenleyen şu XML kodlarını layout kodlarının arasına yazınız:

```
<EditText
    android:id="@+id/MU_EditText_id"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Şifrenizi yazınız."
    android:textSize="30sp"
    android:inputType="textPassword"
    android:maxLines="3"/>
```

5. Adım: XML görünüm kodları yazıldıktan sonra Design ekranında görünüm belirir (Görsel 2.11). Klavyeden Shift+F10 tuşlarıyla uygulamanın ön izlemesini yapınız.

SIRA SİZDE: Mobil uygulama geliştirme ortamında tasarım ekranına kullanıcının metin girebildiği XML kodlarıyla kullanıcıya öneri metni sunan "Mesleğinizi yazınız." metin görünümünü yerleştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama geliştirme programını çalıştırdı.		
2. Çalışma alanında activity_main.xml dosyasını seçti.		
3. Hem kod ekranını hem de Design ekranını görmek için Code, Split ve Design sekmelerinden Split sekmesini tıkladı.		
4. Kod ekranında EditText görünümünü düzenleyen XML kodlarını layout kodlarının arasına yazdı.		
5. Shift+F10 tuşlarıyla uygulamanın ön izlemesini çalıştırdı.		

2.5.3. Button

Düğme bileşeni, üzerine tıklandığında veya dokunulduğunda bir eylem gerçekleştirir. TextView bileşeniyle aynı niteliklere sahip olmakla beraber Button görünümünün kendine has bazı nitelikleri de vardır. Button görünümüne ait niteliklerden en çok kullanılanı Tablo 2.3'te verilmiştir.

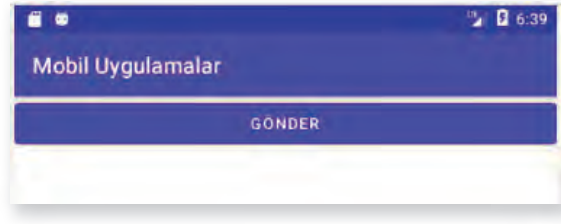
Tablo 2.3: Button Görünümüne Ait Nitelik

Nitelik	Tanım
android:onClick	Düğme tıklandığı zaman çalıştırılacak Java metodunun ismi yazılır.

Button görünümünün bir yerleşim içinde XML kodu şu şekildedir:

```
<Button
    android:id="@+id/MU_Button_id"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Gönder"
    android:onClick="java_metot"
    />
```

Button XML kodu bir yerleşim içine yazılıp emülatörde ön izleme yapılırsa ekranda bir düğme bileşeni görülür (Görsel 2.13).



Görsel 2.13: Button ön izleme

SIRA SİZDE: Mobil uygulama geliştirme ortamında tasarım ekranına üzerinde "Tamam" yazan ve tıklandığında "onayla" Java metodunu çağıran Button görünümünü yerleştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Kod ekranında Button görünümünü düzenleyen XML kodlarını layout kodlarının arasına yazdı.		
2. Shift+F10 tuşlarıyla uygulamanın ön izlemesini çalıştırdı.		
3. Emülatörde çalışan uygulamadaki Button görünümüne tıkladı.		

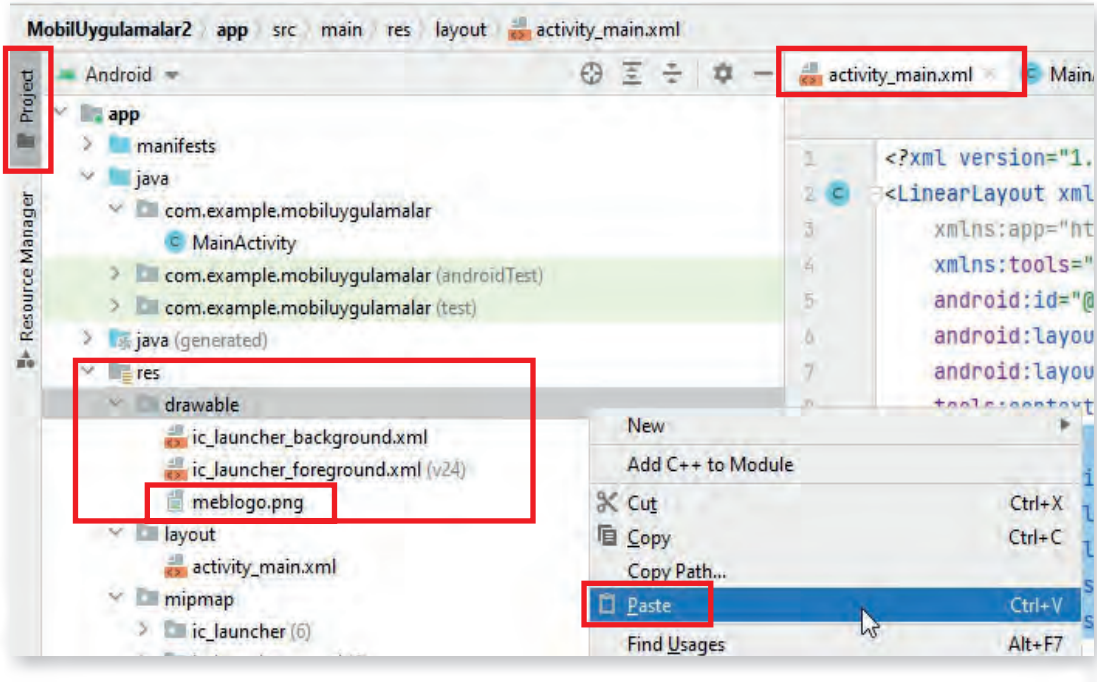
2.5.4. ImageView

ImageView, kullanıcı arayüzünde resim gösterilmesi gerektiğinde kullanılan görünümdür. ImageView görünümüne ait niteliklerden en çok kullanılanları Tablo 2.4'te verilmiştir.

Tablo 2.4: ImageView Görünümüne Ait Nitelikler

Nitelik	Tanım
android:maxHeight	Resmin maksimum yüksekliği belirlenir.
android:maxWidth	Resmin maksimum genişliği belirlenir.
android:src	Ekranda gösterilecek resmin kaynağı belirtilir.
android:scaleType	Resmin boyutlandırılması veya taşınması kontrol edilir. • **center:** Resim, merkeze yerleştirilir fakat ölçeklendirme yapılmaz. • **centerCrop:** Resim, eşit şekilde ölçeklendirilir. • **centerInside:** Resim, kapsayıcı içine yerleştirilir ve resmin kenarları ile kapsayıcının kenarları temas ettirilmez. Resim, kapsayıcının içinde bulunur. • **fitCenter:** Resim, merkezden ölçeklendirilir. • **fitEnd:** Resim, kapsayıcının sonundan, bir başka deyişle sağ taraftan ölçeklendirilir. • **fitStart:** Resim, kapsayıcının başlangıcından, bir başka deyişle sol taraftan ölçeklendirilir. • **fitXY:** Resmin, kapsayıcının tamamına doldurulması sağlanır. Resmin en ve boy oranı bozularak, resim gerilerek veya sıkıştırılarak kapsayıcıya yerleştirilir. • **matrix:** Çizim yapılırken görüntü matrisiyle resmin ölçeklendirilmesi için kullanılır.
android:tint	Resmin renklendirilmesi sağlanır.

Mobil uygulamaya yerleştirilmek istenen görüntü dosyası üzerinde sağ tuşla tıklanır ve **Kopyala** seçeneği seçilir. Resim dosyası kopyalandıktan sonra mobil uygulama geliştirme ortamında **Project** sekmesi seçili hâle getirilir. Dizinlerden **res** içinde **drawable** sağ tıklanır ve **Paste** seçeneği işaretlenerek resim bu dizine yapıştırılır. Yapıştırılan **meblogo.png** resim dosyasının ismi, XML'de kaynak dosya adı belirtilirken **android:src="@drawable/meblogo"** şeklinde yazılır (Görsel 2.14).



Görsel 2.14: Resim dosyasının drawable dizinine yerleştirilmesi

ImageView görünümünün bir yerleşim içinde XML kodu şu şekildedir:

```
<ImageView
    android:id="@+id/MU_ImageView_id"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scaleType="fitCenter"
    android:src="@drawable/meblogo"
/>
```

ImageView görünümü kullanılarak mobil uygulamada resim gösterilmesi için XML kodlar yazılıp ön izleme yapıldığında ekranda resim görülür (Görsel 2.15).

Görsel 2.15: ImageView ön izleme

5. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında tasarım ekranına XML kodlarıyla resim görünümünü ekranın sağına yerleştiriniz.

1. Adım: Mobil uygulama ekranına yerleştirilecek resme fare sağ tuşu ile tıklayınız.

2. Adım: Açılan menü listesinden Kopyala seçeneğini tıklayınız.

3. Adım: Mobil uygulama geliştirme ortamını çalıştırınız.

4. Adım: Çalışma ekranında dosya ve dizinlere ulaşmak için ekranın sol üst kenarındaki Project sekmesini tıklayınız.

5. Adım: Sırasıyla **app>res** dizinleri açınız**.**

6. Adım: res dizinindeki drawable dizinine sağ tıklayarak açılan seçeneklerden Paste seçiniz.

7. Adım: Kod ekranında ImageView görünümünü düzenleyen şu XML kodlarını layout kodlarının arasına yazınız:

```
<ImageView
    android:id="@+id/MU_ImageView_id"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scaleType="fitEnd"
    android:src="@drawable/meblogo"
/>
```

8. Adım: XML görünüm kodları yazıldıktan sonra Design ekranında görünüm belirir. Klavyeden Shift+F10 tuşlarıyla uygulamanın ön izlemesini yapınız.

SIRA SİZDE: Mobil uygulama geliştirme ortamında tasarım ekranına XML kodlarıyla resim görünümünü ekranın soluna yerleştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Mobil uygulama ekranına yerleştirilecek resme fare sağ tuşu ile tıkladı.		
2. Açılan menü listesinden Kopyala seçeneğini tıkladı.		
3. Mobil uygulama geliştirme ortamını çalıştırdı.		
4. Çalışma ekranında dosya ve dizinlere ulaşmak için ekranın sol üst kenarındaki Project sekmesini tıkladı.		
5. Sırasıyla app>res dizinlerini açtı.		
6. res dizinindeki drawable dizinine sağ tıklayarak açılan seçeneklerden Paste seçti.		
7. Kod ekranında ImageView görünümünü düzenleyen XML kodlarını layout kodlarının arasına yazdı.		
8. Klavyeden Shift+F10 tuşlarıyla ön izleme yaptı.		

2.5.5. CheckBox

Mobil uygulamada kullanıcıya onay kutuları sunarak bunlardan bir tanesini, birkaçını veya hepsini seçebilmesi gereken durumlarda kullanılan görünümdür. CheckBox görünümüne ait niteliklerden en çok kullanılanı Tablo 2.5'te verilmiştir.

Tablo 2.5: CheckBox Görünümüne Ait Nitelik

Nitelik	Tanım
android:checked	Onay kutusunun işaretli olması istenirse "true" değeri atanır.

İki seçenekle ekranda yer alan onay kutusu oluşturulması ve ikinci onay kutusunun seçili olması için yazılacak XML kodu şu şekildedir:

```
<CheckBox
   android:id="@+id/MU_cb1_id"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:checked="false"
   android:text="BİLİŞİM TEKNOLOJİLERİ ALANI" />
  <CheckBox
   android:id="@+id/MU_cb2_id"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:checked="true"
   android:text="YAZILIM GELİŞTİRME" />
```

ImageView görünümü kullanılarak mobil uygulamada resim gösterilmesi için XML kodlar yazılıp ön izleme yapıldığında ekranda resim görülür (Görsel 2.16).

Görsel 2.16: CheckBox ön izleme

SIRA SİZDE: Mobil uygulama geliştirme ortamında tasarım ekranına XML kodlarıyla "Millî Eğitim Bakanlığı" metnine sahip bir onay kutusu yerleştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Kod ekranında Button görünümünü düzenleyen XML kodlarını layout kodlarının arasına yazdı.		
2. Shift+F10 tuşlarıyla uygulamanın ön izlemesini çalıştırdı.		
3. Emülatörde çalışan uygulamadaki CheckBox görünümüne tıkladı.		

2.5.6. ProgressBar

ProgressBar, bir işlemin ilerleme durumunu göstermek için kullanılan görünümdür. Varsayılan olarak ProgressBar, dönen bir çember şeklinde mobil uygulamasında görüntülenir. ProgressBar görünümüne ait niteliklerden en çok kullanılanları Tablo 2.6'da verilmiştir.

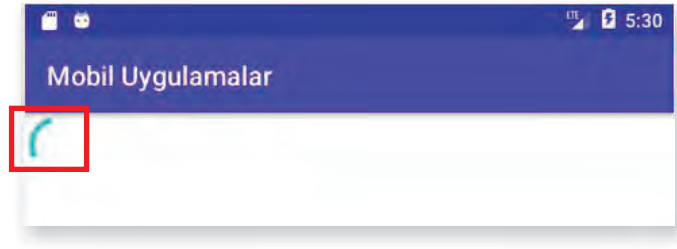
Tablo 2.6: ProgressBar Görünümüne Ait Nitelikler

Nitelik	Tanım
android:minHeight	ProgressBar görünümünün olabilecek en düşük yüksekliği belirlenir.
android:minWidth	ProgressBar görünümünün olabilecek en düşük genişliği belirlenir.
android:max	ProgressBar görünümünün alabileceği en yüksek değer belirtilir.
android:progress	İlerleme değeri 0 ile en yüksek değer arasında bir tam sayı olarak belirlenir.

ProgressBar ile ilerlemeyi göstermek için Determinate (Belirli) ve Indeterminate (Belirsiz) olmak üzere iki biçim çeşidi vardır. Indeterminate ProgressBar, XML kodlamayla şu şekilde oluşturulur:

```
<ProgressBar
    android:id="@+id/MU_ProgressBar_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true"/>
```

Yazılan XML kodlar emülatörde ön izleme yapıldığında cihazın ekranında dönen bir çember görüntülenir (Görsel 2.17).



Görsel 2.17: ProgressBar ön izleme

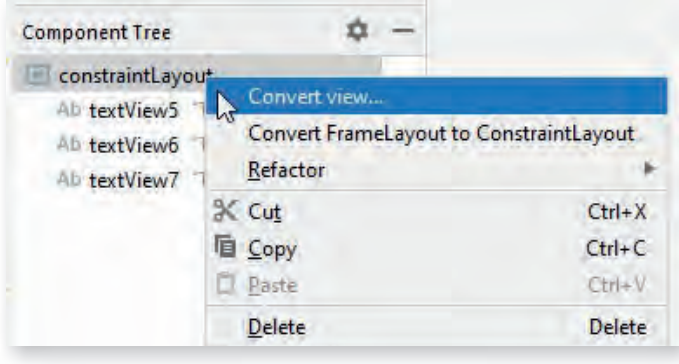
2.6. LAYOUT ÇEŞİTLERİ

Layout, alt görünümlerin ekranda nasıl konumlandırılacağını kontrol eden kapsayıcı bir yerleşimdir. Bu alt görünümler; TextView, Button veya ImageView gibi görünüm bileşenleridir. Mobil uygulama geliştirilirken layout kullanılarak görünümler düzenli bir şekilde ekrana yerleştirilir.

Kullanıcı arayüzü tasarlamak için Android SDK'nin sağladığı layout yerleşimleri vardır. Bunlardan bazıları şunlardır:

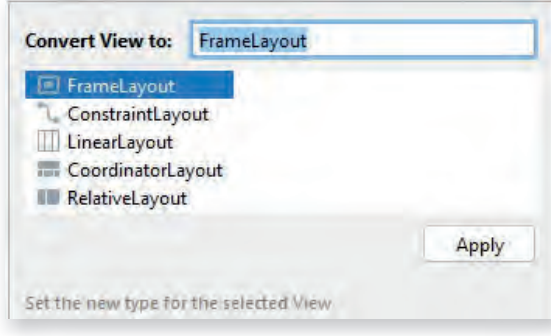
- ConstraintLayout (Kısıtlamalı Yerleşim)
- LinearLayout (Doğrusal Yerleşim)
- RelativeLayout (Bağıl Yerleşim)
- FrameLayout (Çerçeve Yerleşim)

Mobil uygulama geliştirme ortamında yerleşimlerin türünü değiştirmek için Component Tree paneli kullanılır. Component Tree paneli açık değilse ismi dikey yazılı olan sekmeye tıklanır. Açılan panelin en üstündeki layout üzerinde sağ fare tuşuna basılır. Açılan listeden Convert view... menüsü seçilir (Görsel 2.18).



Görsel 2.18: Component Tree paneli

Bu aşamada istenen yerleşim ismi seçilir ve Apply düğmesine tıklanır (Görsel 2.19).



Görsel 2.19: Layout çeşitleri

6. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında yerleşim türünü LinearLayout olarak değiştiriniz.

1. Adım: Fare ok işareti Component Tree panelinden değiştirilecek yerleşim isminin üzerindeyken sağ tuşa basınız.

2. Adım: Açılan menü listesinden Convert view… seçeneğini tıklayınız.

3. Adım: Gelen liste penceresinden LinearLayout yerleşimini seçiniz.

4. Adım: Apply düğmesine tıklayınız.

5. Adım: Klavyeden Shift+F10 tuşlarıyla uygulamanın ön izlemesini yapınız.

SIRA SİZDE: Mobil uygulama geliştirme ortamında yerleşim türünü RelativeLayout olarak değiştiriniz.

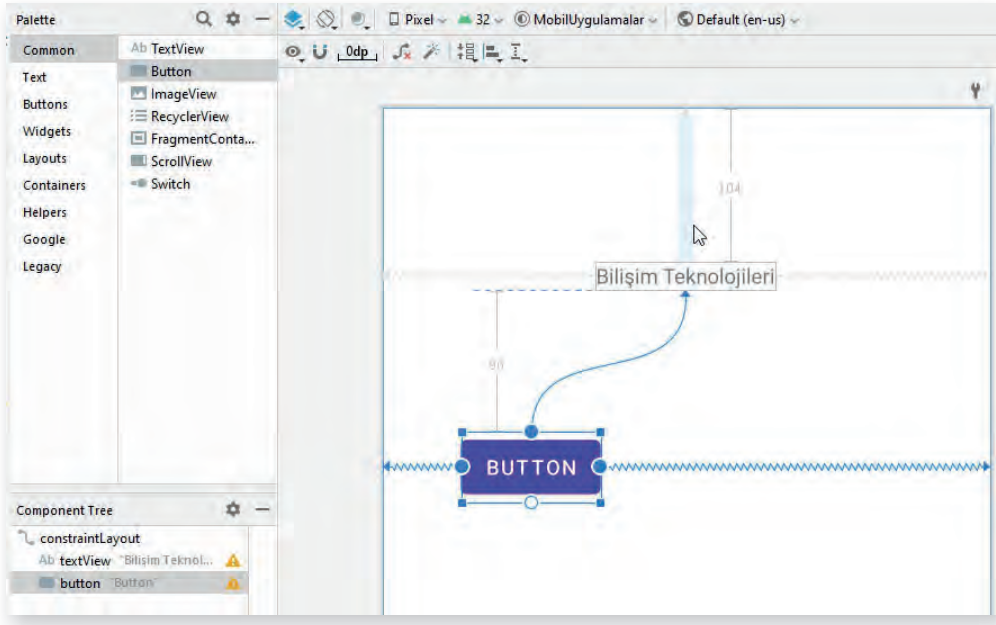
DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Fare ok işareti Component Tree panelinden değiştirilecek yerleşim isminin üzerindeyken sağ tuşa bastı.		
2. Açılan menü listesinden Convert view… seçeneğini tıkladı.		
3. RelativeLayout yerleşimini seçti.		
4. Apply düğmesine tıkladı.		
5. Klavyeden Shift+F10 tuşlarıyla ön izleme yaptı.		

2.6.1. ConstraintLayout

ConstraintLayout, Android işletim sisteminin 7. sürümü ile sunulan bir yerleşimdir. En yeni yerleşim türüdür ve getirdiği yeniliklerle en çok kullanılanıdır. Esnek olduğu, alt görünümleri ekrandan istenen yere konumlandırdığı ve çoğu yerleşim gereksinimlerini karşıladığı için tercih edilir. Basit kısıtlama ayarlarıyla bileşenler birbirinin içine yerleştirilmeden hızlı ve kolay bir şekilde kullanıcı arayüzü tasarlanabilir. Ayrıca Android Studio tasarım editörü kullanılarak basit sürükle bırak işlemleriyle tüm bir layout tasarlanabilir (Görsel 2.20).



Görsel 2.20: ConstraintLayout içinde TextView ve Button görünümleri

Palette paneli içinde sürükle bırak yöntemiyle alt görünümler tasarım ekranına yerleştirilir. Alt görünümün etrafındaki yuvarlaklar kullanılarak sürükle bırak yöntemiyle ya kenar boşluklarına ya da diğer görünümlere bağlanır. Bu sayede ekranın istenen bir konumuna görünüm yerleştirilebilir.

ConstraintLayout yerleşimi oluşturulmak istenirse XML kod olarak şunlar yazılır:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
       <!--
              Bu bölüme View ve ViewGroup
              XML kodları yazılır.
       -->
</androidx.constraintlayout.widget.ConstraintLayout>
```

UYARI: XML kodları yazılırken **<!--** ve **-->** sembolleri arasına yorum satırları yazılabilir. Yorum satırları, kodların çalışmasını etkilemez.

7. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında ConstraintLayout yerleşimine iki adet Button görünümü ekleyiniz ve constraint ayarlarını yaparak alt alta yerleştiriniz.

1. Adım: activity_main.xml dosyasının içindeki tüm kodları siliniz.

2. Adım: activity_main.xml dosyasının içinde ConstraintLayout oluşturmak için şu XML kodlarını yazınız:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Adım: Palette panelini kullanarak birinci Button görünümünü sürükle bırak ile tasarım ekranına yerleştiriniz.

4. Adım: Palette panelini kullanarak ikinci Button görünümünü sürükle bırak ile tasarım ekranına yerleştiriniz.

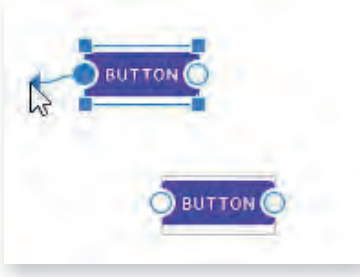
5. Adım: Birinci Button görünümünün sol tarafındaki yuvarlağa tıklayarak oluşan oku sürükle bırak ile sol kenara taşıyınız (Görsel 2.21).

6. Adım: Sol kenara yapışan birinci Button görünümünü sürükle bırak yöntemiyle sağa doğru ekranın ortasına getiriniz.

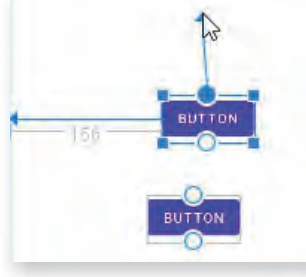
7. Adım: Birinci Button görünümünün üst tarafındaki yuvarlağa tıklayarak oluşan oku sürükle bırak ile üst kenara taşıyınız (Görsel 2.22).

9. Adım: Üst kenara yapışan birinci Button görünümünü sürükle bırak yöntemiyle aşağı doğru ekranın ortasına getiriniz.

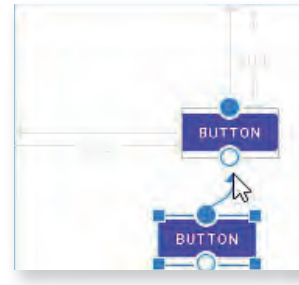
10. Adım: İkinci Button görünümünün üst tarafındaki yuvarlağa tıklayarak oluşan oku birinci Button görünümünün alt tarafına taşıyınız (Görsel 2.23).



Görsel 2.21: Sol constraint ayarı



Görsel 2.22: Üst constraint ayarı

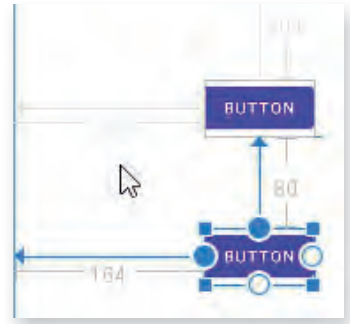


Görsel 2.23: İkinci Button üst constraint ayarı

11. Adım: Üstteki Button görünümüne yapışan ikinci Button görünümünü sürükle bırak yöntemiyle aşağı doğru ekranın ortasına getiriniz.

12. Adım: İkinci Button görünümün sol tarafındaki yuvarlağa tıklayarak oluşan oku sürükle bırak ile sol kenara taşıyınız.

13. Adım: Sol kenara yapışan ikinci Button görünümünü sürükle bırak yöntemiyle sağa doğru ekranın ortasına getiriniz (Görsel 2.24).

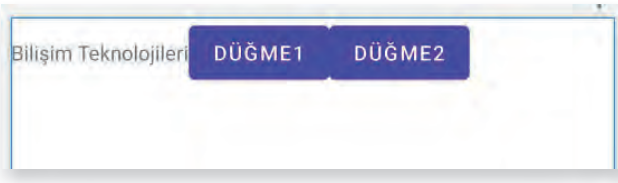


Görsel 2.24: Görünümler

UYARI: ConstraintLayout yerleşiminde birinci görünüme ikinci bir görünüm bağlandığında ikinci görünüm, birinci görünüme göre konumlandırılır. Bir görünüm sol kenar boşluğuna bağlanırsa bu görünümün sol kenar boşluğu korunur, sağ kenar boşluğu değişebilir.

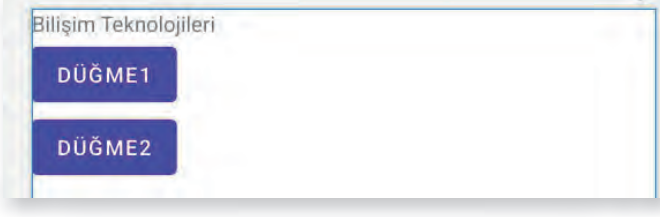
2.6.2. LinearLayout

LinearLayout, seçilen yatay veya dikey yönlendirmeye göre alt görünümleri tek bir satır veya tek bir sütuna yerleştiren bir yerleşim türüdür. LinearLayout yatay özelliği kullanılarak Palette panelinden getirilen TextView, Button1 ve Button2 alt görünümleri tek bir satırda yan yana yerleştirilir (Görsel 2.25).



Görsel 2.25: LinearLayout (Yatay)

LinearLayout dikey özelliği kullanılarak Palette panelinden getirilen TextView, Button1 ve Button2 alt görünümleri tek bir sütunda alt alta yerleştirilir (Görsel 2.26).



Görsel 2.26: LinearLayout (Dikey)

LinearLayout yerleşimi oluşturulmak istenirse XML kod olarak şunlar yazılır:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
        <!--
                Bu bölüme View ve ViewGroup
                XML kodları yazılır.
        -->
</LinearLayout>
```

UYARI: LinearLayout yerleşimini dikey ayarlamak için **android:orientation="vertical"** veya yatay ayarlamak için **android:orientation="horizantal"** komutları yazılır.

8. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında LinearLayout yerleşimine üç adet CheckBox görünümünü yatay olarak ekleyiniz.

1. Adım: activity_main.xml dosyasının içindeki tüm kodları siliniz.

2. Adım: activity_main.xml dosyasının içinde yatay LinearLayout oluşturmak için şu XML kodlarını yazınız:

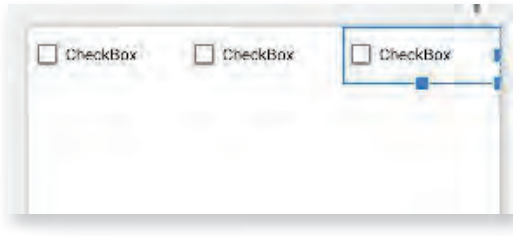
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizantal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</LinearLayout>
```

3. Adım: Palette panelini kullanarak Buttons bölümünü seçiniz.

4. Adım: Birinci CheckBox görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız.

5. Adım: İkinci CheckBox görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız.

6. Adım: Üçüncü CheckBox görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız (Görsel 2.27).

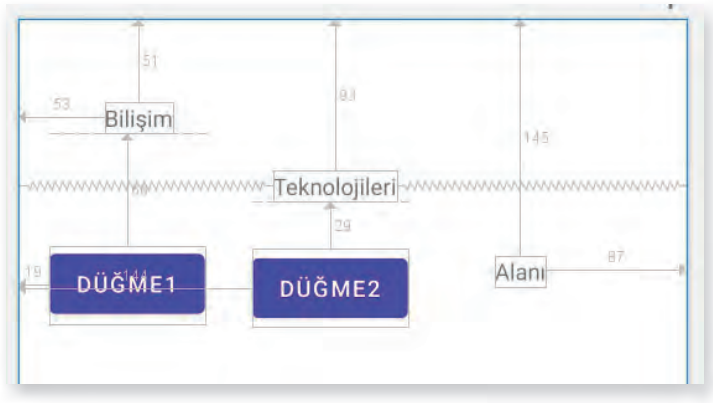


Görsel 2.27: Yatay LinearLayout yerleşimi

2.6.3. RelativeLayout

RelativeLayout, ConstraintLayouttan sonra en esnek yerleşimdir. Alt görünümler, ekranda kenar uzaklıklarına göre yerleştirilebildiği gibi birbirlerine bağlı olarak da yerleştirilebilir.

RelativeLayout içindeki üç adet TextView alt görünümü, kenarlardan mesafelerine göre tasarım ekranına yerleştirilir. Düğme1 alt görünümü, "Bilişim" TextView görünümüne bağlanır. Düğme2 alt görünümü ise "Teknolojileri" TextView görünümüne bağlanır (Görsel 2.28).



Görsel 2.28: RelativeLayout

Bir alt görünümü diğer bir alt görünüme göre düzenlemek için Above (Yukarıda), Below (Aşağıda), Left (Solda) ve Right (Sağda) ifadeleri kullanılır.

RelativeLayout yerleşimi oluşturulmak istenirse XML kod olarak şunlar yazılır:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
      <!--
            Bu bölüme View ve ViewGroup
            XML kodları yazılır.
      -->
</RelativeLayout>
```

9. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında RelativeLayout yerleşimine üç adet RadioButton görünümü ekleyiniz.

1. Adım: activity_main.xml dosyasının içindeki tüm kodları siliniz.

2. Adım: activity_main.xml dosyasının içinde yatay RelativeLayout oluşturmak için şu XML kodlarını yazınız:

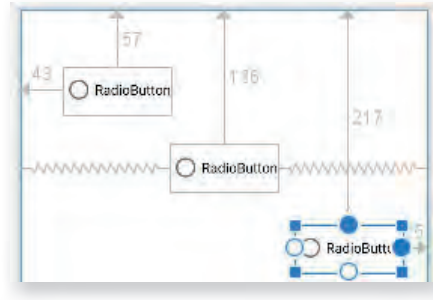
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</RelativeLayout>
```

3. Adım: Palette panelini kullanarak Buttons bölümünü seçiniz.

4. Adım: Birinci RadioButton görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız.

5. Adım: İkinci RadioButton görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız.

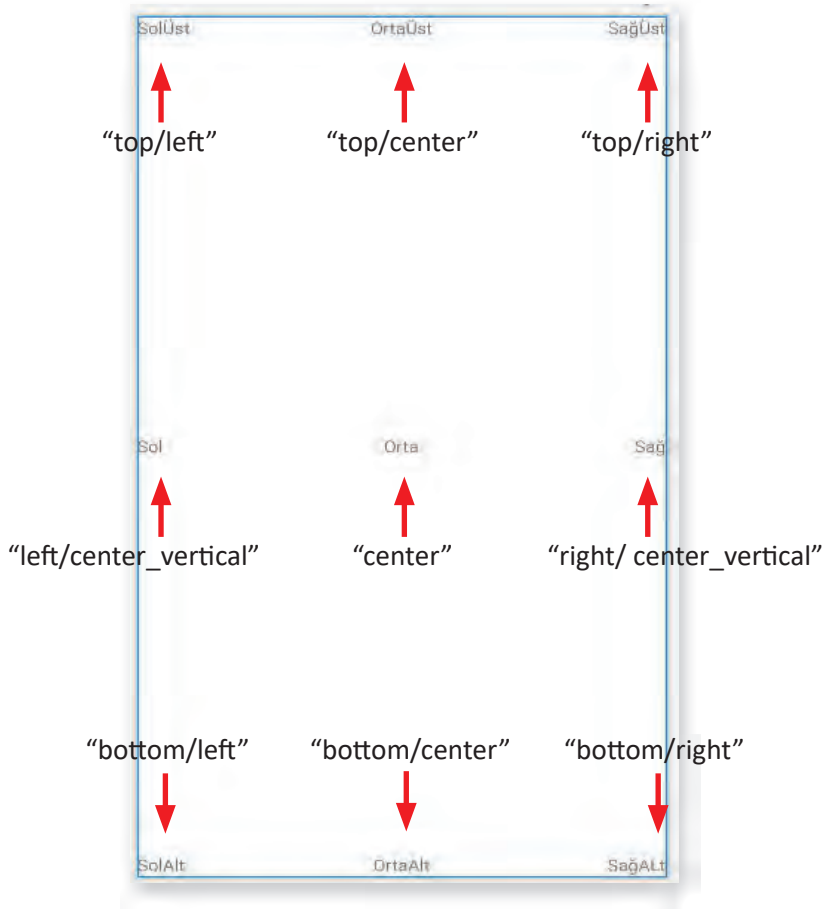
6. Adım: Üçüncü RadioButton görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız (Görsel 2.29).



Görsel 2.29: RelativeLayout yerleşimi

2.6.4. FrameLayout

FrameLayout, alt görünümleri en basit şekilde organize edebilecek yerleşim türüdür. Ekranın bir bölümü kaplanır ve alt görünüme atanacak **android:layout_gravity** niteliğiyle görünümlerin konumlandırılması sağlanır (Görsel 2.30). Bu nitelik atanmazsa alt görünümler üst üste yerleştirilir.



Görsel 2.30: FrameLayout yerleşimi parametreleri

FrameLayout yerleşimi oluşturulmak istenirse XML kod olarak şunlar yazılır:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
      <!--
            Bu bölüme View ve ViewGroup
            XML kodları yazılır.
      -->
</FrameLayout>
```

10. UYGULAMA: İşlem adımlarına göre mobil uygulama geliştirme ortamında FrameLayout yerleşimine üç adet TextView görünümü ekleyiniz.

1. Adım: activity_main.xml dosyasının içindeki tüm kodları siliniz.

2. Adım: activity_main.xml dosyasının içinde FrameLayout oluşturmak için şu XML kodlarını yazınız:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</FrameLayout>
```

3. Adım: Palette panelini kullanarak Common bölümünü seçiniz.

4. Adım: Birinci TextView görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız.

5. Adım: İkinci TextView görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız.

6. Adım: Üçüncü TextView görünümünü sürükle bırak yöntemiyle tasarım ekranına bırakınız.

7. Adım: Birinci TextView görünümüne XML **android:layout_gravity="top|left"** kodunu ekleyiniz.

8. Adım: İkinci TextView görünümüne XML **android:layout_gravity="center"** kodunu ekleyiniz.

9. Adım: Üçüncü TextView görünümüne XML **android:layout_gravity="top|right"** kodunu ekleyiniz (Görsel 2.31).

Görsel 2.31: FrameLayout yerleşimi

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Apps ile mobil işletim sisteminde çalışan uygulamalar ifade edilir.

2. ( ) Eylem çubuğunda uygulama ismi, activity ismi veya simgeleri, ek görünümler ve etkileşimli nesneler bulunur.

3. ( ) ViewGroup içinde başka bir ViewGroup yer alır.

4. ( ) İçeriğin görüntülenmesi için gereken kadar alanı kaplayacağı wrap_content ile ifade edilir.

5. ( ) Seçili görünümün nitelikleri Palette panelinden değiştirilir.

6. ( ) CheckBox ile mobil uygulama ekranına resim yerleştirilir.

7. ( ) Layout kullanılarak görünümler düzenli bir şekilde ekrana yerleştirilir.

B) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

8. Aşağıdakilerden hangisi taşınabilir cihazlara ait ekranlardan biri değildir?

A) Home screen B) All apps C) Recent screen
D) Lock screen E) Status Bar

9. Aşağıdakilerden hangisi bir mobil uygulama ekran yapısı içinde yer almaz?

A) Status Bar B) Palette paneli C) Action Bar
D) Navigation Bar E) Uygulama ismi

10. Aşağıdakilerden hangisi bir mobil uygulamada kullanıcıya metin göstermek için kullanılan görünümdür?

A) TextView B) Button C) CheckBox
D) ProgressBar E) ImageView

11. Aşağıdakilerden hangisi bir mobil uygulamada kullanıcıya resim göstermek için kullanılan görünümdür?

A) EditText B) ImageView C) CheckBox
D) ProgressBar E) Button

12. Aşağıdakilerden hangisi CheckBox görünümüne ait niteliklerden "android:Checked" ile belirtilir?

A) Görünümün seçili olması veya olmaması sağlanır.
B) Görünümün genişliği ayarlanır.
C) Görünümün yüksekliği ayarlanır.
D) Görünümün metni belirlenir.
E) Görünüm resminin kaynağı belirtilir.

13. Aşağıdaki yerleşimlerden hangisi görünümlerin ekrana yatay ve dikey şekilde konumlandırılmasını sağlar?

A) LinearLayout B) ConstraintLayout C) FrameLayout
D) RelativeLayout E) Home screen

3. ÖĞRENME BİRİMİ

TEMEL KOMUTLAR

KONULAR

3.1. VERİ

3.2. DEĞİŞKENLER

3.3. VERİ TİPLERİ

3.4. SABİTLER

3.5. İSİMLENDİRME KURALLARI

3.6. OPERATÖRLER

3.7. HATA AYIKLAMA

3.8. HATA DÜZELTME

NELER ÖĞRENECEKSİNİZ?

- Java dilinde değişken kavramını tanımlama
- Değişken türlerini sıralama
- Değişkenleri, isimlendirme kurallarına uygun kullanma
- Sabit kavramını tanımlama
- Sabitleri, isimlendirme kurallarına uygun kullanma
- Operatörlerin kullanım amaçlarını açıklama
- Değişkenlere değer atama
- Matematiksel operatörleri kullanma
- Hata türlerini tanımlama
- Hata ayıklamayı açıklama
- Logcat panelini kullanma
- Yakalanan hataları düzeltme

ANAHTAR KELİMELER

- Aritmetiksel operatörler
- Atama operatörleri
- Değişken
- Hata ayıklama
- İlkel veri tipleri
- Karşılaştırma operatörleri
- Logcat
- Mantıksal operatörler
- Referans veri tipleri
- Sabit
- Veri
- Veri tipleri

HAZIRLIK ÇALIŞMALARI

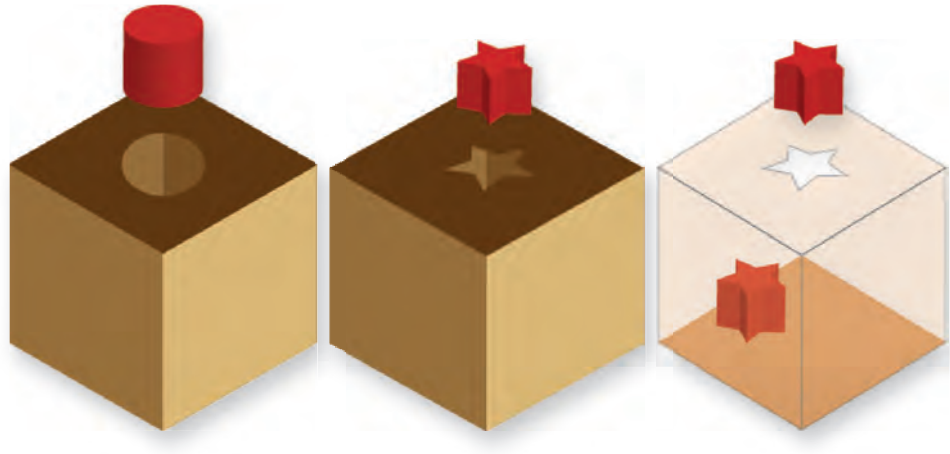
1. Mobil programlar içinde kullanılan verilerin muhafaza edildiği yerler hakkındaki düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Kalıcı hafıza ile geçici hafıza arasındaki farklılıklar neler olabilir? Arkadaşlarınızla değerlendiriniz.
3. Unicode ve ASCII tablosunu araştırınız. Edindiğiniz bilgileri arkadaşlarınızla paylaşınız.

3.1. VERİ

Akıllı cihazlarda uygulamalar tarafından işlenmek üzere hafızada tutulan her türlü ham bilgiye **veri** denir. Veriler tek başlarına bir şey ifade etmez. Verilerin bir anlam ifade etmesi için veri üzerinde işlem yapılması gereklidir. Tek başına 60 sayısının bir anlamı yoktur. Bu sayı, üzerinde yapılacak işleme göre anlam kazanır. Örneğin bir öğrencinin mobil uygulamalar dersine ait not ortalaması 60 olarak ifade edilirse ders geçme notu 50'den büyük olduğu için öğrenci mobil uygulamalar dersinden geçmiş sayılır.

3.2. DEĞİŞKENLER

Yazılım geliştirme sürecindeki en temel konulardan biri değişkenlerdir. Akıllı cihazlar işlem yapabilmek için verilere ihtiyaç duyar. Akıllı cihazlar bu verileri hafızlarında muhafaza etmelidir çünkü gerektiğinde bu verilere ulaşabilmeli, verilerin üzerinde işlem yapabilmeli ve gerektiğinde verileri değiştirebilmelidir. Hafızada tutulan, gerektiğinde değerleri değişebilen bilgilere programcılık dünyasında **değişken** adı verilir. Değişkenler, içinde verileri tutan kutular olarak düşünülebilir. Bu kutuların büyüklüğü, içine alacağı verilere göre değişir (Görsel 3.1).



Görsel 3.1: Değişkenlerdeki veri farklılıkları

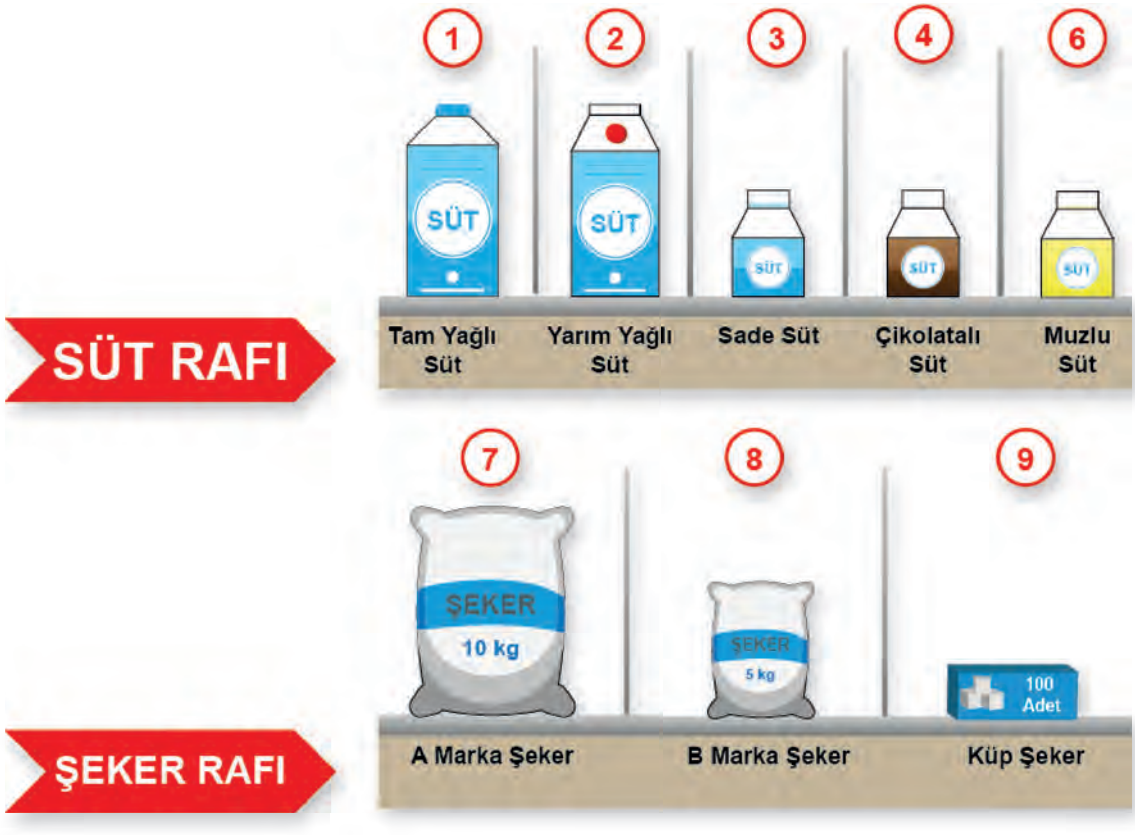
Değişkenlerin dört özelliği bulunur (Görsel 3.2). Bu özellikler şu şekilde sıralanır:

- **Değişkenin Türü:** Kutuya konulacak verinin büyüklüğüne göre kutu tipi (sayı, yazı vb.)
- **Değişkenin Adı:** Kutunun ismi (YAZILI1, İSİM1)

- **Değişkenin Değeri:** Kutunun içeriği (90, ÖMER)
- **Değişkenin Adresi:** Kutunun yeri (1 No.lu raf, 2 No.lu raf)

Görsel 3.2: Değişkenler ve özellikleri

Mobil cihazların hafızası bir marketin reyonlarına benzetilebilir (Görsel 3.3). Bir markette farklı ürün türlerine ait farklı raflar bulunur. Her rafın içine yerleştirilecek ürün türüne göre belirli bir bölme ayarlanır. Her bir bölmeye en büyük ürün de en küçük ürün de konulabilir. Bu ürünler farklı olduğu için üzerinde yapılacak işlemler de farklılık gösterir.



Görsel 3.3: Değişkenler ve raflar benzetimi

SIRA SİZDE: Görsel 3.3'teki değişkenlerin türünü, adresini ve değerini verilen örneklerdeki gibi yazınız.

Değişkenin Adı	Değişkenin Türü	Değişkenin Değeri	Değişkenin Adresi
Tam Yağlı Süt	Süt	1 Kutu Süt	1 No.lu Bölme
A Marka Şeker	Şeker	Şeker 10 kg	7 No.lu Bölme

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Değişken adını yazdı.		
2. Değişken türünü yazdı.		
3. Değişken değerini yazdı.		
4. Değişken adresini yazdı.		

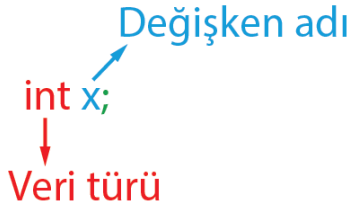
Tanımlanan veri türlerine göre mobil cihazın hafızasında yerler tahsis edilir. Ayrılan bu alanlar, veri türüne göre farklı büyüklüktedir. Mobil cihaz hafızasının verimli kullanılması için mobil program içindeki veri türleri iyi analiz edilmelidir.

Veri türlerinin farklılığından dolayı bu veriler üzerinde gerçekleştirilecek işlemler de farklılık gösterir (Görsel 3.4). Sayı veri türleri ile toplama, çıkarma, çarpma ve bölme gibi matematiksel işlemler gerçekleştirilir. Metin veri türleri ile bir metni büyük veya küçük harflere çevirme gibi metinsel işlemler gerçekleştirilir.

Görsel 3.4: Sayısal ve metinsel işlemler

3.2.1. Değişken Yapısı

Java kodlarıyla değişken tanımlama iki kısımdan oluşur. İlk kısım, saklanmak istenen bilgiye göre veri türüdür, ikinci kısım ise değişkenin adıdır (Görsel 3.5).



Görsel 3.5: Değişken yapısı

NOT:

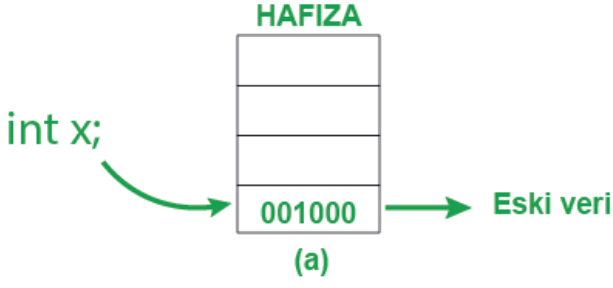
Java programlama dillinde noktalı virgül, derleyiciye talimatın nerede bittiğini gösterir. Böylelikle Java kodları bir satırda veya birden çok satırda yazılabilir.

3.2.2. Değişken Tanımlama ve Atama

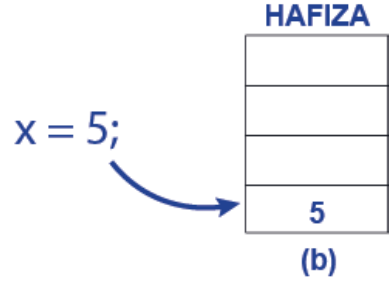
Değişkenin adını ve türünü belirtmeye **değişken tanımlama (declaration)** denir (Görsel 3.6a). Değişken tanımlama yapıldığında hafızada değişken türüne göre boş bir yer ayrılır. Bu yerin adresi ile değişkenin adı aynı şeyi ifade eder.

Değişkenin içindeki değerin değiştirilmesi işlemine **değişkene değer atama (assignment)** denir (Görsel 3.6b). Değişkene değer atama işleminde değişkene ait hafızadaki değer, yeni değer ile değiştirilir. Kod olarak değişkeni başlatma işlemi ile aynıdır fakat arka planda yapılan işlemler farklıdır.

Mobil uygulama geliştirme ortamında bir değişken tanımlanıp değişkenin içine bir değer atanmadığında program hata verir. Bunun nedeni, değişken tanımlandığında hafızada ayrılan bölümde daha önceden bir verinin var olma olasılığıdır. Java dili ile mobil programı yazılırken ilkel değişkenlere mutlaka bir değer ataması yapılmalıdır.



Görsel 3.6a: Değişken tanımlama



Görsel 3.6b: Değişkene değer atama

Değişken tanımlama ve değer atama işlemleri birlikte yapılabilir.

```
int x = 5;
```

NOT:

Bazı programlama dillerinde değişken tanımlanması yapıldığında değişkenin ilk değeri, o değişken türüne göre otomatik olarak verilir. Sayısal veri türleri için bu varsayılan değer 0'dır.

SIRA SİZDE:

Tablodaki kodlara göre değişkeni başlatma ve değişkene değer atama işlemine onay işareti koyunuz.

Kod	Değişken Başlatma	Değişken Atama
int puan;		
puan = 80;		
int puan = 80;		

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

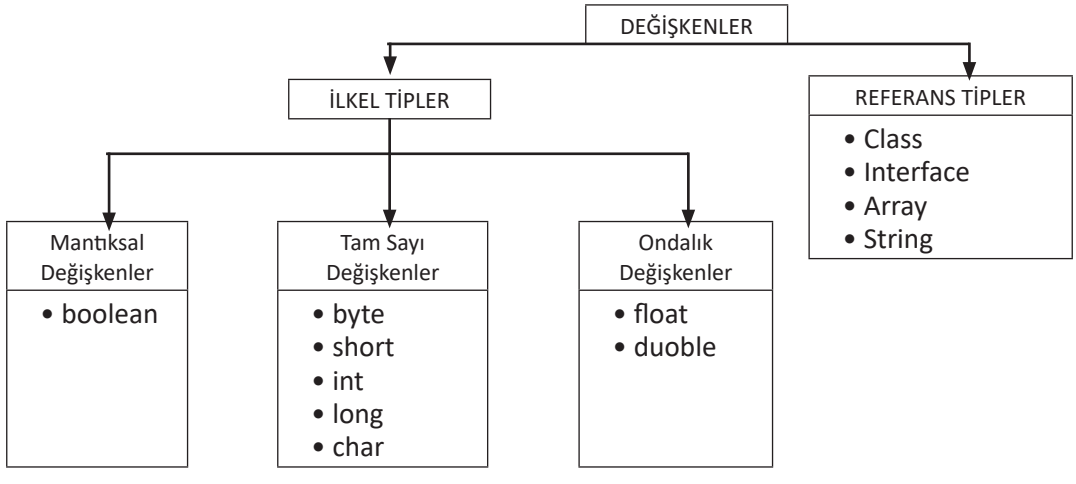
KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Değişken başlatma kodlarını doğru olarak buldu.		
2. Değişkene değer atama kodlarını doğru olarak buldu.		

3.3. VERİ TİPLERİ

Java programlama dili, statik ve kesin olarak yazılmış (Strongly Typed) bir programlama dili olarak tanımlanır. Bu durum, değişkenin oluşturulması sırasında değişken türünün tanımlaması gerektiği ve daha sonra değişken türünün değiştirilemeyeceği anlamına gelir. Değişken türü ile değişkenin değeri farklı kavramlardır.

Java'da ilkel (primitive) veri tipleri ve referans veri tipleri olmak üzere iki değişken grubu vardır (Görsel 3.7).



Görsel 3.7: Değişken tipleri

3.3.1. İlkel (Temel) Veri Tipleri

İlkel veri tipleri, içinde bir seferde tek bir değer tutan veri tipleridir. Java programlama dilinde ilkel veri tipleri program içinde ilk oluşturuldukları andan itibaren bir değere sahip olmak zorundadır.

NOT: İlkel veri tipleri küçük harflerle yazılır.

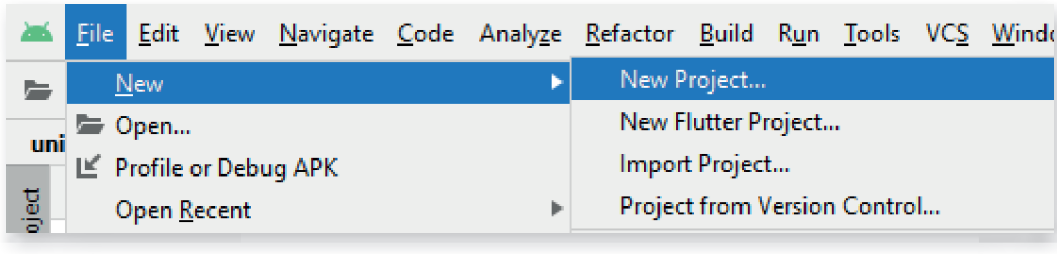
3.3.1.1. Mantıksal Veri Tipi

Java'da boolean olarak isimlendirilen bir adet mantıksal veri tipi vardır.

boolean: boolean veri tipinin true (doğru) ve false (yanlış) olmak üzere sadece iki değeri vardır. boolean veri tipi, evet veya hayır şeklinde cevaplanabilecek soruların cevapları şeklinde düşünülebilir.

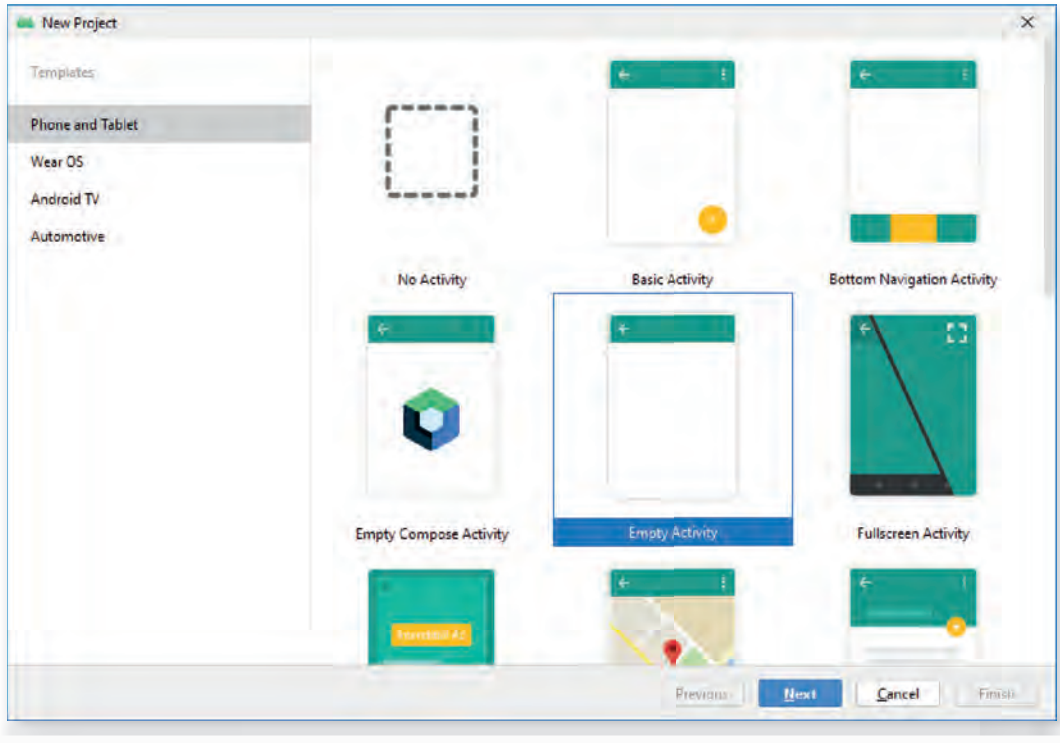
1. UYGULAMA: İşlem adımlarına göre ilkel veri tiplerinin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında File menüsünden New>New Project komutunu tıklayınız (Görsel 3.8).



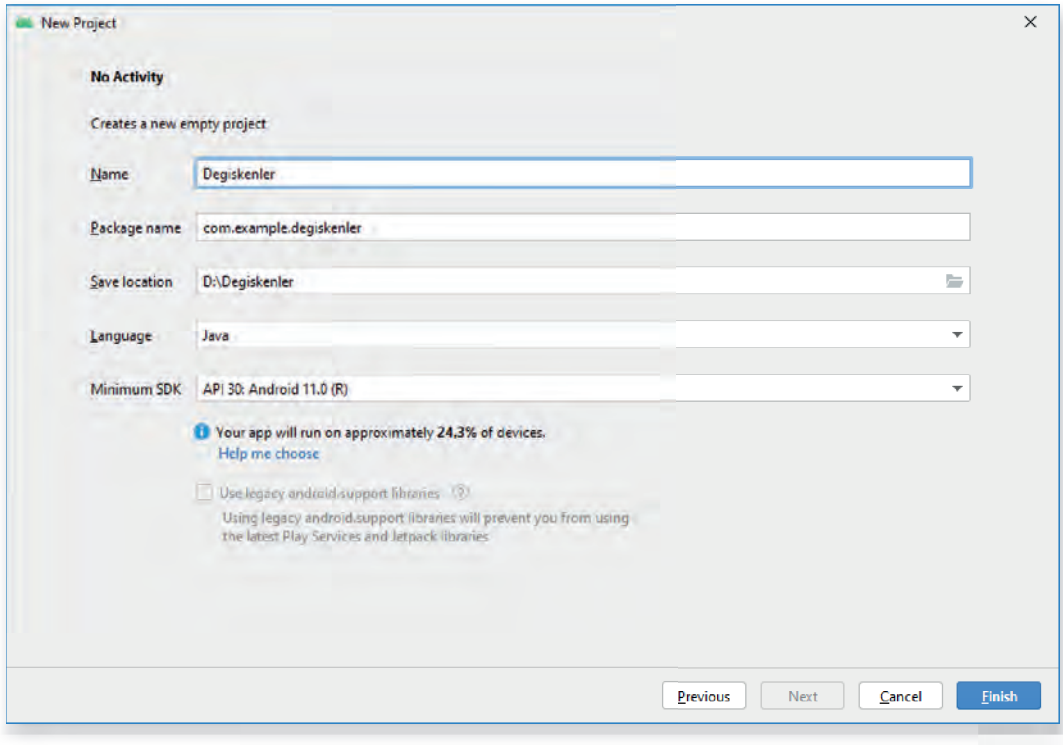
Görsel 3.8: New Project komutu

2. Adım: New Project penceresinden Empty Activity seçeneğini seçip Next düğmesine tıklayınız (Görsel 3.9).



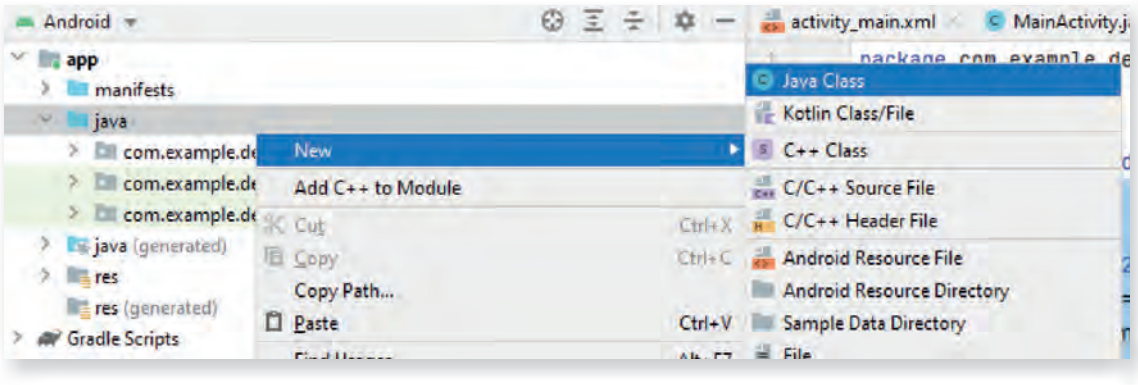
Görsel 3.9: New Project penceresi

3. Adım: Ekrana gelen pencereden Name kısmına uygulamanıza vereceğiniz adı giriniz. Projenin kaydedileceği yolu Save Location kısmına giriniz. Finish düğmesine basınız (Görsel 3.10).



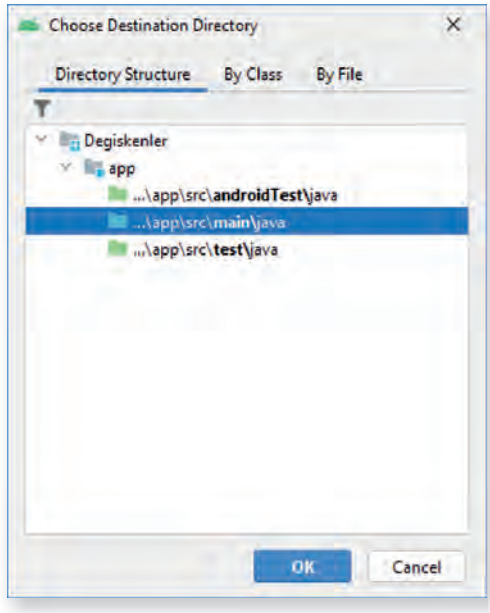
Görsel 3.10: New Project No Activity penceresi

4. Adım: Project penceresinde app>java üzerine sağ tıklayıp New>Java Class'ı tıklayınız (Görsel 3.11).



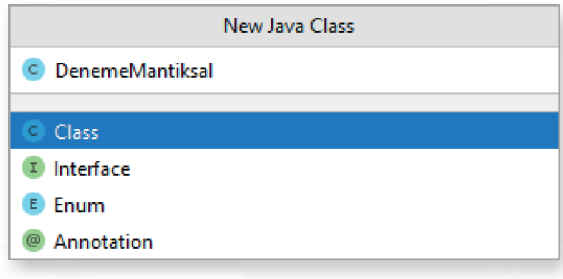
Görsel 3.11: Yeni Java sınıfı oluşturma

5. Adım: Choose Destination Directory penceresinde sınıfınızın kaydedileceği klasör olarak app\src\main\java'yı seçiniz ve OK düğmesine tıklayınız (Görsel 3.12).



Görsel 3.12: Choose Destination Directory penceresi

6. Adım: New Java Class penceresinde Name kısmına "DenemeMantiksal" yazarak yeni bir sınıf oluşturunuz (Görsel 3.13).



Görsel 3.13: New Java Class penceresi

NOT: Java'da sınıf isimleri büyük harfle başlar.

7. Adım: DenemeMantiksal sınıfı içine şu kodu yazınız:

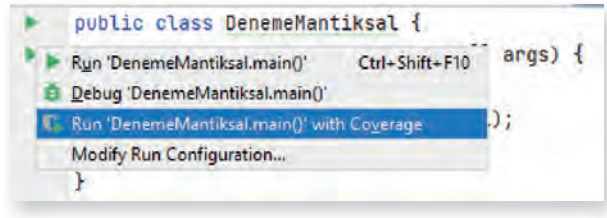
```
public class DenemeMantiksal {
    public static void main(String[] args) {
        boolean degisken1 = true;
        System.out.println(degisken1);
    }
}
```

NOT: `public static void main(String[ ] args)` yapısının oluşturulması için mobil uygulama geliştirme programında **main** yazılıp TAB tuşuna basılmalıdır.

NOT: `System.out.println` kendisine iletilen bilgiyi yazdırmak için kullanılır.

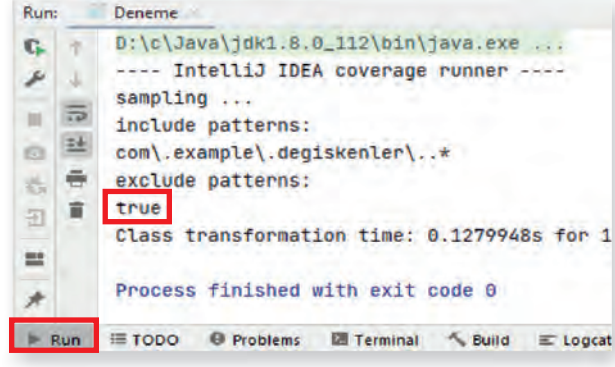
NOT: `System.out.println` yapısının oluşturulması için mobil uygulama geliştirme programında **sout** yazılıp TAB tuşuna basılmalıdır.

8. Adım: Yazılan programı çalıştırmak için Run menüsünden Run '.....' with Coverage komutunu tıklayınız (Görsel 3.14). Aynı işlem, kod editöründeki yeşil çalıştırma düğmesine tıklanarak da gerçekleştirilebilir. Proje, Run komutuyla çalıştırılırsa hata penceresiyle karşılaşılır. Bunun nedeni, bir Android yaşam döngüsünde projeyi çalıştırmak için main fonksiyonunun olmamasıdır.



Görsel 3.14: Run with Coverage komutu

9. Adım: Cover penceresinde çalışan kodun çıktısını inceleyiniz (Görsel 3.15).



Görsel 3.15: Cover penceresi

SIRA SİZDE: "degisken2" adında yeni bir değişken tanımlayarak, değişkenin içine **false** değerini aktarıp yazdırınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

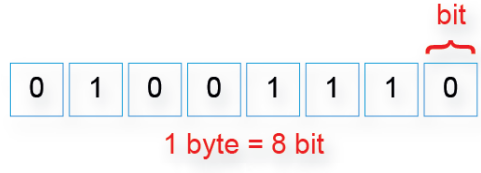
KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Project penceresinde New Java Class komutunu seçti.		
5. Java sınıfı için hedef klasör seçti.		
6. Java sınıfına "DenemeMantiksal" adını belirledi.		
7. Mantıksal değişken oluşturup değişkenin değerini atadı.		
8. Run with Coverage komutuyla uygulamayı çalıştırdı.		
9. Cover penceresinde uygulama çıktısını gözlemledi.		

3.3.1.2. Tam Sayı Veri Tipleri

Değer olarak tam sayıları içinde barındırabilen veri tipleridir. Büyüklüklerine göre farklı sayı aralıklarına sahiptirler.

NOT: Bilgisayarlardaki en küçük veri boyutu bit'tir. Bitler 0 veya 1 ile temsil edilir (Görsel 3.16).



Görsel 3.16: Byte ile bit ilişkisi

Tam sayı veri türleri şunlardır (Tablo 3.1):

- **byte:** Değeri -128 ile 127 arasında olabilen tam sayı veri tipidir. Hafızada 1 byte (8 bit) yer kaplar.
- **short:** Değeri -32.768 ile 32.767 arasında olabilen tam sayı veri tipidir. Hafızada 2 byte yer kaplar.
- **int:** Değeri -2.147.483.648 ile 2.147.483.647 arasında olabilen tam sayı veri tipidir. Hafızada 4 byte yer kaplar. Programcılıkta en çok kullanılan tam sayı veri tipidir.

- **long:** Değeri -9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arasında olabilen tam sayı veri tipidir. Hafızada 8 byte yer kaplar.
- **char:** Türkçe karakter anlamına gelen İngilizce Character kelimesinin kısaltmasıdır. Bu değişken, içinde Unicode olarak sadece bir karakter barındırır. Değer aralığı 0 ile 65.535 arasındadır. Hafızada 2 byte yer kaplar.

Tablo 3.1: Tam Sayı Veri Türleri

Tam Sayı Türü	Kapladığı Alan	Değer Aralığı
byte	1 byte (8 bit)	-127,…,127
short	2 byte	-32768,…,32767
int	4 byte	-2147483648,…,2147483647
long	8 byte	-9223372036854775808,…,9223372036854775807
char	2 byte	0,…,65535 arası Unicode ile kodlanmış karakter

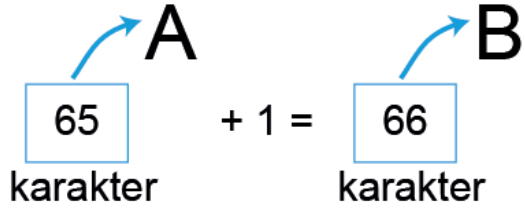
Tam sayı veri tipleri büyüklükleri Görsel 3.17'de verilmiştir.

Görsel 3.17: Tam sayı veri türlerinde büyüklük karşılaştırması

Unicode, her bir karakterin sayısal olarak ifade edildiği endüstri standardıdır. Örnek olarak Unicode ile kodlanmış A karakterinin değeri 65'tir.

Tanımlanan bir char veri türü, hafızada değer olarak sayısal şekilde ifade edilir. Bundan dolayı char veri tipi, tam sayı veri tipleri içinde gruplandırılır.

Değişkenin tipi, değişken üzerinde gerçekleştirilecek işlemleri de belirtir. "char" veri tipi sayısal olduğu için üzerinde aritmetiksel işlemler gerçekleştirilir (Görsel 3.18).



Görsel 3.18: char veri türü

NOT:

Herhangi bir metin editörü açılıp, klavyeden Alt tuşu basılıyken klavyenin sağ tarafındaki rakamlardan 65 yazılarak Alt tuşu bırakıldığında A harfi elde edilir. Klavye üzerinde olmayan karakterler bu yolla yazılabilir.

ETKİNLİK : Verilen karakterleri ASCII tablosundan bulup yazınız.

Karakter	ASCII Kodu
~	
&	
|	
@	

ETKİNLİK: Sırasıyla 77, 69 ve 66 değerlerine sahip char veri tipleri ile oluşan kelimeyi yazınız.

ASCII Kodu	Karakter
77	
69	
66	

2. UYGULAMA: İşlem adımlarına göre tam sayı veri tiplerinin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "DenemeTamSayilar" adında yeni bir sınıf oluşturunuz.

2. Adım: DenemeTamSayilar sınıfı içine şu kodu yazınız:

```
public class DenemeTamSayilar {
    public static void main(String[] args) {
        byte kucukSayi = 127;
        short kisaSayi = 32767;
        int tamSayi = 2147483647;
        long uzunSayi = 9223372036854775807L;
        System.out.println("byte:  "+ kucukSayi);
        System.out.println("short: " + kisaSayi);
        System.out.println("int:   " + tamSayi);
        System.out.println("long:  " + uzunSayi);
    }
}
```

NOT:

Java programlama dilinde, long veri türünde yazılan sayının sonuna büyük L harfi yazılır. Java, sayının sonuna L yazılmaz ise sayıyı otomatik olarak int veri türüne çevirmeye çalışır. Sayı, int veri türünün alabileceği değerler dışına taşarsa program hata verir.

3. Adım: Yazdığınız kodu Run menüsünden Run 'DenemeTamSayilar.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
byte:  127
short: 32767
int:   2147483647
long:  9223372036854775807
```

SIRA SİZDE: İşlem adımlarına göre ikinci uygulamayı yeniden tasarlayınız.

- Uygulamadaki uzunSayi adlı değişkenin değerini 255 şeklinde değiştirip çalıştırınız.
- Uygulamadaki uzunSayi adlı değişkenin değerini 2147483648 şeklinde değiştirip çalıştırınız.
- Uygulamadaki uzunSayi adlı değişkenin değerini 2147483648L şeklinde değiştirip çalıştırınız.

Sonuçları tabloya yazınız.

Değer	Sonuç
255	
2147483648	
2147483648L	

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Değişkenin değerini 255 olarak değiştirdi.		
2. Uygulama sonucunu 255 olarak buldu.		
3. Değişkenin değerini 2147483648 olarak değiştirdi.		
4. Uygulama sonucu olarak "integer number too large" hatasını buldu.		
5. Değişkenin değerini 2147483648L olarak değiştirdi.		
6. Uygulama sonucunu 2147483648 olarak buldu.		

3. UYGULAMA: İşlem adımlarına göre char veri tipinin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "DenemeChar" adında yeni bir sınıf oluşturunuz.

2. Adım: DenemeChar sınıfı içine şu kodu yazınız:

```
public class DenemeChar {
    public static void main(String[] args) {
        char karakter = 'A';
        System.out.println("Karakter: " + karakter);
        karakter = 'A' + 1;
        System.out.println("Karakter: " + karakter);
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'DenemeChar.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
Karakter: A
Karakter: B
```

SIRA SİZDE: Üçüncü uygulamadaki toplama kısmında yer alan sayıyı artırarak çıkışı küçük harfle 'a' olacak şekilde veren sayıyı bulunuz (25 ile 35 arasındaki sayıları deneyiniz.).

Sayı

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Toplam sayısını değiştirdi.		
2. Doğru sayıyı buldu.		

4. UYGULAMA: İşlem adımlarına göre char olarak verilen değişkenin ASCII kodunu bulan bir uygulamayı tasarlayınız.

1. Adım: "CharAscii" adında yeni bir sınıf oluşturunuz.

2. Adım: CharAscii sınıfı içine şu kodu yazınız:

```
public class CharAscii {
    public static void main(String[] args) {
        char karakter = 'a';
        int ascii = (int) karakter;
        System.out.println("Karakter: " + karakter);
        System.out.println("ASCII kodu: " + ascii);
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'CharAscii.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
Karakter:   a
ASCII kodu: 97
```

SIRA SİZDE: ASCII kodu verilen int tipindeki değişkenin karakterini bulan bir uygulamayı tasarlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Project penceresinde New Java Class komutunu seçti.		
5. Java sınıfı için hedef klasör seçti.		
6. Java sınıfına "AsciiChar" adını belirledi.		
7. int veri tipinde bir değişken tanımlayarak değer atadı.		
8. char veri tipinde bir değişken tanımladı.		
9. char değişkenine tür dönüşümü yaparak değer atadı.		
10. System.out.println komutu ile ASCII kodunu yazdırdı.		
11. System.out.println komutu ile karakteri yazdırdı.		
12. Run …. With Coverage ile uygulamayı çalıştırdı.		
13. Cover penceresinde uygulama çıktısını gözlemledi.		

3.3.1.3. Ondalık Veri Tipleri

Değer olarak içinde ondalık sayıları barındıran veri tipleridir (Tablo 3.2).

- **float:** Değer atanırken sayının sonuna **f** veya **F** yazılır.
- **double:** Değer atanırken sayının sonuna **d** veya **D** yazılır.

Tablo 3.2: Ondalık Veri Tipleri

Ondalık Sayı Türü	Kapladığı Alan	Değer Aralığı
float	4 byte	3.4e-038, … , 3.4e+038
double	8 byte	1.7e-308, … , 1.7e+308

5. UYGULAMA: İşlem adımlarına göre ondalık sayı veri tiplerinin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "DenemeOndalikliSayilar" adında yeni bir sınıf oluşturunuz.

2. Adım: DenemeOndalikliSayilar sınıfı içine şu kodu yazınız:

```
public class DenemeOndalikliSayilar {
    public static void main(String[] args) {
        float ondalik1 = 1f/3f;
        double ondalik2 = 1d/3d;
        System.out.println("float: (1/3) = " + ondalik1);
        System.out.println("double: (1/3) = " + ondalik2);
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'DenemeOndalikliSayilar.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

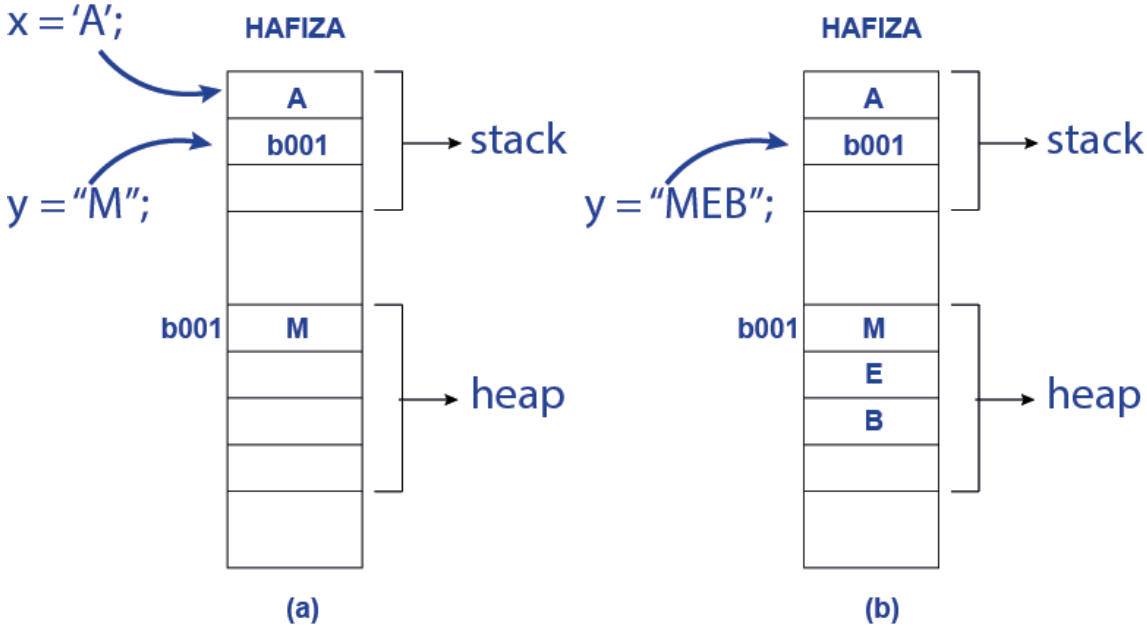
```
float: (1/3) = 0.33333334
double: (1/3) = 0.3333333333333333
```

3.3.2. Referans Veri Tipleri

İlkel ve referans değişkenler, akıllı cihazın stack hafıza adı verilen bölümünde bulunur. Görsel 3.19'un a ile belirtilen kısmında görüldüğü gibi ilkel veri tipleri stack hafıza içinde değerlerini barındırır.

İlkel veri tipleri, içinde birden fazla veri barındıramaz. İlkel veri tipi olarak tanımlanmış bir değişkenin boyutu hafızada sabittir, daha sonra değiştirilemez. Uygulama, içinde birden fazla veri tutacak yapılara ihtiyaç duyar. Bu verilerin boyutu sabit değildir. Uygulama çalışırken boyutu artırılabilir şekilde olmalıdır. Referans veri tipleri bu ihtiyaca cevap verir.

Referans veri tipleri de stack hafızada yer alır. İlkel veri tiplerinden farklı olarak içinde değerleri yoktur. Görsel 3.19'un b ile belirtilen kısmında görüldüğü gibi değer yerine içinde bir hafıza adresi barındırır.



Görsel 3.19: Stack ve heap hafıza

Görsel 3.19'daki x değişkeni bir ilkel değişkendir ve stack hafıza içinde yer alır. Değişkenin değerini de kendi içinde barındırır. Boyutu sabittir ve değiştirilemez. Referans veri tipi olan y değişkeni içerik olarak bir adres barındırır. Bu adresin gösterdiği heap hafıza bölümünde uygun alan y değişkeni için ayrılır. Görsel 3.19a'da y referans değişkenin değeri "M" harfidir. Buna uygun olarak heap hafızada bir harflik alan ayrılır. Görsel 3.19b'de y referans değişkenin değeri "MEB" olarak değiştirilmiştir. Buna uygun olarak heap bölümünde ayrılan alan da artırılmıştır.

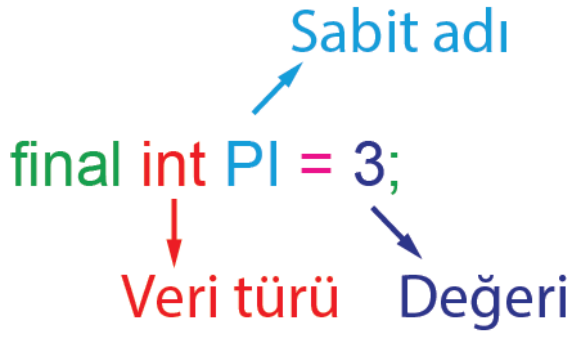
En çok kullanılan referans tipi, karakter dizisidir (String). String, içinde birden fazla karakter barındırabilen referans veri tipidir.

ÖRNEK

String isim = "Mustafa";

3.4. SABİTLER

Sabit, değeri atandıktan sonra değeri değiştirilemeyen bir değişken türüdür. Sabit tanımlandığında atanan değer programın herhangi bir yerinde değiştirilemez, değiştirilmeye çalışıldığında program hata verir. Bir değişken türü olması nedeniyle aynı değişken gibi tanımlanır fakat başına **final** ayrılmış kelimesi (keyword) getirilir (Görsel 3.20).



Görsel 3.20: Sabit tanımlama söz dizimi

NOT:

Sabitlere isim verilirken büyük harfleri kullanmak, Java programcıları arasında bir gelenek hâline gelmiştir. Bu şekilde kullanım zorunlu değildir fakat kodu okuyan herkesin bu değişkenin bir sabit olduğunu anlamasını kolaylaştırır.

6. UYGULAMA: İşlem adımlarına göre sabitin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "DenemeSabitler" adında yeni bir Java sınıfı tanımlayınız.

2. Adım: DenemeSabitler sınıfı içine şu kodu yazınız:

```
public class DenemeSabitler {
    public static void main(String[] args) {
        final int PI = 3;
        int yariCap = 5;
        System.out.println("Çevre = " + (2*PI*yariCap));
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'DenemeSabitler.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
Çevre = 30
```

SIRA SİZDE: Altıncı uygulamadaki PI sabitini 3.14 olacak şekilde gerekli değişiklikleri yapıp çalıştırınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Project penceresinde New Java Class komutunu seçti.		
5. Java sınıfı için hedef klasör seçti.		
6. Java sınıfına "DenemeSabitler" adını belirledi.		
7. PI sabitine uygun değişken türünü belirledi.		
8. PI sabitine 3.14 sayısını atadı.		
9. Run with Coverage komutuyla uygulamayı çalıştırdı.		
10. Cover penceresinde uygulama çıktısını gözlemledi.		

3.5. İSİMLENDİRME KURALLARI

Her programlama dilinde uyulması gereken isimlendirme kuralları ve standartları vardır. Kurallara mutlaka uyulması gerekirken standartlara uyulma zorunluluğu yoktur. İsimlendirme kurallarına uyulmadığında derleyici hatası meydana gelir ve uygulama çalışmaz. İsimlendirme standartlarına uyulmadığında herhangi bir hata ile karşılaşılmaz.

İsimlendirme kuralları şunlardır:

- İsimlerde boşluk kullanılmaz.
- İsimlerde $ ve _ karakterleri dışında özel karakterler kullanılmaz.
- İsimler sayı ile başlamaz.
- İsimler büyük ve küçük harf duyarlıdır. Bir uygulama içinde yer alan "ad" değişkeni ile "Ad" değişkeni farklıdır.
- Ayrılmış anahtar kelimeler (reserved keywords) isim olarak kullanılmaz. İsimlendirmede if, else, int, char gibi programa ait ifadeler ayrılmış anahtar kelimelerdir.

Birçok isimlendirme standardı vardır. Bunlardan üç tanesi Java programlama dilinde sıklıkla kullanılır.

1. **Camel Case (Deve Gösterimi):** Devenin hörgücüne benzetilmesi dolayısıyla böyle bir isim verilmiştir. İlk harf küçük olacak şekilde diğer sözcüklerin ilk harfinin büyük yazılması ile elde edilir.

ÖRNEK

kullanıcıAdıSoyadı, notOrtalaması, büyükKenar

2. **Pascal Case (Paskal Gösterimi):** Camel Case'e benzer. Tüm sözcüklerin ilk harfinin büyük yazılması ile elde edilir.

ÖRNEK

KullaniciAdiSoyadi, NotOrtalamasi, BuyukKenar

3. **Screaming Snake Case (Çığlık Atan Yılan Gösterimi):** Bütün harflerin büyük yazıldığı, sözcükler arası boşluk yerine _ kullanıldığı gösterim şeklidir.

ÖRNEK

PI_SAYISI, MAAS_KATSAYISI, SAAT_UCRETI

İsimlendirme standartları kodun okunabilirliğini artırmak için getirilmiştir. Java kodları yazılırken kullanılan isimlendirme standartları şunlardır:

- Türkçe karakter (ç, ğ, ı, ö, ş, ü, Ç, Ğ, İ, Ö, Ş, Ü) kullanmaktan kaçınılmalıdır.
- Proje isimleri Pascal Case şeklinde olmalıdır.
- Sınıf isimleri Pascal Case şeklinde olmalıdır.
- Değişken isimleri Camel Case şeklinde olmalıdır.
- Sabitler Screaming Snake Case şeklinde olmalıdır.
- Paket isimleri Pascal Case şeklinde olmalıdır.
- Metot isimleri Camel Case şeklinde olmalıdır.

SIRA SİZDE: Tabloda verilen isimlerin karşısına isimlendirme kuralları ve standartlarına göre isimleri tekrar yazınız.

İsim	Türü	Doğru GösterimI
Öğrenci Adı	Değişken	
Araba Sınıfı	Sınıf	
1. not	Değişken	
Tahmin Oyunu	Proje Adı	
Geçme Notu	Sabit	
Kullanıcı İşlemleri	Paket Adı	
Puan Hesapla	Metot Adı	

DEĞERLENDİRME: Çalışmalarınız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. "Öğrenci Adı" değişkenine isimlendirme kurallarına ve standartlarına uygun isim verdi.		
2. "Araba Sınıfı" sınıfına isimlendirme kurallarına ve standartlarına uygun isim verdi.		
3. "1. not" değişkenine isimlendirme kurallarına ve standartlarına uygun isim verdi.		
4. "Tahmin Oyunu" projesine isimlendirme kurallarına ve standartlarına uygun isim verdi.		
5. "Geçme Notu" sabitine isimlendirme kurallarına ve standartlarına uygun isim verdi.		
6. "Kullanıcı İşlemleri" paket adına isimlendirme kurallarına ve standartlarına uygun isim verdi.		
7. "Puan Hesapla" metoduna isimlendirme kurallarına ve standartlarına uygun isim verdi.		

3.6. OPERATÖRLER

Java dilinde kullanılan operatörlerin bazıları şunlardır:

- Aritmetik operatörler
- Atama operatörleri
- Karşılaştırma operatörleri
- Mantıksal operatörler

3.6.1. Aritmetik Operatörler

Aritmetik operatörler, matematiksel işlemleri gerçekleştirmek için kullanılır (Tablo 3.3).

Tablo 3.3: Aritmetik Operatörler

Operatör	İsim	Açıklama	Örnek
+	Toplama	İki değeri toplar.	x + y
-	Çıkarma	Bir değeri diğerinden çıkarır.	x - y
*	Çarpma	İki değeri çarpar.	x * y
/	Bölme	Bir değeri diğer değer böler.	x / y
%	Mod alma	Soldaki değeri sağdaki değere bölerek kalanı bulur.	x % y
++	Artırma	İşlenen değeri 1 artırır.	x++
- -	Eksiltme	İşlenen değeri 1 azaltır.	x--

7. UYGULAMA: İşlem adımlarına göre aritmetiksel operatörlerin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "AritmetikselOperatorler" adında yeni bir Java sınıfı tanımlayınız.

2. Adım: AritmetikselOperatorler sınıfı içine şu kodu yazınız:

```
public class AritmetikselOperatorler {
    public static void main(String[] args) {
        int x = 10;
        int y = 3;
        int toplam = x + y;
        int fark = x - y;
        int carpim = x * y;
        int bolme = x / y;
        int mod = x % y;
        x++;
        y--;
        System.out.println("Toplam: " + toplam);
        System.out.println("Fark: " + fark);
        System.out.println("Çarpım: " + carpim);
        System.out.println("Bölme: " + bolme);
        System.out.println("Mod Alma: " + mod);
        System.out.println("Artırma: " + x);
        System.out.println("Azaltma: " + y);
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'AritmetikselOperatorler.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
Toplam: 13
Fark: 7
Çarpım: 30
Bölme: 3
Mod Alma: 1
Artırma: 11
Azaltma: 2
```

SIRA SİZDE: Yedinci uygulamadaki **x değeri 13, y değeri 5** olacak şekilde değiştirildiğinde oluşacak sonuçları kâğıt üzerinde hesaplayıp aşağıdaki tabloya yazınız. Gerekli kodlamayı yaptıktan sonra çalıştırıp sonuçları karşılaştırınız.

İşlem	Tahmininiz	Sonuç
Toplam		
Fark		
Çarpım		
Bölme		
Mod Alma		
Artırma		
Azaltma		

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Toplam tahminini ve uygulama çıktısını doğru olarak buldu.		
2. Fark tahminini ve uygulama çıktısını doğru olarak buldu.		
3. Çarpım tahminini ve uygulama çıktısını doğru olarak buldu.		
4. Bölme tahminini ve uygulama çıktısını doğru olarak buldu.		
5. Mod alma tahminini ve uygulama çıktısını doğru olarak buldu.		
6. Artırma tahminini ve uygulama çıktısını doğru olarak buldu.		
7. Azaltma tahminini ve uygulama çıktısını doğru olarak buldu.		

3.6.2. Atama Operatörleri

Değişkenlere değer atama için atama operatörleri kullanılır (Tablo 3.4.).

Tablo 3.4: Atama Operatörleri

Operatör	Örnek	Benzeri	Açıklama
=	x = 5	x = 5	Basit atama operatörüdür. Eşitliğin sağındaki değeri soldaki değişkenin içine aktarır.
+=	x += 5	x = x + 5	Topla ve ata operatörüdür. Değişkeni eşitliğin sağındaki değer kadar artırır.
-=	x -= 5	x = x -5	Çıkart ve ata operatörüdür. Değişkeni eşitliğin sağındaki değer kadar azaltır.
*=	x *= 5	x = x * 5	Çarp ve ata operatörüdür. Değişkeni eşitliğin sağındaki değer ile çarpıp atama işlemini gerçekleştirir.
/=	x /= 5	x = x / 5	Böl ve ata operatörüdür. Değişkeni eşitliğin sağındaki değere bölüp atama işlemini gerçekleştirir.
%=	x %= 5	x = x % 5	Mod al ve ata operatörüdür. Eşitliğin sağındaki değere göre değişkenin modunu alıp atama işlemini gerçekleştirir.

8. UYGULAMA: İşlem adımlarına göre atama operatörlerinin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "AtamaOperatorleri" adında yeni bir Java sınıfı tanımlayınız.

2. Adım: AtamaOperatorleri sınıfı içine şu kodu yazınız:

```
public class AtamaOperatorleri {
    public static void main(String[] args) {
        int x = 15;
        System.out.println("x = " + x);
        x += 3;
        System.out.println("x += 3 : " + x);
        x -= 2;
        System.out.println("x -= 2 : " + x);
        x *= 2;
        System.out.println("x *= 2 : " + x);
        x /= 4;
        System.out.println("x /= 4 : " + x);
        x %= 2;
        System.out.println("x %= 2 : " + x);
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'AtamaOperatorleri.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
x = 15
x += 3 : 18
x -= 2 : 16
x *= 2 : 32
x /= 4 : 8
x %= 2 : 0
```

SIRA SİZDE:

Sekizinci uygulamadaki **x değeri 13** olacak şekilde değiştirildiğinde oluşacak sonuçları kâğıt üzerinde hesaplayıp aşağıdaki tabloya yazınız. Gerekli kodlamayı yaptıktan sonra çalıştırıp sonuçları karşılaştırınız.

İşlem	Tahmininiz	Sonuç
Atama		
Topla ve ata		
Çıkart ve ata		
Çarp ve ata		
Böl ve ata		
Mod al ve ata		

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. "Atama" tahminini ve uygulama çıktısını doğru olarak buldu.		
2. "Topla ve ata" tahminini ve uygulama çıktısını doğru olarak buldu.		
3. "Çıkart ve ata" tahminini ve uygulama çıktısını doğru olarak buldu.		
4. "Çarp ve ata" tahminini ve uygulama çıktısını doğru olarak buldu.		
5. "Böl ve ata" tahminini ve uygulama çıktısını doğru olarak buldu.		
6. "Mod al ve ata" tahminini ve uygulama çıktısını doğru olarak buldu.		

3.6.3. Karşılaştırma Operatörleri

Karşılaştırma operatörleri iki değeri karşılaştırmak için kullanılır (Tablo 3.5). Sonuç olarak boolean bir değer meydana gelir. Karşılaştırma işlemi doğru ise true, değil ise false değeri oluşur.

Tablo 3.5: Karşılaştırma Operatörleri

Operatör	Adı	Örnek
==	Eşit mi?	x == y
!=	Farklı mı? (Eşit değil mi?)	x != y
>	Büyük mü?	x > y
<	Küçük mü?	x < y
>=	Büyük veya eşit mi?	x >= y
<=	Küçük veya eşit mi?	x <= y

9. UYGULAMA: İşlem adımlarına göre karşılaştırma operatörlerinin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "KarsilastirmaOperatorleri" adında yeni bir Java sınıfı tanımlayınız.

2. Adım: KarsilastirmaOperatorleri sınıfı içine şu kodu yazınız:

```
public class KarsilastirmaOperatorleri {
    public static void main(String[] args) {
        int x = 15;
        int y = 8;
        System.out.println("x ile y eşit mi : " + (x == y));
        System.out.println("x ile y farklı mı : " + (x != y));
        System.out.println("x, y'den büyük mü : " + (x > y));
        System.out.println("x, y'den küçük mü : " + (x < y));
        System.out.println("x, y'den büyük veya eşit mi : " + (x >= y));
        System.out.println("x, y'den küçük veya eşit mi : " + (x <= y));
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'KarsilastirmaOperatorleri.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
x ile y eşit mi : false
x ile y farklı mı : true
x, y'den büyük mü : true
x, y'den küçük mü : false
x, y'den büyük veya eşit mi : true
x, y'den küçük veya eşit mi : false
```

SIRA SİZDE:

Dokuzuncu uygulamadaki **x değeri 8** olacak şekilde değiştirildiğinde oluşacak sonuçları kâğıt üzerinde hesaplayıp aşağıdaki tabloya yazınız. Gerekli kodlamayı yaptıktan sonra çalıştırıp sonuçları karşılaştırınız.

İşlem	Tahmininiz	Sonuç
==		
!=		
>		
<		
>=		
<=		

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. "= =" tahminini ve uygulama çıktısını doğru olarak buldu.		
2. "!=" tahminini ve uygulama çıktısını doğru olarak buldu.		
3. ">" tahminini ve uygulama çıktısını doğru olarak buldu.		
4. "<" tahminini ve uygulama çıktısını doğru olarak buldu.		
5. ">=" tahminini ve uygulama çıktısını doğru olarak buldu.		
6. "<=" tahminini ve uygulama çıktısını doğru olarak buldu.		

3.6.4. Mantıksal Operatörler

Mantıksal operatörler iki veya daha fazla karşılaştırma işlemini değerlendirmek için kullanılır (Tablo 3.6). Sonuç olarak boolean bir değer meydana gelir.

Tablo 3.6: Mantıksal Operatörler

Operatör	Adı	Örnek	Açıklama
&&	Ve	(x == 5 && y ==6)	x, 5'e eşit ve y, 6'ya eşit midir? True değeri alması için her iki koşulun da doğru olması gerekir.
||	Veya	(x == 5 || y ==6)	x, 5'e eşit veya y, 6'ya eşit midir? True değeri alması için her iki koşuldan sadece birinin doğru olması yeterlidir.
!	Mantıksal değil	!(x == 5)	x, 5'e eşit midir ifadesinin mantıksal tersini alır. Parantez içindeki ifade **true** ise mantıksal tersini alarak **false** sonucunu üretir. Parantez içindeki ifade **false** ise mantıksal tersini alarak **true** sonucunu üretir.

10. UYGULAMA: İşlem adımlarına göre mantıksal operatörlerin kullanıldığı bir uygulamayı tasarlayınız.

1. Adım: "MantiksalOperatorler" adında yeni bir Java sınıfı tanımlayınız.

2. Adım: MantiksalOperatorler sınıfı içine şu kodu yazınız:

```
public class MantiksalOperatorler {
    public static void main(String[] args) {
        int x = 5;
        int y= 8;
        System.out.println("x 10'dan büyük ve y 10'dan küçük mü : " + (x > 20 && y < 20));
        System.out.println("x 10'dan büyük ve y 10'dan küçük mü tersi : " + !(x > 20 && y < 20));
        System.out.println("x 10'dan büyük veya y 10'dan küçük mü : " + (x > 20 || y <20));
        System.out.println("x 10'dan büyük veya y 10'dan küçük mü tersi: " + !(x > 20 || y <20));
    }
}
```

3. Adım: Yazdığınız kodu Run menüsünden Run 'MantiksalOperatorler.main()' with Coverage komutuyla çalıştırınız.

Ekran Çıktısı

```
x 10'dan büyük ve y 10'dan küçük mü : false
x 10'dan büyük ve y 10'dan küçük mü tersi : true
x 10'dan büyük veya y 10'dan küçük mü : true
x 10'dan büyük veya y 10'dan küçük mü tersi: false
```

SIRA SİZDE:

Onuncu uygulamadaki **x değeri 15** olacak şekilde değiştirildiğinde oluşacak sonuçları kâğıt üzerinde hesaplayıp aşağıdaki tabloya yazınız. Gerekli kodlamayı yaptıktan sonra çalıştırıp sonuçları karşılaştırınız.

İşlem	Tahmininiz	Sonuç
(x > 10 && y < 10)		
!(x > 10 && y < 10)		
(x > 10 || y < 10)		
!(x > 10 || y < 10)		

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. (x > 10 && y < 10) işleminin tahminini ve uygulama çıktısını doğru olarak buldu.		
2. !(x > 10 && y < 10) işleminin tahminini ve uygulama çıktısını doğru olarak buldu.		
3. (x > 10 || y < 10) işleminin tahminini ve uygulama çıktısını doğru olarak buldu.		
4. !(x > 10 || y < 10) işleminin tahminini ve uygulama çıktısını doğru olarak buldu.		

3.7. HATA AYIKLAMA

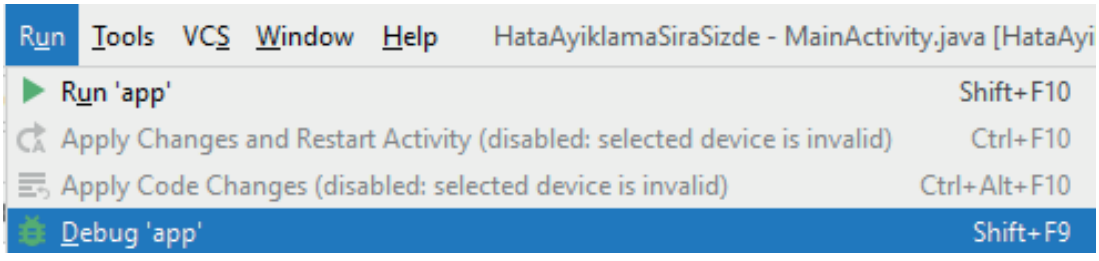
Mobil uygulama geliştirirken çeşitli hatalardan dolayı uygulama istenen şekilde çalışmayabilir. Bu hatalar genellikle iki şekilde meydana gelir.

1. **Yazım Yanlışı Hatası (Syntax Error):** Bu hatalar mobil uygulama geliştirme programı tarafından otomatik olarak tespit edilir. Mobil uygulama geliştirme yazılımı uygulamanın çalıştırılmasına izin vermez. Öncelikle bu yazım yanlışı hatalarının giderilmesi gerekir.

2. **Çalışma Zamanı Hatası (Run-Time Error):** Uygulamanın çalışması sırasında meydana gelen hatalardır. Bu tür hataların ne zaman meydana geleceğini önceden tespit etmek zordur. Çalışma zamanında hata vermeyen bir uygulama bazen bir sonraki çalıştırılmasında hata verebilir. Örneğin internet üzerinden anlık mesajlaşma uygulaması, internet bağlantısı kontrol kodlarını barındırmadan bir mesaj gönderildiğinde çalışma zamanı hatası verir.

Çalışma zamanı hatalarını gidermek için **hata ayıklama (Debug)** işlemi yapılır. Hata ayıklama işlemi için uygulama, hata ayıklama modunda çalıştırılmalıdır. Hata ayıklama modu uygulamada gerçekleşen olayları günlük kayıtlarından (log) izleyebilmeyi, kod satılarına durak noktası (break-point) koyup kodu adım adım çalıştırabilmeyi, uygulama içindeki değişken ve nesnelerin durumlarını izleyebilmeyi sağlar. Çalışma zamanında uygulamanın davranışlarını izlemek için uygulama, hata ayıklama modunda çalıştırılmalıdır. Görsel 3.21 ve Görsel 3.22'de hata ayıklama modunda uygulamayı başlatma verilmiştir.

Görsel 3.21: Hata ayıklama (Debug) düğmesi



Görsel 3.22: Hata ayıklama menü komutu

3.7.1. Logcat

Logcat, çalışma zamanında uygulama ile ilgili çıktıları izlemeye yarayan geliştirici aracıdır (Görsel 3.23). Logcat penceresini açmak için menüden View>Tool Windows>Logcat komutu seçilir.

Görsel 3.23: Logcat ekranı

Görsel 3.23'te 1 numara ile gösterilen kısım, uygulamanın çalıştığı cihazı gösterir. 2 numaralı kısım, çalıştırılan uygulamayı gösterir. 3 numaralı kısım, log sınıfını gösterir (Tablo 3.7).

Tablo 3.7: Log Sınıfları

Log Sınıfı	Anlamı	Kullanımı
Verbose	Ayrıntılı	Log.v("etiket","mesaj");
Debug	Hata Ayıklama	Log.d("etiket","mesaj");
Info	Bilgi	Log.i("etiket","mesaj");
Warn	Uyarı	Log.w("etiket","mesaj");
Error	Hata	Log.e("etiket","mesaj");

Görsel 3.23'te 4 numaralı kısım ise günlük kaydında filtreleme yapmak için kullanılır. Genellikle etiket ile filtreleme yapılır.

Log komutu iki kısımdan oluşur. Birinci bölüm, TAG (etiket) olarak adlandırılır. İkinci kısım ise mesajdan oluşur. Logcat ekranında birçok mesaj bulunur. Bu mesajlara farklı etiketler verilerek birbirinden ayrılması sağlanır.

11. UYGULAMA: İşlem adımlarına göre EditTexte girilen sayıya 2 ekleyen ve Logcat ekranında yapılan işlemleri gösteren bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: activity_main.xml içine şu kodu yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="İşlem Yap"
        android:onClick="islemYap"
        />
</LinearLayout>
```

3. Adım: MainActivity.java dosyasına şu kodu yazınız:

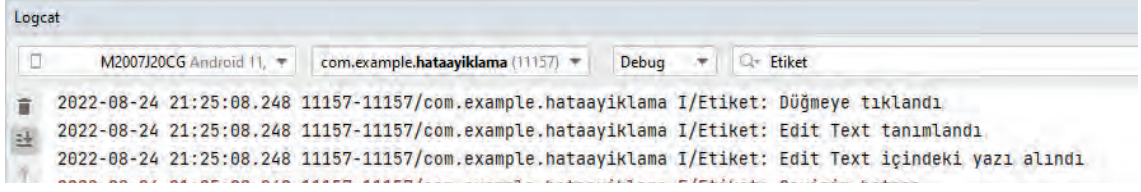
```
public class MainActivity extends AppCompatActivity {
    private  final String TAG = "Etiket";
    private int sayi = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG,"debug (hata ayıklama)");
    }
    public void islemYap(View view) {
        Log.i(TAG,"Düğmeye tıklandı");
        EditText editText = (EditText) findViewById(R.id.editText);
```

```
        Log.i(TAG,"Edit Text tanımlandı");
        String s1 =  editText.getText().toString();
        Log.i(TAG,"Edit Text içindeki yazı alındı");
        sayi = Integer.parseInt(s1);
        Log.i(TAG,"Yazı sayıya çevrildi");
        sayi = sayi + 2;
        Log.i(TAG,"sayıya 2 eklendi");
    }
}
```

4. Adım: Debug'a tıklayarak uygulamayı çalıştırınız.

5. Adım: Metin kutusuna Merhaba yazıp İşlem Yap düğmesine tıklayınız.

6. Adım: Görsel 3.24'teki Logcat ekranında düğmeye tıklanma, EditText tanımlama ve EditText içindeki yazı alınma çalıştığı hâlde yazıyı sayıya çevirme ve sayıya 2 eklemenin çalışmadığını, hatanın yazıyı sayıya çevirme işleminde olduğunu gözlemleyiniz.

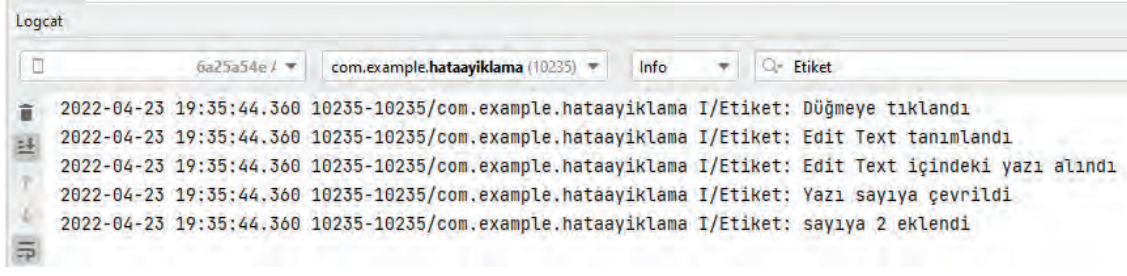


Görsel 3.24: Logcat ekranı

7. Adım: Debug modunda tekrar uygulamayı çalıştırınız.

8. Adım: Metin kutusuna herhangi bir sayı yazıp İşlem Yap düğmesine tıklayınız.

9. Adım: Logcat ekranını gözlemleyiniz (Görsel 3.25).



Görsel 3.25: Logcat ekranı

SIRA SİZDE: Birbirinden farklı etiketler ile Log sınıflarını kullanan bir uygulama tasarlayınız ve Logcat ekranında filtreleme uygulayarak bu Logları bulunuz.

DEĞERLENDİRME: Çalışmalarınız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

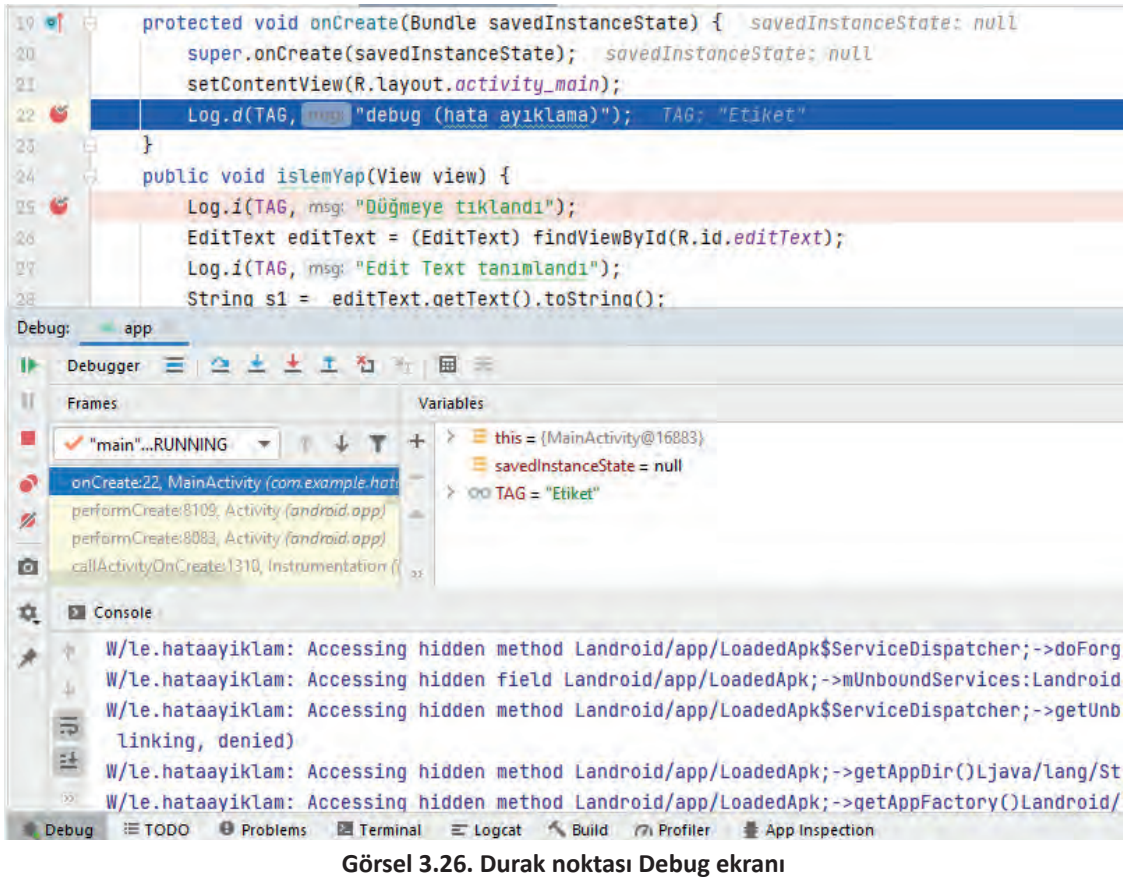
KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Log.v komutunu kullandı.		
2. Log.d komutunu kullandı.		
3. Log.i komutunu kullandı.		
4. Log.w komutunu kullandı.		
5. Log.e komutunu kullandı.		
6. Logcat ekranında filtreleme yaptı.		

3.7.2. Durak Noktası

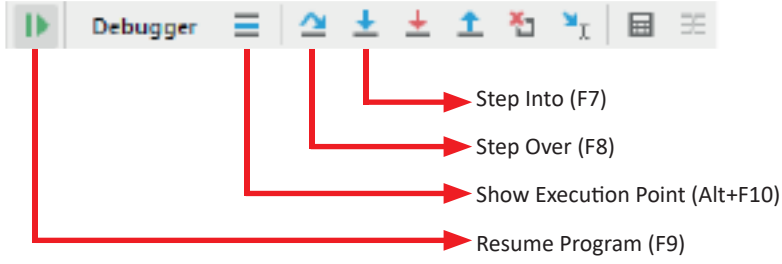
Hata ayıklama yöntemlerinden biri de durak noktası (break-point) eklemektir. Durak noktası eklemek için ilgili kodun satır numarasının hemen yanına fare ile tıklamak yeterlidir. Kod satır numarasının yanında kırmızı bir daire belirir. Durak noktasına tekrar tıklandığında durak noktası silinir.

Hata ayıklama modunda iken çalıştırılacak kod sırası durak noktası konulan yere geldiğinde uygulama çalışmaya ara verir. Uygulamanın o anki durumu hakkında Debug penceresinden bilgi verilir (Görsel 3.26). Variables bölümünde değişkenlerin durumu yer alır.



Görsel 3.26. Durak noktası Debug ekranı

Uygulamanın çalışmaya devam etmesi için Debug penceresinde Resume Program komutu veya klavyeden F9 tuşuna basılır. Programı adım adım çalıştırmak için Step Into ve Step Over düğmeleri kullanılır (Görsel 3.27).



Görsel 3.27: Adım adım çalıştırma düğmeleri

3.7.3. Değişken İzleme (Watch)

Uygulama kodu durak noktasına geldiğinde Debug penceresinde Variables bölümünden değişkenlerin durumu izlenebilir. Buraya özellikle durumu izlenecek değişkenler eklenir. Kod ekranında izlenecek değişken üzerine sağ tıklanıp, Add to Watches komutu verilerek gözcü eklenir.

3.8. HATA DÜZELTME

Çalışma zamanı hatalarını düzeltmek için birçok yöntem vardır. Bunlardan en çok kullanılanı Try-Catch-Finally yapısıdır. Hata verebilecek kodlar Try (Dene) bloku içine alınır. Böylelikle herhangi bir istisna (Exception) meydana geldiğinde kod çalışmaya devam eder. Bir istisna meydana gelirse Catch (Yakala) bloku içindeki kodlar çalışır. Finally (En sonunda) bloku herhangi bir istisna oluşup oluşmadığına bakılmaksızın Try-Catch-Finally yapısında en son olarak çalıştırılacak kodların bulunduğu bölümdür. Finally blokunun kullanılması zorunlu değildir.

Catch blokunda FileNotFoundException (Dosya bulunamadı istisnası), ArithmeticException (Aritmetik istisnası) vb. özel istisnalar yakalanabileceği gibi genel istisnalar (Exception) da yakalanabilir.

Try-Catch-Finally blokunun kullanımı şu şekildedir:

```
try {

}
catch (İstisnaTipi istisnaAdi){

}
finally {

}
```

12. UYGULAMA: İşlem adımlarına göre EditTexte girilen sayıya 2 ekleyen uygulamayı Try-Catch-Finally ile yeniden tasarlayınız.

1. Adım: On birinci uygulamadaki MainActivity.java kodunu şu şekilde değiştiriniz:

```
public class MainActivity extends AppCompatActivity {
    private  final String TAG = "Etiket";
    private int sayi = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void islemYap(View view) {
        Log.i(TAG,"Düğmeye tıklandı");
        EditText editText = (EditText) findViewById(R.id.editText);
        Log.i(TAG,"Edit Text tanımlandı");
        String s1 =  editText.getText().toString();
        Log.i(TAG, "Edit Text içindeki yazı alındı");
        try {
            sayi = Integer.parseInt(s1);
            Log.i(TAG,"Yazı sayıya çevrildi");
        }
        catch (Exception e){
            Log.e(TAG,"Çevirim hatası");
            sayi = 0;
        }
        finally {
            sayi = sayi + 2;
            Log.i(TAG, "(finally) sayı = " +sayi);
        }
    }
}
```

2. Adım: Debug'a tıklayarak uygulamayı çalıştırınız.

3. Adım: Metin kutusuna Merhaba yazıp İşlem Yap düğmesine tıklayınız.

4. Adım: Logcat ekranını gözlemleyiniz.

```
2022-04-24 06:06:08.561 32743-32743/com.example.hataayiklama I/Etiket: Düğmeye tıklandı
2022-04-24 06:06:08.561 32743-32743/com.example.hataayiklama I/Etiket: Edit Text tanımlandı
2022-04-24 06:06:08.561 32743-32743/com.example.hataayiklama I/Etiket: Edit Text içindeki yazı alındı
2022-04-24 06:06:08.562 32743-32743/com.example.hataayiklama E/Etiket: Çevirim hatası
2022-04-24 06:06:08.562 32743-32743/com.example.hataayiklama I/Etiket: (finally) sayı = 2
```

SIRA SİZDE: İki sayıyı birbirine bölen uygulamayı Try-Catch-Finally yapısını kullanarak tasarlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Uygulamaya iki adet EditText ekledi.		
2. Uygulamaya bir adet TextView ekledi.		
3. Uygulamaya bir adet Button ekledi.		
4. Button tıklama olayını yazdı.		
5. Button tıklama olayına bölme işlemi kodlarını yazdı.		
6. Metni sayıya çevirme kodları için Try-Catch-Finally yapısını kullandı.		
7. Sıfıra bölme hatası için Try-Catch-Finally yapısını kullandı.		
8. Logcat ekranını kullandı.		

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Değişkenler; değişken türü, adı, adresi ve değerinden oluşur.

2. ( ) char değişken türü, referans veri türüdür.

3. ( ) String değişken türü, referans veri türüdür.

4. ( ) boolean, mantıksal veri türüdür.

5. ( ) int ve byte, tam sayı veri türleridir.

6. ( ) float sayi = 3.14d; doğru bir tanımlamadır.

7. ( ) Sabit tanımlamak için değişkenin sonuna final ifadesi yazılır.

B) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

8. Aşağıdakilerden hangisi isimlendirme kurallarına uygun bir değişken tanımlamadır?

A) yazılı-1 B) 1.yazili C) YAZILI D) yazili1 E) yazili 1

9. Aşağıdakilerden hangisi isimlendirme standartlarına uyularak kullanılacak bir değişken ismidir?

A) kullaniciAdi B) KULLANICIADI C) KullaniciAdi
D) kullaniciadi E) kullaniciADI

10. Mod alma işlemi için aşağıdaki operatörlerden hangisi kullanılır?"

A) / B) % C) & D) | E) #

11. Aşağıdaki atama operatörlerinden hangisi önce topla sonra atama işlemini yap anlamına gelir?

A) =+ B) += C) == D) ++ E) -=

12. Aşağıdaki karşılaştırma operatörlerinden hangisi <u>yanlıştır</u>?

A) == B) >= C) >> D) < E) !=

13. Aşağıdaki mantıksal operatörlerden hangisi "ve" anlamına gelir?

A) || B) == C) %% D) && E) !

14. Aşağıdakilerden hangisi log kayıtlarına bir hata mesajı bırakmak için kullanılır?

A) Log.d B) Log.i C) Log.v D) Log.w E) Log.e

4. ÖĞRENME BİRİMİ

KARAR VE DÖNGÜ YAPILARI

KONULAR

4.1. KARAR YAPILARI

4.2. DÖNGÜ YAPILARI

NELER ÖĞRENECEKSİNİZ?

- Karşılaştırma operatörlerinin ve mantıksal operatörlerin kullanım alanları
- if, if-else, else-if, switch-case karar ifadeleri ve kullanım alanları
- İç içe karar yapıları
- Döngü kavramı
- for, while, do-while döngü yapılarının kullanımı
- İç içe döngü yapıları oluşturma

ANAHTAR KELİMELER

- do-while
- Döngü yapıları
- else
- for
- if
- Karar ifadeleri
- switch-case
- while

HAZIRLIK ÇALIŞMALARI

1. Gündelik hayatta karşılaştığınız ve seçenekler arasında tercih yapmak zorunda kaldığınız bir olay karşısında aldığınız kararların nedenlerini arkadaşlarınızla tartışınız. Program yazarken bu kararları nasıl kullanabileceğiniz hakkında örnekler veriniz.
2. Program yazarken döngü kullanmanın avantajları neler olabilir? Arkadaşlarınızla paylaşınız.

4.1. KARAR YAPILARI

Üst seviye yazılım geliştirme aşamalarının vazgeçilmezi olan karar ifadeleri, belirlenen koşul veya koşulların sonuçlarına göre yazılımların gidiş yolunu çizen yapılardır. Bu yapılar sayesinde oluşturulan uygulamalar daha esnektir. Örneğin not girişlerinin yapıldığı bir uygulamada, ortalama hesaplandıktan sonra öğrencinin dersi geçip geçmediğini belirlemek için ortalamanın, dersi geçme sınırından büyük mü yoksa küçük mü olduğu karşılaştırma operatörleri ile kontrol edilmelidir. Buna bağlı olarak öğrencinin dersi geçtiğine veya dersten kaldığına karar verilmelidir. Yazılım geliştirme ortamında Java programlama diliyle uygulama oluşturulurken "if" ve "switch-case" karar yapıları kullanılır.

4.1.1. if Karar Yapısı

Programlama dillerinde en sık kullanılan if karar yapısı kelime olarak **eğer** anlamına gelir. Yapı oluşturulacağı zaman blok içindeki kodlar çalıştırılmak istendiğinde "eğer şart sağlanıyorsa" ifadesi if karar yapısı ile karşılanır. Yazılan öncülün sonucunda evet (true) cevabı verilirse if karar yapısının blokları arasındaki işlem yapılır. Hayır (false) cevabı verilirse if karar yapısının blokları arasına uğramadan işlemlere devam edilir.

```
if(şart ifadesi){
        //Şart sonucu true ise yapılacaklar
}
```

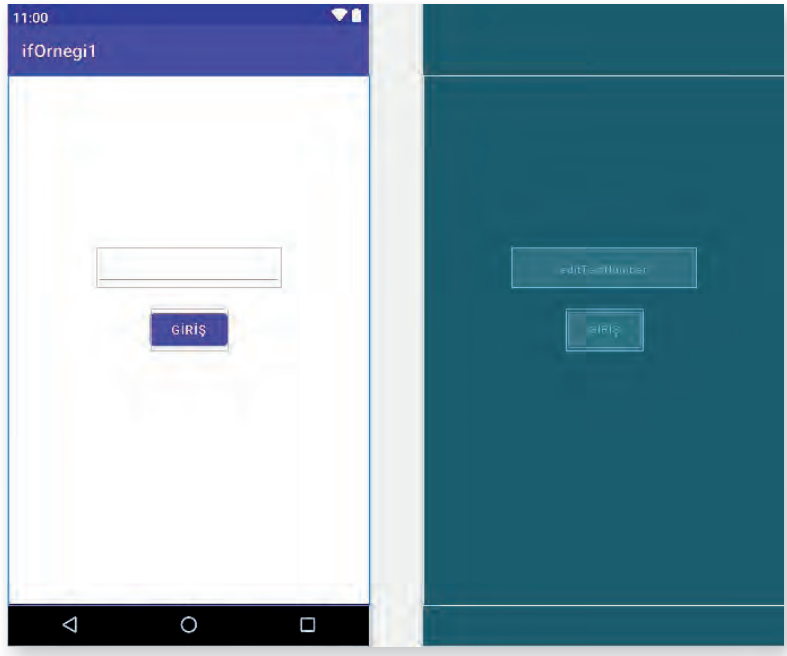
1. UYGULAMA: İşlem adımlarına göre TextView ögesine not girişi yapıp butona tıklandığında notun 100'den büyük olması durumunda "Yanlış Giriş" Toast mesajı veren bir uygulama tasarlayınız.

1. Adım: Yazılım geliştirme ortamında Empty Activity açınız.

2. Adım: İçeriye bir adet sadece sayı girilen "EditText" bir adet de "Button" ögesi ekleyiniz. Daha sonra "Infer Constraints" butonuna tıklayarak ögelerin yerini sabitleyiniz (Görsel 4.1).

3. Adım: Button ögesinin "onClick" özelliğine "kontrolEt" yazınız (Butona tıklandığında çalışacak "kontrolEt" isimli metot sonradan yazılacaktır.).

4. Adım: MainActivity'yi açınız ve eklenen ögeleri "findViewById" ile onCreate metoduna ait blokların içinde tanımlayınız.



Görsel 4.1: Not kontrol uygulaması tasarımı

```
public class MainActivity extends AppCompatActivity {
    EditText number;
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button=findViewById(R.id.button);
        number=findViewById(R.id.editTextNumber);
    }
}
```

5. Adım: MainActivity sınıfının içinde, onCreate metodunun dışında olacak şekilde, geri dönüş değeri olmayan "kontrolEt" isminde bir metot tasarlayınız. Parametre olarak da View sınıfından view isimli bir nesne türetiniz. Bu metodun ismi, bir görünüme tıklandığında çalışacağı için butonun onClick özelliği içinde yazılmalıdır.

6. Adım: Oluşturulan metot içinde integer türünden "sayi" isimli bir değişken tasarlayıp içine EditTextten gelen değeri alınız. Değeri alırken EditTextten gelen "String" tipini öncelikle "int" veri tipine çeviriniz. Karar yapısını kullanarak sayi isimli değişkenin içindeki değerin 100'den büyük olması durumunda şu Toast mesajını gönderiniz:

```
public void kontrolEt(View view){
    int sayi = Integer.parseInt(number.getText().toString());
    if(sayi>100){
        Toast.makeText(this, "100'den Büyük Not Olamaz", Toast.LENGTH_SHORT).show();
    }
}
```

7. Adım: Uygulamanın son şekli Görsel 4.2'de verilmiştir. Shift+F10 tuşlarıyla veya Başlat ikonuyla uygulamayı çalıştırınız.

```
package com.atilimciftci.ifOrnegi1;

import ...

public class MainActivity extends AppCompatActivity {

    EditText number;
    Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = findViewById(R.id.button);
        number = findViewById(R.id.editTextNumber);
    }

    public void kontrolEt(View view){
        int sayi = Integer.parseInt(number.getText().toString());
        if(sayi>100){
            Toast.makeText( context: this, text: "100'den Büyük Not Olamaz",Toast.LENGTH_LONG).show();
        }
    }
}
```

Görsel 4.2: Not kontrol uygulaması MainActivity

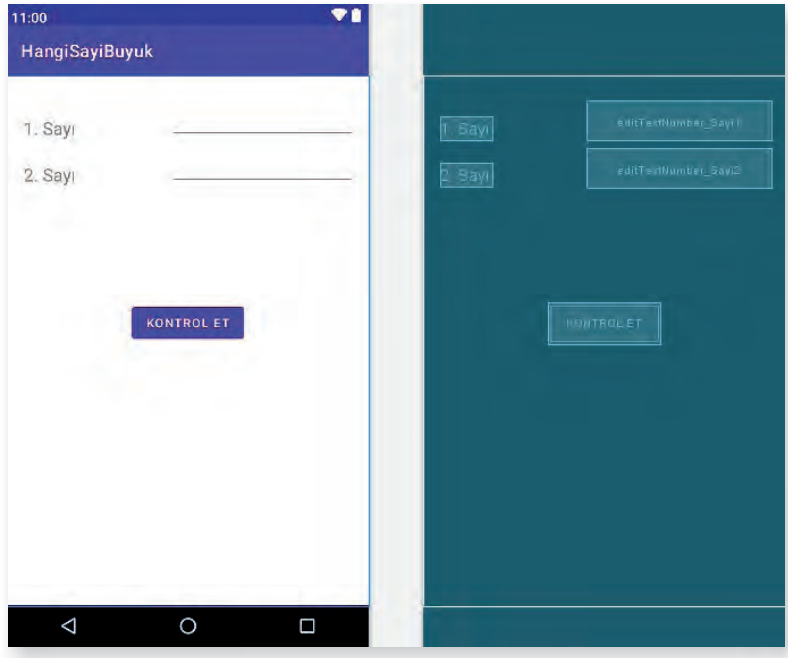
NOT: **kontrolEt** ismi verilip yazılan metot ile buttonun **onClick** olayına yazılan metot aynıdır. Bunların biri oluşturma amaçlı, diğeri ise oluşturulan o metodu kullanmak için çağırma amaçlı yazılmıştır. Bu nedenle metotların isminin de aynı olması gerekir. Bu örnekte girilen not değeri 100'den büyük olursa küçük bir Toast mesajı çıkacaktır.

NOT: Toast mesajları, küçük birer mesaj kutucuğudur. Bu mesajı oluştururken üç parametreye ihtiyaç vardır. İlk parametre değeri **this**, varolan Activity'yi context olarak hedef gösterir. Bu ifade, **MainActivity.this** olarak da yazılabilir. İkinci parametre, mesajın içeriğini oluşturur. Üçüncü parametre ise mesajın ekranda ne kadar kalacağını belirler.

2. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranında birinci ve ikinci sayı girişlerini yaparak bu sayılardan hangisinin büyük olduğunu bulan ve büyük sayıyı ekranda Toast mesajı ile gösteren uygulamayı tasarlayınız.

1. Adım: Görsel 4.3'te görülen tasarımı Empty Activity açarak hazırlayınız.

2. Adım: İçeriye iki adet sadece sayı girilen "EditText" ögesi, bir adet "Button" ögesi, iki adet de 1. Sayı ve 2. Sayı yazan "TextView" ögesi ekleyiniz. Daha sonra "Infer Constraints" butonuna tıklayarak ögelerin yerini sabitleyiniz (Görsel 4.3).



Görsel 4.3: Sayıları karşılaştırma uygulaması tasarımı

3. Adım: Button ögesinin "onClick" özelliğine "kontrol" yazınız.

4. Adım: MainActivity'yi açınız ve eklenen ögeleri "findViewById" metodu ile "onCreate"i şu şekilde tanımlayınız:

```
public class MainActivity extends AppCompatActivity {

    EditText editText_sayi1;
    EditText editText_sayi2;
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = findViewById(R.id.button_Kontrol);
        editText_sayi1 = findViewById(R.id.editTextNumber_Sayi1);
        editText_sayi2 = findViewById(R.id.editTextNumber_Sayi2);
    }
}
```

5. Adım: MainActivity sınıfının içinde, onCreate metodunun dışında olacak şekilde, geri dönüş değeri olmayan (void) bir metot tasarlayınız. Metoda "kontrol" adını veriniz. Parametre olarak da View sınıfından view isimli bir nesne gönderen şu kodları yazınız:

```
public void kontrol(View view){
    int sayi1 = Integer.parseInt(editText_sayi1.getText().toString());
    int sayi2 = Integer.parseInt(editText_sayi2.getText().toString());
    if(sayi1>sayi2){
      Toast.makeText(MainActivity.this,"1. Sayı Daha Büyüktür.",Toast.LENGTH_
LONG).show();
    }
    if(sayi2>sayi1){
      Toast.makeText(MainActivity.this,"2. Sayı Daha Büyüktür.",Toast.LENGTH_
LONG).show();
    }
    if(sayi1==sayi2){
      Toast.makeText(MainActivity.this,"İki Sayı Birbirine Eşittir.",Toast.LEN-
GTH_LONG).show();
    }
}
```

6. Adım: Oluşturulan metot içinde integer türünden "sayi1" ve "sayi2" değişkenleri tasarlayıp değişkenlerin içine EditTextlerden gelen değerleri alınız. Değerleri alırken EditTextten gelen "String" tipini öncelikle "int" veri tipine çeviriniz. Karar yapısını üç defa kullanarak 1. sayının büyük olması durumunda bir Toast mesajı, 2. sayının büyük olması durumunda farklı bir Toast mesajı ve sayıların birbirlerine eşit olması durumunda da ayrı bir Toast mesajı gönderiniz.

7. Adım: Uygulamanın son şekli Görsel 4.4'te verilmiştir. Shift+F10 tuşlarıyla veya Başlat ikonuyla uygulamayı çalıştırınız.

NOT: Bundan sonraki uygulamalarda eklenen nesnelerin varsayılan isimleri (id) değiştirilerek işlem yapılacaktır. Bu uygulamada da bu nedenle EditTextlerin ve buttonun id'leri değiştirilmiş ve uygun isimler verilmiştir. Bu isimler de ögelerin başlatması yapılırken yazılan findViewByld metodunda görülür.

```
package com.atilimciftci.hangisayibuyuk;

import ...

public class MainActivity extends AppCompatActivity {

    EditText editText_sayi1;
    EditText editText_sayi2;
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = findViewById(R.id.button_Kontrol);
        editText_sayi1 = findViewById(R.id.editTextNumber_Sayi1);
        editText_sayi2 = findViewById(R.id.editTextNumber_Sayi2);
    }

    public void kontrol(View view){
        int sayi1 = Integer.parseInt(editText_sayi1.getText().toString());
        int sayi2 = Integer.parseInt(editText_sayi2.getText().toString());
        if(sayi1>sayi2){
            Toast.makeText( context: MainActivity.this, text: "1. Sayı Daha Büyüktür.",Toast.LENGTH_LONG).show();
        }
        if(sayi2>sayi1){
            Toast.makeText( context: MainActivity.this, text: "2. Sayı Daha Büyüktür.",Toast.LENGTH_LONG).show();
        }
        if(sayi1==sayi2){
            Toast.makeText( context: MainActivity.this, text: "İki Sayı Birbirine Eşittir.",Toast.LENGTH_LONG).show();
        }
    }
}
```

Görsel 4.4: MainActivity sayfasında sayıların karşılaştırılması

4.1.2. if-else Karar Yapısı

Oluşturulan şartın "true" olması durumunda if karar yapısının blokları arasındaki işlem gerçekleşir. Şart sonucu "false" ise uygulamanın çalışma prensibi, oluşturulan if blokuna giriş yapılmadan sonrasındaki satırların çalışmaya devam etmesi şeklindedir. Buradaki problem, şart "true" olduğunda blok içine girilip işlemler yapıldıktan sonra blok dışındaki ifadelerin de çalıştırılmasıdır. "else" ifadesi **aksi hâlde** anlamı taşır. "if-else" karar yapısında ise "true" ve "false" durumları ayrı ayrı yazılır. Bu sayede "true" ile "false" işlemleri birbirinden tamamen ayrılıp iki farklı yolda işlenebilir.

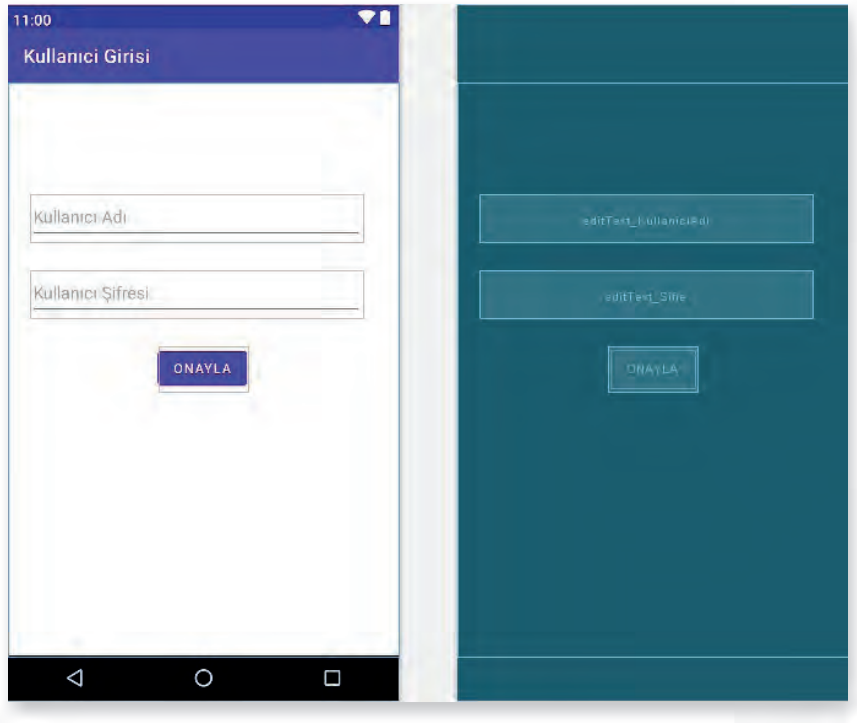
```
if(şart ifadesi){
       //Şart sonucu true ise yapılacaklar
}
else{
    //Şart sonucu false ise yapılacaklar
}
```

3. UYGULAMA: İşlem adımlarına göre bir EditText ile kullanıcıdan "kullanıcı adı" ve "şifre" isteyiniz. Girilen kullanıcı adı ve şifresi, programın içinde belirtilen kullanıcı adı ve şifreye eşitse Toast mesajı ile "Giriş Başarılı", eşit değilse "Giriş Başarısız" mesajı veriniz.

1. Adım: Yeni bir proje açıp Empty Activity ekleyiniz. Görsel 4.5'teki tasarımı oluşturunuz. Tasarıma bir adet "PlainText" ögesi, bir adet "Password" ögesi bir adet de "Button" ögesi ekleyiniz. PlainTextin id'sine "editText_KullaniciAdi", hint özelliğine de "Kullanıcı Adı :" yazınız. Password ögesinin id'sine "editText_Sifre", hint özelliğine de "Şifre :" yazınız. Buttonun id'sini "button_Onayla", text özelliğini ise "Onayla" şeklinde değiştiriniz. Ardından "Infer Constraints" seçeneğine tıklayınız.

NOT: "PlainText" ve "Password" ögeleri aslında birer EditText ögesi olmasına rağmen farklı özellikleri nedeniyle ayrıştırılır.

NOT: Hint özelliği, o ögeye tıklanmadığı sürece görünen fakat tıklandığında kaybolan, ipucu şeklinde yer alan text mesajıdır.



Görsel 4.5: Kullanıcı girişi uygulama tasarımı

2. Adım: Button nesnesinin "onClick" özelliğine "onayla" yazınız.

3. Adım: MainActivity'yi açınız ve eklenen ögeleri "findViewById" metodu ile onCreate blokları içinde şu şekilde tanımlayınız:

```
public class MainActivity extends AppCompatActivity {
    EditText editText_KullaniciAdi, editText_Sifresi;
    Button button_Onayla;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editText_KullaniciAdi = findViewById(R.id.editText_KullaniciAdi);
        editText_Sifresi = findViewById(R.id.editText_Sifre);
        button_Onayla = findViewById(R.id.button_Onayla);
    }
}
```

4. Adım: MainActivity sınıfının içinde, onCreate metodunun dışında olacak şekilde, geri dönüş değeri olmayan bir metot tasarlayınız. Metoda "onayla" adını veriniz. Parametre olarak da View sınıfından "view" isimli bir nesne gönderiniz.

5. Adım: MainActivity sınıfının blokları arasına "kullanıcıAdi" ve "sifre" isminde iki adet sabit tanımlayıp istediğiniz kullanıcı adı ve şifre bilgilerini buraya giriniz.

```
final String kullaniciAdi = "atilimciftci";
final String sifre = "12345";
```

6. Adım: "onayla" isminde geri dönüş değeri olmayan (void) bir metot tanımlayınız. editText_KullaniciAdi ve editText_Sifre ögelerinden aldığınız bilgileri, bu metot blokları içinde String tipte tanımladığınız "kullaniciAdiGiris" ve "sifreGiris" isimli değişkenlerin içine atınız.

```
public void onayla(View view){
    String kullaniciAdiGiris = editText_KullaniciAdi.getText().toString();
    String sifreGiris = editText_Sifresi.getText().toString();
}
```

7. Adım: if-else karar yapısında sabit olarak tanımlanan "kullaniciAdi" ile yine sabit olarak tanımlanan "sifre" bilgilerini "kullaniciAdiGiris" ve "sifreGiris" değişkenleri ile karşılaştırınız. Karşılaştırma işleminde "ve operatörünü (&&)" kullanınız.

```
if(kullaniciAdi.equals(kullaniciAdiGiris) && sifre.equals(sifreGiris)) {
}
else {
}
```

NOT: Diğer programlama dillerinde sıklıkla kullanılan "==" karşılaştırma operatörü ile tam olarak referans eşitliği kontrol edilir. Bir başka deyişle iki taraftaki sorgulanan bilgilerin aynı object türünden olup olmadığı kontrol edilir. Bu nedenle Java programlama dilinde String tipteki bilgiler bu karşılaştırma operatörü ile değil, **equals()** metodu ile karşılaştırılır. Bu metot ile değer eşitliği kontrol edilir. Bir başka deyişle kıyaslanan iki String ifadenin içeriklerinin aynı olup olmadığına bakılır.

8. Adım: Karşılaştırılan ifadelerin "true" olması durumunda "Kullanıcı Girişi Başarılı", "false" olması durumunda ise "Kullanıcı Adı veya Şifresi Hatalı" şeklinde bir Toast mesajı hazırlayınız.

```
if(kullaniciAdi.equals(kullaniciAdiGiris) && sifre.equals(sifreGiris)) {
    Toast.makeText(this,"Kullanıcı Girişi Başarılı",Toast.LENGTH_LONG).show();
}
else {
    Toast.makeText(this,"Kullanıcı Adı veya Şifresi Hatalı",Toast.LENGTH_LONG).
show();
}
```

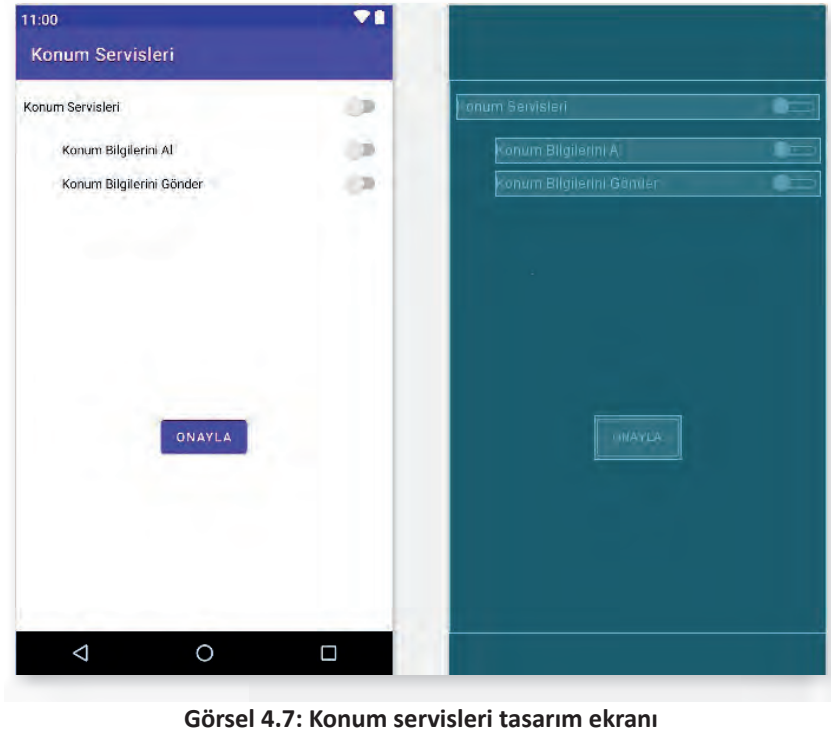
9. Adım: Uygulamanın son şekli Görsel 4.6'da verilmiştir. Buna göre Shift+F10 tuşlarıyla veya Başlat ikonuyla uygulamayı çalıştırınız.

```
package com.atilimciftci.kullancigirisi;

import ...

public class MainActivity extends AppCompatActivity {

    EditText editText_KullaniciAdi, editText_Sifresi;
    Button button_Onayla;

    final String kullaniciAdi = "atilimciftci";
    final String sifre = "12345";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editText_KullaniciAdi = findViewById(R.id.editText_KullaniciAdi);
        editText_Sifresi = findViewById(R.id.editText_Sifre);
        button_Onayla = findViewById(R.id.button_Onayla);
    }

    public void onayla(View view){
        String kullaniciAdiGiris = editText_KullaniciAdi.getText().toString();
        String sifreGiris = editText_Sifresi.getText().toString();

        if(kullaniciAdi.equals(kullaniciAdiGiris) && sifre.equals(sifreGiris)) {
            Toast.makeText( context: this, text: "Kullanıcı Girişi Başarılı",Toast.LENGTH_LONG).show();
        }
        else {
            Toast.makeText( context: this, text: "Kullanıcı Adı veya Şifresi Hatalı",Toast.LENGTH_LONG).show();
        }
    }
}
```

Görsel 4.6: MainActivity sayfasında kullanıcı girişi

4. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranına üç adet "switch" ögesi ekleyiniz. Birinci switch ögesi olan "Konum Servisleri" açık ise "Konum Bilgilerini Al" ve "Konum Bilgilerini Gönder" switchlerini kontrol ederek açık ve kapalı olanları Toast mesajında gösteren uygulamayı tasarlayınız.

1. Adım: Üç adet switch ve bir adet button ile Görsel 4.7'yi oluşturunuz. Birinci switchi; id'si "switch_KonumServisi", layouth_with özelliği "match_parent", Left Margin özelliği ise 8 dp olacak şekilde ayarlayınız. İkinci switchi; id'si "switch_KonumAl", layouth_with özelliği "match_parent", Left Margin özelliği ise 50 dp olacak şekilde ayarlayınız. Üçüncü switchi; id'si "switch_KonumGonder", layouth_with özelliği "match_parent", Left Margin özelliği ise 50 dp olacak şekilde ayarlayınız.



Görsel 4.7: Konum servisleri tasarım ekranı

2. Adım: Oluşturulan ögeleri MainActivity sayfasında tanımlayıp onCreate blokları içinde findViewById metodu ile initialize (başlatma) ediniz.

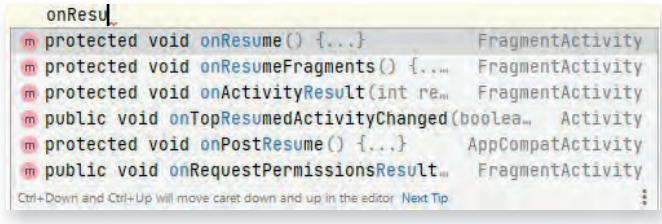
```
Switch konumServisleri,konumGonder,konumAl;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    konumServisleri = findViewById(R.id.switch_KonumServisi);
    konumAl = findViewById(R.id.switch_KonumAl);
    konumGonder = findViewById(R.id.switch_KonumGonder);
}
```

3. Adım: Uygulama ilk açıldığında konum servislerinin durumunu kontrol etmek için bir if-else karar yapısı uygulayınız. Bunun için onCreate metodu içine Görsel 4.8'deki kodu ekleyiniz.

```
// İlk açılacağı zaman kontrol edilecektir
if(konumServisleri.isChecked()){
    konumAl.setVisibility(View.VISIBLE);
    konumGonder.setVisibility(View.VISIBLE);
}
else
{
    konumAl.setVisibility(View.GONE);
    konumGonder.setVisibility(View.GONE);
}
```

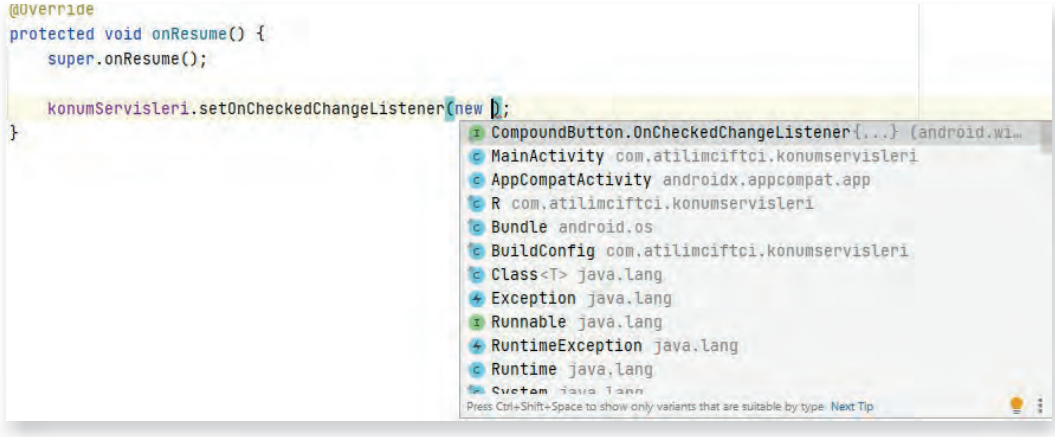
Görsel 4.8: Konum servisleri switchini kontrol kodu

4. Adım: Yaşam döngülerinden olan "onResume" metodunu oluşturunuz. Konum servisleri switchi anlık olarak kapatılıp açıldığında alt switchlerin ona göre hareketler yapması istenir. Bunun için onCreate metodundan çıkılıp MainActivity sınıfı bloklarındayken onResume yazılırsa Görsel 4.9'da olduğu gibi bir görüntü elde edilir. Enter tuşuna basıldığında onResume yaşam döngüsü oluşturulur.



Görsel 4.9: onResume yaşam döngüsü

5. Adım: Oluşturulan onResume içinde "konumServisleri" isimli switchin anlık durumunun değişimi kontrol edilmelidir. Bunun için de switchin **"setOnCheckedChangeListener"** metodu kullanılır. Bunlar tıpkı onCreate, onResume gibi interface'den gelen gövdeli metotlardır. Bu nedenle bunlar da override edilir. Kullanımı için "onResume" metodu blokları içindeyken "konumServisleri.setOnCheckedChangeListener()" yazınız. Parametre girmek için Görsel 4.10'da görüldüğü gibi **new** yazınız. Çıkan "onCheckedChangeListener" seçeneğine Enter tuşu ile basınız ve Görsel 4.11'de görülen yapıyı oluşturunuz.



Görsel 4.10: setOnCheckedChangeListener seçimi

```
@Override
protected void onResume() {
    super.onResume();

    konumServisleri.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {

        }
    });
}
```

Görsel 4.11: OnCheckedChangeListener yapısı

6. Adım: "onCheckedChanged" metodunun blokları arasına girilerek karar ve kontrol yapıları oluşturulur. Buradaki kontroller bire bir şekilde onCreate blokları arasında yazılan if-else karar yapılarıdır. Öncelikle konumServisleri switchinin açık olup olmadığı kontrol edilir. Açık olursa alt switchlerin (Konum Al ve Konum Gönder) görünümü aktif, kapalı olursa da pasif edilir. Bunun için "setVisibility" metodunu kullanarak içine View sınıfından parametre gönderiniz (Görsel 4.12).

```
@Override
protected void onResume() {
    super.onResume();
    konumServisleri.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
            if(konumServisleri.isChecked()){
                konumAl.setVisibility(View.VISIBLE);
                konumGonder.setVisibility(View.VISIBLE);
            }
            else
            {
                konumAl.setVisibility(View.GONE);
                konumGonder.setVisibility(View.GONE);
            }
        }
    });
}
```

Görsel 4.12: Switchlerin kontrolü

7. Adım: Buton işleri için buttonun onClick özelliğine "onayla" yazınız ve MainActivity'yi açıp orada da geri dönüş değeri olmayan bir metot tanımlayınız. Bu metodun içinde switchin kontrolünü yapınız. Switch pasif durumdaysa diğer alt switchlere bakmadan Toast mesajı ile "Konum Servisleri Kapalı" mesajı veriniz. Switch aktif durumdaysa alt switchleri kontrol ediniz ve bu switchlerin durumunu Toast mesajı ile bildiriniz (Görsel 4.13).

```
public void onayla(View view){
    if(!konumServisleri.isChecked())
    {
        Toast.makeText( context: this, text: "Konum Servisleri Kapalı",Toast.LENGTH_LONG).show();
    }else {
        if(konumAl.isChecked()==true && konumGonder.isChecked()==true){
            Toast.makeText( context: this, text: "Konum Al ve Konum Gönder Açık",Toast.LENGTH_LONG).show();
        }
        if(konumAl.isChecked()==true && konumGonder.isChecked()==false){
            Toast.makeText( context: this, text: "Konum Al Açık Konum Gönder Kapalı",Toast.LENGTH_LONG).show();
        }
        if(konumAl.isChecked()==false && konumGonder.isChecked()==true){
            Toast.makeText( context: this, text: "Konum Al Kapalı Konum Gönder Açık",Toast.LENGTH_LONG).show();
        }
        if(konumAl.isChecked()==false && konumGonder.isChecked()==false){
            Toast.makeText( context: this, text: "Konum Al ve Konum Gönder Kapalı",Toast.LENGTH_LONG).show();
        }
    }
}
```

Görsel 4.13: onayla metodunun içi

NOT: konumServisleri önünde bulunan "!" işareti durumu terslemek için kullanılır. Bir başka deyişle olumlu olan durumu olumsuza çevirir. Ayrıca else karar yapısı içinde farklı durumlarda ne yapılacağını anlatmak için if'ler yeniden kullanılır ve iç içe karar yapıları oluşturulur.

MainActivity içindeki kodların bütün hâli Görsel 4.14'te görülür.

```java
package com.atilimciftci.konumservisleri;

import ...

public class MainActivity extends AppCompatActivity {

    Switch konumServisleri,konumGonder,konumAl;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        konumServisleri = findViewById(R.id.switch_KonumServisi);
        konumAl = findViewById(R.id.switch_KonumAl);
        konumGonder = findViewById(R.id.switch_KonumGonder);

        // İlk Açılacağı zaman Kontrol edeceğiz.
        if(konumServisleri.isChecked()){
            konumAl.setVisibility(View.VISIBLE);
            konumGonder.setVisibility(View.VISIBLE);
        }
        else
        {
            konumAl.setVisibility(View.GONE);
            konumGonder.setVisibility(View.GONE);
        }
    }
    @Override
    protected void onResume() {
        super.onResume();
        konumServisleri.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                if(konumServisleri.isChecked()){
                    konumAl.setVisibility(View.VISIBLE);
                    konumGonder.setVisibility(View.VISIBLE);
                }
                else {
                    konumAl.setVisibility(View.GONE);
                    konumGonder.setVisibility(View.GONE);
                }
            }
        });
    }
    public void onayla(View view){
        if(!konumServisleri.isChecked())
        {
            Toast.makeText( context: this, text: "Konum Servisleri Kapalı",Toast.LENGTH_LONG).show();
        }else {
            if(konumAl.isChecked()==true && konumGonder.isChecked()==true){
                Toast.makeText( context: this, text: "Konum Al ve Konum Gönder Açık",Toast.LENGTH_LONG).show();
            }
            if(konumAl.isChecked()==true && konumGonder.isChecked()==false){
                Toast.makeText( context: this, text: "Konum Al Açık Konum Gönder Kapalı",Toast.LENGTH_LONG).show();
            }
            if(konumAl.isChecked()==false && konumGonder.isChecked()==true){
                Toast.makeText( context: this, text: "Konum Al Kapalı Konum Gönder Açık",Toast.LENGTH_LONG).show();
            }
            if(konumAl.isChecked()==false && konumGonder.isChecked()==false){
                Toast.makeText( context: this, text: "Konum Al ve Konum Gönder Kapalı",Toast.LENGTH_LONG).show();
            }
        }
    }

}
```

Görsel 4.14: MainActivity sayfasının içeriği

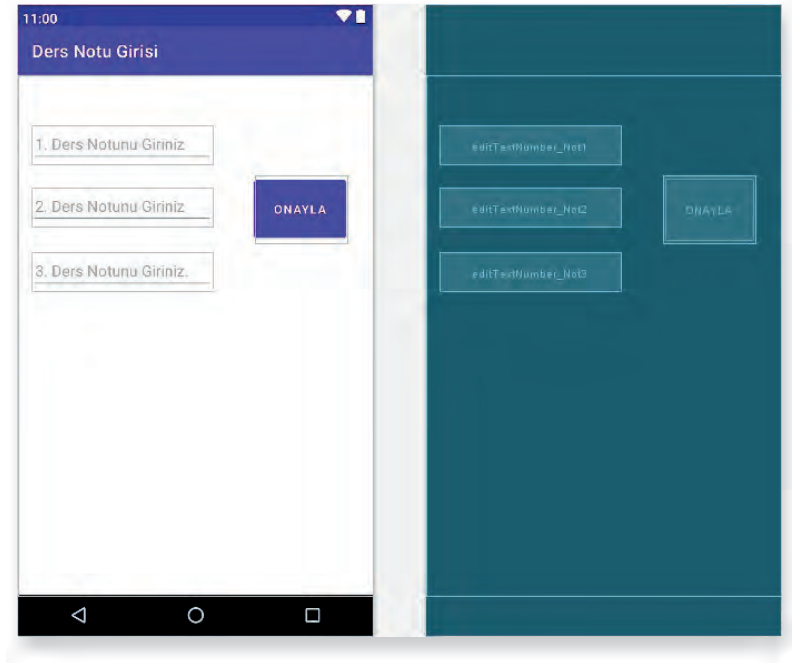
4.1.3. if-else-if Karar Yapısı

Birden fazla şartın olduğu durumlarda "if-else-if" karar yapısı kullanılır. Örneğin dördüncü uygulama için onayla isimli metot içinde dört adet şart vardır (Görsel 4.14). Bunlar, tek tek "if" ile yazılabileceği gibi "if-else-if" yapısıyla da oluşturulabilir. "if-else-if" karar yapısı şu şekildedir:

```
if(şart){
        //true ise yapılacaklar
}else if(şart)
        //true ise yapılacaklar
}else if(şart)
        //true ise yapılacaklar
}
```

5. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranında kullanıcı tarafından bir derse ait üç not girişi yapıldığında dersin ortalamasını alan ve Toast mesajı ile ortalamanın karşılığını veren uygulamayı tasarlayınız.

1. Adım: Görsel 4.15'teki tasarımı yaparak "Infer Constraints" ile otomatik constraint veriniz. Üç adet number ögesi (EditText), 1 adet button ögesi ekleyiniz. EditTextlerin id'lerini "editTextNumber_Not1", "editTextNumber_Not2", "editTextNumber_Not3" şeklinde, buttonun id'sini ise "button_Onayla" biçiminde oluşturunuz. EditTextlerin hint özelliğine ilgili yazıları giriniz. Buttonun onClick özelliğine de "onayla" yazınız.



Görsel 4.15: Not girişi uygulama tasarımı

2. Adım: MainActivity sınıfında nesneleri tanımlayıp onCreate blokları içinde başlatınız.

3. Adım: "onayla" isminde void tipteki metodu tanımlayarak View sınıfından view isimli bir parametre gönderiniz.

4. Adım: Metot içinde integer tipte **not1**, **not2**, **not3** isimlerindeki değişkenleri tanımlayınız. EditTextlerden gelen verileri integer veri tipine çevirerek oluşturulan değişkenlerin içlerine atınız. Float tipte **ort** isminde bir değişken tanımlayıp girilen üç notun hesaplamasını bu değişkende yapınız.

5. Adım: if-else-if karar yapısını oluşturarak Görsel 4.16'daki gibi şartları yazınız.

```
if (ort>=0 && ort<25){
    Toast.makeText( context: this, text: "0 ile kaldınız.",Toast.LENGTH_LONG).show();
}else if(ort>=25 && ort<50){
    Toast.makeText( context: this, text: "1 ile kaldınız.",Toast.LENGTH_LONG).show();
}else if(ort>=50&& ort<60){
    Toast.makeText( context: this, text: "2 ile geçtiniz.",Toast.LENGTH_LONG).show();
}else if(ort>=60 && ort<70){
    Toast.makeText( context: this, text: "3 ile geçtiniz.",Toast.LENGTH_LONG).show();
}else if(ort>=70 && ort<85){
    Toast.makeText( context: this, text: "4 ile geçtiniz.",Toast.LENGTH_LONG).show();
}else if(ort>=85 && ort<=100){
    Toast.makeText( context: this, text: "5 ile geçtiniz. Tebrikler",Toast.LENGTH_LONG).show();
}else
{
    Toast.makeText( context: this, text: "Girilen Not Bilgileri Hatalıdır.",Toast.LENGTH_LONG).show();
}
```

Görsel 4.16: Not girişi uygulaması if-else-if karar yapısı

6. Adım: MainActivity sınıfının içi Görsel 4.17a ve Görsel 4.17b'deki hâle geldikten sonra uygulamayı başlatarak test ediniz.

```
package com.atilimciftci.dersnotugirisi;

import ...

public class MainActivity extends AppCompatActivity {

    EditText editTextNot1,editTextNot2,editTextNot3;
    Button buttonOnayla;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editTextNot1 = findViewById(R.id.editTextNumber_Not1);
        editTextNot2 = findViewById(R.id.editTextNumber_Not2);
        editTextNot3 = findViewById(R.id.editTextNumber_Not3);
    }
    public void onayla(View view){
        int not1 = Integer.parseInt(editTextNot1.getText().toString());
        int not2 = Integer.parseInt(editTextNot2.getText().toString());
        int not3 = Integer.parseInt(editTextNot3.getText().toString());
```

Görsel 4.17a: MainActivity sayfasında ders notu girişi 1. kısım

```
        float ort = (not1+not2+not3)/3;

        if (ort>=0 && ort<25){
            Toast.makeText( context: this, text: "0 ile kaldınız.",Toast.LENGTH_LONG).show();
        }else if(ort>=25 && ort<50){
            Toast.makeText( context: this, text: "1 ile kaldınız.",Toast.LENGTH_LONG).show();
        }else if(ort>=50&& ort<60){
            Toast.makeText( context: this, text: "2 ile geçtiniz.",Toast.LENGTH_LONG).show();
        }else if(ort>=60 && ort<70){
            Toast.makeText( context: this, text: "3 ile geçtiniz.",Toast.LENGTH_LONG).show();
        }else if(ort>=70 && ort<85){
            Toast.makeText( context: this, text: "4 ile geçtiniz.",Toast.LENGTH_LONG).show();
        }else if(ort>=85 && ort<=100){
            Toast.makeText( context: this, text: "5 ile geçtiniz. Tebrikler",Toast.LENGTH_LONG).show();
        }else
        {
            Toast.makeText( context: this, text: "Girilen Not Bilgileri Hatalıdır.",Toast.LENGTH_LONG).show();
        }
    }
}
```

Görsel 4.17b: MainActivity sayfasında ders notu girişi 2. kısım

4.1.4. Switch-Case Yapısı

Switch-case, bir başka karar yapısıdır. Burada olay, switch içinde verilen ifadenin değeri hangi case içinde yer alırsa o case blokunun çalışması üzerine kuruludur. Her case ifadesi "break" komutuyla bitirilir. "break" komutu, işlem bittikten sonra bir alt case ifadesi çalıştırılmadan switch-case yapısından çıkmak için kullanılır. C dili zamanından bu yana değişiklik göstermeyen bu karar yapısı, birçok farklı dilde aynı şekilde kullanılır (Tablo 4.1). Switch ifadesinde belirtilen parametrenin veri türü ile case içinde belirtilen verinin türü aynı olmak zorundadır.

Tablo 4.1: Switch-Case Kullanımı

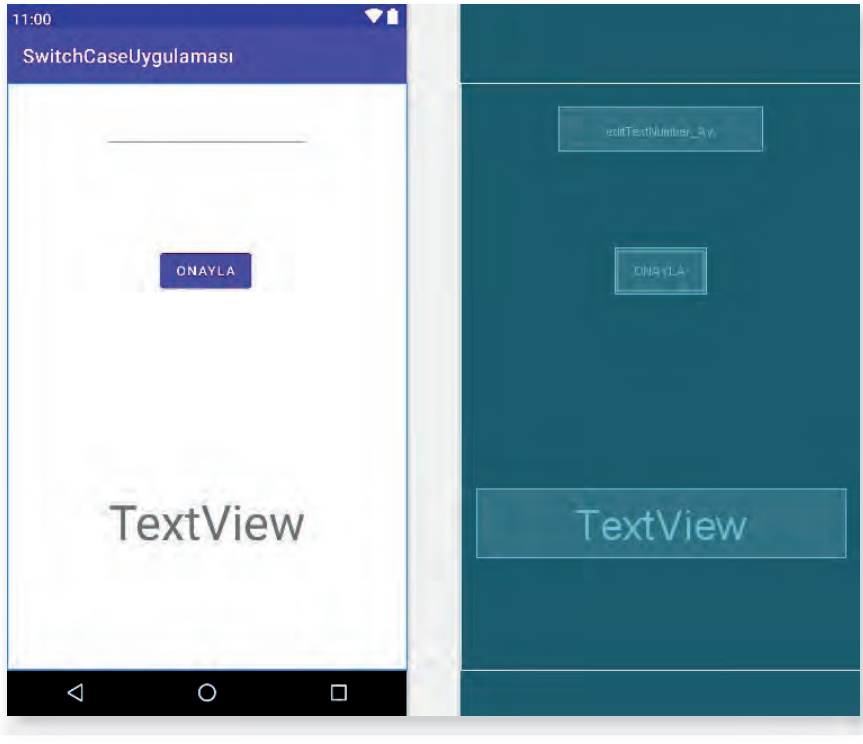
switch (integer ifade) {
case integer değer :
//kodlar
break;
case integer değer :
//kodlar
break;
case integer değer :
//kodlar
break;
default:
//kodlar
}
	switch (String ifade) {
	case "metinsel değer":
	//kodlar
	break;
	case "metinsel değer":
	//kodlar
	break;
	case "metinsel değer":
	//kodlar
	break;
	default:
	//kodlar
	}
		switch (char ifade) {
		case 'char değer' :
		//kodlar
		break;
		case 'char değer':
		//kodlar
		break;
		case 'char değer':
		//kodlar
		break;
		default:
		//kodlar
		}

NOT: Default değeri, case ifadelerinden hiçbiri uygun değilse çalışacak blok kısmıdır.

6. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranında kullanıcı tarafından ay bilgisi bir EditTexte sayı olarak girildikten sonra "Onayla" buttonuna basıldığı zaman hangi ay olduğunu yazıyla bir TextViewde gösteren mobil uygulamayı switch-case karar yapısını kullanarak tasarlayınız.

1. Adım: File>New>New Project sekmesinden yeni proje açınız ve Empty Activity seçiniz.

2. Adım: Açılan ekranda activity_main içine Görsel 4.18'deki ekran tasarımını yapınız. Tasarımda bir adet "Number", bir adet "Button" ve bir adet de "TextView" bulundurunuz. Number ögesini tasarım ekranının üst kısmına sabitleyiniz. Boşluk olarak 24 dp bırakınız. Aynı şekilde sağ ve sol yanı da çerçeveye sabitleyerek sağdan ve soldan tasarıma ortalayınız. Boşluk değeri olarak 0 dp bırakınız. Button ögesini de Number nesnesine sabitleyip arada 100 dp bırakınız. Soldan ve sağdan sabitleyip onu da tasarımda ortalayınız. TextView ögesinin layout_width (genişlik) özelliğini "match_parent" şeklinde ayarlayıp, tasarım içinde tek satırda tek öge olarak kendini büyütmesini sağlayınız. Sağdan ve soldan 8 dp boşluk bırakarak çerçeveye sabitleyiniz. Üstten ise buttona sabitleyip arada 200 dp bırakınız. Yazı tipi boyutu textSize özelliğini 50 sp olarak ayarlayıp, "textAlignment" özelliğinden "Align Center"ı işaretleyerek TextView yerleşimini ortalayınız.



Görsel 4.18: Ayların gösterilmesi uygulaması

3. Adım: Number ögesinin id'sini "editTextNumber_Ay" şeklinde değiştiriniz. Button ögesinin id'sini "button_Onayla", text özelliğini "Onayla" olarak değiştiriniz. "onClick" özelliğine de "onayla" yazınız.

4. Adım: Ögeleri Görsel 4.19'da görüldüğü şekilde initialize yapınız.

```
public class MainActivity extends AppCompatActivity {
    EditText number;
    Button button;
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        number = findViewById(R.id.editTextNumber_Ay);
        button = findViewById(R.id.button_Onayla);
        textView = findViewById(R.id.textView_Ay);
    }
```

Görsel 4.19: Ögelerin initialize yapılması

5. Adım: onayla isimli buttona tıklandığı zaman çalışacak metodu Görsel 4.20'de olduğu gibi hazırlayınız. Görselde sayfa ikiye ayrılmış durumdadır. Bu ayrılma kodlama esnasında yapılmaz, case 7 ifadesi case 6'dan sonra alt alta yazılarak işleme devam edilir. "onayla" isimli metodun içinde switch-case oluşturulur. Metot içinde önce sayı okutulur sonra switch ile karşılaştırma yapılır. Case olarak gelebilecek değerler 1 ile 12 arasında değişir. Bunlar dışında bir değer gelirse de default olarak ayrılan bölüm çalışır. Buttona her tıklamada number nesnesinin içeriğinin de silinmesi için "number.setText("");" kodunu yazınız ve metot bittikten sonra uygulamayı çalıştırınız.

```
public void onayla(View view){
    int sayiAy = Integer.parseInt(number.getText().toString());
    String ay="";
    number.setText("");
    switch (sayiAy){
        case 1:
            ay="OCAK";
            break;
        case 2:
            ay="ŞUBAT";
            break;
        case 3:
            ay="MART";
            break;
        case 4:
            ay="NİSAN";
            break;
        case 5:
            ay="MAYIS";
            break;
        case 6:
            ay="HAZİRAN";
            break;
        case 7:
            ay="TEMMUZ";
            break;
        case 8:
            ay="AĞUSTOS";
            break;
        case 9:
            ay="EYLÜL";
            break;
        case 10:
            ay="EKİM";
            break;
        case 11:
            ay="KASIM";
            break;
        case 12:
            ay="ARALIK";
            break;
        default:
            ay="YANLIŞ BİLGİ";
            break;
    }
    textView.setText(ay);
}
```

Görsel 4.20: "onayla" isimli metodun içeriği

SIRA SİZDE: Mobil uygulama ekranında EditText ile yaş bilgisi, switch ögesi ile de ehliyet kursunu tamamlama bilgisi istenen kişi için buttona basıldığında yaşı 18 ve daha büyükse ve ehliyet sınavından geçmişse ekrana Toast mesajı olarak "Ehliyet Alabilirsiniz", değilse "Ehliyet Alamazsınız" mesajı veren uygulamayı hazırlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Palette menüsünü kullanarak yaş bilgisini girmek için EditText ögesi oluşturdu.		
2. Palette menüsünü kullanarak ehliyet kursu tamamlama bilgisi için switch ögesi oluşturdu.		
3. Palette menüsünü kullanarak Button ögesi oluşturdu.		
4. Ögelerin ConstraintLayout içinde sınırlarını belirledi.		
5. Ögeleri yaşam döngüsü içinde initalize etti.		
6. Buttona tıkladığında çalışacak metodu hazırladı.		
7. Gereğine uygun olacak biçimde "if" karar yapısını oluşturdu.		
8. Şart yapısında ve (&&) operatörünü amacına uygun olarak kullandı.		
9. Toast mesaj ögesini amacına uygun olarak oluşturdu.		

4.2. DÖNGÜ YAPILARI

Döngüler, uygulama oluşturma aşamasında belirli şartlara bağlı olarak yazılan kod blokunu istendiği sayıda tekrar tekrar çalıştırmayı sağlayan yapılardır. Duruma göre bir kod iki defa çalıştırılabileceği gibi binlerce defa ve hatta sonsuz defa da çalıştırılabilir. Burada iş aslında uygulamanın tasarımcısının ne düşündüğü ile ilgilidir. Örneğin 30 kişinin bulunduğu bir sınıfın bilgilerinin girilmesi istendiğinde 30 kişi için ayrı ayrı isteme kodu yazılması yerine tek satır kodu 30 defa çalışacak bir döngüye sokarak bu iş daha kolay yapılabilir. Bilgilerin belirli bir şarta göre 30 kişi için getirilmesi istediğinde 30 ayrı "if" karar yapısı yazılması yerine 30 defa çalışacak bir döngünün blokları arasında tek bir "if" karar yapısı oluşturulabilir.

NOT: Uygulama parçacığı üretmekle uygulamanın bütününü oluşturmak arasında fark vardır. Uygulama parçacığı üretilirken yazılan kodlar az, oluşturulan yapılar kısa görünebilir ancak bir uygulamanın bütünü oluşturulurken binlerce satır kodlama yapılabilir. Her kod tek sefer çalışıp geçilirse veya istenen her bilgi için bir isteme kodu yazılırsa bu binlerce satırlık kod, on binlerce satırlık hâle dönüşür.

4.2.1. Sayaçlar

Döngülerin çalışma mekanizması içinde yapının kaç defa çalıştığının veya çalışacağının tutulması gerekir. Bu işlemi kontrol eden, döngünün çalışma aralığını veya kaç defa çalıştığını anlık olarak hafızasına alan yapıya **sayaç** adı verilir. Sayaçlar yapısı itibarıyla birer değişkendir. Tek fark, sayaç döngü yapısı içinde tanımlanırsa döngü dışına çıkıldığında tanımlanan sayaç yok olacak şekilde ayarlanır.

Sayaçlar, döngü içinde sürekli artan veya sürekli azalan bir yapıda olmalıdır. Bu nedenle sayaçlar tanımlanırken en uygun olan yol tercih edilir. Uygulama tasarlanırken bir değişkenin değerini artırmak için "+", azaltmak için "-" aritmetiksel operatörler kullanılır. Bir x değişkeninin değerini 1 artırmak için "x=x+1" ifadesi yazılmalıdır. Bu ifadenin bir çeşidi "x++", bir diğeri de "x+=1" ifadesidir.

Tablo 4.2'de görüldüğü gibi bir değişkenin değerini artırmak veya azaltmak için birden fazla seçenek olabilir. Bu seçeneklerden uygun olanı seçilip sayaç için döngü yapısında uygulanır.

Tablo 4.2: Değişkenlerde Aritmetiksel İşlemler

x değişkeninin değerini 1 artırmak için	x=x+1	x+=1	x++
x değişkeninin değerini 5 artırmak için	x=x+5	x+=5	Yok
x değişkeninin değerini 1 azaltmak için	x=x-1	x-=1	x--
x değişkeninin değerini 5 azaltmak için	x=x-5	x-=5	Yok

Sayaçlar yapı itibarıyla birer değişken olduğu için sayaçların isimlendirilmesinde değişken oluşturma kuralları geçerlidir ve sayaçlara verilecek isim önemli değildir. Sayaçlar isimlendirilirken uygun yöntem, programın akışına göre sayaçlara bir isim belirlemektir. Sayaçlar genellikle programlamada "i" veya "sayac" değişken ismi ile ifade edilir.

4.2.2. for Döngüsü

for, uygulama tasarımlarında en çok kullanılan döngü çeşididir. Genellikle tekrar sayısının belli ve net olduğu ifadelerde kullanılır. Döngü yapısı oluşturulurken üç parametre değerine ihtiyaç vardır. Bunlardan ilki, sayaç tanımlama parametresi; ikincisi, şart oluşturma parametresi; üçüncüsü de sayacın artırma veya azaltma işlemi olan parametredir. Şart ifadesinin sonucu "true" olduğu sürece for döngüsünün blokları tekrar tekrar çalışır. for döngüsü şu şekilde tanımlanır:

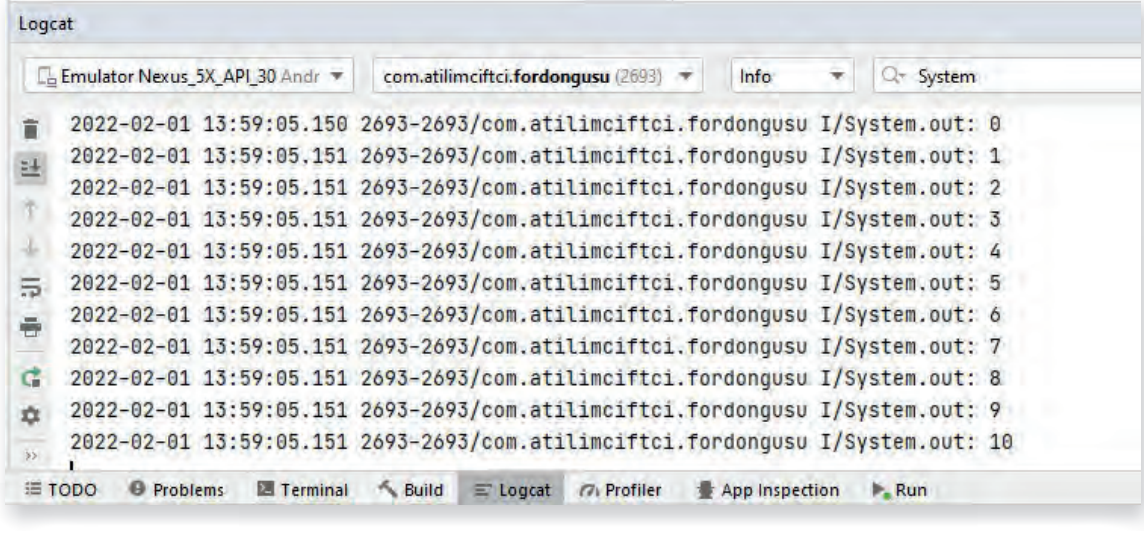
```
for( sayaç tanımlama; şart ifadesi; sayaç arttırma veya azaltma)
{
        //kodlar
}
```

Örneğin 1'den 10'a kadar olan sayıların Logcat ekranında gösterilmesi istendiğinde uygulama geliştirme ortamında kodlar onResume yaşam döngüsünün içine Görsel 4.21'deki gibi yazılır.

```
protected void onResume() {
    super.onResume();
    for (int i=0;i<=10;i++)
    {
        System.out.println(i);
    }
}
```

Görsel 4.21: for döngüsü

Görsel 4.21'de olduğu gibi "`int i=0`" komutu ile **i** isminde bir sayaç tanımlanır. Döngü yapısında yer alan bloklar arasındaki kodlar, sayacın değeri 0'dan başlayarak 10'a eşit veya 10'dan küçük olduğu sürece "`i<=0`" komutu ile tekrar tekrar çalışır. Her çalışma sonucunda tekrar başa dönüldüğünde ise "`i++`" komutu ile sayaç 1 sayı artarak yeniden şart ifadesinde karşılaştırılır. Blok içine yazılan "`System.out.println(i);`" komutu da Logcat ekranında Görsel 4.22'de olduğu gibi 0 ile 10 arasındaki değerleri gösterir.



Görsel 4.22: Logcat ekran çıktısı

NOT:

for döngüsü için çalışma mantığı incelendiğinde yanda göründüğü gibi parametre bölümleri numaralandırılırsa 1 numarada sayaç tanımlama bölümü görülür. Birinci bölüm, for döngüsü çalışır çalışmaz ilk olarak işleme girer ve döngü ne kadar tekrar ederse etsin sadece 1 defa çalışır. İki numaralı bölüm, şart ifadesinin yazılacağı bölümdür. İkinci bölüm, sayaç tanımlamasından hemen sonra çalışır. Sonuç true gelirse bloklar (kaşlı ayraç içi) arasına giriş yapar ve kodların çalıştırılmasını sağlar. Şart sonucu false gelirse de blok içine girmeden direkt for döngüsü dışına gider. Üç numarada da sayaç artırma veya azaltma bölümü yer alır. Döngü çalıştırıldıktan sonra blok içindeki işlem yapılmışsa, bir başka deyişle blok arasına girilirse işlem bittikten sonra bu bölüm devreye girer, belirtildiği şekilde değeri azaltır veya artırır. Hemen ardından tekrar şart ifadesine (2 numara) bakılır ve süreç bu sırayla devam eder.

```
for( (1); (2); (3); )
{
        //kodlar
}
```

7. UYGULAMA: İşlem adımlarına göre 1'den 50'ye kadar olan sayıları Logcat ekranında gösteren mobil uygulamayı tasarlayınız.

1. Adım: File>New>New Project sekmesinden yeni proje açınız ve Empty Activity seçiniz.

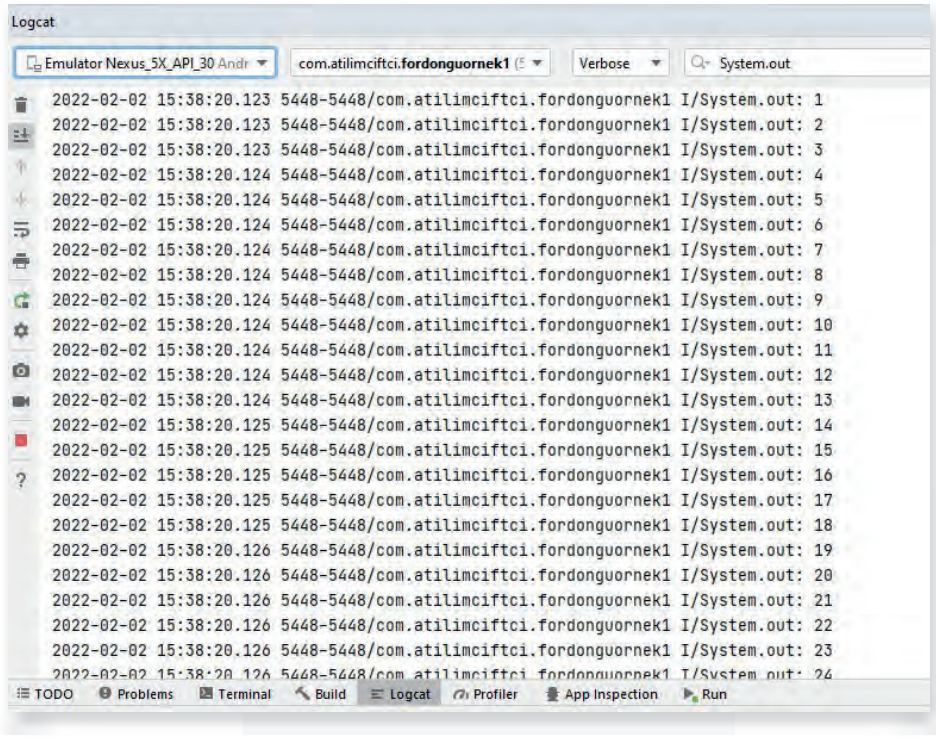
2. Adım: MainActivity sınıfını açınız. "onResume" yaşam döngüsü metodunu oluşturunuz.

3. Adım: Yaşam döngüsünün içine bir for döngüsü oluşturunuz. Sayacı 1'den başlatıp sayaç 50'den küçük veya 50'ye eşit oldukça blok içindeki kodları çalıştıran ve her defasında sayacı 1 sayı artıran şekilde düzenleyiniz (Görsel 4.23).

```
package com.atilimciftci.fordonguornek1;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onResume() {
        super.onResume();
        for (int i=1;i<=50;i++){
            System.out.println(i);
        }
    }
}
```

Görsel 4.23: 1'den 50'ye kadar olan sayıları yazdıran uygulama

4. Adım: Uygulamayı çalıştırdığınızda uygulama geliştirme programına dönerek Logcat ekranını açınız. Arama çubuğuna "System.out" yazarak çıkan sonuçları filtreleyiniz. Tek satır kod yazılmasına rağmen (System.out.println(i)) 50 defa aynı kodun çalıştırıldığını gözlemleyiniz (Görsel 4.24).



Görsel 4.24: 1'den 50'ye kadar olan sayıları ekranda yazdıran program Logcat ekranı

8. UYGULAMA: İşlem adımlarına göre yedinci uygulamaya benzer şekilde 0'dan 100'e kadar olan sayılardan üçün katı olanları Logcat ekranında gösteren mobil uygulamayı tasarlayınız.

NOT:

Bu işlem, iki farklı yöntemle yapılabilir. Birinci yöntemde artış değeri üç olacak şekilde ayarlanır. İkinci yöntemde ise if karar yapısı da for döngüsüne dâhil edilerek kullanılır.

1. Yöntem

1. Adım: File>New>New Project sekmesinden yeni proje açınız ve Empty Activity seçiniz.

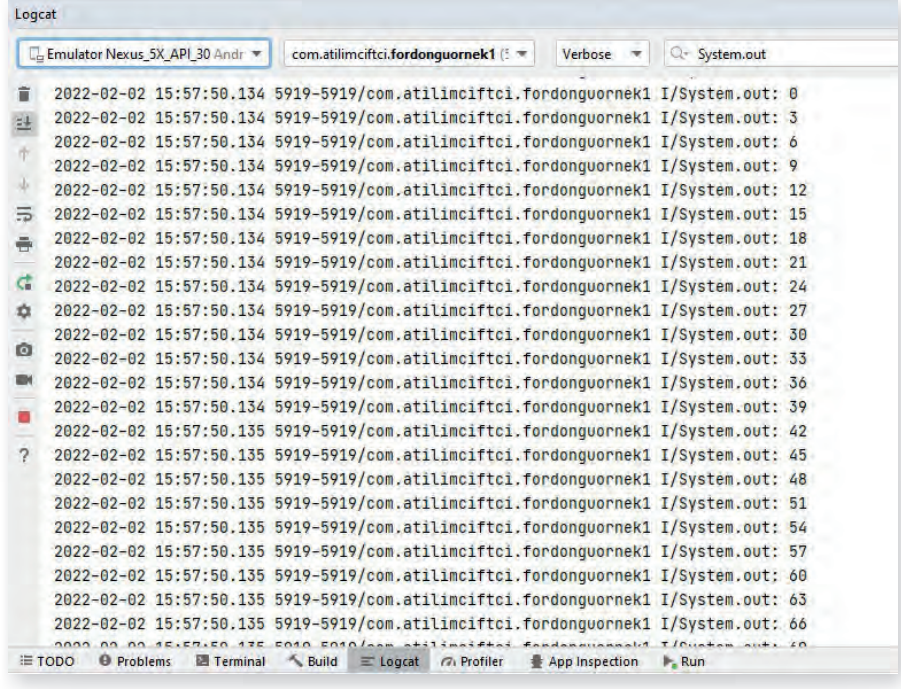
2. Adım: MainActivity sınıfını açınız. "onResume" yaşam döngüsü metodunu oluşturunuz.

3. Adım: Yaşam döngüsünün içine bir for döngüsü oluşturunuz. Sayacı 0'dan başlatıp sayaç 100'den küçük veya 100'e eşit oldukça blok içindeki kodları çalıştıran ve her defasında sayacı 3 sayı artıran olacak şekilde düzenleyiniz (Görsel 4.25).

```
package com.atilimciftci.fordonguornek1;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onResume() {
        super.onResume();
        for (int i=0;i<=100;i+=3){
            System.out.println(i);
        }
    }
}
```

Görsel 4.25: MainActivity sayfasında 1'den 100'e kadar üçün katı olan sayıları gösteren uygulama

4. Adım: Uygulamayı çalıştırdığınızda Logcat ekranını açınız. Arama çubuğuna "System.out" yazarak çıkan sonuçları filtreleyiniz. Tek satır kod yazılmasına rağmen (System.out.println(i)) kodun tekrar tekrar çalıştığını gözlemleyiniz (Görsel 4.26).



Görsel 4.26: 1'den 100'e kadar üçün katı olan sayıların Logcat ekranında gösterilmesi

2. Yöntem

1. Adım: File>New>New Project sekmesinden yeni proje açınız ve Empty Activity seçiniz.

2. Adım: MainActivity sınıfını açınız. "onResume" yaşam döngüsü metodunu oluşturunuz.

3. Adım: Yaşam döngüsünün içine bir for döngüsü oluşturunuz. Sayacı 0'dan başlatıp sayaç 100'den küçük veya 100'e eşit oldukça blok içindeki kodları çalıştıran ve her defasında sayacı 1 sayı artıran olacak şekilde düzenleyiniz (Görsel 4.25).

4. Adım: for döngüsünün blokları arasına if karar yapısı oluşturunuz. Gelen her "i" sayacının 3 ile modunu aldırınız. İşlemin sonucu "0" olduğu takdirde ekrana yazdırma işlemi yapacak kodu yazınız (Görsel 4.27).

```
package com.atilimciftci.fordonguornek1;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onResume() {
        super.onResume();
        for (int i=0;i<=100;i++){
            if(i%3==0){
                System.out.println(i);
            }
        }
    }
}
```

Görsel 4.27: MainActivity sayfasında 1'den 100'e kadar üçün katı olan sayıları gösteren uygulama

5. Adım: Uygulamayı çalıştırdığınızda Logcat ekranını açınız. Arama çubuğuna "System.out" yazarak çıkan sonuçları filtreleyiniz. Gelen sonucun Görsel 4.26 ile aynı şekilde olduğunu gözlemleyiniz.

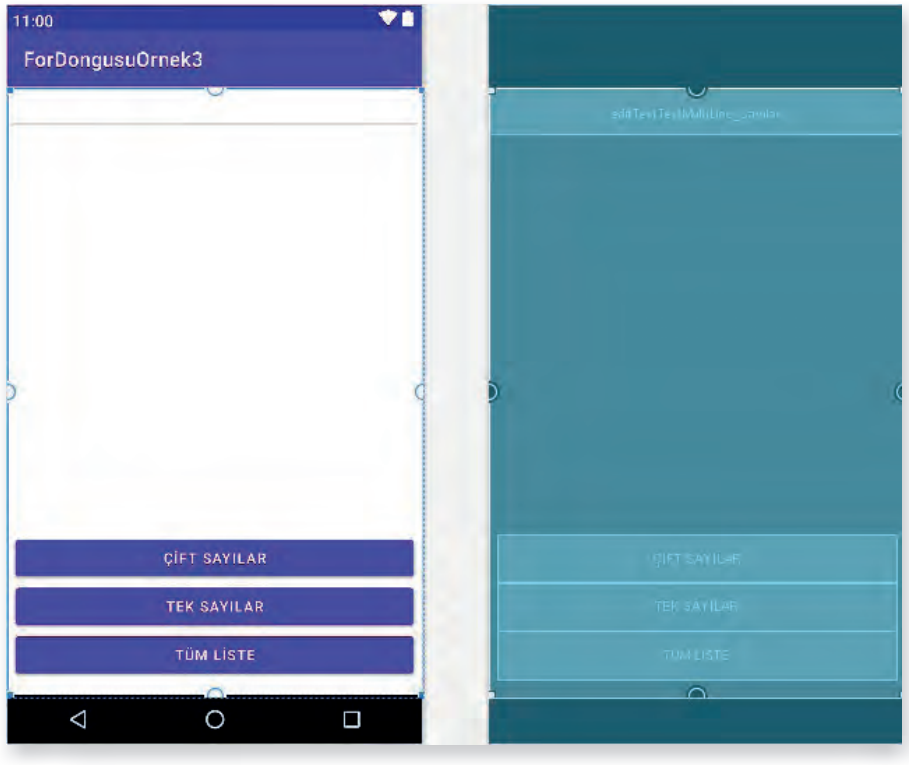
NOT:

Uygulama geliştirme aşamasında uygulanabilecek birden fazla yöntem vardır. Bu yöntemler aynı sonuca götürse de bunların bazısı daha uygun çalışmaya sahiptir. Sekizinci uygulamada yazılan for döngüsü 1. yöntemde 34 defa çalışırken 2. yöntemde 101 defa çalışır.

9. UYGULAMA: İşlem adımlarına göre mobil ekrandaki üç adet Button ile seçecek şekilde 0'dan 15'e kadar olan sayıların çift olanlarını, tek olanlarını ve hepsini alt alta yazdıran uygulamayı tasarlayınız.

1. Adım: File>New>New Project sekmesinden yeni proje açınız ve Empty Activity seçiniz.

2. Adım: Görsel 4.28'de görülen ekran tasarımını yapınız. Üst tarafta bir "Multiline Text" yer alır. Alt tarafta da üç adet button bulunur. Multiline Text'in "ConstraintLayout"ununu uygulama penceresine soldan ve sağdan yerleştirip 8 dp boşluk bırakınız. Yukarıdan da ekranın üstüne bağlayarak 8 dp boşluk bırakınız. "layout_width" özelliğine (genişlik özelliği) tamamını kaplaması için "match_parent" yazınız. Yükseklik değeri olan "layout_height" özelliğine ise yazıldığı kadar genişlemesini sağlayan "wrap_content" yazınız.



Görsel 4.28: Sayıları gösteren uygulama

3. Adım: Buttonları da ekleyerek aynı şekilde genişlik özelliği olan "layout_width"e "match_parent", yükseklik özelliği olan "layout_height"e "wrap_content" yazınız. En alttaki buttonu yerleşimin alt çerçevesine bağlayıp 16 dp boşluk bırakınız. Soldan ve sağdan da 8 dp boşluk bırakarak yan çerçeveye sabitleyiniz. Diğer buttonları da sırasıyla birbirine bağlayarak hizalama işlemini bitiriniz.

4. Adım: Multiline Text'in id özelliğini **"**editTextTextMultiLine_Sayilar**"** şeklinde veriniz. Buttonların id özelliklerini de sırasıyla "button_CiftSayilar", "button_TekSayilar", "button_TumSayilar" şeklinde değiştiriniz. Ayrıca buttonların "onClick" özelliklerine üstten sırayla "ciftSayilar", "tekSayilar", "tumSayilar" olacak şekilde yazınız.

5. Adım: EditTexti ve buttonları tanımlayarak başlatmasını (initialize) onCreate metodunda yapınız (Görsel 4.29).

```
public class MainActivity extends AppCompatActivity {
    EditText editText_Sayilar;
    Button button_TekSayi,button_CiftSayi,button_TumSayi;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText_Sayilar = findViewById(R.id.editTextTextMultiLine_Sayilar);
        button_CiftSayi =findViewById(R.id.button_CiftSayilar);
        button_TekSayi = findViewById(R.id.button_TekSayilar);
        button_TumSayi = findViewById(R.id.button_TumSayilar);
    }
}
```

Görsel 4.29: Nesneleri initialize aşaması

6. Adım: Buttonlara ait üç adet metodu yazınız.

7. Adım: Çift sayılar için yazılan metodun blokları arasına Multiline Text'i temizlemesi için gereken "`editText_Sayilar.setText("");`" kodunu yazınız. Bu komut, her button farklı bir amacı aynı Text üzerinde yapacağı için gereklidir. Ardından 0'dan 15'e kadar sayacı birer birer artırarak 16 defa dönen bir döngü açınız ve içinde if şart yapısını oluşturunuz. Sayacın modunu alınız. İşlem sonucu sıfıra eşit olursa modu alınan sayacın çift olduğu anlamına gelir. Sayaç çift olduğunda ekrana sayıyı yazmasını isteyiniz (Görsel 4.30).

```
public void ciftSayilar(View view){
    editText_Sayilar.setText("");
    for (int i=0;i<=15;i++){
        if(i%2==0){
            editText_Sayilar.setText(editText_Sayilar.getText()+" "+i);
        }
    }
}
```

Görsel 4.30: Çift sayılar buttonunun metodu

8. Adım: Yedinci adımdaki gibi tek sayılar buttonu için de bir metot yazınız. Bu metot içinde çift sayılar metoduna göre sadece şart ifadesi değişecektir. Şart ifadesini "mod işlemi sonucu 0'dan farklı olursa" şeklinde düzenleyerek tek olan sayıları gösteriniz (Görsel 4.31).

```
public void tekSayilar(View view){
    editText_Sayilar.setText("");
    for (int i=0;i<=15;i++){
        if(i%2!=0){
            editText_Sayilar.setText(editText_Sayilar.getText()+"\n"+i);
        }
    }
}
```

Görsel 4.31: Tek sayılar buttonunun metodu

9. Adım: Yedinci adımdaki gibi tüm sayılar buttonu için de bir metot hazırlayınız. Bu metot içinde hiç şart ifadesi yazmayınız. Döngü içindeki tüm sayıları Multi Text'te gösteriniz (Görsel 4.32).

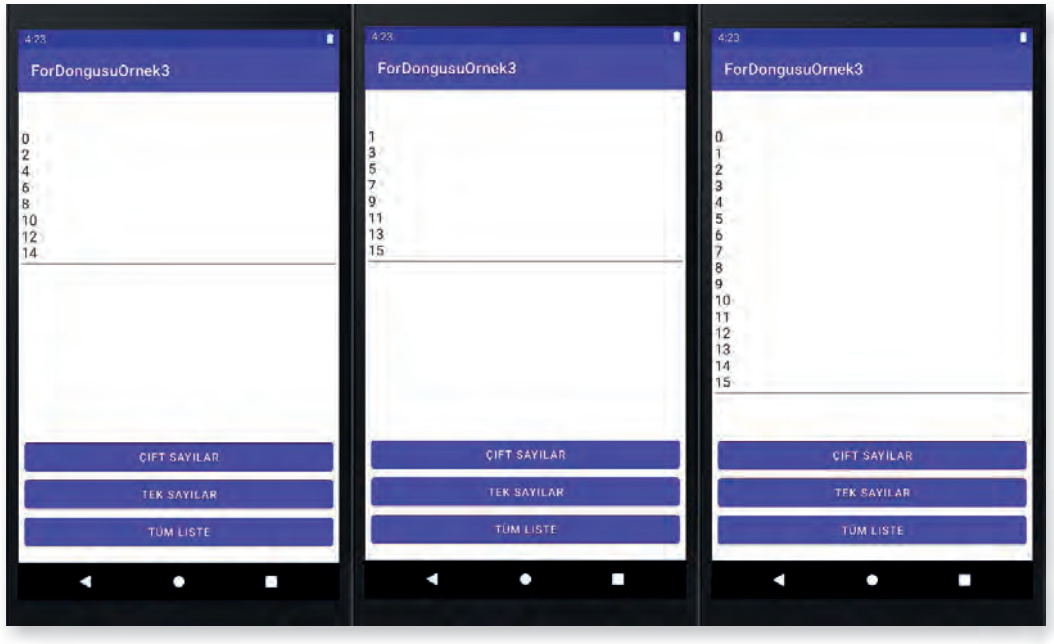
```
public void tumSayilar(View view){
    editText_Sayilar.setText("");
    for (int i=0;i<=15;i++){
            editText_Sayilar.setText(editText_Sayilar.getText()+"\n"+i);
    }
}
```

Görsel 4.32: Tüm sayılar buttonunun metodu

10. Adım: Tüm adımlar yapıldığına göre uygulamayı çalıştırınız. Uygulama çalıştırıldığında ve buttonlara tıklandığında çıkan sonuçlar da Görsel 4.33'te verilmiştir. Oluşturulan MainActivity'nin son şekli Görsel 4.34'te görülür.

UYARI: View, Button, EditTextlerden birinin altı kırmızı çizgi ile çizilmişse ona ait sınıf MainActivity içine import edilmelidir. Bu işlem için uyarı verilen yazının üstüne gelinip **Alt+Enter** tuşlarına birlikte basılırsa çözüm sağlanır.

NOT: "break" komutu tıpkı switch-case karar yapısında olduğu gibi burada da blok kırma görevinde yer alır. Döngünün belirli bir şarta bağlı olarak zamanından önce bitirilmesi istenirse "break" komutu aynı şekilde kullanılabilir. Örneğin bu komut sayesinde veri tabanında 10.000 kişinin yer aldığı bir listede T.C. Kimlik Numarası ile arama yapılırsa ve aranan kayıt tüm liste bitirilmeden bulunursa listenin devamını aratmaya gerek kalmayacağı için döngü orada bitirilebilir. Bu sayede de gereksiz iş yükü önlenir.



Görsel 4.33: Sayıları gösteren uygulama ekran çıktısı

```
import ...

public class MainActivity extends AppCompatActivity {
    EditText editText_Sayilar;
    Button button_TekSayi,button_CiftSayi,button_TumSayi;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText_Sayilar = findViewById(R.id.editTextTextMultiLine_Sayilar);
        button_CiftSayi =findViewById(R.id.button_CiftSayilar);
        button_TekSayi = findViewById(R.id.button_TekSayilar);
        button_TumSayi = findViewById(R.id.button_TumSayilar);
    }

    public void ciftSayilar(View view){
        editText_Sayilar.setText("");
        for (int i=0;i<=15;i++){
            if(i%2==0){
                editText_Sayilar.setText(editText_Sayilar.getText()+"\n"+i);
            }
        }
    }
    public void tekSayilar(View view){
        editText_Sayilar.setText("");
        for (int i=0;i<=15;i++){
            if(i%2!=0){
                editText_Sayilar.setText(editText_Sayilar.getText()+"\n"+i);
            }
        }
    }
    public void tumSayilar(View view){
        editText_Sayilar.setText("");
        for (int i=0;i<=15;i++){
                editText_Sayilar.setText(editText_Sayilar.getText()+"\n"+i);
        }
    }
}
```

Görsel 4.34: MainActivity sayfasında sayıları gösteren uygulama

SIRA SİZDE: Dokuzuncu uygulamadaki ekran tasarımına benzer şekilde 1 ile 100 arasındaki sayılar için "8'e Tam Bölünebilen", "12'ye Tam Bölünebilen" ve "15'e Tam Bölünebilen" sayılar için üç adet Button, sayıları göstermek için de bir adet ListView tasarlayarak uygulamayı geliştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Palette menüsünü kullanarak istenen üç adet Button ögesini oluşturdu.		
2. Palette menüsünü kullanarak sayıları göstermek için ListView ögesi oluşturdu.		
3. Ögelerin ConstraintLayout içinde sınırlarını belirledi.		
4. Ögeleri yaşam döngüsü içinde initalize etti.		
5. Buttonlara tıklandığında çalışacak metodu hazırladı.		
6. for döngü yapısını amacına uygun olarak hazırladı.		
7. for içinde if karar yapısını amacına uygun olacak bir biçimde oluşturdu.		
8. Buttonlara tıklandığında ListView içinde sayıları doğru biçimde gösterdi.		

4.2.3. while Döngüsü

Bir veya birden fazla kod blokunun belirtilen şart sağlandığı sürece tekrar ettiği döngü yapısıdır. while içinde belirtilen şart sonucu **true** olduğu sürece bloklar (kaşlı ayraç içi) arasındaki kod tekrar tekrar çalışır. Bu döngü yapısında sayaç, for döngü yapısında olduğu gibi parantez içinde belirtilmez. while yapısından önce sayacın tanımlanması gerekir. Ayrıca sayacı artırma veya azaltma işlemleri de blok içindeki kodlarla birlikte yapılır. Bu nedenle de for döngüsünden ayrışır. Döngü yapısı, for döngüsüne göre daha esnek biçimde kurulur.

Yapının oluşturulması genellikle aşağıdaki şekilde olduğu gibi yapılır. Sayacı artırma veya azaltma işlemleri unutulursa çok kolay bir biçimde sonsuz bir döngüye girecek ve mantık hatasına neden olabilecek bir tasarım yapısı bulunur.

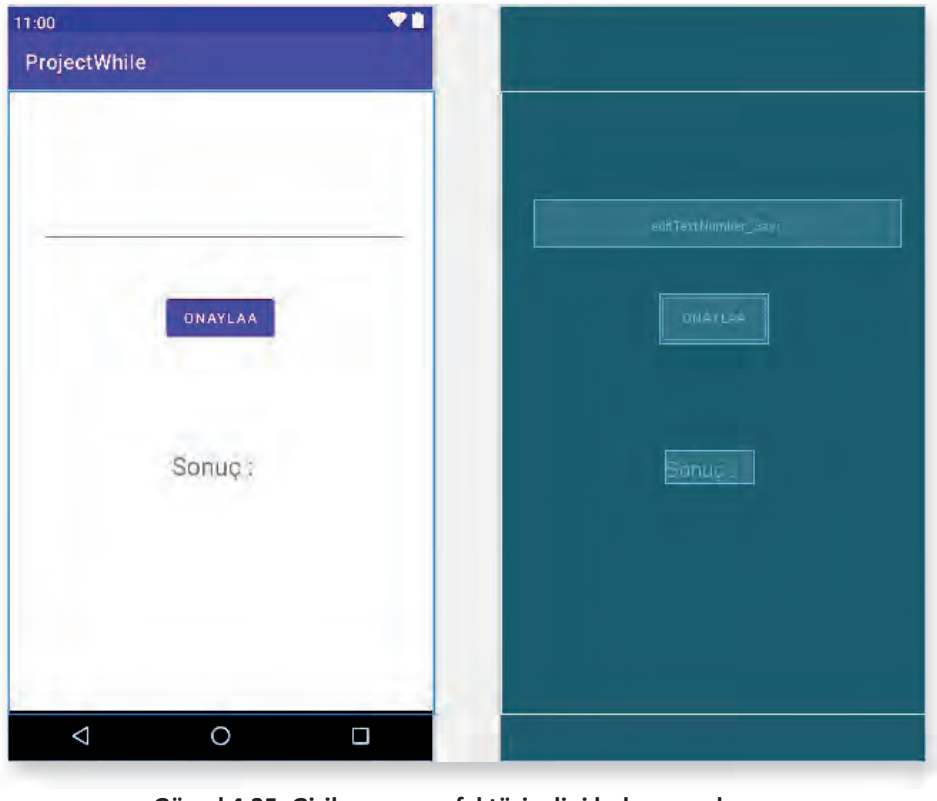
```
//sayaç tanımlaması
while(şart)
{
      //kodlar
      //Sayaç artırma veya azaltma
}
```

10. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranındaki EditTextNumber objesine girilen sayının faktöriyelini hesaplayan bir uygulama tasarlayınız.

NOT:

Bir sayının faktöriyeli, 1'den o sayıya kadar olan tüm sayıların çarpılması ile oluşur. Örneğin 5'in faktöriyeli, 1 x 2 x 3 x 4 x 5 = 120 şeklinde ifade edilebilir.

1. Adım: Görsel 4.35'teki uygulama ekranını tasarlayınız. Bu tasarım ekranında bir adet EditTextNumber objesi, bir adet Button ve bir adet TextView bulunur. EditTextNumber objesinin id'sini **editTextNumber_Sayi** şeklinde, onayla buttonunun id'sini **button_Onayla** şeklinde, TextView'in id'sini de textView_Sonuc şeklinde yapınız. EditTextNumber objesinin hint özelliğine "Sayınızı Giriniz" şeklinde yazı yazınız. Buttonun da **text** özelliğine "ONAYLA" şeklinde, TextView'in text özelliğine de "Sonuç :" şeklinde yazınız. Button tıklandığında çalışacak olan metodu aktif etmek için buttonun **onClick** özelliğine "onayla" ifadesi ekleyiniz.



Görsel 4.35: Girilen sayının faktöriyelini bulan uygulama

2. Adım: MainActivity içinde eklenen objelerin tanımlanması ve başlatılması işlemlerini yapınız. Class bloku altında TextView, EditText ve Buttonu tanımlayarak onCreate yaşam döngüsü içinde bunların başlatmasını yapınız.

```
EditText number;
Button button;
TextView textView;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    number = findViewById(R.id.editTextNumber_Sayi);
    button = findViewById(R.id.button_Onayla);
    textView = findViewById(R.id.textView_Sonuc);
}
```

3. Adım: Button tıklandığı zaman çalışacak olan onayla isimli metodu tanımlayınız. Metodun içine EditTexte girilen sayı değerini okuması için integer bir değişken tasarlayınız. Bu değişkeni de EditTextten okuttuğunuz veriyi integer veri tipine çevirerek doldurunuz.

```
public void onayla(View view) {
    int sayi = Integer.parseInt(number.getText().toString());
}
```

4. Adım: Tanımlanan **sayi** değişkeninin altında **sayac** ve **sonuc** isimli iki adet değişken tanımlayınız. Değişkenlerden sayac int veri tipinde tanımlanıp içine 1 değeri atanırken, sonuc isimli değişken long veri tipinde tanımlanarak içine 1 değeri atanır. Özellikle 10'dan sonra girilebilecek her değer oldukça büyük sonuçlar döndürür. Ayrıca çarpmada 0 yutan eleman olduğu için de sonucun ilk değeri 1 olarak girilir.

5. Adım: while döngüsünü hazırlayınız. Bu döngüyü oluşturmak için while yazıp parantezlerini açınız. Parantezin içine şart olarak "**sayac<=sayi**" şeklinde yazınız. Burada sayaç 1'den başlayarak (ilk çarpım değeri 1 olacağı için) girilen sayı değerine eşit oluncaya kadar döner. while döngüsünün blokları içinde ise "**sonuc= sonuc*sayac**" yazarak, her bir dönmede sonuç değeri ile sayaç değerini çarpıp tekrar sonucun içine atmayı sağlayınız. Bloklar içindeyken sayacı bir artırıp işleme devam ediniz.

6. Adım: while döngüsünün dışına çıkarak textView ögesinin setText metodunda sonucu göstermesi için "onayla" metodunun içinde fakat while döngüsü dışında "**textView.setText("Sonuç: "+sonuc);**" kodunu ekleyiniz.

```
public void onayla(View view) {
    int sayi = Integer.parseInt(number.getText().toString());
    int sayac=1;
    long sonuc=1;
    while(sayac<=sayi){
        sonuc=sonuc*sayac;
        sayac++;
    }
    textView.setText("Sonuç: "+sonuc);
}
```

MainActivity'nin son hâli Görsel 4.36'da yer alırken uygulamanın son çıktısı Görsel 4.37'de gösterildiği şekilde olacaktır.

```
public class MainActivity extends AppCompatActivity {
    EditText number;
    Button button;
    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        number = findViewById(R.id.editTextNumber_Sayi);
        button = findViewById(R.id.button_Onayla);
        textView = findViewById(R.id.textView_Sonuc);
    }

    public void onayla(View view) {
        int sayi = Integer.parseInt(number.getText().toString());
        int sayac=1;
        long sonuc=1;
        while(sayac<=sayi){
            sonuc=sonuc*sayac;
            sayac++;
        }
        textView.setText("Sonuç: "+sonuc);
    }
}
```

Görsel 4.36: MainActivity son hâli

Görsel 4.37: Program çıktısı

NOT:

Bu ve diğer uygulamalarda textView.setText metodunun içinde sadece int değer yazdırılmak istenirse uygulama çöker. Java yazılım dilinde bunu yapabilmenin yolu, String ifadeye integer ifadenin eklenmesidir. Bu nedenle sadece sonuç olarak gösterilmek istenirse setText parantezi içine **“”+sonuc** şeklinde belirtilmelidir.

4.2.4. do-while Döngüsü

Bu döngü yapısı aslında **while** döngüsüne çok benzer. Tek fark, while döngüsünde blok içinde yazılan kodlar şarta bağlı olarak çalıştığı için şart sağlanmadıkça döngü çalışmaz. Doğal olarak döngünün hiç çalışmama ihtimali vardır. Yapı gereği **do-while** döngüsü ise blok içine yazılan kodları en az bir defa çalıştırır ve sonrasında şarta bakar. Şart sağlandığı sürece döngü çalışmaya devam eder. Bir başka deyişle do-while döngüsü, blokları arasındaki kodları en az bir defa çalıştırırken while döngüsü, blokları arasındaki kodları şart sonucu true olmazsa çalıştırmaz.

do-while döngüsünün yapısı aşağıda gösterildiği şekildedir. while döngü yapısında olduğu gibi sayaç, do ifadesinden önce tanımlanır ve yapıya ait bloklar içinde artırma veya azaltma işlemi gerçekleştirilir. Kodlar çalıştırıldıktan sonra şart incelenir ve şart sonucu true gelirse tekrar do ifadesine dönerek işlem yeniden gerçekleştirilir.

```
do
{
    //kodlar
}while(şart);
```

NOT:

do-while döngü yapısında diğer yapılardan farklı olarak “;” işareti kullanılır. while ifadesinin şart sonu “;” işaretiyle ile bitirilir.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Şart ifadesi "true" olarak sağlandığı zaman if içindeki kodlar çalıştırılır.

2. ( ) <,>,==,<=,>=,!= karakterleri aritmetiksel operatörlerdir.

3. ( ) Maksimum dört adet for döngüsü iç içe yazılır.

4. ( ) while döngüsünde yazılan şart ifadesi şartı sağlamasa da döngü en az bir defa çalıştırılır.

5. ( ) Karar ifadelerinde şart yazılırken "||" operatörü kullanılırsa şartlardan birinin "true" olması yeterlidir.

6. ( ) Java dilindeki String veri tipinde karşılaştırma yapılırken "==" ifadesi ile karşılaştırmak yeterlidir.

B) Aşağıdaki kodlar çalıştırıldığında elde edilecek textView değerlerini ilgili başlıkta verilen kutucuğa yazınız.

7. Aşağıdaki kodlar çalıştırıldığında textView ögesine eklenecek değerler nelerdir?

```
textView.setText("");
for(int i=0;i<50;i++){
    if(i==18 || i==27){
        continue;
    }
    if(i%3==0){
        textView.setText(textView.getText()+" "+ i);
    }
}
```

textView Değerleri

8. Aşağıdaki kodlar çalıştırıldığında textView ögesine eklenecek değerler nelerdir?

```
int sayac = 0,sonuc=1;
while (sayac<7)
{
    sayac = sayac+1;
    sonuc = sonuc * sayac;
    textView.setText(textView.getText() + " " +sonuc);
}
```

textView Değerleri

C) Aşağıdaki cümlelerde oluşturulması istenen kodları boş bırakılan yerlere yazınız.

9. Eğer **sayi1** değişkeni **sayi2** değişkeninden büyük eşit ve **sayi3** değişkeninden de küçükse karar ifadesini Java diline uygun kod olarak oluşturunuz.

10. Daha öncesinde tanımlanan ve içinde String veri tutan **editText_Adi** ögesindeki değer, adiSoyadi içindeki değere eşit ise karar ifadesini Java diline uygun kod olarak oluşturunuz.

11. Daha öncesinde tanımlanan ve içinde 10 değeri tutan **sayac** değişkeni vardır. 0'dan sayac değişkeninin içindeki değere kadar olan tam sayıları Logcat ekranına yazdıran programın kodlarını Java diline uygun bir biçimde oluşturunuz.

D) Aşağıdaki cümlelerde tasarlanması istenen kodları boş bırakılan yerlere yazınız.

12. Üç farklı EditTextten üç not girişi yapıldığında en yüksek notu textView ögesinde gösteren uygulamanın karar kodlarını tasarlayınız.

13. EditText ögesine gün bilgisi yazı ile girildiğinde her gün için farklı bir günaydın mesajı tanımlayan programın kodlarını switch-case kullanarak tasarlayınız.

5. ÖĞRENME BİRİMİ

GELİŞMİŞ KOMUTLAR

KONULAR

5.1. METOT

5.2. SINIF VE NESNE KAVRAMLARI

5.3. KAPSÜLLEME (ENCAPSULATION)

5.4. KALITIM (INHERITANCE)

5.5. ÇOKBİÇİMLİLİK (POLYMORPHISM)

5.6. DİZİLER

NELER ÖĞRENECEKSİNİZ?

- Metot yapıları
- Metotlara parametre gönderme
- Metotlar için aşırı yükleme
- Sınıf yapısı
- Nesne oluşturma
- Sınıfların erişim düzeyleri
- Kapsülleme yapısı
- Getter ve Setter yapısı
- Kalıtım yapısı
- Alt ve üst sınıf yapısı
- Çokbiçimlilik yapısı
- Çokbiçimlilik ve kalıtım arasındaki bağlantı
- Dizi yapısı
- Dizileri ile ilgili işlemler

ANAHTAR KELİMELER

- Alt sınıf
- ArrayList
- Aşırı yükleme
- Çokbiçimlilik
- Dizi
- DRY
- Getter
- Kalıtım
- Kapsülleme
- Metot
- Nesne
- Parametre
- Private
- Protected
- Public
- Setter
- Sınıf
- Üst sınıf
- Yapıcı metotlar

HAZIRLIK ÇALIŞMALARI

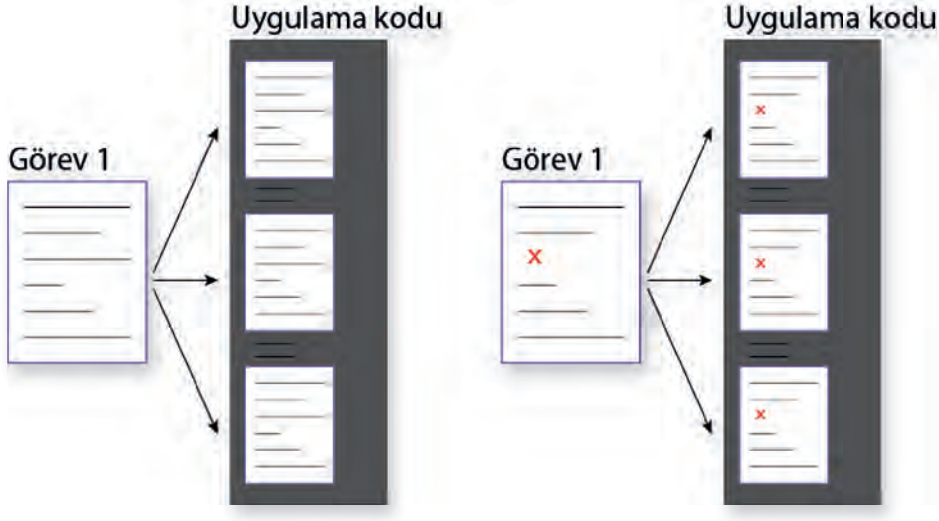
1. Mobil uygulamada ve günlük hayatta karşılaşılan problemlerin çözümündeki ortak noktalar neler olabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Uygulama geliştirilirken neden farklı yöntemler vardır? Arkadaşlarınızla değerlendiriniz.

5.1. METOT

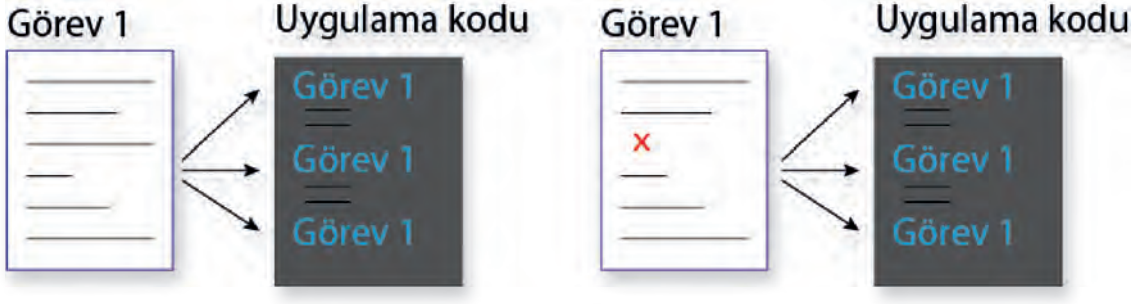
Programlama dünyasında birçok avantaja sahip yazılım prensipleri vardır. Bu ilkelerden en çok bilineni DRY (Don't Repeat Yourself) "kendini tekrar etme" prensibidir. DRY prensibinin amacı, kod tekrarını önlemektir.

Aynı görevleri gerçekleştiren kodlar birden çok yerde yazıldığında bu görevdeki değişiklik, kodların tamamında değişiklik yapılması anlamına gelir (Görsel 5.1a). Bu durum, daha uzun kod satırlarına, daha uzun süreye ve hata yapma ihtimalinin artmasına neden olur.

Aynı görevleri gerçekleştiren kodların bir blok hâline getirilmesi ve buna uygulama içinde referans verilmesiyle DRY prensibi uygulanır (Görsel 5.1b). Bu kod bloku içinde herhangi bir değişiklik yapılması durumunda uygulama kodunda düzenlemeye gerek kalmaz.



Görsel 5.1a: DRY prensibi uygulanmadan kod yazımı



Görsel 5.1b: DRY prensibi uygulanmış kod yazımı

Aynı kod satırlarının tekrar tekrar yazılmasına gerek kalmadan çağrılabileceği kullanışlı kod parçalarına **metot** denir. Metot, belirli bir görevi yerine getiren bağımsız kod bloku olarak da ifade edilir. Metotlar, birçok kod satırı içerebilen uygulamadaki büyük ve karmaşık hesapları daha yönetilebilir parçalara böler.

5.1.1. Metot Yapısı

Bir metot genellikle altı bölümden oluşur (Görsel 5.2).

- **Erişim Belirleyici (Düzeyleri):** Metoda uygulama içinden nerelerden erişebileceğini ifade eder.
- **Geri Dönüş Tipi:** Metodun geriye döndüreceği tip değeridir. Geri dönüş tipi olarak void belirlenmişse geriye bir değer döndürmez.
- **Metot Adı:** Metoda verilen isimdir. Java isimlendirme standartlarına göre Camel Case şeklinde yazılır.
- **Parametre Listesi:** Metoda gönderilecek değerleri ifade eder. Birden fazla değer gönderilecek ise aralarına virgül kullanılır. Metoda değer gönderilmeyecek ise parantez içine herhangi bir şey yazılmaz.
- **Metot Gövdesi:** Küme parantezleri arasındaki kısımdır. Bu bölüme metodun amaçladığı işi gerçekleştirilecek kodlar yazılır.
- **Geri Dönüş Değeri:** Metot işlevini yerine getirdikten sonra çağrıldığı yere geri döndüreceği değerin yazıldığı kısımdır. Geri dönüş değerinden önce return komutu yazılır. Metot herhangi bir değer geri döndürmüyor ise return komutunun kullanılması zorunlu değildir.

```
Erişim_belirleyici Geri_dönüş_tipi metot_adı ( Parametre_listesi) {
//Metot Gövdesi
//return Metot geri dönüş değeri
}
```

Görsel 5.2: Metot yapısı

5.1.2. Değer Döndürmeyen Metotlar

Bir metot, yaptığı işlem sonucunda herhangi bir değer geri döndürmüyor ise bu metoda değer döndürmeyen metot denir. Bir metodu değer döndürmeyen metot şeklinde tanımlamak için geri dönüş tipi olarak void yazılır.

ÖRNEK

```
private void selamVer() {
     Toast.makeText(getApplicationContext(),"Merhaba",Toast.LENGTH_LONG).show();
}
```

1. UYGULAMA: İşlem adımlarına göre Buttonlara tıklandığında Toast mesaja "Merhaba" yazan bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: activity_main.xml içine şu kodu yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="10px"
        android:text="Selam Ver 1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="10px"
        android:text="Selam Ver 2" />
</LinearLayout>
```

3. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.metotkavrami;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn1 = findViewById(R.id.button1);
        btn1.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View view) {
                selamVer();
            }
        });
        Button btn2 = findViewById(R.id.button2);
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                selamVer();
            }
        });
    }
    private void selamVer() {
        Toast.makeText(getApplicationContext(),
"Merhaba",Toast.LENGTH_LONG).show();
    }
}
```

4. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

5. Adım: SELAM VER düğmesine tıklayınız.

SIRA SİZDE: Bir Buttondan oluşan bir uygulama tasarlayınız. Buttona tıklandığında iki sayıyı toplayan kodu metot kullanarak yazınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Uygulama tasarım ekranına bir Button ekledi.		
5. MainActivity.java dosyasında Button tıklama olayını yazdı.		
6. Button tıklama olayında "topla" metodunu çağırdı.		
7. Metot adı "topla" olacak şekilde bir metot oluşturdu.		
8. Toplama kodlarını doğru şekilde yazdı.		
9. Toplama sonucunu Toast mesajına doğru olarak yazdı.		
10. Run düğmesiyle uygulamayı çalıştırdı.		

5.1.3. Değer Döndüren Metotlar

Bazı metotların yaptığı işlem sonucunda çağrıldığı yere değer döndürmesi gerekir. Metot, geri döndüreceği değerin türüne göre tanımlanmalıdır. Değer geri döndürmek için metot içinde **return** ifadesi kullanılır.

ÖRNEK

```
private int topla() {
   return 5+10;
}
```

2. UYGULAMA: İşlem adımlarına göre butona tıklandığında Toast mesaja iki sayının toplamını yazan bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında iki EditText ve bir Button oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <EditText
        android:id="@+id/editText1"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="5"
        android:inputType="number"
        />
    <EditText
        android:id="@+id/editText2"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="10"
        android:inputType="number"/>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Topla" />
</LinearLayout>
```

3. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.metotkavrami2;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn1 = findViewById(R.id.button1);
        EditText editText1 = findViewById(R.id.editText1);
        EditText editText2 = findViewById(R.id.editText2);
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int sonuc = topla();
    Toast.makeText(getApplicationContext(),
                 Integer.toString(sonuc),
                 Toast.LENGTH_LONG).show();
            }
        });
    }
    private int topla() {
        return 5 + 10;
    }
}
```

4. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

5. Adım: TOPLA düğmesine tıklayınız.

5.1.4. Parametre Alan Metotlar

Bazı metotların işlem yapabilmesi için metot dışından bilgi alması gerekir. Metodun aldığı bu bilgilere parametre denir. Parametre, metot isminden sonra parantez içine bir değişken tanımlama gibi yazılır. Birden fazla parametre varsa parametreler arasına virgül kullanılır.

ÖRNEK

```
private int topla(int sayi1, int sayi2) {
    return sayi1 + sayi2;
}
```

Metot çağrılırken parametreler girilmelidir.

ÖRNEK

```
topla(5,10);
```

3. UYGULAMA: İşlem adımlarına göre iki sayının toplamını parametre kullanan metot ile gerçekleştiren bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında iki EditText ve bir Button oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <EditText
        android:id="@+id/editText1"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="5"
        android:inputType="number"
        />
    <EditText
        android:id="@+id/editText2"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="10"
        android:inputType="number"/>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Topla" />
</LinearLayout>
```

3. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.metotkavrami3;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn1 = findViewById(R.id.button1);
        EditText editText1 = findViewById(R.id.editText1);
        EditText editText2 = findViewById(R.id.editText2);
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int sayi1;
                sayi1 = Integer.parseInt(editText1.getText().toString());
                int sayi2;
                sayi2 = Integer.parseInt(editText2.getText().toString());
                int sonuc = topla(sayi1,sayi2);
                Toast.makeText(getApplicationContext(),
Integer.toString(sonuc),Toast.LENGTH_LONG).show();
            }
        });
    }
    private int topla(int sayi1, int sayi2) {
        return sayi1 + sayi2;
    }
}
```

4. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

5. Adım: TOPLA düğmesine tıklayınız.

SIRA SİZDE: Dört Button ve iki EditTextten oluşan bir uygulama tasarlayınız. Düğmelere tıklandığında toplama, çıkarma, çarpma ve bölme işlemini gerçekleştiren kodu parametre alan metotlar ile oluşturunuz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Uygulama tasarım ekranına dört Button ekledi.		
5. Uygulama tasarım ekranına iki EditText ekledi.		
6. MainActivity.java dosyasında Button tıklama olaylarını yazdı.		
7. Metin kutularındaki sayıları tip dönüşümü yaparak birer değişkene alan kodları yazdı.		
8. Parametreleri alacak şekilde toplama metodunu yazdı.		
9. Parametreleri alacak şekilde çıkarma metodunu yazdı.		
10. Parametreleri alacak şekilde çarpma metodunu yazdı.		
11. Parametreleri alacak şekilde bölme metodunu yazdı.		
12. Toplama kodlarını doğru şekilde yazdı.		
13. Çıkarma kodlarını doğru şekilde yazdı.		
14. Çarpma kodlarını doğru şekilde yazdı.		
15. Bölme kodlarını doğru şekilde yazdı.		
16. Sonucu Toast mesajına doğru olarak yazdı.		
17. Run düğmesine tıklayarak uygulamayı çalıştırdı.		

5.1.5. Metotlarda Aşırı Yükleme

Uygulama geliştirme standartları açısından metot adları önemlidir. Metotlar, yapacakları işlemlere göre isimlendirilmelidir. Bu standart, kod okunabilirliğini artırır. Örneğin topla adlı bir metodun yapacağı işin toplama olacağı tahmin edilebilir fakat bu metodun ismi islemYap şeklinde yazılırsa metodun yapacağı işin tahmin edilmesi güçtür. Bu nedenle metodun yapacağı işi anlamak için metodun koduna bakılmalıdır.

Benzer işlemleri yapan metotlara isim verilirken zorluklar yaşanabilir. Örneğin iki sayının toplamını yapan metoda topla ismi verildiğinde üç sayının toplamını bulan başka bir metoda isim verilirken zorluk yaşanır. Bu uygulamada metotlar daha fazla sayı toplayan şeklinde düşünüldüğünde kodlama işlemi güçleşir. Bu tür problemleri çözmek için **Metot Aşırı Yükleme (Method Overloading)** kavramı devreye girer. Metot aşırı yükleme, aynı isme sahip metot tanımlamaya izin verir. Metotları birbirinden ayırmak için parametre sayısı ve tipleri kullanılır.

ÖRNEK

```
private int topla(int sayi1, int sayi2) {
    return sayi1 + sayi2;
}
```

```
topla(5, 10);
```

`private int topla(int sayi1, int sayi2, int sayi3) {`
`    return sayi1 + sayi2 + sayi3;`
`}`
	`topla(5, 10, 15);`

`private float topla(float sayi1, float sayi2) {`
`    return sayi1 + sayi2;`
`}`
	`topla(2.3, 1.7);`

4. UYGULAMA: İşlem adımlarına göre iki ve üç sayının toplamını metot aşırı yükleme ile gerçekleştiren bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında üç EditText ve iki Button oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <EditText
        android:id="@+id/editText1"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="5"
        android:inputType="number"/>
    <EditText
        android:id="@+id/editText2"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="10"
        android:inputType="number"/>
    <EditText
        android:id="@+id/editText3"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:text="10"
        android:inputType="number"/>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="2 Sayıyı Topla" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="3 Sayıyı Topla" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Sonuç:"/>
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

3. Adım: MainActivity.java dosyasına şu kodu yazınız:

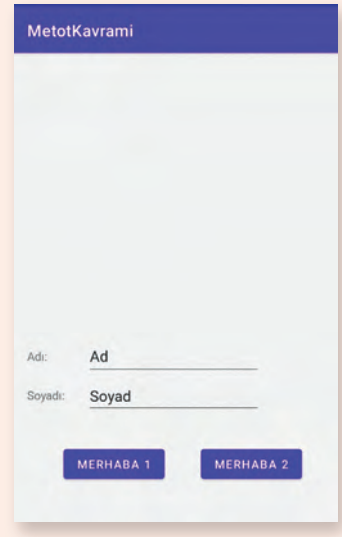
```
package com.example.metotkavrami3;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn1 = findViewById(R.id.button1);
        EditText editText1 = findViewById(R.id.editText1);
        EditText editText2 = findViewById(R.id.editText2);
        EditText editText3 = findViewById(R.id.editText3);
        TextView textView = findViewById(R.id.textView);
        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int sayi1;
                sayi1 = Integer.parseInt(editText1.getText().toString());
                int sayi2;
                sayi2 = Integer.parseInt(editText2.getText().toString());
                int sonuc = topla(sayi1,sayi2);
                textView.setText(Integer.toString(sonuc));
            }
        });
```

```
        Button btn2 = findViewById(R.id.button2);
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int sayi1;
                sayi1 = Integer.parseInt(editText1.getText().toString());
                int sayi2;
                sayi2 = Integer.parseInt(editText2.getText().toString());
                int sayi3;
                sayi3 = Integer.parseInt(editText3.getText().toString());
                int sonuc = topla(sayi1,sayi2,sayi3);
                textView.setText(Integer.toString(sonuc));
            }
        });
    }

    private int topla(int sayi1, int sayi2) {
        return sayi1 + sayi2;
    }
    private int topla(int sayi1, int sayi2,int sayi3) {
        return sayi1 + sayi2 + sayi3;
    }
}
```

4. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

5. Adım: Düğmelere tıklayınız.

SIRA SİZDE: İki Button ve iki EditTextten oluşan bir uygulama tasarlayınız (Görsel 5.3). Birinci düğmeye tıklandığında "Merhaba Ad", ikinci düğmeye tıklandığında "İyi günler Ad Soyad" şeklinde mesaj veren kodları metot aşırı yükleme ile oluşturunuz.



Görsel 5.3: Uygulama tasarımı

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

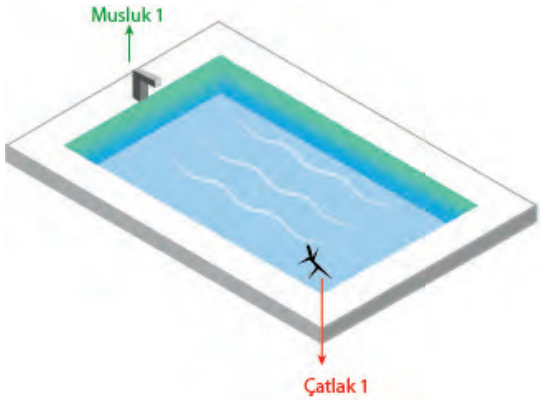
KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Uygulama tasarım ekranına iki Button ekledi.		
5. Uygulama tasarım ekranına iki EditText ekledi.		
6. Uygulama kodunda findViewById yöntemiyle button1'i tanımladı.		
7. Uygulama kodunda findViewById yöntemiyle button2'yi tanımladı.		
8. button1'e tıklanma olayını ekledi.		
9. editTextAd içindeki yazıyı bir değişkene atadı.		
10. editTextSoyad içindeki yazıyı bir değişkene atadı.		
11. Tek parametreli selamVer metodunu oluşturdu.		
12. İki parametreli selamVer metodunu oluşturdu.		
13. Toast mesaj kodunu yazdı.		
14. Run düğmesine tıklayarak uygulamayı çalıştırdı.		

5.2. SINIF VE NESNE KAVRAMLARI

Java programlama dili, daha güçlü bir yapı sağlamak amacıyla prosedürel ve nesne yönelimli kavramlarını bir arada sunar. **Prosedürel programlama**, geleneksel programlama yöntemidir ve programın yukarıdan aşağıya doğru sırasıyla komutları çalıştırması anlamına gelir. Program, prosedürel programlamada bir bütün hâlinde yazılır. Program büyüdükçe bu bütünlüğü sağlamak zorlaşır. Program yazımı bittikten sonra program üzerinde yapılacak bir değişiklik, programın bütünlüğüne zarar verebilir. Uygulamanın yapacağı işlemler karmaşıklaştıkça geleneksel programlama yöntemi uygulama geliştirmeyi zorlaştırır.

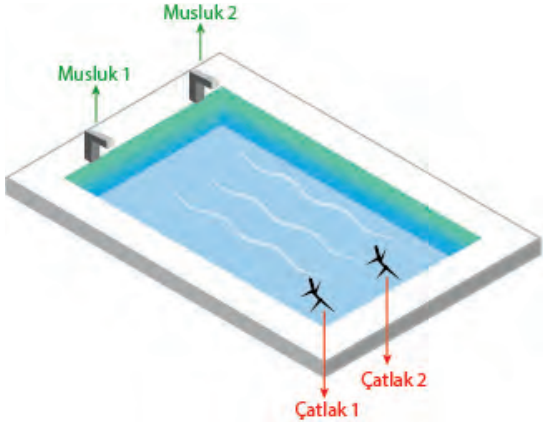
Program yazma işleminin ilk adımı, biri problemi analiz etmektir. Karmaşık problemleri geleneksel yöntemlerle analiz etmek zordur. Geleneksel yöntemin yerine problemi günlük hayattaki nesnelere benzeten nesne tabanlı programlama önerilir. **Nesne Tabanlı Programlama (Object Oriented Programming)**, problemi bütün olarak değil de parçalar hâlinde ele alır. Problemin her bir parçası nesne olarak ifade edilir. Birçok nesne birbiriyle etkileşime geçerek bütünü oluşturur. Bu durum, problemi daha iyi analiz etmeye ve daha hızlı çözüm üretmeye yarar. Örneğin bir havuzu su ile doldurma probleminde havuzu doldurmak için bir musluk ve havuzun dibinde suyu dışarı sızdıran bir çatlak olduğu varsayılırsa havuzun büyüklüğüne, musluktan ve çatlaktan akan suyun hızlarına göre bir matematik formülü ile bu havuzun kaç saate dolacağı bulunabilir (Görsel 5.4).




```
int musluk1=4;
int catlak1=5;
float birimDolum = 1/musluk1-1/catlak1;
float dolumSaati = 1/birimDolum;
```

Görsel 5.4: Basit havuz problemi

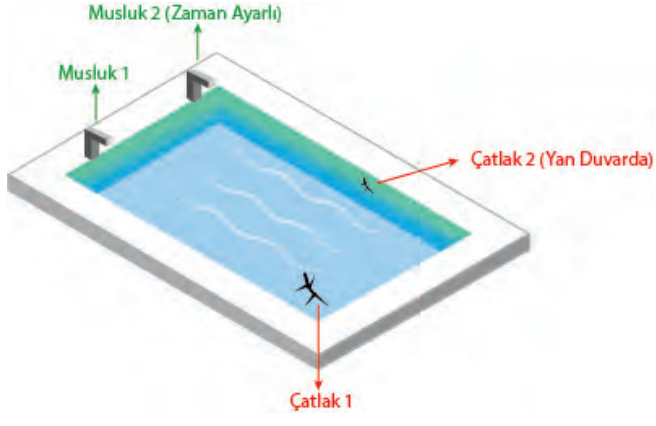
Problem daha karmaşık hâle getirildiğinde, musluk ve çatlak sayıları artırıldığında elde edilecek formül de karmaşıklaşır (Görsel 5.5).




```
int musluk1=4;
int catlak1=5;
int musluk2=3;
int catlak2=6;
float birimDolum = 1/musluk1+1/musluk2
-1/catlak1-1/catlak2;
float dolumSaati = 1/birimDolum;
```

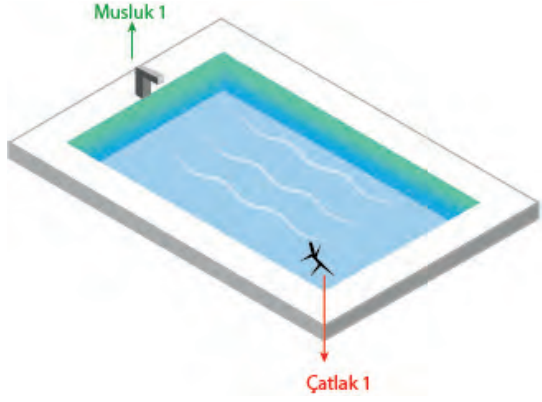
Görsel 5.5: Karmaşık havuz problemi

Musluklardan birinin zaman ayarlı olduğu, çatlaklardan birinin havuzun zemininde değil de yan duvarda olduğu varsayılırsa problem çözümü için izlenecek yöntem de değişir. Havuzdaki ikinci çatlaktan hemen su sızmaz. Havuzun su seviyesi bu çatlağa varıncaya kadar çatlağın havuza bir etkisi olmaz. Problem karmaşıklaştıkça problemin çözümü daha da zorlaşır (Görsel 5.6).

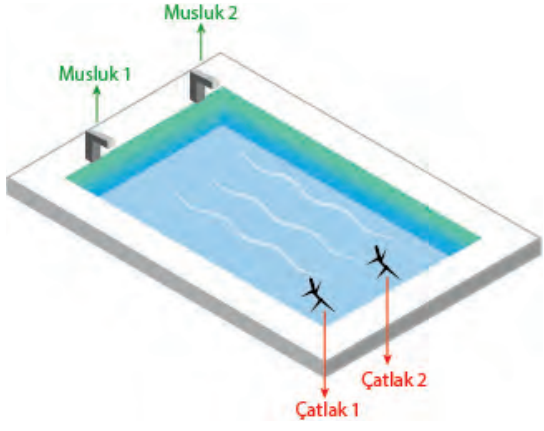


Görsel 5.6: Çözümü zor havuz problemi

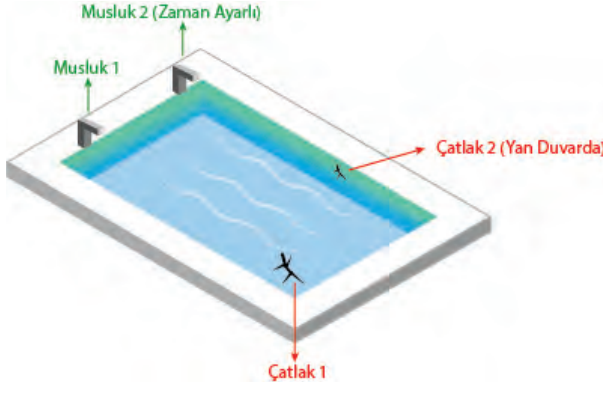
Problem çözümünde musluk ve çatlak birer nesne olarak düşünüldüğünde havuzu modellemek basitleşir. Havuz, musluk ve çatlak için birer sınıf modellemek, gerçek hayata benzetildiği için işlemleri kolaylaştırır. Problemin çözümünde nesne kullanımı, geleneksel programlamaya göre daha fazla kod yazılmasına karşın programın yönetilebilirliğini artırır. Problemin karmaşıklığı arttıkça sadece etkilenecek sınıflar üzerinde düzenlemeler yapılması, programın yönetilmesini kolaylaştırır (Görsel 5.7).




```
public class Havuz  {….}
public class Musluk{.…}
public class Catlak  {….}
Havuz havuz = new Havuz();
Musluk musluk1 = new Musluk(4);
catlak catlak1 = new Catlak(5);
havuz.ekle(musluk1,catlak1);
float dolumSaati = havuz.dolumSaati();
```




```
public class Havuz  {….}
public class Musluk{.…}
public class Catlak  {….}
Havuz havuz = new Havuz();
Musluk musluk1 = new Musluk(4);
catlak catlak1 = new Catlak(5);
Musluk musluk2 = new Musluk(3);
catlak catlak2 = new Catlak(6);
havuz.ekle(musluk1,catlak1,musluk2,catlak2);
float dolumSaati = havuz.dolumSaati();
```




```
public class Havuz  {….}
public class Musluk{.…}
public class Catlak  {….}
Havuz havuz = new Havuz();
Musluk musluk1 = new Musluk(4);
catlak catlak1 = new Catlak(5);
Musluk musluk2 = new Musluk(3,2);
//musluk2 2 saat sonra açılacak
Catlak catlak2 = new Catlak(6,4);
//catlak2 havuzun dibine olan mesafesi
havuzun yüksekliğinin 1/4'ü kadar
havuz.ekle(musluk1,catlak1,musluk2,catlak2);
float dolumSaati = havuz.dolumSaati();
```

Görsel 5.7: Nesne tabanlı programlama bakış açısıyla havuz problemi

5.2.1. Sınıf (Class)

Sınıf, nesneleri oluşturmak için kullanılan şablonlar veya prototiplerdir. Bu, bir evi yapmak için kullanılan ev planına benzetilebilir. Bu plana bakılarak evler inşa edilir. Üretilen evlerin genel görünüşü aynı olmakla birlikte renkleri farklılık gösterebilir (Görsel 5.8).

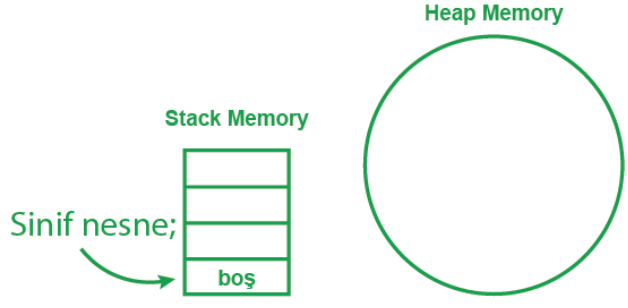
Görsel 5.8: Ev planından ev üretimi

Sınıflar, kurabiye kalıpları gibi de düşünülebilir. Kurabiye kalıbından çıkan şekiller ana hatlarıyla birbirlerine benzer fakat hamur içeriği ve kullanılan süsleme gibi özellikleri farklılık gösterebilir. Kurabiye kalıbı, sınıfı temsil ederken kalıptan türetilen kurabiyeler ise nesneleri ifade eder (Görsel 5.9).

Görsel 5.9: Kurabiye adam kalıbı ile kurabiye üretme

5.2.2. Nesne (Object) Oluşturma

Programlamada sadece sınıf oluşturmak yeterli değildir. Sınıf tek başına kullanılamaz. Sınıflar prototip olduğu için sınıflar üzerinde işlem yapmak mümkün değildir. Programda sınıflardan üretilecek nesnelere ihtiyaç duyulur. Nesneler birer varlık olarak ifade edildiği için nesneler üzerinde işlem yapmak mümkündür. Nesneler tanımlanırken ilkel değişkenler gibi tanımlama yapılır. Bu tanımlama sonucu yeni bir nesne oluşmaz. Nesne adı için hafızada yer ayrılır fakat bu hafıza yerinde herhangi bir değer bulunmaz (Görsel 5.10).

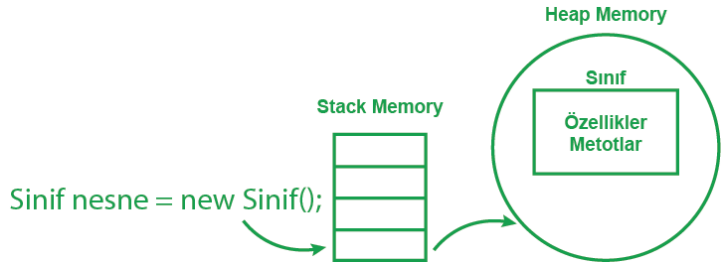


Görsel 5.10: Nesne tanımlamada hafıza durumu

Nesne tanımlama:

SinifAdi nesneAdi;

Yeni bir nesne tanımlamak için **"new"** anahtar kelimesi kullanılır. New anahtar kelimesi ile sınıftan bir nesne türetilir (Görsel 5.11).



Görsel 5.11: Nesne oluşturmada hafıza durumu

Nesne oluşturma:

SinifAdi nesneAdi = new SinifAdi();

5.2.3. Sınıf Yapısı

Sınıflar dört bölümden oluşur (Görsel 5.12).

- **Erişim Belirleyici:** Sınıfa nereden ulaşılabileceğini ifade eder.
- **Class Anahtar Kelimesi:** Oluşturulan kodun sınıf olduğunu ifade etmek için class anahtar kelimesi kullanılır.
- **Sınıf Adı:** Sınıfa verilen isimdir. İsimlendirme standartlarına göre Pascal Case şeklinde isimlendirme yapılır.
- **Sınıf Gövdesi:** Sınıfın özellik ve metotlarının bulunduğu kısımdır.

```
Erişim_belirleyici class SınıfAdı {
//Sınıf Gövdesi
}
```

Görsel 5.12: Sınıf yapısı

5. UYGULAMA: İşlem adımlarına göre sınıf oluşturarak bir elektrikli cihazın aylık tüketimini bulan bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

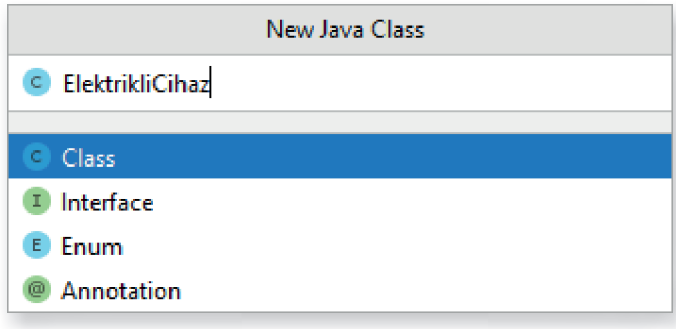
2. Adım: Görsel 5.13'te verilen uygulama ekranını oluşturan şu kodu activity_main.xml içine yazınız:

Görsel 5.13: Uygulama ekran tasarımı

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" >
    <EditText
        android:id="@+id/editTextCihazAdi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Cihaz Adı"
        android:inputType="textPersonName"
        android:minHeight="48dp" />
    <EditText
        android:id="@+id/editTextKw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Cihaz Gücü (KW)"
        android:inputType="number"
        android:minHeight="48dp" />
    <EditText
        android:id="@+id/editTextGunlukKullanimSaat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Günlük kullanılan saat"
        android:inputType="number"
        android:minHeight="48dp" />
```

```
    <Button
        android:id="@+id/buttonHesapla"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hesapla"/>
    <TextView
        android:id="@+id/textViewSonuc"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=" "/>
</LinearLayout>
```

3. Adım: File>New>Java Class menü komutu ile "ElektrikliCihaz" adıyla yeni bir sınıf oluşturunuz (Görsel 5.14).



Görsel 5.14: Yeni sınıf oluşturma

4. Adım: ElektrikliCihaz.java dosyasına şu kodu yazınız:

```
package com.example.elektriktuketimi;
public class ElektrikliCihaz {
    public String cihazAdi;
    public int cihazKW;
    public int gunlukSaatKullanim;
    public int aylikTuketim() {
        int aylikKullanim;
        aylikKullanim = cihazKW*gunlukSaatKullanim*30;
        return aylikKullanim;
    }
}
```

5. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.elektriktuketimi;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnHesapla = findViewById(R.id.buttonHesapla);
        btnHesapla.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                EditText editTextCihazAdi = findViewById(R.id.editTextCihazAdi);
                EditText editTextKW = findViewById(R.id.editTextKw);
                EditText editTextGunlukKullanimSaat = findViewById(R.id.edit-
TextGunlukKullanimSaat);
                TextView textViewSonuc = findViewById(R.id.textViewSonuc);
                String cihazAdi = editTextCihazAdi.getText().toString();
                int kw = Integer.parseInt(editTextKW.getText().toString());
                int saat = Integer.parseInt(editTextGunlukKullanimSaat.get-
Text().toString());
                ElektrikliCihaz cihaz1 = new ElektrikliCihaz();
                cihaz1.cihazAdi = cihazAdi;
                cihaz1.cihazKW = kw;
                cihaz1.gunlukSaatKullanim=saat;
                int sonuc = cihaz1.aylikTuketim();
                textViewSonuc.setText(Integer.toString(sonuc));
            }
        });
    }
}
```

6. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

7. Adım: Değerleri girerek HESAPLA düğmesine tıklayınız.

SIRA SİZDE: "ElektrikliCihaz" sınıfını düzenleyerek elektrik tüketim bedelini hesaplayan uygulamayı oluşturunuz. Elektrik tüketim bedeli 1 kWh için 1,37 TL'dir.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Uygulama tasarım ekranına üç EditText ekledi.		
5. Uygulama tasarım ekranına bir Button ekledi.		
6. Uygulama tasarım ekranına bir TextView ekledi.		
7. "ElektrikliCihaz" adında yeni bir sınıf oluşturdu.		
8. "ElektrikliCihaz" sınıfının özelliklerini belirledi.		
9. "ElektrikliCihaz" sınıfına "tuketimBedeliHesapla" metodunu yazdı.		
10. Uygulama kodunda findViewById yöntemiyle nesneleri tanımladı.		
11. Button nesnesine tıklanma olayını ekledi.		
12. Nesnelerdeki yazıları değişkenlere atadı.		
13. "ElektrikliCihaz" sınıfından cihaz1 adlı yeni bir nesne türetti.		
14. cihaz1 nesnesinin özelliklerine değer atadı.		
15. cihaz1 nesnesinin tuketimBedeliHesapla metodunu çağırdı.		
16. Tüketim bedelini TextView içinde gösterdi.		
17. Run düğmesine tıklayarak uygulamayı çalıştırdı.		

5.2.4. Erişim Belirleyiciler (Erişim Düzeyleri)

Java programlama dilinde erişim belirleyiciler; bir sınıfın, özelliğin veya metodun erişilebilirliğini veya kapsamını belirtir. Dört tür Java erişim belirleyici vardır (Tablo 5.1).

1. **Private (Özel):** Erişim düzeyi sınıf düzeyindedir. Sınıf dışından erişilemez.
2. **Default (Varsayılan):** Varsayılan erişim düzeyi paketle sınırlıdır. Paket dışından erişilemez. Herhangi bir erişim belirleyici kullanılmazsa bu varsayılan erişim düzeyidir.
3. **Protected (Korumalı):** Bir alt sınıf aracılığıyla hem paket içinden hem de paket dışından erişilir.
4. **Public (Genel):** Her yerden ulaşılabilen erişim belirleyicisidir.

Tablo 5.1: Erişim Belirleyiciler

Erişim Belirleyici	Sınıf İçinde	Paket İçinde	Alt Sınıfla Aynı Paket İçinde	Alt Sınıfla Aynı Paket Dışında	Genel
Public	Evet	Evet	Evet	Evet	Evet
Protected	Evet	Evet	Evet	Evet	Hayır
Default	Evet	Evet	Evet	Hayır	Hayır
Private	Evet	Hayır	Hayır	Hayır	Hayır

5.2.4.1. Private (Özel) Erişim Belirleyici

Özel erişim belirleyicisine yalnızca sınıfın üyeleri ulaşabilir.

ÖRNEK

A sınıfı içinde private olarak değer özelliği ve değiştir metodu tanımlanmıştır. Aynı sınıf içinde değiştir metodu private özelliğe ulaşılabilir. B sınıfı içinden private özellik veya metoda ulaşılmak istendiğinde uygulama hata verir.

```
public class A {
    private int deger = 10;
    private void degistir(){
        deger=15;
    }
}
```

```
public class B {
    A a = new A();
    a.deger = 12; //Hata verir.
    a.degistir(); //Hata verir.

}
```

5.2.4.2. Default (Genel) Erişim Belirleyici

Herhangi bir erişim belirleyici kullanılmadığında default belirleyici tanımlanır.

ÖRNEK

Paket1 içindeki değer özelliği ve değiştir metodu default olarak tanımlanmıştır. Paket2 içinden bu özellik veya metoda ulaşmak mümkün değildir. Bu özellik veya metoda erişilmeye çalışıldığında uygulama hata verir.

```
package paket1;
public class A{
    int deger = 10;
    void degistir(){
        deger=15;
    }
}
```

```
package paket2;
import paket1.*;
public class B{
    A a = new A();
    a.deger = 12; //Hata verir.
    a.degistir(); //Hata verir.
}
```

5.2.4.3. Protected (Korumalı) Erişim Belirleyici

Paket dışından özellik ve metotlara erişim bir alt sınıf aracılığıyla mümkündür. Alt sınıf tanımlamadan özellik ve metotlara paket dışından erişilemez.

ÖRNEK

Paket2 içindeki B sınıfı, A sınıfının alt sınıfıdır. Extends anahtar kelimesiyle B sınıfına A sınıfının alt sınıfı olduğu bilgisi verilmiştir. B sınıfı farklı paket içinde olmasına rağmen protected olan özellik veya metoda erişebilmiştir. Oysa C sınıfı protected özellik veya metoda erişmeye çalışınca uygulama hata verir.

```
package paket1;
public class A{
    protected int deger = 10;
    protected void degistir(){
        deger=15;
    }
}
```

```
package paket2;
import paket1.*;
public class B extends A{
    A a = new A();
    a.deger = 12;
    a.degistir();
}
public class C{
    A a = new A();
    a.deger = 12; //Hata verir.
    a.degistir(); //Hata verir.
}
```

5.2.4.4. Public (Genel) Erişim Belirleyici

Public erişim belirleyicisine her yerden ulaşmak mümkündür. Herhangi bir kısıtlaması yoktur.

ÖRNEK

Paket2 içindeki B sınıfı, paket1 içindeki A sınıfının public özelliklerine veya metotlarına erişebilir.

```
package paket1;
public class A{
    public int deger = 10;
    public void degistir(){
        deger=15;
    }
}
```

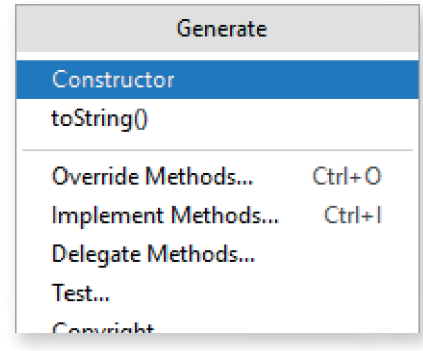
```
package paket2;
import paket1.*;
public class B {
    A a = new A();
    a.deger = 12;
    a.degistir();
}
```

5.2.5. Kurucu veya Yapıcı Metotlar (Constructors)

Yeni bir nesne oluşturulduğunda otomatik olarak çalıştırılan metotlardır. Genellikle nesne oluşturulduğunda başlangıç ayarlarının yapılması için kullanılır. Kurucu metotlar, ait olduğu sınıf ile aynı isimle tanımlanır. Bu metotların geri dönüş tipleri yoktur.

```
public class SinifA {
    public SinifA() {
    }
}
```

Mobil uygulama geliştirme programında sınıfa ait dosya açıkken Alt+Ins tuşlarına basılarak yapıcı metot eklenebilir (Görsel 5.15).



Görsel 5.15: Yapıcı metot oluşturma

Metotları aşırı yükleme ile birden fazla kurucu metot tanımlanabilir.

```
public class SinifA {
private int sayi1;
private int sayi2;
    public SinifA() {
       sayi1=0;
       sayi2=0;
    }
    public SinifA (int sayi1) {
        this.sayi1 = sayi1;
        this.sayi2 = sayi1;
    }
    public SinifA (int sayi1, int sayi2) {
        this.sayi1 = sayi1;
        this.sayi2 = sayi2;
    }
}
```

```
SinifA a1 = new SinifA();
// a1 nesnesi için
// sayi1=0
// sayi2=0

SinifA a2 = new SinifA(2);
// a2 nesnesi için
// sayi1=2
// sayi2=2

SinifA a3 = new SinifA(3,4);
// a1 nesnesi için
// sayi1=3
// sayi2=4
```

6. UYGULAMA: İşlem adımlarına göre dörtgen sınıfı oluşturarak yapıcı metotlar ile kare ve dikdörtgenin alanını bulan bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında iki EditText, bir Button ve bir TextView oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <EditText
        android:id="@+id/editTextKisaKenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:hint="Kısa Kenar"/>
    <EditText
        android:id="@+id/editTextUzunKenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:hint="Uzun Kenar"/>
```

```
    <Button
        android:id="@+id/buttonKareAlani"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Kare Alanı" />
    <Button
        android:id="@+id/buttonDikdortgenAlani"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dikdörtgen Alanı"/>
    <TextView
        android:id="@+id/textViewAlan"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Alan"/>
</LinearLayout>
```

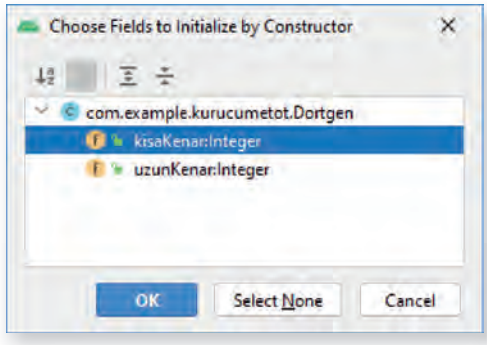
3. Adım: "Dortgen" adıyla yeni bir sınıf oluşturunuz.

4. Adım: Dortgen.java dosyasına şu kodu yazınız:

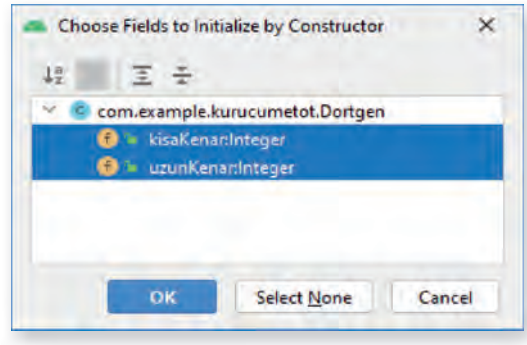
```
package com.example.kurucumetot;
public class Dortgen {
    public Integer kisaKenar;
    public Integer uzunKenar;
    public Integer alanBul(){
        Integer alan;
        alan = kisaKenar * uzunKenar;
        return alan;
    }
}
```

5. Adım: Dortgen.java dosyasında Alt+Ins tuşlarına basarak kisaKenar parametreli bir kurucu metot oluşturunuz (Görsel 5.16).

6. Adım: Dortgen.java dosyasında Alt+Ins tuşlarına basarak kisaKenar ve uzunKenar parametreli bir kurucu metot oluşturunuz (Görsel 5.17).



Görsel 5.16: Tek parametreli yapıcı metot oluşturma



Görsel 5.17: Çok parametreli yapıcı metot oluşturma

7. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.kurucumetot;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText editTextKisaKenar = findViewById(R.id.editTextKisaKenar);
        EditText editTextUzunKenar = findViewById(R.id.editTextUzunKenar);
        Button buttonKareAlani = findViewById(R.id.buttonKareAlani);
        Button buttonDikdortgenAlani = findViewById(R.id.buttonDikdortgenAlani);
        TextView textViewAlan = findViewById(R.id.textViewAlan);

        buttonKareAlani.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Integer kenar;
                Integer alan;
                kenar = Integer.parseInt(editTextKisaKenar.getText().toString());
                Dortgen kare = new Dortgen(kenar);
                alan = kare.alanBul();
                textViewAlan.setText(alan.toString());
            }
        });
        buttonDikdortgenAlani.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Integer kisaKenar;
                Integer uzunKenar;
                Integer alan;
                kisaKenar = Integer.parseInt(editTextKisaKenar.getText().toString());
                uzunKenar = Integer.parseInt(editTextUzunKenar.getText().toString());
                Dortgen dikdortgen = new Dortgen(kisaKenar,uzunKenar);
                textViewAlan.setText(dikdortgen.toString());
            }
        });
    }
}
```

8. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

SIRA SİZDE: Üçgen sınıfında eşkenar, ikizkenar ve çeşitkenar üçgenler için kurucu metotları oluşturunuz. Üçgenin çevresini hesaplayan bir "cevreBul" metodu yazınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Uygulama tasarım ekranına üç EditText ekledi.		
5. Uygulama tasarım ekranına üç Button ekledi.		
6. Uygulama tasarım ekranına bir TextView ekledi.		
7. "Ucgen" adında yeni bir sınıf oluşturdu.		
8. "Ucgen" sınıfına ait kenar1, kenar2 ve kenar3 özelliklerini belirledi.		
9. "Ucgen" sınıfına "cevreBul" adında metot oluşturdu.		
10. "Ucgen" sınıfına tek parametreli kurucu metot oluşturdu.		
11. "Ucgen" sınıfına iki parametreli kurucu metot oluşturdu.		
12. "Ucgen" sınıfına üç parametreli kurucu metot oluşturdu.		
13. Uygulama kodunda findViewById yöntemiyle nesneleri tanımladı.		
14. Düğmelere tıklanma olaylarını ekledi.		
15. "Ucgen" sınıfından eskenarUcgen adlı yeni bir nesne türetti.		
16. "Ucgen" sınıfından ikizkenarUcgen adlı yeni bir nesne türetti.		
17. "Ucgen" sınıfından cesitkenarUcgen adlı yeni bir nesne türetti.		
18. Çevre bilgisini TextView içinde gösterdi.		
19. Run düğmesine tıklayarak uygulamayı çalıştırdı.		

5.3. KAPSÜLLEME (ENCAPSULATION)

Nesne tabanlı programlama kavramlarından biri de kapsüllemedir. Kapsülleme, sınıf içindeki özelliklerin saklanarak doğrudan erişime kapatılması anlamına gelir. Buradaki asıl amaç, özellikleri gizlemek yani güvenlik değildir. Kapsülleme, sınıflar arasındaki bağımlılığı en aza indirir. Kapsüllemenin amacının daha iyi anlaşılabilmesi için kapsülleme üç açıdan incelenir.

1. **Doğruluk Açısından:** Sınıf içindeki özelliklerin sınıf dışına açık bırakılması iyi bir fikir değildir. Örneğin public erişim belirleyicisi ile dışarıya açık bırakılan bir Personel sınıfının yaş özelliğine -10 girilmesi sorunlara neden olur. Gerçek hayatta herhangi bir personelin yaşının -10 olması mümkün değildir.

```
public class Personel{
    public int yas;
}
```

```
public class Muhasebe{
    Personel personel1 = new Personel();
    personel1.yas = -10;
}
```

2. **Güvenlik Açısından:** Sınıf tanımlanırken güvenlik düşünülmelidir. Örneğin Personel sınıfındaki maaş özelliği public erişim belirleyici olarak tanımlandığında sınıf dışından maaş bilgisi istendiği gibi değiştirilebilir. Bu tür durumlar güvenlik açığı oluşturur.

```
public class Personel{
    public int yas;
    public int maas = 4253;
}
```

```
public class Muhasebe{
    Personel personel1 = new Personel();
    personel1.yas = -10;
    personel1.maas = 10000;
}
```

3. **Bağımlılık Açısından:** Kapsüllemenin asıl amacı, bağımlılıkları en aza indirmektir. Nesneler birbirleriyle etkileşim içindedir. Bağımlılık, bir nesnenin diğer nesneler ile olan ilişkisidir. İlişki ne kadar sıkı olursa bağımlılık o kadar artar. Örneğin Personel sınıfı içindeki yaş özelliğinin ismi değiştirildiğinde Muhasebe sınıfından personel sınıfına bağımlık olduğu için uygulama hata verir.

```
public class Personel{
    public int personelYas;
    public int maas = 4253;
}
```

```
public class Muhasebe{
    Personel personel1 = new Personel();
    personel1.yas = -10; //Hata verir.
    personel1.maas = 10000;
}
```

Kapsülleme; kodun yeniden kullanılabilirliğini, esnekliğini ve sürdürebilirliğini artıran avantajlara sahiptir.

- **Esneklik:** Kapsüllenmiş kodu yeni gereksinimlere göre değiştirmek esnek ve kolaydır. Örneğin bir kişinin maaş bilgisini belirleme gereksinimi değişirse **setMaas()** Setter metodunu değiştirmek yeterlidir.
- **Yeniden Kullanılabilirlik:** Aynı veya farklı uygulamalarda kapsüllenmiş kod yeniden kullanılabilir.
- **Sürdürülebilirlik:** Kapsüllenmiş uygulama kodu üzerindeki bir değişiklik, uygulamanın diğer bölümlerini etkilemez.

5.3.1. Kapsülleme Yapısı

Kapsülleme iki adımda gerçekleştirilir. Bu adımlar şunlardır:

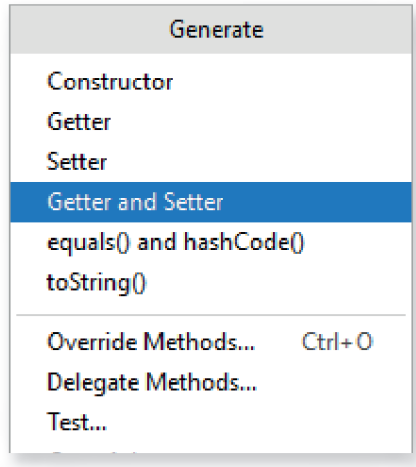
- Sınıf içindeki özelliklerin erişim belirleyicisi private olarak ayarlanır.
- Bu private üyelere erişmek ve değerlerini değiştirmek için sırasıyla public Getter ve Setter yöntemleri tanımlanır.

```
public class Personel{
    private int yas;
public int getYas() {
    return yas;
}
public void setYas(int yas) {
    this.yas = yas;
}
}
```

```
public class Muhasebe{
    Personel personel1 = new Personel();
    personel1.setYas(18);
}
```

5.3.2. Getter ve Setter Metotları

Mobil uygulama hazırlama programı, Getter ve Setter metotlarını tanımlamaya yardımcı olur. Getter ve Setter eklenecek sınıf dosyası açıkken Code menüsünden Generate komutu seçilir veya klavyeden Alt+Ins tuşlarına basılır. Ekrana gelen Generate penceresinden Getter and Setter seçilir (Görsel 5.18). İstenen özellikler seçilerek OK düğmesine tıklanır.



Görsel 5.18: Getter ve Setter metotlarını oluşturma

ÖRNEK

```
public class Personel {
    private String ad;
    public String getAd() {
        return ad;
    }
    public void setAd(String ad) {
        this.ad = ad;
    }
}
```

Setter metodu parametresi ile kapsüllenen özellik aynı adı taşır. Kod içinde karışıklık yaşanmaması için this deyimi kullanılır. This, üzerinde işlem yapılan nesneyi ifade eder. Sınıf içinde yer alan "this.ad" kodu, sınıfın private ile kapsüllenmiş ad özelliğidir.

Getter metodu, kapsüllenmiş özelliğin değerini almaya yarar. Getter metodunda özellik değiştirilmeden alınabileceği gibi manipüle ederek de alınabilir.

ÖRNEK

Personel sınıfında ad bilgisine ulaşılırken ad bilgisi büyük harfe çevrilerek alınır.

```
public class Personel {
    private String ad;
    public String getAd() {
        return ad.toUpperCase();
    }
}
```

Setter metodu, kapsüllenmiş özelliğin değerini girmeye yarar. Setter metodu da manipüle edilmeye açıktır.

ÖRNEK

Personel sınıfında ad bilgisi girilirken adın başına Sayın kelimesi getirilir.

```
public class Personel {
    private String ad;
    public void setAd(String ad) {
    this.ad = "Sayın " + ad;
}
}
```

7. UYGULAMA: İşlem adımlarına göre personel yaş bilgisinin girildiği bir uygulama tasarlayınız. Personel sınıfının yaş özelliğini kapsülleyiniz. Personel sınıfının yaş özelliğini 18'den küçük, 55'ten büyük girilmeyecek şekilde ayarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında bir EditText, bir Button ve bir TextView oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
```

```
    <EditText
        android:id="@+id/editTextYas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:minHeight="48dp"
        android:minWidth="100dp"
        android:hint="Yaş" />
    <Button
        android:id="@+id/buttonKaydet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Kaydet"/>
    <TextView
        android:id="@+id/textViewSonuc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Sonuç"/>
</LinearLayout>
```

3. Adım: "Personel" adıyla yeni bir sınıf oluşturunuz.

4. Adım: Personel.java dosyasına şu kodu yazınız:

```
package com.example.kapsulleme;
public class Personel {
    private int yas;
}
```

5. Adım: Alt+Ins tuşlarına basarak Generate penceresinden Getter and Setter komutunu çalıştırınız.

6. Adım: setYas metodunu şu kod ile değiştiriniz:

```
public void setYas(int yas) {
    if(yas<18 || yas>55){
        Log.e("Kapsülleme", "Hatalı yaş");
        yas = 18;
    }
    this.yas = yas;
}
```

7. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.kapsulleme;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText editTextYas = findViewById(R.id.editTextYas);
        Button buttonKaydet = findViewById(R.id.buttonKaydet);
        TextView textViewSonuc = findViewById(R.id.textViewSonuc);
        buttonKaydet.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int yas = Integer.parseInt(editTextYas.getText().toString());
                Personel personel = new Personel();
                personel.setYas(yas);
                textViewSonuc.setText(Integer.toString(personel.getYas()));
            }});
    }
}
```

8. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

9. Adım: Değerleri girerek KAYDET düğmesine tıklayınız.

SIRA SİZDE: "Personel" sınıfına maaş özelliğini kapsüllemeyle ekleyiniz. Maaş değerini 4.253'ten az olmayacak şekilde ayarlayınız.

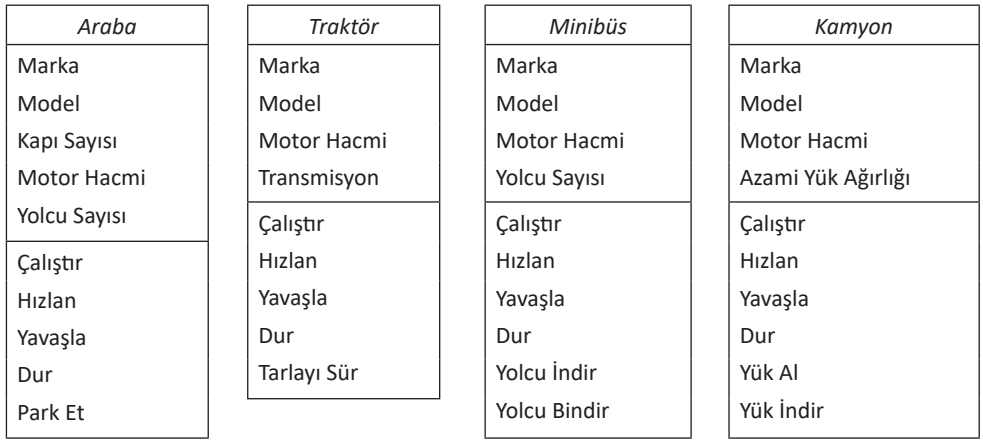
DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Uygulama tasarım ekranına iki EditText ekledi.		
5. Uygulama tasarım ekranına bir Button ekledi.		
6. Uygulama tasarım ekranına bir TextView ekledi.		
7. "Personel" adında yeni bir sınıf oluşturdu.		
8. "Personel" sınıfının özelliklerini belirledi.		
9. Maaş özelliğini erişim belirleyici olarak private seçti.		
10. Maaş için Getter ve Setter metotlarını oluşturdu.		
11. Uygulama kodunda findViewById yöntemiyle nesneleri tanımladı.		
12. Button nesnesine tıklanma olayını ekledi.		
13. Maaş bilgisini değişkenlere atadı.		
14. "Personel" sınıfından "personel" adlı yeni bir nesne türetti.		
15. "personel" nesnesinin setMaas metodu ile maaş bilgisini gönderdi.		
16. Personel maaşını getMaas ile aldı.		
17. Maaş bilgisini TextView içinde gösterdi.		
18. Run düğmesine tıklayarak uygulamayı çalıştırdı.		

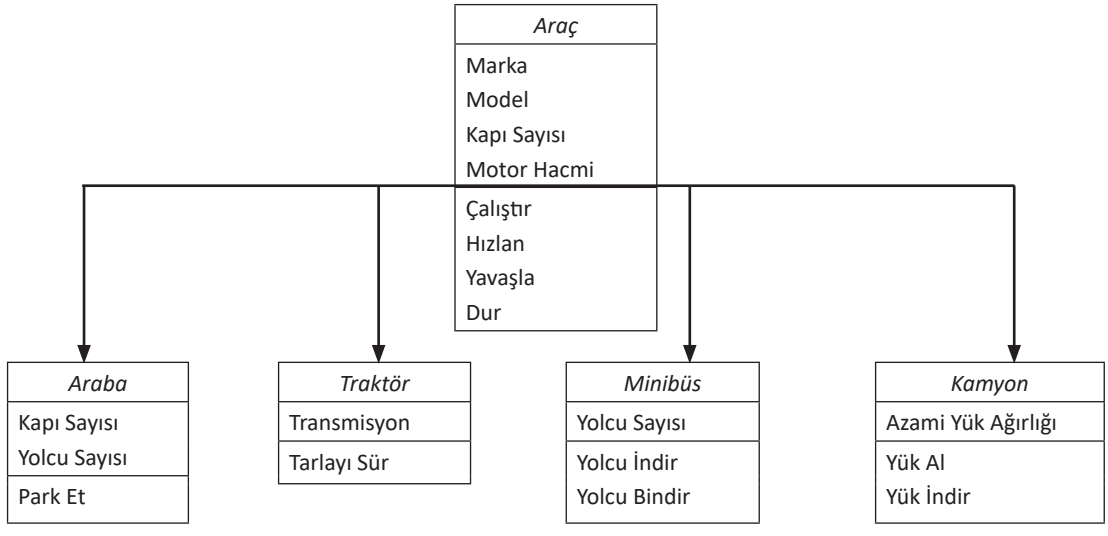
5.4. KALITIM (INHERITANCE)

Bir sınıf yapısının özellik ve metotlarını farklı bir sınıfa aktarma işlemine kalıtım denir. Kalıtım, çocukların ebeveynlerinden saç rengi, boy, göz rengi vb. özellikleri almaları gibi düşünülebilir. Örneğin trafikteki araçlar ile ilgili bir uygulama yapılması istenirse bazı araçlar (araba, traktör, minibüs ve kamyon) sınıf olarak modellenmelidir. Her araç için ayrı bir sınıf tanımlanmalıdır. Görsel 5.19'da bu sınıflara ait özellikler ve metotlar verilmiştir. Buradaki bazı özellik ve metotlar ortaktır. Sınıfların bu şekilde oluşturulması kod tekrarına neden olur.



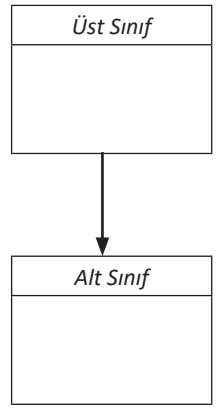
Görsel 5.19: Çeşitli araçlara ait sınıflar

Kod tekrarını engellemek için ortak özellik ve metotlara sahip bir üst sınıf tanımlanmalıdır. Diğer sınıflar, bu ortak özellikleri üst sınıftan miras almalıdır (Görsel 5.20).



Görsel 5.20: Üst sınıftan özellik ve metotları miras alma

Özellik ve metotlarını aktaran sınıfa üst sınıf denir. Özellik ve davranışların aktarıldığı sınıfa ise alt sınıf denir (Görsel 5.21).



Görsel 5.21: Üst sınıf, alt sınıf

Alt sınıf tanımlanırken "extends" kodu ile alt sınıfın üst sınıftan özellik ve metotları miras alacağı belirtilir.

```
public class AltSınıf extends ÜstSınıf {
}
```

8. UYGULAMA: İşlem adımlarına göre trafik araçlarını kalıtım kullanarak modelleyen bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında bir TextView, sekiz Button oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <TextView
        android:id="@+id/textViewBilgi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Araç Bilgisi"/>
    <Button
        android:id="@+id/buttonArabaKapi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Araba Kapı Sayısı"/>
```

```
    <Button
        android:id="@+id/buttonArabaHiz"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Araba maksimum hızı"/>
    <Button
        android:id="@+id/buttonArabaCalistir"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Araba'yı Çalıştır"/>
    <Button
        android:id="@+id/buttonArabaIseGit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Araba İşe Git"/>
    <Button
        android:id="@+id/buttonMinibusKapi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Minibüs Kapı Sayısı"/>
    <Button
        android:id="@+id/buttonMinibusHiz"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Minibüs maksimum hızı"/>
    <Button
        android:id="@+id/buttonMinibusCalistir"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Minibüs'ü Çalıştır"/>
    <Button
        android:id="@+id/buttonMinibusYolcuIndir"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Minibüs Yolcu İndir"/>
</LinearLayout>
```

3. Adım: "Arac" adıyla yeni bir sınıf oluşturunuz.

4. Adım: Arac.java dosyasına şu kodu yazınız:

```
package com.example.kalitim;
public class Arac {
    private Integer kapiSayisi;
    private Integer maksimumHiz;
    public Integer getKapiSayisi() {
        return kapiSayisi;
    }
```

```
    public void setKapiSayisi(Integer kapiSayisi) {
        this.kapiSayisi = kapiSayisi;
    }
    public Integer getMaksimumHiz() {
        return maksimumHiz;
    }
    public void setMaksimumHiz(Integer maksimumHiz) {
        this.maksimumHiz = maksimumHiz;
    }
    public String kapiSayisiniGoster(){
        return "Aracın kapı sayısı :" + this.kapiSayisi.toString();
    }
    public String maksimumHizGoster(){
        return "Aracın maksimum hızı :" + this.maksimumHiz.toString();
    }
    public String calistir(){
        return "Araç çalışıyor";
    }
}
```

5. Adım: "Araba" adıyla yeni bir sınıf oluşturunuz.

6. Adım: Araba.java dosyasına şu kodu yazınız:

```
package com.example.kalitim;
public class Araba extends Arac {
    public String iseGit(){
        return "Araba işe gidiyor";
    }
}
```

7. Adım: "Minibus" adıyla yeni bir sınıf oluşturunuz.

8. Adım: Minibus.java dosyasına şu kodu yazınız:

```
public class Minibus extends Arac{
    public String yolcuIndir(){
        return "Yolcular indiriliyor";
    }
}
```

9. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.kalitim;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView textViewBilgi = findViewById(R.id.textViewBilgi);
        //Araba düğmeleri
        Button buttonArabaKapi = findViewById(R.id.buttonArabaKapi);
        Button buttonArabaHiz = findViewById(R.id.buttonArabaHiz);
        Button buttonArabaCalistir = findViewById(R.id.buttonArabaCalistir);
        Button buttonArabaIseGit = findViewById(R.id.buttonArabaIseGit);
        //Minibüs düğmeleri
        Button buttonMinibusKapi = findViewById(R.id.buttonMinibusKapi);
        Button buttonMinibusHiz = findViewById(R.id.buttonMinibusHiz);
        Button buttonMinibusCalistir = findViewById(R.id.buttonMinibusCalistir);
        Button buttonMinibusYolcuIndir = findViewById(R.id.buttonMinibusYolcuIndir);
        //Yeni bir araba nesnesi oluşturuldu.
        Araba araba = new Araba();
        //Araba koltuk sayısı ve hız tanımlanır.
        araba.setKapiSayisi(5);
        araba.setMaksimumHiz(210);
        //Yeni bir minibüs nesnesi oluşturuldu.
        Minibus minibus = new Minibus();
        //Minibüs koltuk sayısı ve hız tanımlanır.
        minibus.setKapiSayisi(3);
        minibus.setMaksimumHiz(170);
        //Düğme tıklanma olayları
        buttonArabaKapi.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                textViewBilgi.setText(araba.kapiSayisiniGoster());
            }
        });
        buttonArabaHiz.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                textViewBilgi.setText(araba.maksimumHizGoster());
            }
        });
        buttonArabaCalistir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                textViewBilgi.setText(araba.calistir());
            }
        });
        buttonArabaIseGit.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
 public void onClick(View view) {
                textViewBilgi.setText(araba.iseGit());
            }
        });
        buttonMinibusKapi.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                textViewBilgi.setText(minibus.kapiSayisiniGoster());
            }
        });
        buttonMinibusHiz.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                textViewBilgi.setText(minibus.maksimumHizGoster());
            }
        });
        buttonMinibusCalistir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                textViewBilgi.setText(minibus.calistir());
            }
        });
        buttonMinibusYolcuIndir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                textViewBilgi.setText(minibus.yolcuIndir());
            }
        });
    }
}
```

10. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

9. UYGULAMA: İşlem adımlarına göre kalıtım ile eşkenar üçgen ve karenin çevresini bulan bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında bir EditText, iki Button ve bir TextView oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
```

```
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:orientation="vertical">
    <EditText
        android:id="@+id/editTextUzunluk"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:hint="Kenar Uzunluğu"/>
    <Button
        android:id="@+id/buttonUcgen"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Üçgen"/>
    <Button
        android:id="@+id/buttonKare"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Kare"/>
    <TextView
        android:id="@+id/textViewCevre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Çevre"/>
</LinearLayout>
```

3. Adım: "Sekil" adıyla yeni bir sınıf oluşturunuz.

4. Adım: Sekil.java dosyasına şu kodu yazınız:

```
package com.example.geometriksekiller;
public class Sekil {
    public Integer kenarSayisi;
    public Integer kenarUzunlugu;
    public Integer cevre(){
        return kenarSayisi*kenarUzunlugu;
    }
}
```

5. Adım: "Ucgen" adıyla yeni bir sınıf oluşturunuz.

6. Adım: Ucgen.java dosyasına şu kodu yazınız:

```
package com.example.geometriksekiller;
public class Ucgen extends Sekil{
    public Ucgen(Integer uzunluk) {
        this.kenarUzunlugu = uzunluk;
        this.kenarSayisi = 3;
    }
}
```

7. Adım: "Kare" adıyla yeni bir sınıf oluşturunuz.

8. Adım: Kare.java dosyasına şu kodu yazınız:

```
package com.example.geometriksekiller;
public class Kare extends Sekil{
    public Kare(Integer uzunluk) {
        this.kenarUzunlugu = uzunluk;
        this.kenarSayisi = 4;
    }
}
```

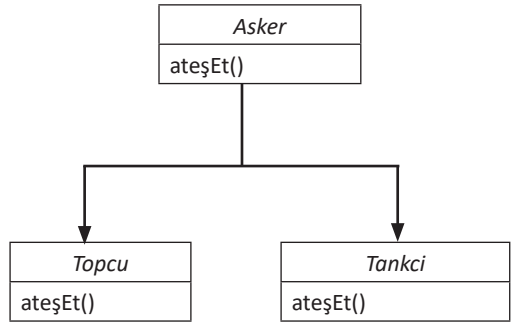
9. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.geometriksekiller;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText editTextUzunluk = findViewById(R.id.editTextUzunluk);
        Button buttonUcgen = findViewById(R.id.buttonUcgen);
        Button buttonKare = findViewById(R.id.buttonKare);
        TextView textViewCevre = findViewById(R.id.textViewCevre);
        buttonUcgen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Integer uzunluk = Integer.parseInt(editTextUzunluk.getText().toString());
                Ucgen ucgen = new Ucgen(uzunluk);
                Integer cevre = ucgen.cevre();
                textViewCevre.setText(cevre.toString());
            }
        });
        buttonKare.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Integer uzunluk = Integer.parseInt(editTextUzunluk.getText().toString());
                Kare kare = new Kare(uzunluk);
                Integer cevre = kare.cevre();
                textViewCevre.setText(cevre.toString());
            }
        });
    }
}
```

10. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

5.5. ÇOKBİÇİMLİLİK (POLYMORPHISM)

Çokbiçimlilik, yapılacak tek bir eylemin farklı şekillerde gerçekleştirilmesini sağlayan özelliklerden biridir. Bir başka deyişle bir nesnenin birden farklı nesne gibi davranmasıdır. Örneğin orduda birçok sınıf bulunur. Bu sınıflardan bazıları topçular ve tankçılardır. Topçular, top atarak savaşan ve topun başından ayrılmayan askerlerdir. Tankçılar ise tank kullanarak savaşan askerlerdir. Topçu ve tankçı askerler, ateş et emri aldıklarında farklı davranışlar sergiler. Topçular, top ile ateş ederken tankçılar, tank ile ateş eder. Görsel 5.22'de asker sınıfından türetilmiş topçu ve tankçı sınıfları verilmiştir. Çokbiçimliliğe ait bir kavram olan metot geçersiz kılma (method overriding) ile kalıtım yoluyla türemiş sınıflarda aynı isimli metotlar farklı komutlar çalıştırabilir. Askerler, topçular ve tankçılar aynı emri alırlar fakat farklı davranışlar sergiler.



Görsel 5.22: Asker sınıfından Topcu ve Tankci sınıflarının türetilmesi

5.5.1. Çokbiçimlilik Yapısı

Üst sınıftan bir alt sınıf türetilirken metotların geçersiz kılınması için @Override anahtar kelimesi kullanılmalıdır.

ÖRNEK

ÜST SINIF

```
public class Asker {
    public String atesEt(){
        return "Asker eteş etti";
    }
}
```

ALT SINIF

```
public class Tankci extends Asker{
    @Override
    public String atesEt() {
        return "Tankçı ateş etti";
    }
}
```

Asker sınıfında atesEt metodu mevcuttur. Asker sınıfından türetilmiş Tankci sınıfında da atesEt metodu vardır. Tankci sınıfında @Override anahtar kelimesi ile bir üst sınıfa ait atesEt metodu geçersiz kılınmıştır.

Türetilecek alt sınıflarda bazı metotların geçersiz kılınmasının engellenmesi gerekebilir. Bunun için üst sınıftaki metodun erişim belirleyicisine final anahtar kelimesi eklenmelidir.

ÖRNEK

ÜST SINIF	ALT SINIF

`public class Asker {`
`  public final String dur(){`
`  return "Asker durdu";`
`  }`
`}`
	Alt sınıfta dur metodu tanımlanmak istendiğinde hata verir.

10. UYGULAMA: İşlem adımlarına göre asker sınıfından türetilmiş tankçı sınıfıyla nesne oluşturan bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında iki Button ve bir TextView oluşturan şu kodu activity_main.xml içine yazınız:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <TextView
        android:id="@+id/textViewMesaj"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Mesaj"/>
    <Button
        android:id="@+id/buttonAsker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Asker Ateş Et" />
    <Button
        android:id="@+id/buttonTankci"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tankçı Ateş Et" />
</LinearLayout>
```

3. Adım: "Asker" adıyla yeni bir sınıf oluşturunuz.

4. Adım: Asker.java dosyasına şu kodu yazınız:

```
public class Asker {
    public String atesEt(){
        return "Asker ateş etti";
    }
}
```

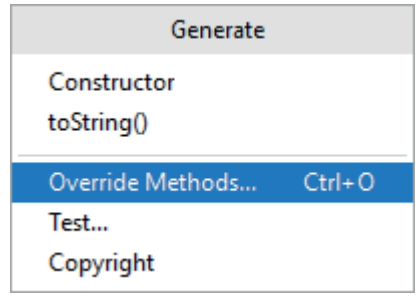
5. Adım: "Tankci" adıyla yeni bir sınıf oluşturunuz.

6. Adım: Tankci.java dosyasını açınız.

7. Adım: Tankci sınıfını Asker sınıfından türetecek şu kodu yazınız:

```
public class Tankci extends Asker{

}
```

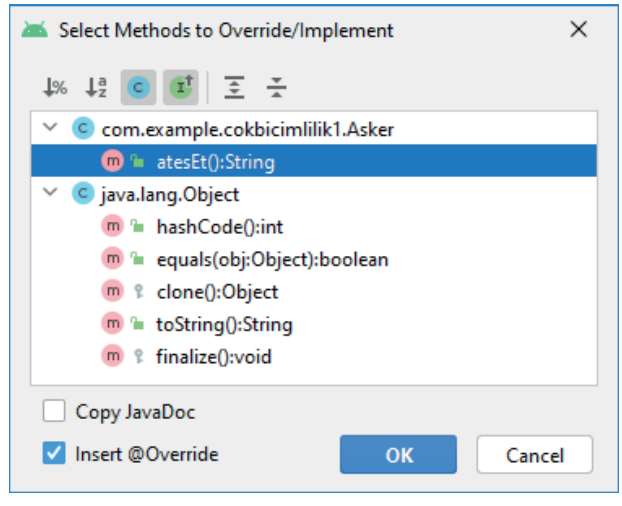
8. Adım: Alt+Ins tuşlarına basarak Generate penceresini açınız (Görsel 5.23).



Görsel 5.23: Generate penceresi

9. Adım: Override Methods komutunu tıklayınız.

10. Adım: Ekrana gelen Select Methods to Override/Implement penceresinde atesEt() komutunu seçip OK düğmesine tıklayınız (Görsel 5.24).



Görsel 5.24: Select Methods to Override/Implement penceresi

11. Adım: Tankci.java dosyasına şu kodu yazınız:

```
public class Tankci extends Asker{
    @Override
    public String atesEt() {
        return "Tankçı ateş etti";
    }
}
```

12. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.cokbicimlilik1;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    public Asker asker;
    public Tankci tankci;
    String mesaj = "";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        asker = new Asker();
        tankci = new Tankci();
        TextView textViewMesaj = findViewById(R.id.textViewMesaj);
        Button buttonAsker = findViewById(R.id.buttonAsker);
        Button buttonTankci = findViewById(R.id.buttonTankci);
        buttonAsker.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mesaj = asker.atesEt();
                textViewMesaj.setText(mesaj);
            }
        });
        buttonTankci.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mesaj = tankci.atesEt();
                textViewMesaj.setText(mesaj);
            }
        });
    }
}
```

13. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

SIRA SİZDE: Asker sınıfından türemiş bir Topcu sınıfı ekleyen bir uygulama tasarlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

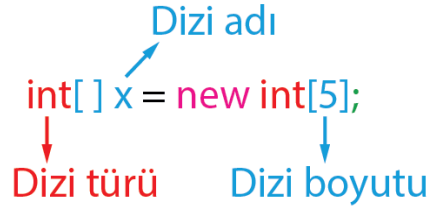
KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Open Project komutunu tıkladı.		
2. Onuncu uygulama projesini açtı.		
3. Uygulama tasarım ekranına buttonTopcu adında bir Button ekledi.		
4. Topcu sınıfını oluşturdu.		
5. Topcu sınıfını Asker sınıfından türeten kodu yazdı.		
6. Metot geçersiz kılma ile atesEt metodunu yazdı.		
7. Uygulama kodunda findViewById yöntemiyle "buttonTopcu" nesnesini tanımladı.		
8. Topcu sınıfından yeni bir nesne oluşturdu.		
9. "buttonTopcu" nesnesine tıklanma olayını ekledi.		
10. "buttonTopcu" nesnesinin tıklanma olayına topcu.atesEt( ) komutunu veren kodları yazdı.		
11. Run düğmesine tıklayarak uygulamayı çalıştırdı.		

5.6. DİZİLER

Diziler, içinde birden fazla değer barındırabilen değişkenlerdir. Uygulama geliştirilirken birçok değer ile işlem yapılması gerekebilir. Her değer için ayrı bir değişken tanımlamak zordur. Diziler sayesinde tek değişken tanımlanarak bu değişken içinde birden fazla değer kaydedilir.

Diziler bir değişken gibi tanımlanır. Değişken türünün yanına köşeli parantez işareti kullanılır. New anahtar kelimesinin yanına değişken türü ve köşeli parantez içine dizinin boyutu yazılır (Görsel 5.25).



Görsel 5.25: Dizi yapısı

Dizilerin türü, ilkel veri türü olabileceği gibi herhangi bir sınıf da olabilir.

ÖRNEK

```
Asker[] askerler = new Asker[10];
// İçinde 10 adet Asker sınıfı nesne barındıran bir dizi
```

Dizilerin boyutu oluşturulduktan sonra değiştirilemez, sabittir. Dizilerin içindeki her bir elemanın konumunu bildiren bir sıra numarası verilir. Bu sıra numarasına indis adı verilir. Java'da indisler 0'dan başlar.

5.6.1. Diziye Değer Atama

Dizilere değer atamanın birden fazla yöntemi mevcuttur. İlk yöntem, dizi oluşturulurken değer atamasının yapılmasıdır. Bu yöntemde dizinin boyut değeri girilmez. Girilen elemanların sayısına göre dizinin boyutu otomatik olarak ayarlanır.

ÖRNEK

```
Integer[ ] sayilar = new Integer[]{1,3,5,7,9}; //Dizinin boyutu 5'tir.
```

Dizilere değer atamanın diğer yöntemi ise dizi oluşturulduktan sonra değer atamaktır. Dizinin elamanına değer atamak için dizi indis numarası kullanılır.

ÖRNEK

```
Integer[ ] sayilar = new Integer[5]; //Dizinin boyutu 5'tir.
sayilar[0] = 2; //Dizinin 0'ıncı elemanı
sayilar[1] = 4; //Dizinin 1'inci elemanı
sayilar[2] = 6; //Dizinin 2'nci elemanı
sayilar[3] = 8; //Dizinin 3'üncü elemanı
sayilar[4] = 10; //Dizinin 4'üncü elemanı
```

11. UYGULAMA: İşlem adımlarına göre diziler ile 6 sayıyı toplayan bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Uygulama ekranında iki TextView, bir EditText ve iki Button oluşturan şu kodu activity_main.xml içine yazınız:

```
<TextView
    android:id="@+id/textViewIndis"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="İndis: 0"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.1" />
<TextView
    android:id="@+id/textViewToplam"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Toplam: 0"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textViewIndis" />
<EditText
    android:id="@+id/editTextSayi"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:ems="10"
    android:inputType="number"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textViewToplam" />
<Button
    android:id="@+id/buttonEkle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Ekle"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextSayi" />
<Button
    android:id="@+id/buttonSifirla"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Sıfırla"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/buttonEkle" />
```

3. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.diziler;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    public Integer[] notlar = new Integer[6];
    public int indis =0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView textViewIndis = findViewById(R.id.textViewIndis);
        TextView textViewToplam = findViewById(R.id.textViewToplam);
        EditText editTextSayi = findViewById(R.id.editTextSayi);
        Button buttonEkle = findViewById(R.id.buttonEkle);
        Button buttonSifirla = findViewById(R.id.buttonSifirla);
        buttonEkle.setOnClickListener(view -> {
            if (indis <6){
                int sayi;
                sayi = Integer.parseInt(editTextSayi.getText().toString());
                notlar[indis]=sayi;
                indis++;
                textViewIndis.setText("İndis: " + Integer.toString(indis));
                int toplam = 0;
                for (int i = 0; i < indis; i++) {
                    toplam += notlar[i];
                }
                textViewToplam.setText("Toplam: " + Integer.toString(toplam));
                editTextSayi.getText().clear();
            }
            else{
                textViewIndis.setText("Dizi dolu");
            }
        });
        buttonSifirla.setOnClickListener(view -> {
            indis=0;
            textViewIndis.setText("İndis: 0");
            textViewToplam.setText("Toplam: 0");
            notlar = new Integer[6];
        });
    }
}
```

4. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

5.6.2. ArrayList

Dizilerin boyutu sabittir. Uygulama çalışırken dizilerin boyutu artırılamaz ve azaltılamaz. Mobil uygulama geliştirilirken uygulama çalışma zamanı esnasında boyutu daha önceden tahmin edilemeyecek dizi yapılarına ihtiyaç duyulur. Bu ihtiyacı karşılayacak birçok yapı vardır. Bunlardan en çok kullanılanı ArrayList yapısıdır. ArrayList, istenen sayıda öge eklenen veya silinen dinamik yapıdır. ArrayList yapısını kullanmak için uygulama paketine import komutu ile java.util.ArrayList eklenmelidir. ArrayList yapısına öge olarak sadece nesneler eklenebilir, ilkel veri türleri eklenemez. ArrayList tanımlaması Görsel 5.26'da verilmiştir.

Görsel 5.26: ArrayList yapısı

5.6.2.1. ArrayList Yapısına Öge Ekleme (Add)

ArrayList yapısının sonuna öge eklemek için add metodu kullanılır.

ÖRNEK

```
ArrayList<Integer> aListesi = new ArrayList< >();
aListesi.add(2);
aListesi.add(4);
aListesi.add(6);
//aListesi içinde sırasıyla 2, 4 ve 6 vardır.
```

ArrayList yapısında araya öge eklemek için add metodunda iki parametre gönderilir. Birinci parametre eklenecek indis sırası, ikinci parametre ise eklenecek nesnedir.

ÖRNEK

```
ArrayList<Integer> aListesi = new ArrayList< >();
aListesi.add(2);
aListesi.add(4);
aListesi.add(6);
aListesi.add(1,3); //1 No.lu indise 3 sayısını ekler.
//aListesi içinde sırasıyla 2, 3, 4 ve 6 vardır.
```

5.6.2.2. ArrayList Yapısından Öge Silme (Remove)

ArrayList yapısından öge silmek için remove metodu kullanılır. Remove metodu iki farklı şekilde gerçekleştirilir. İlk kullanımda parametre olarak listeden çıkarılacak öge verilir. İkinci kullanımda ise parametre olarak çıkarılacak ögenin indisi verilir.

ÖRNEK

```
ArrayList<Integer> aListesi = new ArrayList< >();
aListesi.add(2);
aListesi.add(4);
aListesi.add(6);
aListesi.remove(1); //1 No.lu indisteki 4 sayısı listeden çıkartılır.
//aListesi içinde sırasıyla 2 ve 6 vardır.
```

ÖRNEK

```
ArrayList<Ogrenci> ogrenciler= new ArrayList< >();
ogrenciler.add(ogrenci1);
ogrenciler.add(ogrenci2);
ogrenciler.add(ogrenci3);
ogrenciler.remove(ogrenci2);
//Öğrenciler listesi içinde sırasıyla ogrenci1 ve ogrenci3 vardır.
```

5.6.2.3. ArrayList Yapısını Temizleme (Clear)

ArrayList içindeki tüm ögeleri silmek için clear metodu kullanılır.

ÖRNEK

```
ArrayList<Integer> aListesi = new ArrayList< >();
aListesi.add(2);
aListesi.add(4);
aListesi.add(6);
aListesi.clear();
//aListesi içinde hiçbir öge yoktur.
```

5.6.2.4. ArrayList Yapısında Arama (Contains)

ArrayList içindeki bir ögeyi aramak için contains metodu kullanılır. Contains metoduna parametre olarak aranacak nesne gönderilir.

ÖRNEK

```
ArrayList<Integer> aListesi = new ArrayList< >();
aListesi.add(2);
aListesi.add(4);
aListesi.add(6);
if(aListesi.contains(4))
       System.out.println("Var");
else
       System.out.println("Yok");
```

5.6.2.5. ArrayList Yapısını Kopyalama (Clone)

ArrayList yapısının tam olarak bir kopyasını almak için clone metodu kullanılır. Clone metodunu kullanmak için tip dönüşümü yapılmalıdır.

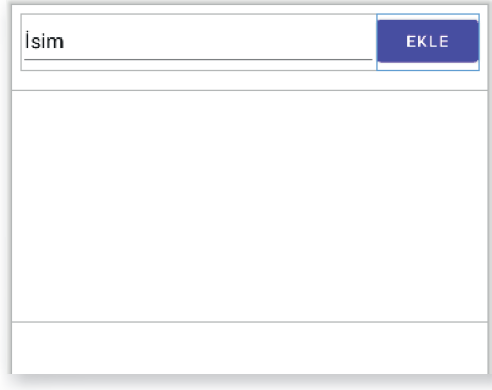
ÖRNEK

```
ArrayList<Integer> aListesi = new ArrayList< >();
aListesi.add(2);
aListesi.add(4);
aListesi.add(6);
ArrayList<Integer> aListeKopyasi = (ArrayList<Integer>) aListesi.clone();
```

12. UYGULAMA: İşlem adımlarına göre eklenen isimleri ListViewde gösteren ve ListViewde tıklanan ismi listeden silen bir uygulama tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Görsel 5.27'de verilen uygulama ekranını oluşturan şu kodu activity_main.xml içine yazınız.



Görsel 5.27: Uygulama ekranı

```
<EditText
    android:id="@+id/editTextAd"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:inputType="textPersonName"
    android:minHeight="48dp"
    android:text="İsim"
    app:layout_constraintEnd_toStartOf="@+id/buttonEkle"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<Button
    android:id="@+id/buttonEkle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:text="Ekle"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<ListView
    android:id="@+id/listViewAdlar"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_marginTop="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextAd"
    app:layout_constraintVertical_bias="0.02" />
```

3. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.arraylistkullanimi;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {
    Button buttonEkle;
    EditText editTextAd;
    ListView listViewAdlar;
    ArrayList<String> isimlerListesi  = new ArrayList<>();
    ArrayAdapter<String> adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, isim-
lerListesi);
        buttonEkle = findViewById(R.id.buttonEkle);
        editTextAd = findViewById(R.id.editTextAd);
        listViewAdlar = findViewById(R.id.listViewAdlar);
        listViewAdlar.setAdapter(adapter);
        buttonEkle.setOnClickListener(view -> {
            String ad = editTextAd.getText().toString();
            isimlerListesi.add(ad);
            adapter.notifyDataSetChanged();
            editTextAd.getText().clear();
        });
```

```
        listViewAdlar.setOnItemClickListener((adapterView, view, i, l) -> {
            isimlerListesi.remove(i);
            adapter.notifyDataSetChanged();
        });
    }
}
```

4. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

SIRA SİZDE: Öğrenci numarası ve adından bir öğrenci sınıfı oluşturunuz. Öğrenci sınıfı nesnelerini alacak şekilde bir liste tanımlayınız. Bir Button ile öğrenci sınıfından yeni nesne oluşturup bu listeye ekleyiniz. Uygulama ekranında bu listeyi gösteren bir uygulama tasarlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. New Project komutunu tıkladı.		
2. Empty Activity proje türünü seçti.		
3. Uygulama adını belirledi.		
4. Uygulama tasarım ekranına iki EditText ekledi.		
5. Uygulama tasarım ekranına bir Button ekledi.		
6. Uygulama tasarım ekranına bir ListView ekledi.		
7. “Ogrenci” adında yeni bir sınıf oluşturdu.		
8. “Ogrenci” sınıfına numara özelliğini ekledi.		
9. “Ogrenci” sınıfına ad özelliğini ekledi.		
10. Uygulama kodunda findViewById yöntemiyle nesneleri tanımladı.		
11. ArrayList oluşturdu.		
12. ArrayAdepter oluşturdu.		
13. ListView nesnesine ArrayAdepter nesnesini bağladı.		
14. Button nesnesine tıklanma olayı ekledi.		
15. Button nesnesi tıklanma olayına listeye ekleme kodunu yazdı.		
16. Run düğmesine tıklayarak uygulamayı çalıştırdı.		

13. UYGULAMA: İşlem adımlarına göre asker, tankçı ve topçu nesnelerini kullanarak iki kişilik basit bir oyun uygulaması tasarlayınız.

1. Adım: Mobil uygulama geliştirme programında Empty Activity olacak şekilde bir proje oluşturunuz.

2. Adım: Görsel 5.28'de verilen uygulama ekranını oluşturan şu kodu activity_main.xml dosyası içine yazınız.

Görsel 5.28: Uygulama ekranı

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.andro-
id.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="SAVAŞ OYUNU"
        android:textSize="24dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="52dp"
        android:layout_marginTop="28dp"
        android:text="1. OYUNCU"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="72dp"
        android:text="2. OYUNCU"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@+id/textView2" />
    <ProgressBar
        android:id="@+id/progressBar1"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="125dp"
        android:layout_height="20dp"
        android:layout_marginStart="24dp"
        android:layout_marginTop="12dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        android:max="100"
        android:progress="100"
        />
    <ProgressBar
        android:id="@+id/progressBar2"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="125dp"
        android:layout_height="20dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="44dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3"
        android:max="100"
        android:progress="100"
        />
    <Button
        android:id="@+id/buttonTopcu1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="20dp"
        android:text="Topçu Ateşi"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/progressBar1" />
    <Button
```

```
        android:id="@+id/buttonTankci1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="20dp"
        android:text="Tankçı Ateşi"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonTopcu1" />
    <Button
        android:id="@+id/buttonTopcu2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="36dp"
        android:text="Topçu Ateşi"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/progressBar2" />
    <Button
        android:id="@+id/buttonTankci2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="32dp"
        android:text="Tankçı Ateşi"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonTopcu2" />
    <Button
        android:id="@+id/buttonReset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="Baştan Başla"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonTankci1" />
    <TextView
        android:id="@+id/textViewIsabet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="44dp"
        android:text="0"
        android:textSize="24dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonReset" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Adım: Asker sınıfı oluşturunuz.

4. Adım: Asker.java dosyasına şu kodu yazınız:

```
package com.example.savasoyunu;
import java.util.Random;
public class Asker {
    private Integer atesGucu;
    public Integer getAtesGucu() {
        return atesGucu;
    }
    public void setAtesGucu(Integer atesGucu) {
        this.atesGucu = atesGucu;
    }
    private Integer sans;
    public Integer getSans() {
        return sans;
    }
    public void setSans(Integer sans) {
        this.sans = sans;
    }
    public Asker(Integer atesGucu, Integer sans) {
        this.atesGucu = atesGucu;
        this.sans = sans;
    }
    public Integer atesEt(Oyuncu dusman){
        Random random = new Random();
        int isabet = random.nextInt(sans)*atesGucu;
        dusman.setCan(dusman.getCan()-isabet);
        return isabet;
    }
}
```

5. Adım: Tankci sınıfını oluşturunuz.

6. Adım: Tankci.java dosyasına şu kodu yazınız:

```
package com.example.savasoyunu;
public class Tankci extends Asker {
    public Tankci() {
        super(5,5);
    }
}
```

7. Adım: Topcu sınıfını oluşturunuz.

8. Adım: Topcu.java dosyasına şu kodu yazınız:

```
package com.example.savasoyunu;
public class Topcu extends Asker{
    private boolean topIsindi = false;
    public int getAtisSayisi() {
        return atisSayisi;
    }
    public void setAtisSayisi(int atisSayisi) {
        this.atisSayisi = atisSayisi;
```

```
}
    public int atisSayisi;
    @Override
    public Integer atesEt(Oyuncu dusman) {
        if (topIsindi){
            topIsindi = false;
            atisSayisi=0;
            return 0;
        }
        atisSayisi++;
        if (atisSayisi==3){
            topIsindi=true;
        }
        int isabet = super.atesEt(dusman);
        return isabet;
    }
    public Topcu() {
        super(3,7);
        atisSayisi =0;
    }
}
```

9. Adım: Oyuncu sınıfını oluşturunuz.

10. Adım: Oyuncu.java dosyasına şu kodu yazınız:

```
package com.example.savasoyunu;
public class Oyuncu {
    private String isim;
    private Integer can;
    public String getIsim() {
        return isim;
    }
    public void setIsim(String isim) {
        this.isim = isim;
    }
    public Integer getCan() {
        return can;
    }
    public void setCan(Integer can) {
        this.can = can;
    }
    public Oyuncu(String isim, Integer can) {
        this.isim = isim;
        this.can = can;
    }
    public Asker tankci = new Tankci();
    public Asker topcu = new Topcu();
}
```

11. Adım: MainActivity.java dosyasına şu kodu yazınız:

```
package com.example.savasoyunu;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {
    Oyuncu oyuncu1;
    Oyuncu oyuncu2;
    Button buttonTank1;
    Button buttonTank2;
    Button buttonTopcu1;
    Button buttonTopcu2;
    Button buttonReset;
    ProgressBar progressBar1;
    ProgressBar progressBar2;
    TextView textViewIsabet;
    Integer isabet;
    ArrayList<Oyuncu> oyuncular = new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        oyuncu1 = new Oyuncu("Oyuncu 1",100);
        oyuncu2 = new Oyuncu("Oyuncu 2",100);
        oyuncular.add(oyuncu1);
        oyuncular.add(oyuncu2);
        buttonTank1 = findViewById(R.id.buttonTankci1);
        buttonTank2 = findViewById(R.id.buttonTankci2);
        buttonTopcu1 = findViewById(R.id.buttonTopcu1);
        buttonTopcu2 = findViewById(R.id.buttonTopcu2);
        buttonReset = findViewById(R.id.buttonReset);
        progressBar1 = findViewById(R.id.progressBar1);
        progressBar2 = findViewById(R.id.progressBar2);
        textViewIsabet = findViewById(R.id.textViewIsabet);
        buttonTank1.setOnClickListener(view -> {
            isabet = oyuncu1.tankci.atesEt(oyuncu2);
            progressBar2.setProgress(oyuncu2.getCan());
            textViewIsabet.setText(isabet.toString());
            oyuncuKontrolet();
        });
        buttonTopcu1.setOnClickListener(view -> {
            isabet = oyuncu1.topcu.atesEt(oyuncu2);
            progressBar2.setProgress(oyuncu2.getCan());
```

```
                    textViewIsabet.setText(isabet.toString());
            oyuncuKontrolet();
        });
        buttonTank2.setOnClickListener(view -> {
            isabet = oyuncu2.tankci.atesEt(oyuncu1);
            progressBar1.setProgress(oyuncu1.getCan());
            textViewIsabet.setText(isabet.toString());
            oyuncuKontrolet();
        });
        buttonTopcu2.setOnClickListener(view -> {
            isabet = oyuncu2.topcu.atesEt(oyuncu1);
            progressBar1.setProgress(oyuncu1.getCan());
            textViewIsabet.setText(isabet.toString());
            oyuncuKontrolet();
        });
        buttonReset.setOnClickListener(view -> {
            oyuncu1.setCan(100);
            oyuncu2.setCan(100);
            progressBar1.setProgress(oyuncu1.getCan());
            progressBar2.setProgress(oyuncu2.getCan());
            textViewIsabet.setText("0");
        });
    }
    private void oyuncuKontrolet() {
        for (Oyuncu oyuncu:oyuncular) {
            if (oyuncu.getCan()<=0 )
                textViewIsabet.setText("Oyun Bitti");
        }
    }
}
```

12. Adım: Run düğmesine tıklayarak uygulamayı çalıştırınız.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) DRY prensibinin amacı, kod tekrarının önlenmesidir.
2. ( ) Değer döndüren metotlarda return anahtar kelimesi kullanılmalıdır.
3. ( ) Overriding, metot aşırı yükleme anlamına gelir.
4. ( ) Yeni bir nesne oluşturmak için new anahtar kelimesi kullanılır.
5. ( ) Private erişim belirleyicisi ile özellik ve metotlara paket içinden erişilir.
6. ( ) Final anahtar kelimesi ile metot geçersiz kılma engellenir.
7. ( ) Getter metodu, değişkene değer atamak için kullanılır.

B) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

8. **Aşağıdakilerden hangisi geri dönüş değeri olmayan bir metot tanımlamasıdır?**

   A) Integer B) void C) null D) boş E) String

9. **Aşağıda verilen metot aşırı yüklemelerinden hangisi "int topla(int sayi1)" metodu için yanlış tanımlanmıştır?**

   A) int topla(int sayi2)
   B) int topla()
   C) int topla(float sayi1)
   D) int topla(int sayi1, int sayi2)
   E) String topla(int sayi1)

10. **Aşağıdaki erişim belirleyicilerinden hangisi genel olarak her yerden erişilebilir anlamına gelir?**

    A) Private
    B) Default
    C) Protected
    D) Public
    E) void

11. **Kalıtım için aşağıdakilerden hangisi yanlıştır?**

    A) Alt sınıf tanımlanırken extends anahtar kelimesi kullanılır.

    B) Alt ve üst sınıfta aynı isimde metot olabilir.

    C) Üst sınıf, alt sınıftan özellik ve metotları miras alır.

    D) Alt sınıf, üst sınıfın tüm özellik ve metotlarını miras alır.

    E) Üst sınıftan birden fazla sınıf türetilebilir.

12. Integer[ ] sayilar = new Integer[5];

sayilar[0] = 1;

sayilar[1] = 3;

sayilar[2] = 5;

sayilar[3] = 7;

sayilar[4] = 9;

Yukardaki tanımlamaya göre 7 değerini almış dizi değişkeni aşağıdaki seçeneklerin hangisinde verilmiştir?

A) sayilar[0] B) sayilar[1] C) sayilar[2]

D) sayilar[3] E) sayilar[4]

6. ÖĞRENME BİRİMİ

UYGULAMA TASARIMI

KONULAR

6.1. PROJE YAPILANDIRMA AYARLARI

6.2. VIEW BINDING

6.3. ACTIVITY YAPISI

6.4. ARAYÜZ TASARIMI

6.5. FRAGMENT YAPISI

6.6. MOBİL UYGULAMADA İZİNLER VE İZİN YAPISI

NELER ÖĞRENECEKSİNİZ?

- Oluşturulacak projelerin yapılandırma ayarları
- Gradle dosya yapısının içeriği
- Projelerde kullanıcıdan izin alma işlemleri
- Farklı arayüz tasarımları oluşturma
- Çoklu aktivitelerle çalışma ve aktiviteler arasına bilgi gönderme
- Fragment yapısı
- View Binding yapısı

ANAHTAR KELİMELER

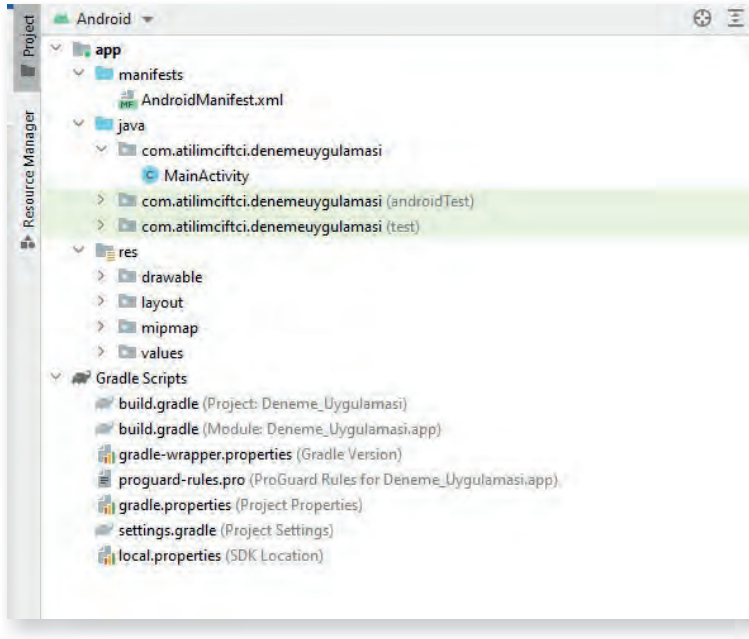
- Activity
- Arayüz
- Fragment
- Gradle
- Permission
- View Binding
- Yapılandırma

HAZIRLIK ÇALIŞMALARI

1. Daha önce kullandığınız mobil uygulamalarda farklı ekranlar arasındaki geçişlerin nasıl olduğunu arkadaşlarınızla değerlendiriniz.
2. Mobil uygulamalar kullanılırken hangi izinler sorulur? Hatırladıklarınızı arkadaşlarınızla paylaşınız.

6.1. PROJE YAPILANDIRMA AYARLARI

Proje olarak yeni bir mobil uygulama tasarımı oluşturulduğunda Project alanında karşılaşılan dosya ve klasör yapısını tanımak, buradaki dosyaların her birinin farklı bir işlevinin olduğunu bilmek, projeye hâkimiyet sağlar (Görsel 6.1).



Görsel 6.1: Mobil uygulama geliştirme için Project alanı

6.1.1. AndroidManifest.xml Dosya Yapısı

AndroidManifest, **manifest** klasörünün içinde yer alan ve mobil uygulama geliştirme ortamında tasarlanan uygulamalar için vazgeçilmez dosya yapısıdır. Bu dosya içinde Activity, Service, Receiver vb. sınıfları ve projeye ait temel bilgiler bulunur. Dosya uzantısı, xml formatında uzantı olduğu için okunması ve değiştirilmesi oldukça kolaydır. Görsel 6.2'de yeni oluşturulmuş bir mobil uygulama geliştirme projesinin AndroidManifest.xml dosya içeriği verilmiştir. Burada kodlar "Tag" (etiket) denilen **< >** işaretleri arasında yazılır. Bu etiketler **< >** şeklinde başlayıp **</>** şeklinde kapatılır. Görsel 6.2'de kodlar içinde etiketler verilmiştir.

activity_main.xml | MainActivity.java | AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.atilimciftci.denemeuygulamasi">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.DenemeUygulamasi">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Görsel 6.2: AndroidManifest.xml içeriği

- **manifest**

Bu etiket ile uygulamaların versiyon numaraları, paket isimleri, uygulama izinleri gibi mobil uygulamanın en temel özellikleri girilir. Projenin içinde yer alan kütüphanelerden minimum ve maksimum sdk sürümlerine kadar bu bölümde ayarlanır.

- **package**

Uygulamanın paket adının bulunduğu yapı türüdür. Genellikle mobil uygulamalar Görsel 6.2'de görüldüğü gibi "com.atilimciftci.denemeuygulamasi" şeklinde bir domain olarak verilir. Bunun amacı, paket isminin **unique** (benzersiz) olarak tanımlanmasıdır.

NOT:

AndroidManifest.xml içinde, manifest etiketi arasında yer alan package ile oluşturulmuş paket adının tamamen uygulama geliştiriciye ait olduğu unutulmamalıdır. Geliştirilen uygulama daha sonra markette yayımlanmak istediğinde de bu paket adı kullanılır. Bu nedenle geliştirilen her uygulamanın paket adının unique bir yapıda olmasına dikkat edilmelidir.

- **application**

Bu etiket ile geliştirilen mobil uygulamanın logosu (android:icon), adı (android:label), teması (android:theme) vb. temel özellikleri ayarlanır. Application etiketleri arasına donanım hızlandırma veya yedekleme gibi temel özellikler de eklenebilir. Görsel 6.2'de "application" etiketi arasına bakıldığında "android:" koduyla birlikte verilen özelliklerin girildiği görülür.

- **activity**

Geliştirilen mobil uygulamaya ait activitylerin ve bu activitylere ait özelliklerin yer aldığı etikettir.

- **intent-filter**

Activity, Service ve Broadcast Receiver bileşenlerine intent tipini belirtmek için kullanılan etikettir. intent-filter içinde <action>, <category>, <data> etiketleri bulunur. intent-filter içine <action> etiketinin mutlaka eklenmesi gerekir.

- **action**

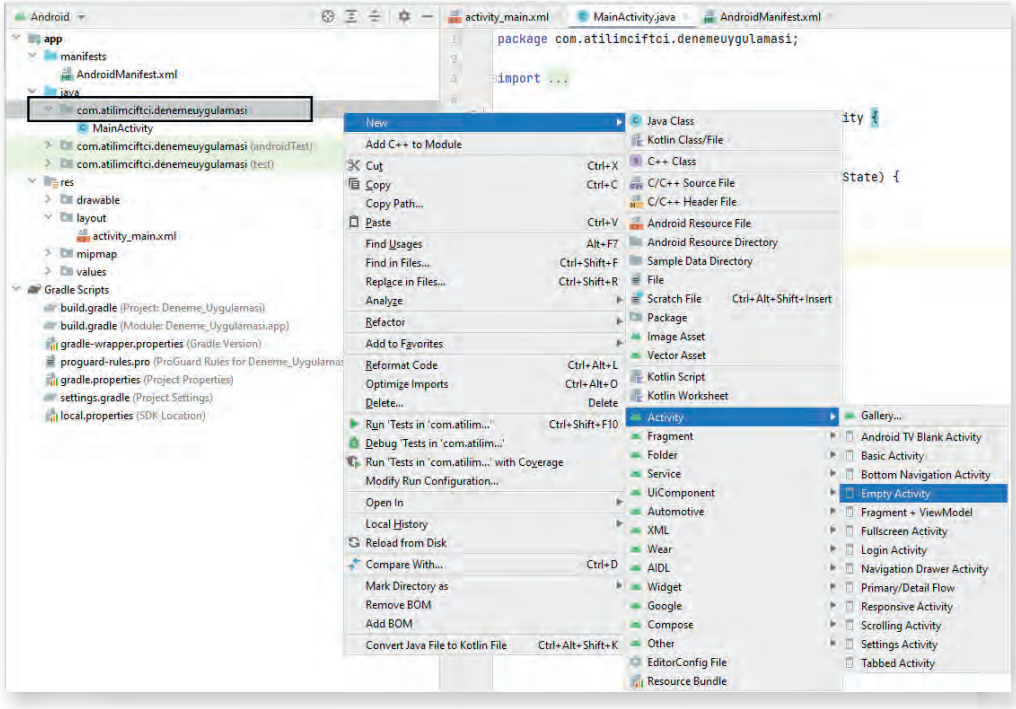
Uygulama çalıştırıldığında ilk olarak çalışacak sınıfı belirtir. Bu etiket, Görsel 6.2'de görüldüğü gibi **<action android:name="android.intent.action.MAIN" />** şeklinde tanımlanır. Bu kodun açılan etiketinin tekrar <action> </action> şeklinde kapatılmasına gerek yoktur. İlk açılan action etiketi içinde action etiketi kapatılmadan önce **/** tagının kullanılması, etiketlemenin kapatıldığını gösterir.

NOT: AndroidManifest.xml içinde verilen özellikler; manifest, package, application, activity, intent-filter, action ile sınırlı değildir. İzinler, Servisler, Broadcast Receiver vb. kullanılmak istenen yapıya göre o yapının etiketleri de AndroidManifest.xml içine eklenmelidir.

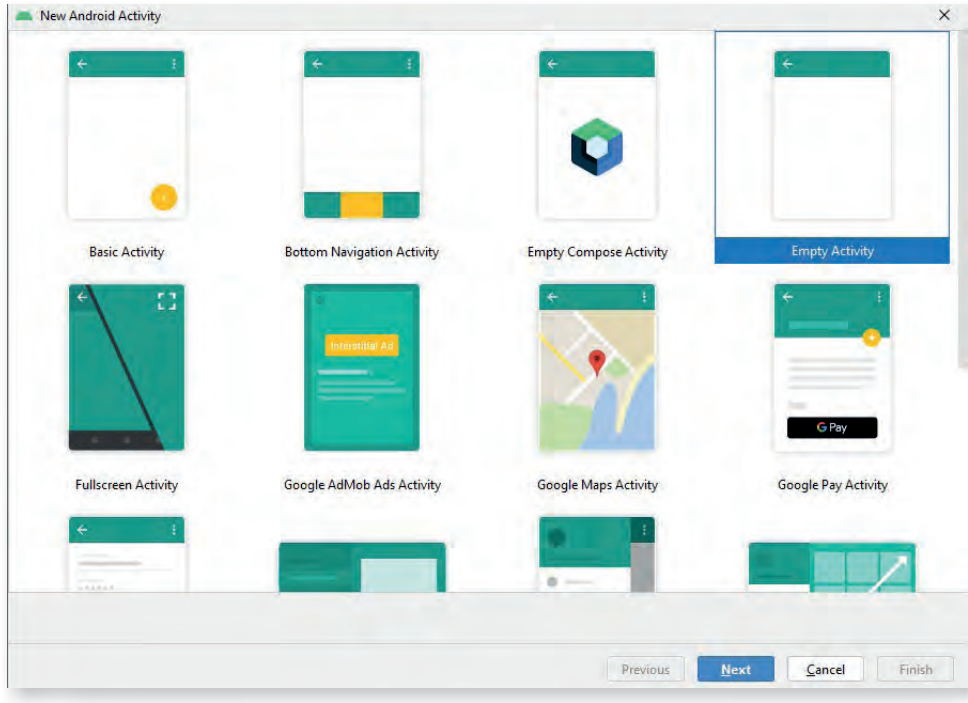
6.1.2. MainActivity

Mobil uygulama geliştirme ortamında oluşturulan projenin ana activitysinin java uzantılı kodlama dosyasıdır. MainActivity, **java** klasörünün içindeki **paket adı** ile gösterilen klasörde yer alır. Geliştirilen uygulamanın bu activity için gerekli olan java kodları bu dosya içine yazılır. Activityler aynı zamanda birer **xml** uzantıya sahip, dizayn (tasarım) dosyaları ile ilişkilidir. Açılan her activity için java uzantılı kod dosyası ve xml uzantılı tasarım dosyası bulunur. MainActivity için olan tasarım dosyası da **res** klasörü içinde bulunan **layout** klasöründeki **activity_main.xml** dosyasıdır.

MainActivity.java için yazılan her kod, activity_main.xml'i ve activity_main.xml içinde yazılan her kod veya oluşturulan her tasarım da MainActivity.java'yı ilgilendirir. Geliştirilen mobil uygulamaya yeni bir Activity eklenmesi için Görsel 6.3'te görüldüğü gibi java klasörü altında yer alan paket ismine farenin sağ tuşu ile tıklanıp, **New>Activity** üzerine gelinerek bir Activity seçilir veya **Gallery** seçeneğine tıklanarak açılan pencerede istenen bir Activity seçilir (Görsel 6.4).

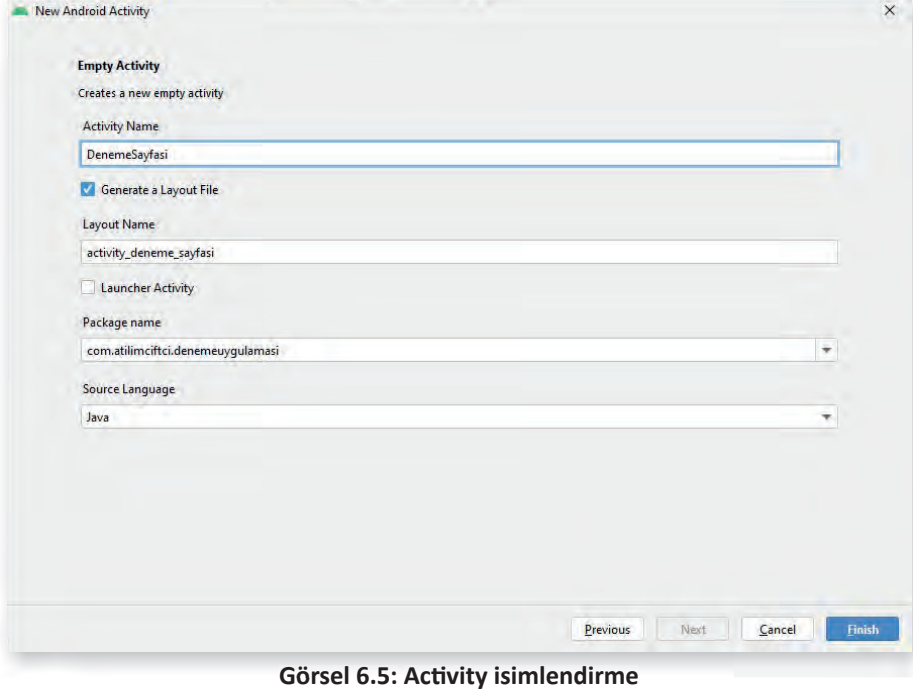


Görsel 6.3: Yeni Activity ekleme



Görsel 6.4: Activity seçenekleri

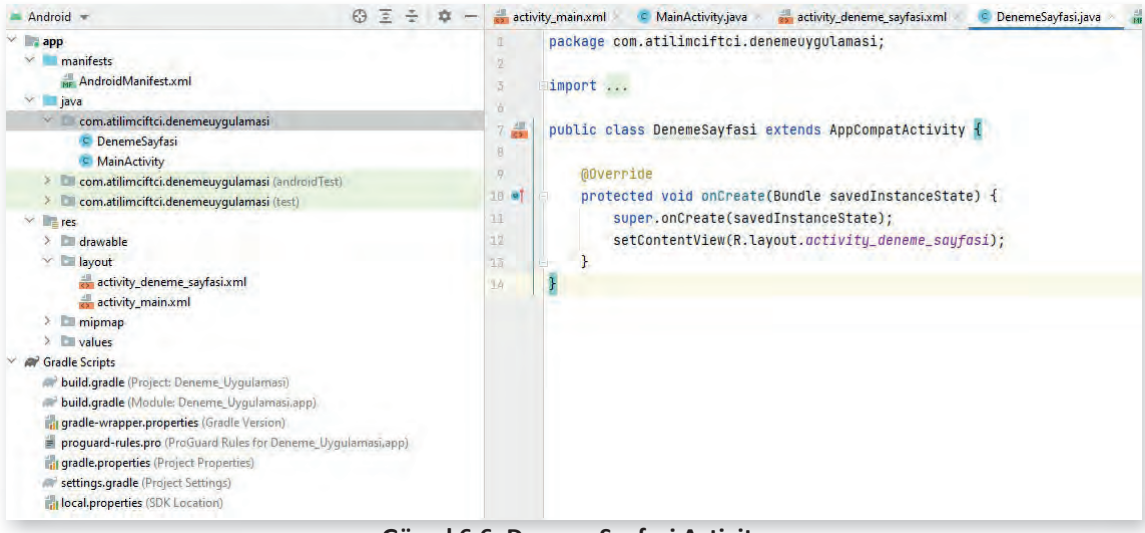
Activity seçimi yapıldıktan sonra Next tuşuna basılır veya seçilen Activity üzerine farenin sol tuşu ile çift tıklanır. Görsel 6.5'te olduğu gibi bir görüntü ile karşılaşılır.



Görsel 6.5: Activity isimlendirme

Activity Name alanına paket isminin altında yer alacak java uzantılı Activity'nin adı yazılır. **Generate a Layout File** seçeneği seçili olursa layout klasörü altında, oluşturulan Activity'ye ait bir tasa-

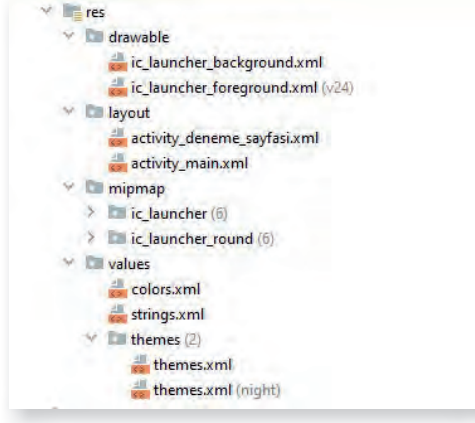
rım dosyası (layout) oluşturulur ve tasarım dosyasının ismi de otomatik olarak Activity Name ile ilişkilendirilir. Package name seçeneğinde ise hangi pakete ait olduğu sorulur. **Source Language** seçeneğinde ise oluşturulacak Activity'nin hangi dili desteklediği seçilir. Java dilinde kodlama ile uygulama tasarlamak için Java seçeneği seçilir ve sonra **Finish** seçeneğine tıklanır. **DenemeSayfasi** isminde java uzantılı yeni bir Activity, paket adı altında oluşturulur. Bu Activitye ait bir tasarım dosyası da layout altında **activity_deneme_sayfasi** ismi ile oluşturulur (Görsel 6.6).



Görsel 6.6: DenemeSayfasi Activity

6.1.3. res

res klasörü, **resources (kaynak)** kelimesinin kısaltılmış biçimi olarak adlandırılır. İçeriğinde yer alan dosya ve klasörler, mobil uygulamanın içerdiği ana dosyaları oluşturur (Görsel 6.7).



Görsel 6.7: res klasörü içeriği

Resources klasörünün altında yer alan klasörlerin her biri farklı bir amaca hizmet eder. Buraya dışarıdan da ekleme yapılabilir.

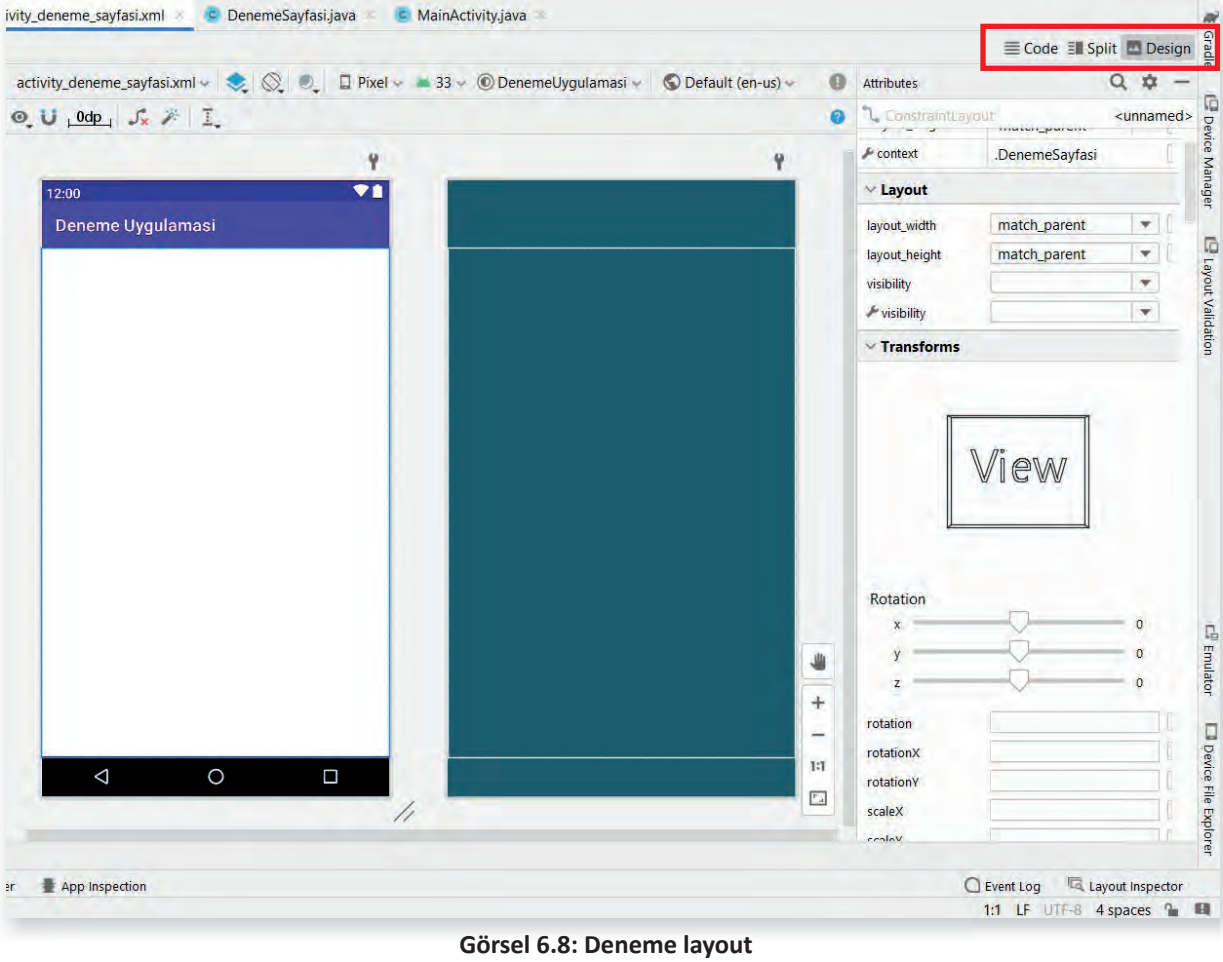
6.1.3.1. drawable

Geliştirilen mobil uygulamaya ait bitmap (.png, .jpg, .gif) ve xml dosyaları **drawable** klasörü içine konumlandırılır. Yüklenen resimler, farklı ekran boyutlarında kullanılmak üzere şu şekilde klasörlenir:

- **/drawable-ldpi:** Düşük yoğunluklu ekranlar için
- **/drawable-mdpi:** Orta yoğunluklu ekranlar için
- **/drawable-hdpi:** Yüksek yoğunluklu ekranlar için
- **/drawable-xhdpi:** Ekstra yüksek yoğunluklu ekranlar için
- **/drawable-xxhdpi:** Ekstra-ekstra yüksek yoğunluklu ekranlar için
- **/drawable-xxxhdpi:** Ekstra-ekstra-ekstra yüksek yoğunluklu ekranlar için

6.1.3.2. layout

layout, mobil geliştirme platformunda geliştirilen uygulamanın görsel arayüzlerinin bulunduğu bölümdür. Oluşturulan bu arayüzler xml formattadır. Bu xml dosyaları Görsel 6.8'de görüldüğü gibi **Code**, **Split** veya **Desing** şekilde düzenlenebilir.



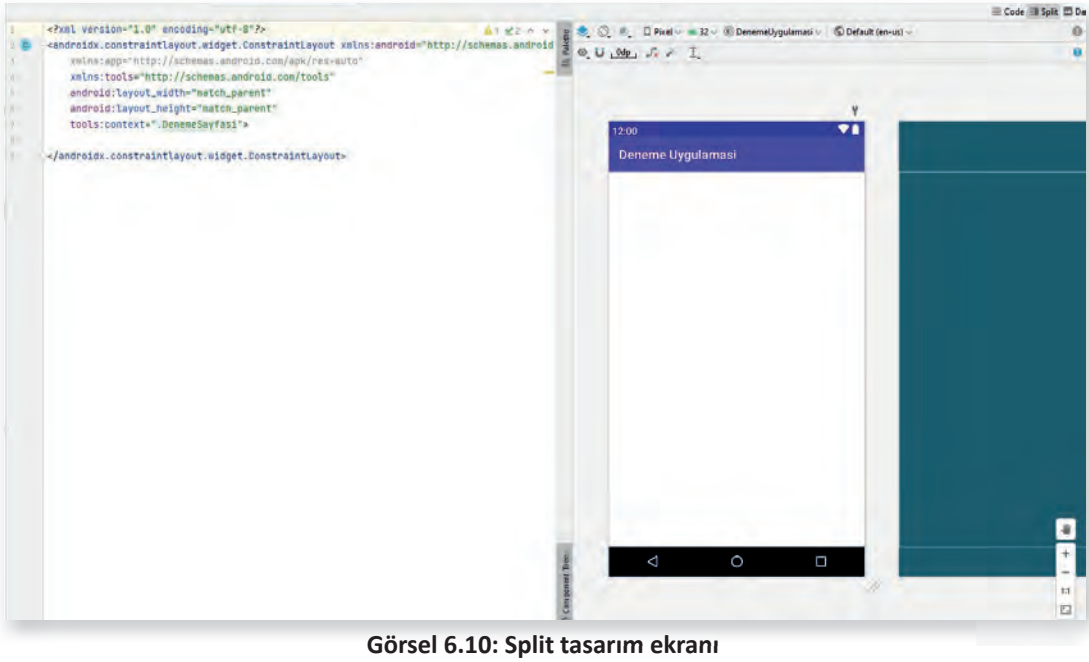
Görsel 6.8: Deneme layout

Code ekranında düzenleme istenirse Görsel 6.9'da olduğu gibi sadece xml kodları ile tasarım yapılan ekranla karşılaşılır. Bu ekrandan yeni ögeler oluşturabilir, bulunan ögelerin özellikleri de değiştirilebilir.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DenemeSayfasi">

</androidx.constraintlayout.widget.ConstraintLayout>
```

Görsel 6.9: Code ekranı

Split seçeneği ile düzenleme istenirse Görsel 6.10'da olduğu gibi hem xml kodlarının hem de tasarım ekranının göründüğü bir yapı ortaya çıkar. Kod ekranında yazılan xml kodları ile eş zamanlı olarak tasarım ekranına eklenen veya düzenlenen öge değişiklik gösterir. Tasarım ekranına eklenen veya değiştirilen öge de kod ekranında eş zamanlı olarak eklenir veya silinir.



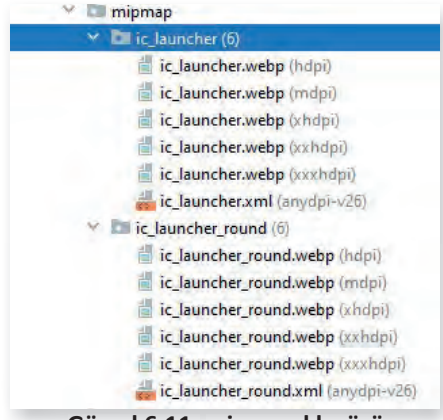
Görsel 6.10: Split tasarım ekranı

Desing ekranında düzenleme istenirse Palette bölümünden ögeler seçilerek tasarım ekranına eklenir ve **Component Tree** bölümünden de ögelerin yerleşimleri ayarlanır. Bu ekranda herhangi bir kodlama bölümü görülmez.

Üç tasarım ekranı da zaman zaman farklı şekillerde kullanılması uygun olan yapıdadır. Farklı işlemler, farklı şekillerde daha portatif çalışabilir.

6.1.3.3. mipmap

mipmap, mobil geliştirme ortamında geliştirilen uygulamanın başlatma simgelerinin yerleştirildiği bölümdür. Görsel 6.11'de görüldüğü gibi **ic_launcher** isimli default iconu farklı şekillerde kaydedilmiştir. Bu kayıtlar, drawable klasörleme biçimine benzer şekilde ve farklı yoğunluktaki (dp) ekran modlarına göre ayrı ayrı çalışır.



Görsel 6.11: mipmap klasörü

Burada eklenen iconlar, AndroidManifest içinde şu şekilde çağrılır:

6.1.3.4. values

```
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
```

values klasörü içinde, mobil uygulama geliştirme ortamında geliştirilen projede kullanılan sabit değerler tutulur. Bir mobil uygulama geliştirildiğinde themes.xml, colors.xml, strings.xml şeklinde default olarak üç dosya oluşur. Tutulan değerler, klasik olarak "Anahtar ↔ Değer" mantığı ile tutulur. Tutulan bu değerlere daha sonra uygulama tasarım ekranlarından (layout) veya Activity-lerden metotlar aracılığı ile ulaşılabilir.

- **themes.xml**

themes.xml, mobil geliştirme platformunda geliştirilen uygulamalarda kullanılan stiller ve temaların tutulduğu bölümdür.

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.DenemeUygulamasi" parent="Theme.MaterialComponents.Day-
Night.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
```

```
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPri-
maryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

- **colors.xml**

colors.xml, geliştirilen mobil uygulamanın içinde kullanılmak üzere renklerin hexadecimal (16'lık sayı tipinde) değerlerini tutar.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

- **strings.xml**

strings.xml, mobil uygulama geliştirme platformunda geliştirilen mobil uygulama içinde kullanılacak stringlerin saklanıldığı dosyadır. Uygulamalarda birden fazla dil desteği sunulacaksa her dil için farklı bir string.xml dosyası oluşturmak gereklidir.

```
<resources>
    <string name="app_name">Deneme Uygulamasi</string>
</resources>
```

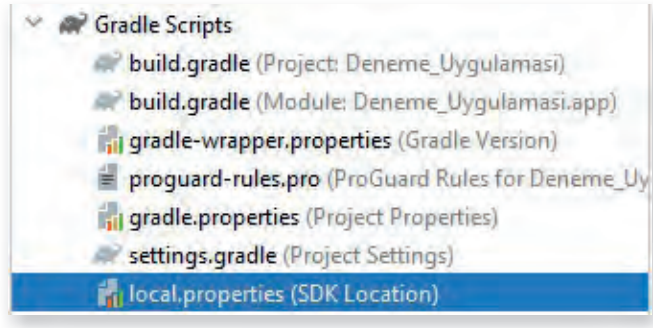
NOT: themes, colors, strings ismindeki xml uzantılı dosyalarda tanımlanan ögeler, layout içinden ulaşılarak kullanılabilir. Bunun için "@dosya_adı/name" ifadesi şeklinde çağrılması mümkündür.

```
android:textColor="@color/purple_500"
android:text="@string/mesaj_bildirimi"
```

NOT: values klasörü içine default değerler dışında dimension.xml (boyut değerleri için), bool.xml (bool tipi değerler için), integer.xml (int veri tipinde değerler tutmak için), integer_array.xml (birden fazla integer tipte sabit değerler için), typed_array.xml (birden fazla farklı tipte sabit değerler için) dosyaları da kullanılabilir ve layout içinden aynı şekilde ulaşılabilir.

6.1.4. Gradle Scripts

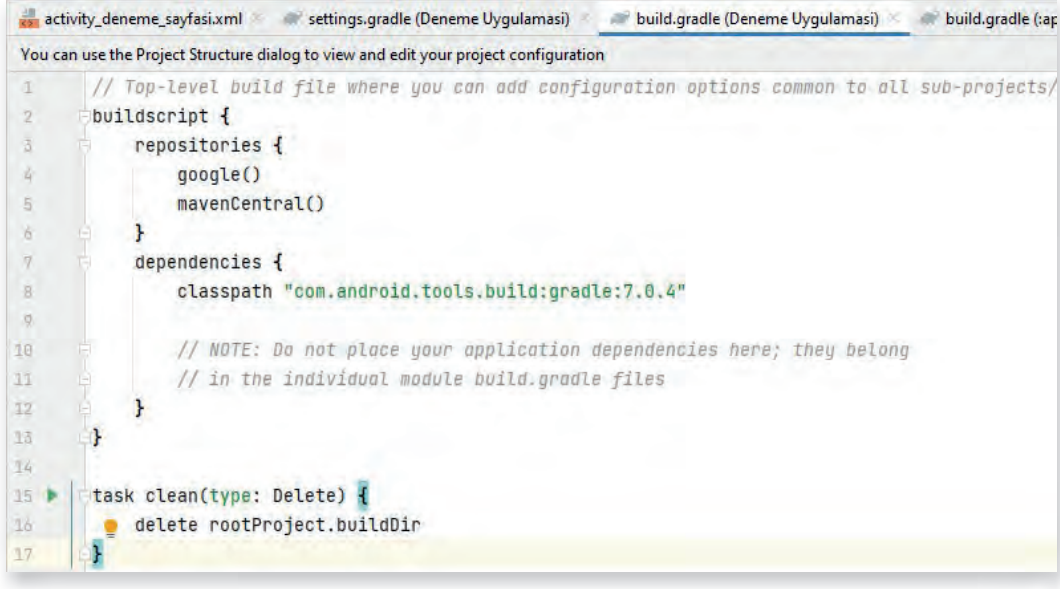
Mobil yazılım geliştirme ortamında Gradle Scripts; geliştirilen uygulamalar için test, inşa, dağıtım gibi yapı sistemini oluşturur. Oluşturulan Gradle dosyaları, proje geliştirme aşamasında yazılım geliştiriciye büyük kolaylıklar sağlar. Gradle dosyaları, apk imzaları ekleyebilir ve hata ayıklama modları düzenleyebilir. Gradle Scripts altında default olarak **build.gradle (Project), build.gradle (Module)** dosyaları bulunur (Görsel 6.12). Bu dosyalar içinde Gradle Scripts'e ait konfigürasyon yapısı bulunur. Project ve Module olan build dosyaları birbiriyle karıştırılmamalıdır.



Görsel 6.12: Gradle Scripts

6.1.4.1. build.gradle (Project)

Procet isimli build.gradle, geliştirilen projeye ait build dosyasıdır. Bu dosya içinde yapılacak herhangi bir değişiklik, mobil uygulama projesindeki tüm modüllere uygulanır. Açılan uygulama, Görsel 6.13'teki yapıya benzer şekildedir.



Görsel 6.13: build.gradle içeriği

NOT:

Gradle, sürekli güncellenir. Bu nedenle Görsel 6.13'teki classpath ile başlayan 7.0.4 zamanla değişiklik gösterir ve ileri versiyonlar uygulama tarafından önerilir.

6.1.4.2. build.gradle (Module)

Bu dosya, mobil geliştirme ortamında geliştirilen projenin her modülünde yer alır. Bu dosya içinde yapılacak değişiklikler, belirli bir uygulama modülünde geçerli olur. İçeriğinde geçerli modül için uygulama kimliği, sürüm adı, sürüm kodları, minimum ve maksimum sdk sürümleri, paket adları vb. içerir. Görsel 6.14'te açılan bir build.gradle (Module) içeriği yer alır.

```
plugins {
    id 'com.android.application'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.atilimciftci.denemeuygulamasi"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

Görsel 6.14: build.gradle (Module) içeriği

- **android:** Mobil uygulamasının oluşturma seçeneklerini yapılandırmak için kullanılır.
  - ► **compileSdk:** Mobil uygulamanın API (Application Programming Interface) seviyesini tanımlamak için kullanılır. Geliştirilen mobil uygulama burada tanımlanan seviyeyi veya daha alt seviyeleri destekler.
- **defaultConfig:** Geliştirilen mobil uygulamanın default konfigürasyon ayarlarını yapılandırmak için kullanılır.
  - ► **applicationId:** Geliştirilen mobil uygulamayı yayınlama sırasında kullanılan benzersiz kimliği tanımlamak için kullanılır.

- ► **minSdk:** Geliştirilen mobil uygulamayı çalıştırabilmek için gereken en düşük API seviyesini belirtir.
- ► **targetSdk:** Geliştirilen mobil uygulamayı test etmek için kullanılan API seviyesini belirtir.
- ► **versionCode:** Geliştirilen mobil uygulamanın sürüm kodudur. Her güncellemede sürüm kodunun bir veya daha fazla artırılması gerekir.
- ► **versionName:** Geliştirilen mobil uygulamanın sürüm adını tanımlar. Her güncellemede rastgele bir oranda artırılması gerekir.

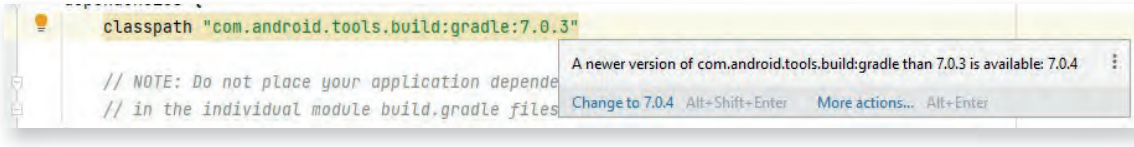
- **buildTypes:** Mobil uygulamalar geliştirilirken, derlenirken ve imzalı versiyon hazırlanırken Gradle'ın kullandığı bazı özelliklerin düzenlenmesini ve kullanılacak bazı konfigürasyonların daha kolay yönetilmesini sağlar.
  - ► **release:** Geliştirilen mobil uygulamada açılan buildTypes, iki adet buildTypes tanımlar. Bunlardan biri release iken diğeri debugtır. Daha çok geliştirme aşamasında kullanılan debug modu, standart olarak açıkça gösterilmez. Release ise pro guard ayarlar uygulayıp standart güvenlik önlemi almayı sağlar.
- **compileOptions:** Geliştirilen mobil uygulamada java derleme seçeneklerinin yer aldığı bölümdür.
- **Dependencies:** Mobil uygulama geliştirme platformunda tasarlanan projenin beraberinde derlenmesi gereken kaynakları içeren bölümdür. Geliştirilen mobil uygulamada kullanılması istenen paketler vb. burada projeye implement edilir.

UYARI: Projede gradle içinde güncelleme varsa build.gradle içine girildiğinde güncellenmesi gereken paket, Görsel 6.15'te olduğu gibi sarı bir arka plan ile belirir.

```
dependencies {
    classpath "com.android.tools.build:gradle:7.0.3"
```

Görsel 6.15: Güncelleme gerektiren paket

Paketin üzerine farenin sol tuşu ile tıklanıp birkaç saniye beklendiğinde yeni sürümünün ne olduğu gösterilir (Görsel 6.16).



Görsel 6.16: Güncellemesi istenen paketin güncel versiyonu

Eski paket versiyonu silinip önerilen yeni versiyon yazıldığında mobil uygulama geliştirme uygulamasının sağ üst köşesinde Görsel 6.17'de olduğu gibi seçenekler görülür.

Görsel 6.17: Sync Now seçeneği

Sync Now seçeneğine basıldığı zaman internet bağlıysa gradle dosyası içindeki paket güncellenir ve mobil uygulamaya senkronize edilir.

NOT: Proje dosya yapıları ve yapılandırma ayarları hakkında daha detaylı ve güncel bilgiler için https://developer.android.com/docs adresinde yer alan dokümanları inceleyiniz.

6.2. VIEW BINDING

View Binding, kullanıcı arayüzü (UI) bileşenlerini içinde barındıran layout dosyası ile ".java" uzantılı kod dosyasını birbirine bağlamayı hızlı ve kolay bir biçimde sağlayan Google'a ait bir kütüphanedir. Bu tarz kütüphanelere de Jetpack ögesi ismi verilmiştir. Görsel 6.18'e benzer bir biçimde, UI'deki (kullanıcı arayüzü) tüm farklı görsel ögelerin bir kütüphanede belirli bir düzen içinde tutulması mantığı ile çalışır.

Görsel 6.18: Kütüphaneleme mantığı

View Binding sayesinde layout dosyası içindeki mobil uygulama tasarım ekranına eklenen ögeler, "findViewBinding" metoduna gerek kalmadan her yerden erişilebilir hâle getirilir. Bu sayede hem hızlı bir çalışma ortamı sağlanır hem de birçok gereksiz kod yazılmaz. Küçük uygulamalarda fark edilmese de büyük uygulamalarda yer alan birçok ögenin findViewBinding ile tek tek eklenmesi çok uygun değildir.

View Binding kütüphanesinden önce çıkartılan ve bu işi yapan Data Binding kütüphanesi de vardır. Data Binding kullanımı, View Binding kadar esnek değildir. Derleme işlemi sırasında zaman açısından olumsuzluklara yol açar. Bu nedenle View Binding kütüphanesi üretilmiş ve bu sorunların da önüne geçilmiştir.

NOT: Jetpack, mobil uygulama geliştiricilerin daha kaliteli uygulamalar yazabilmesi Google tarafından üretilen, kütüphane ve araçları içinde barındıran bir pakettir. Daha detaylı bilgi için https://developer.android.com/jetpack/getting-started adresini ziyaret ederek dokümanları inceleyiniz.

View Binding kullanımı için https://developer.android.com/topic/libraries/view-binding adresine bakılırsa (Mobil uygulama geliştirme aşamasında yenilik ve kullanımdaki güncelleştirmeler, Google tarafından bu sitede paylaşılmaktadır.) **Setup instruction** seçeneğinde, Java dilinde geliştirme yapıldığı için **Groovy** seçeneği seçilir. Görünüm bağlama modül düzeyinde yapılır. Bu bağlamayı etkinleştirmek için de modül düzeyindeki **build.gradle (Module)** dosyası içinde viewBinding true olarak ayarlanır. Bir başka deyişle geliştirilen uygulamada öncelikle build.gradle (Module) dosyası açılır ve şu kod eklenir:

```
android {
    ...
    buildFeatures {
        viewBinding true
    }
}
```

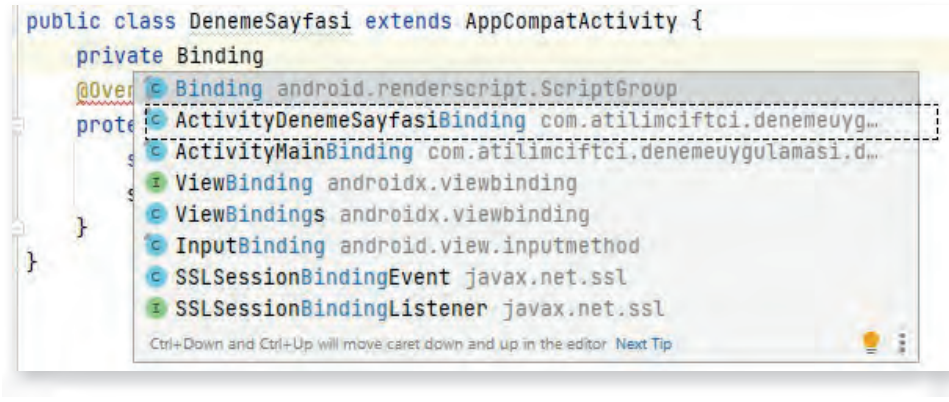
NOT: viewBinding kodu hâlihazırda android bloklarının arasına eklenir. Yeniden bir android bloku açmaya gerek yoktur. Ayrıca buildFeatures blokları da varsa tekrar bu bloku açmaya gerek yoktur. buildFeatures blokları içine **viewBinding true** yazılması yeterlidir.

View Binding için doküman sayfasında aşağıya inildiğinde Activity'de kullanımın nasıl olduğu belirtilir. Bu bölümde Java'ya tıklandığında şu kodlarla karşılaşılır:

```
private ResultProfileBinding binding;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ResultProfileBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
}
```

Kod içindeki `ResultProfileBinding` bölümü, build.gradle dosyasına eklenen viewBinding özelliği sayesinde Binding türünden oluşturulan bir sınıfı temsil eder. Geliştirilen mobil uygulamasında kullanılan java uzantılı her Activity için bu Binding sınıfı oluşturulmuş durumdadır. Yapılması gereken işlem, açılan Activity'nin sınıf blokları içindeyken bu sınıftan bir nesne türetmektir. "private Binding" yazılırsa Görsel 6.19'da görüldüğü gibi önerilerde ilgili Activity'ye uygun Binding sınıfı çıkacaktır. Activity ismi ile Activity ismine eklenmiş Binding kelimesi ile görülen (ör. ActivityDenemeSayfasiBinding) kütüphaneye tıklanarak seçilir. Sonrasında **binding** ismi ile bir nesne oluşturulur.

NOT: Nesne ismi önemli olmamakla birlikte profesyonel düzeyde uygulama geliştiriciler "binding" şeklinde kullanır. Her Activity için bu işlem bir defa yapılacağından ve "private" olarak tanımlanıp sadece bu Activity üzerinden erişilebileceğinden bir sorun oluşturmaz.



Görsel 6.19: ActivityDenemeSayfasiBinding önerisi

Sonrasında mobil uygulamanın içinde bulunulan Activity'ye ait "onCreate" yaşam döngüsünde değişiklik yapılması istenir. "`super.onCreate(savedInstance);`" kodu yerinde kalmakla birlikte, **`setContentView(R.layout.activity_deneme_sayfasi);`** kodunun değiştirilmesi gerekir. Uygulamanın default düzeyinde contentView, layout üzerinde yer alan xml uzantılı kullanıcı arayüzüne bağlanır fakat bu bağlantı kopartılarak yeni bir bağlantı oluşturulur. Activity'nin onCreate yaşam döngüsü blokları arasında bulunan `setContentView(R.layout.activity_deneme_sayfasi);` silinerek "`super.onCreate(savedInstanceState);`" altına kodlar şu şekilde eklenir:

```
binding = ActivityDenemeSayfasiBinding.inflate(getLayoutInflater());
View view = binding.getRoot();
setContentView(view);
```

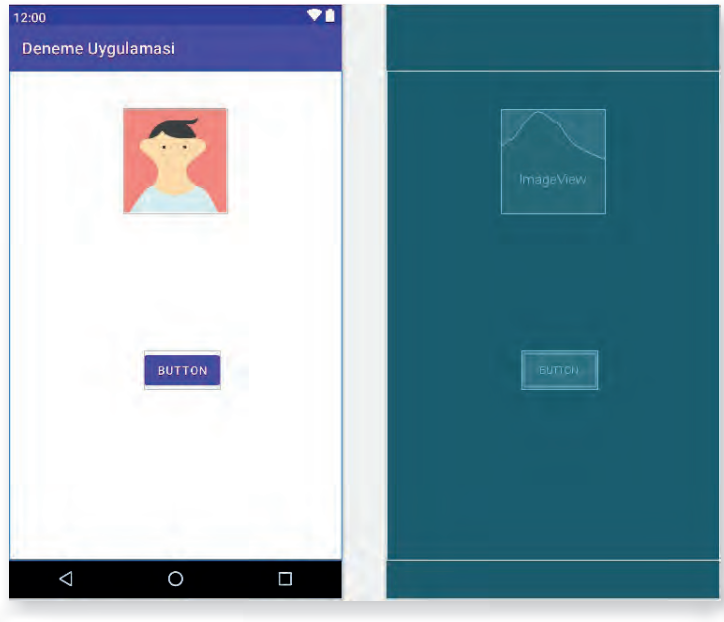
NOT:

"ActivityDenemeSayfasiBinding", Activity'ye eklenen kütüphanenin adıdır. Oluşturulan her Activity'de farklılık gösterir. Oluşturulan Activity ismi neyse o isimle değişiklik gösterir. View kırmızı gösterirse de üzerine farenin sol tuşu ile tıklanıp "Alt+Enter" tuşlarına basılarak View kütüphanesi Activity'ye dâhil edilir.

Eklentinin son şekli Görsel 6.20'de verilmiştir.

Görsel 6.20: Activity'de viewBinding etkinleştirme

Bu işlemden sonra layout üzerindeki her ögeye findViewById yazılmadan erişilebilir. İşlemin doğruluğunu denemek amacıyla "activity_deneme_sayfasi.xml" isimli layout dosyasına Görsel 6.21'de olduğu gibi bir imageView, bir de button nesnesi eklenir. Bu ögelere erişim sağlamak için "DenemeSayfasi.java" isimli Activity'de, **binding**. yazıldığında ögelerin erişilebilir olduğu görülür (Görsel 6.22).



Görsel 6.21: layout içinde görüntü

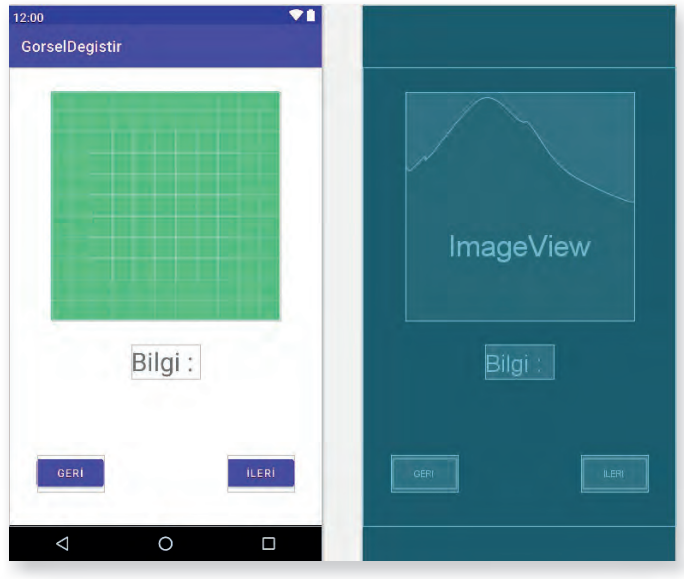
```
package com.atilimciftci.denemeuygulamasi;

import ...

public class DenemeSayfasi extends AppCompatActivity {
    private ActivityDenemeSayfasiBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityDenemeSayfasiBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);

        binding.
    }
}
```

```
getRoot()                                   ConstraintLayout
buttonOnayla                                Button
imageViewGorsel                             ImageView
equals(Object obj)                          boolean
cast                                        ((SomeType) expr)
hashCode()                                  int
toString()                                  String
getClass()   Class<? extends ActivityDenemeSayfasiBinding>
notify()                                    void
notifyAll()                                 void
wait()                                      void
Press Enter to insert, Tab to replace  Next Tip
```

Görsel 6.22: Activity içinde ögelere erişim

11. UYGULAMA: İşlem adımlarına göre butonlara tıklandığında ImageView üzerinde fotoğrafların değiştiği ve altında da fotoğraflara ait bilgilerin oluştuğu uygulamayı viewBinding kullanarak yapınız.

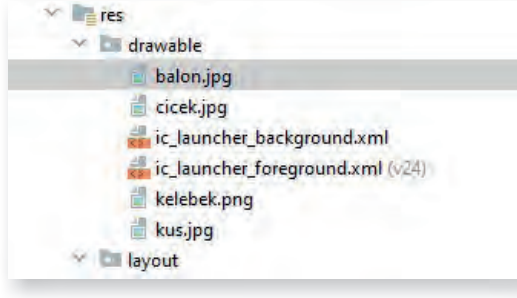
1. Adım: Yeni bir proje açarak projenin ismine "GorselDegistir" yazınız. "activity_main.xml" içine Görsel 6.23'te görüldüğü gibi bir adet ImageView, bir adet TextView ve iki adet Button ögesi ekleyiniz. Geri buttonunun onClick özelliğine "**geriGelme**", ileri buttonunun onClick özelliğine de "**ileriGitme**" yazınız. ConstraintLayout kullanarak Görsel 6.23'tekine benzer şekilde ögeleri kilitleyiniz.



Görsel 6.23: GörselDegistir arayüz tasarımı

2. Adım: İnternetten telifsiz olarak kullanılan dört adet görsel bulunuz. Uygulamada balon, kuş, çiçek ve kelebek görselleri kullanınız. Görsellerinize mümkün olduğunca basit isimler (balon.jpg, cicek.jpg vb.) veriniz ve görselleri isimlendirirken Türkçe karakter kullanmayınız. Mobil uygulama geliştirme platformunda tasarlanan uygulamaların uzun ve karmaşık isimlendirmelerde hatalara yol açtığını unutmayınız.

3. Adım: Bulduğunuz görselleri isimlendirdikten sonra sürükle bırak ile res klasörü altındaki drawable klasörüne atınız. Klasör ismi olarak drawable-v24 vb. gelirse -v24'ü siliniz, klasörün isminin sadece drawable olmasına dikkat ediniz. Ögeler Görsel 6.24'te olduğu gibi geldikten sonra ögelere tıklayarak açınız ve görsellerin kullanılabilir olduğundan emin olunuz. Kullanılmayan görsel varsa o görseli silip yerine yenisini ekleyiniz.

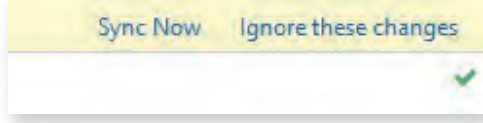


Görsel 6.24: drawable içeriği

4. Adım: Gradle Scripts altındaki build.gradle (Module) dosyasını açınız ve android blokları arasına girerek viewBinding etkinleştirmek için gereken şu kodu ekleyiniz:

```
buildFeatures{
    viewBinding true;
}
```

Daha sonra Görsel 6.25'te görüldüğü gibi mobil geliştirme platformunun sağ üst köşesinde çıkan Sync Now kısmına tıklayarak projeye senkronize edilmesini sağlayınız. İşlem bittikten sonra build.gradle'ı kapatınız.



Görsel 6.25: Senkronizasyon

5. Adım: MainActivity içine dönerek MainActivity sınıfının blokları arasına giriniz ve "private Bindig" yazınız. Öneriler arasında gelen "private ActivityMainBinding"e tıklayınız ve boşluk bırakıp, binding yazarak nesnesini de belirtiniz.

6. Adım: MainActivity sınıfındayken onCreate yaşam döngüsüne giriniz ve "setContentView(R.layout.*activity_main*);" kodunu siliniz. Aynı blok içinde "super.onCreate(savedInstanceState);" kodunun altına geliniz ve şu kodları ekleyiniz:

```
binding = ActivityMainBinding.inflate(getLayoutInflater());
View view = binding.getRoot();
setContentView(view);
```

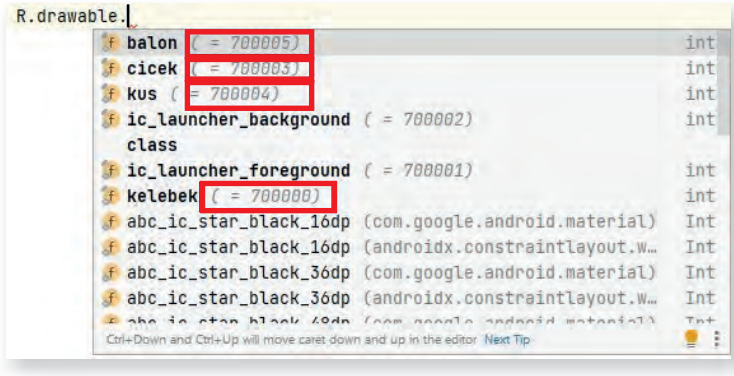
İşlem adımlarının sonucu Görsel 6.26'da verilmiştir.

```
package com.atilimciftci.gorseldegistir;

import ...

public class MainActivity extends AppCompatActivity {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);
    }
}
```

Görsel 6.26: Uygulamada viewBinding etkinleştirme

7. Adım: java klasörü altındaki paket ismine sağ tuş ile tıklayarak yeni bir sınıf oluşturunuz. Sınıfa "Gorsel" ismini veriniz. Bu sınıfta hem görseller hem de görsellerin bilgileri tutulacaktır. Açılan sınıf içine bir adet String veri tipinde "bilgi" isimli değişken, bir adet int veri tipinde "siraNo" isimli bir değişken bir de "int" veri tipinde "foto" isimli bir değişken tanımlayınız.

Görselleri tutacağınız foto isimli değişkenin int olmasının nedeni, drawable klasörüne atılan görsellerin her birine aslında birer "id" verilmesinden kaynaklanır. Bu numaralara Görsel 6.27'de görüldüğü gibi MainActivity'deki onCreate metodu içinde **"R.drawable."** yazılırsa erişilebilir. Görsellere ait id'ler bu sınıf içinde tutulur.



Görsel 6.27: Görsellerin id'leri

8. Adım: Gorsel sınıfının içine bir constructer oluşturunuz. Constructer, bir sınıf çağrıldığında ilk olarak çalıştırılacak metottur. Kodlarınızı şu şekilde yazınız:

```
public Gorsel(String bilgi,int siraNo,int foto){
    this.bilgi=bilgi;
    this.siraNo=siraNo;
    this.foto=foto;
}
```

Görsel sınıfının içeriği Görsel 6.28'de verilmiştir.

```
package com.atilimciftci.gorseldegistir;

public class Gorsel {
    String bilgi;
    int siraNo;
    int foto;

    public Gorsel(String bilgi,int siraNo,int foto){
        this.bilgi=bilgi;
        this.siraNo=siraNo;
        this.foto=foto;
    }
}
```

Görsel 6.28: Gorsel sınıfının içeriği

9. Adım: MainActivity.java sınıfının içine dönünüz. onCreate metodu içinde görselleri ve bilgileri sınıf üzerine ekleyiniz. Kodları şu şekilde yazınız:

```
Gorsel balon = new Gorsel("Sarı Balon",1,R.drawable.balon);
Gorsel cicek = new Gorsel("Mavi Çiçek",2,R.drawable.cicek);
Gorsel kelebek = new Gorsel("Mavi Kelebek",3,R.drawable.kelebek);
Gorsel kus = new Gorsel("Yuvada Kuş",4,R.drawable.kus);
```

İşlemden sonra bu verileri bir ArrayListe yüklemek gerektiğini unutmayınız. Bunun için onCreate metodunun dışına çıkıp, onCreate metodunun üzerinde fakat MainActivity blokları arasında olacak şekilde "**ArrayList<Gorsel> gorselArrayList;**" yazarak Gorsel sınıfından türetilen nesneleri tutacak gorselArrayList isimli bir ArrayList tanımlayınız. Tekrar onCreate içine girerek bu ArrayListi "**gorselArrayList = new ArrayList<>();**" kodlarını yazarak başlatınız. Gorsel sınıfından oluşturulan nesnelerin altında olacak biçimde de bu ArrayListe verileri gönderiniz. Kodları şu şekilde yazınız:

```
gorselArrayList.add(balon);
gorselArrayList.add(cicek);
gorselArrayList.add(kelebek);
gorselArrayList.add(kus);
```

Uygulama açılır açılmaz ilk görselin (0. kayıtlı olan balon) imageView üzerinde, ilk kayda ait bilginin de textView ekranında görünmesi için onCreate blokları arasında şu kodları yazınız:

```
binding.imageViewGorsel.setImageResource(gorselArrayList.get(0).foto);
binding.textViewBilgi.setText("Bilgi : " + gorselArrayList.get(0).bilgi);
```

Ardından buttona tıkladıkça sıra numaralarını tutması için MainActivityde int veri tipinde **seciliSiraNo** isminde bir değişken tanımlayınız ve değişkenin değerini onCreate içinde **0** olarak atayınız. İşlem sonucu Görsel 6.29'da verilmiştir.

```
package com.atilimciftci.gorseldegistir;
import ...

public class MainActivity extends AppCompatActivity {
    private ActivityMainBinding binding;
    ArrayList<Gorsel> gorselArrayList;
    int seciliSiraNo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);

        gorselArrayList = new ArrayList<>();
        //Veri
        Gorsel balon = new Gorsel( bilgi: "Sarı Balon", siraNo: 1,R.drawable.balon);
        Gorsel cicek = new Gorsel( bilgi: "Mavi Çiçek", siraNo: 2,R.drawable.cicek);
        Gorsel kelebek = new Gorsel( bilgi: "Mavi Kelebek", siraNo: 3,R.drawable.kelebek);
        Gorsel kus = new Gorsel( bilgi: "Yuvada Kuş", siraNo: 4,R.drawable.kus);
        //ArrayList Yüklemesi
        gorselArrayList.add(balon);
        gorselArrayList.add(cicek);
        gorselArrayList.add(kelebek);
        gorselArrayList.add(kus);

        binding.imageViewGorsel.setImageResource(gorselArrayList.get(0).foto);
        binding.textViewBilgi.setText("Bilgi : " + gorselArrayList.get(0).bilgi);
        seciliSiraNo=0;
    }
}
```

Görsel 6.29: MainActivity ArrayList hâli

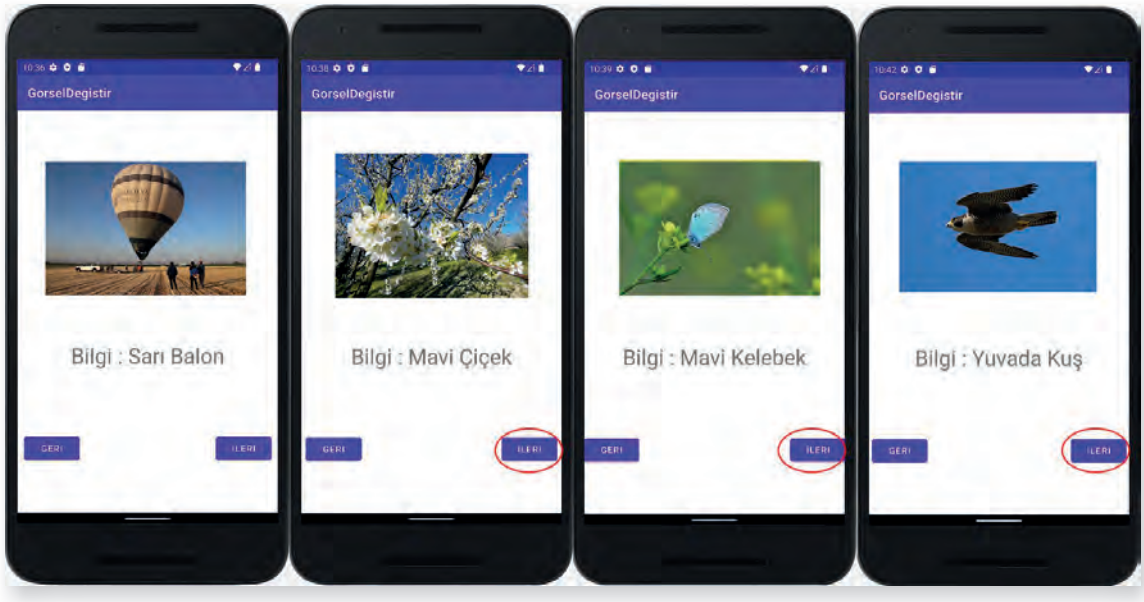
10. Adım: MainActivity.java dosyası içinde MainActivity sınıfı blokları arasına onCreate metodundan çıkarak "geriGelme" ve "ileriGitme" metotlarını hazırlayınız. Görünüme tıklandığı zaman çalışacağı için metotlar, View sınıfından view nesnesi şeklinde bir adet parametre alacaktır. Metotlar içinde secilenSiraNo değerini "geriGelme" metodunda 1 azaltıp "ileriGitme" metodunda 1 artırınız. Bu değişken sayesinde hangi ArrayList bilgisinin getirileceği bilinir. Ayrıca içeride birer kontrol kodu yazarak 0'dan büyük olmamasını veya ArrayListin eleman sayısından büyük olmamasını kontrol ediniz. "geriGelme" metodu için kodlar şu şekildedir:

```
public void geriGelme(View view){
    if(seciliSiraNo>0){
        seciliSiraNo--;
        binding.imageViewGorsel.setImageResource(gorselArrayList.get(seciliSi-
raNo).foto);
        binding.textViewBilgi.setText("Bilgi : " + gorselArrayList.get(secili-
SiraNo).bilgi);
    }
}
```

11. Adım: "ileriGitme" metodu için kodlar şu şekildedir:

```
public void ileriGitme(View view){
    if(seciliSiraNo<gorselArrayList.size()-1){
        seciliSiraNo++;
        binding.imageViewGorsel.setImageResource(gorselArrayList.get(seciliSi-
raNo).foto);
        binding.textViewBilgi.setText("Bilgi : " + gorselArrayList.get(secili-
SiraNo).bilgi);
    }
}
```

Uygulama çalıştırıldığındaki görüntü Görsel 6.30'da verilmiştir.



Görsel 6.30: GorselDegistir uygulama çıktısı

NOT: gorselArrayList.size() metodu, ArrayListin içindeki eleman sayısını verir. Yazılan bu metottan dönen değer 4'tür fakat 4. eleman, 3. indist numarasında olduğu için program son görselde hata verir. Bu hatayı önlemek için **gorselArrayList.size()-1** ifadesi kullanılır.

SIRA SİZDE: Sınıf arkadaşlarınızdan 10 tanesinin "Okul Numarası, Ad, Soyad, Sınıf ve Fotoğraf" bilgilerini "Arkadaslar" isimli bir class ile kodlama esnasında kaydedip, bu bilgileri mobil ekranda buttonlar ile tek tek ekrana getiren uygulamayı viewBinding yöntemini de kullanarak tasarlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. "Arkadaslar" isimli yeni bir sınıf oluşturdu.		
2. "Arkadaslar" isimli sınıfta belirtilen alanları oluşturdu.		
3. "Arkadaslar" isimli sınıfta Constructor oluşturdu.		
4. Görselleri drawable içine uygun biçimde yükledi.		
5. activity_main üzerinde gereken tasarımı oluşturdu.		
6. viewBinding yapısını oluşturdu.		
7. MainActivity sınıfında Arkadaslar sınıfından nesne türeterek içeriğindeki bilgilerin girişini yaptı.		
8. Buttonlara ait metotları oluşturdu.		
9. Metotlar içinde bilgileri ögeler üzerine yükledi.		

6.3. ACTIVITY YAPISI

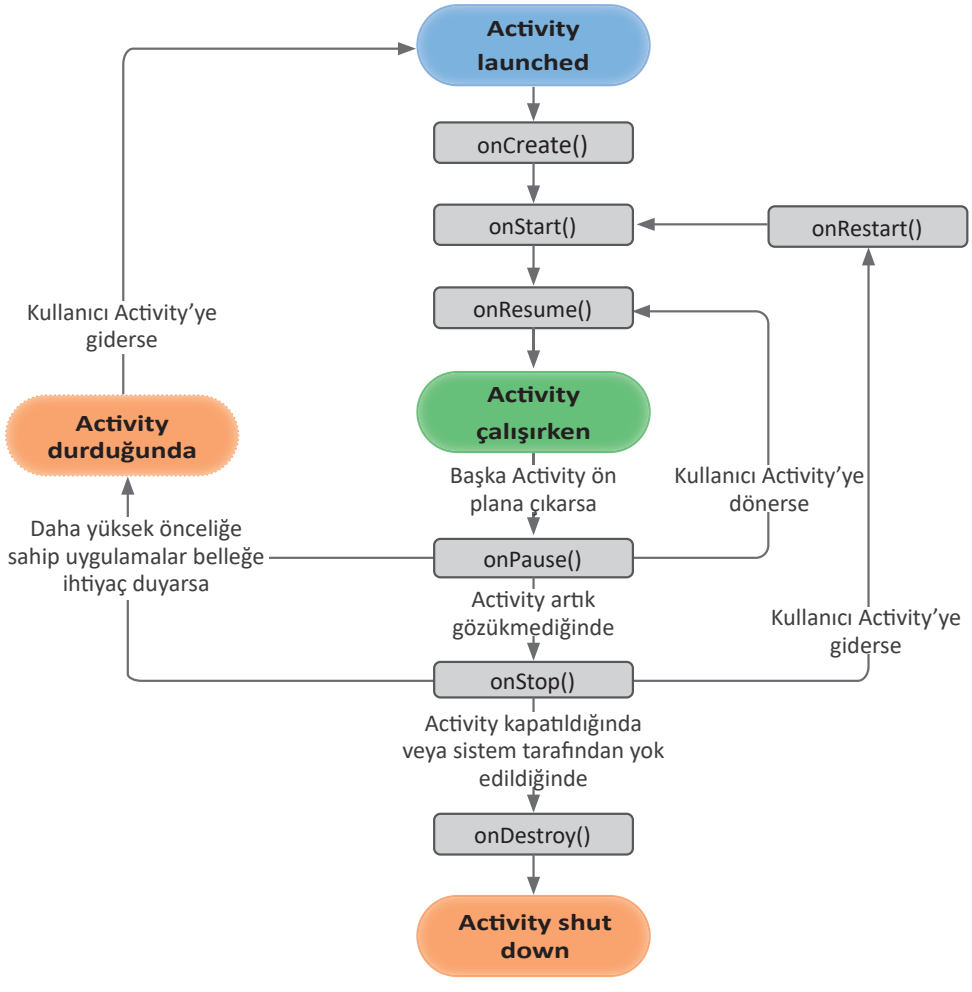
Activity, bir kullanıcı arayüzüne (UI) sahip mobil uygulama ekranını temsil eder. Bir sosyal medya uygulamasında arkadaş listesini gösteren bir Activity, mesajlar sayfasında kişilerin mesajlarını gösteren bir Activity, üye girişi için Activity örnek olarak gösterilebilir. Bir mobil uygulamada en az bir tane Activity olmalıdır. Bu Activity, "Ana Aktivite", "Başlatıcı Aktivite" veya "Main Activity" olarak adlandırılır. Bir mobil uygulama birden fazla Activity içerse de bunlardan biri ana aktivite olmak zorundadır. Uygulama ilk başlatıldığında ana aktivite devreye girmelidir. Birden fazla Activitye sahip mobil uygulamalarda, uygulama ilk çalıştırıldığında yüklenecek Activity, AndroidManifest.xml içinde Görsel 6.31'de belirtilen bölümden değiştirilebilir.

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Görsel 6.31: Main Activity değiştirme

6.3.1. Activity Yaşam Döngüleri (Activity Life Cycle)

Activityler, belirli bir döngü içinde çalışan yapılardır. Ekranın çevrilmesi, uygulamanın arka planına bakılması, başka bir Activity açılması, geri tuşuna basılması vb. eylemler sırasında uygulamanın arka planında yaşam döngüsü denilen bir yapıya göre işlemler yapılır. Yaşam döngüsünde yer alan her bir işleme yönelik bir metot bulunur. Bu metotlar, Görsel 6.32'de verilmiştir.



Görsel 6.32: Activity yaşam döngüsü

6.3.1.1. onCreate()

onCreate(), bir Activity çağrıldığı zaman arka planda ilk olarak devreye giren metottur. Yapı itibarıyla constructera (kurucu metot) benzetilebilir. Uygulama çalışır çalışmaz Activity içinde gerçekleşmesi istenen her olay onCreate metoduna yazılır. Activity her başlatıldığında yalnızca bir defa çalışır.

6.3.1.2. onStart()

onStart(), onCreate() metodu çalıştıktan sonra veya Activity tekrar çalıştırıldıktan sonra devreye girer. Arka plandaki başka bir uygulamaya geçip tekrar mobil uygulamada Activity'ye dönüldüğünde de çalıştırılır.

6.3.1.3. onResume()

onResume(), onStart() metodundan sonra kullanıcının odağı Activity'den uzaklaşıncaya kadar çalışan bölümdür. Örneğin telefon çalarsa, arka plandaki uygulamalar kontrol edilirse, Activity'den çıkılırsa, başka uygulamalara geçiş yapılırsa bu metottan çıkılır. İlgili mobil uygulamada Activity'ye dönülürse onResume() metodu tekrar çalıştırılır.

onPause()

onPause(), Activity arka plana alındığı zaman çalışacak metottur. Uygulama içinde bir olayın arka plana alındığında çalışması istenirse onPause() metodu kullanılır. Uygulama arka plana her alındığında tekrar tekrar çalışır.

6.3.1.5. onStop()

onStop(), Activity arka plana alındığında onPause() metodundan hemen sonra çalışır. onStop() metodu çalıştıktan sonra ya Activity'ye tekrar dönülür ve sırasıyla onStart(), onResume() metotları çalıştırılır ya da Activity kapatılır ve onDestroy() metodu çalıştırılır.

6.3.1.6. onDestroy()

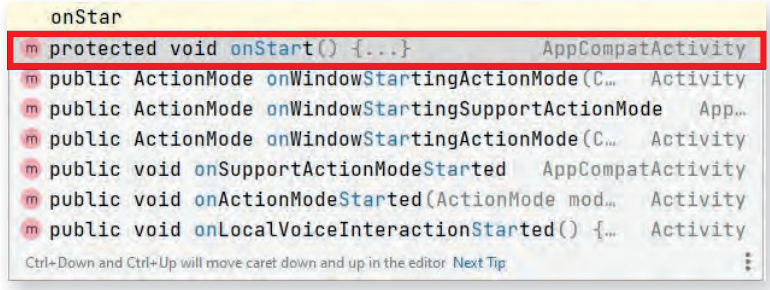
onDestroy(), mobil uygulama veya Activity tamamen yok edildiğinde (kapatıldığında) çalışan metottur.

12. UYGULAMA: İşlem adımlarına göre yaşam döngüsü metotları çalıştığında Logcat ekranında uyarı veren uygulamayı tasarlayınız.

1. Adım: Mobil uygulama geliştirme platformundan yeni bir proje açarak Empty Activity seçiniz.

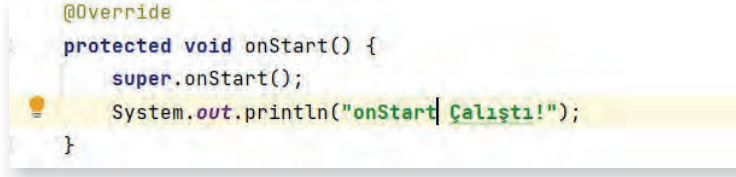
2. Adım: Uygulama ismini My Application olarak bırakınız.

3. Adım: Uygulama tasarım ekranına bir şey eklemeden MainActivity.java ekranında onStart() metodunu tanımlayınız. Bunun için onCreate() metodunun dışına çıkarak "onStart" yazılırsa otomatik tamamlama gelir. Görsel 6.33'te işaretlenen önerinin üzerine gelerek Enter tuşuna basınız.



Görsel 6.33: onStart() metot tamamlaması

4. Adım: Enter tuşundan sonra onStart metodu MainActivity.java dosyasına override edilir. Görsel 6.34'te olduğu gibi "`System.out.println("onStart Çalıştı!");`" şeklinde kodu giriniz.



Görsel 6.34: onStart metodunun içi

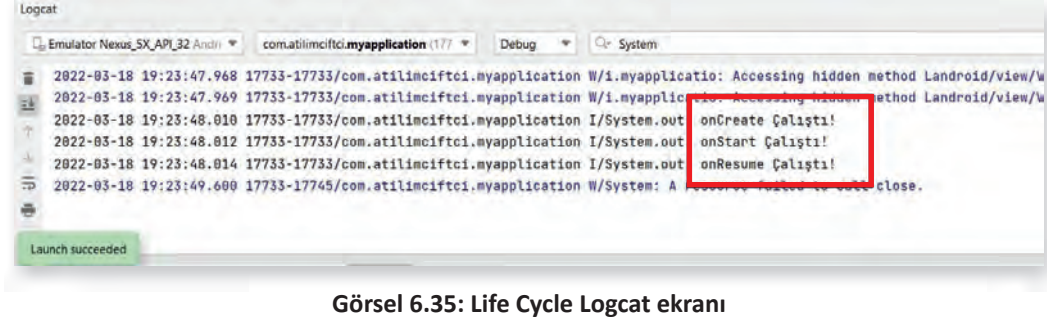
5. Adım: onResume(), onPause(), onStop(), onDestroy() metotları için de aynı işlemleri uygulayınız.

Uygulama metotları şu şekilde görünür:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    System.out.println("onCreate Çalıştı!");
}
@Override
protected void onStart() {
    super.onStart();
    System.out.println("onStart Çalıştı!");
}
@Override
protected void onResume() {
    super.onResume();
    System.out.println("onResume Çalıştı!");
}
@Override
protected void onPause() {
    super.onPause();
    System.out.println("onPause Çalıştı!");
}
```

```
@Override
protected void onStop() {
    super.onStop();
    System.out.println("onStop Çalıştı!");
}
@Override
protected void onDestroy() {
    super.onDestroy();
    System.out.println("onDestroy Çalıştı!");
}
```

6. Adım: Uygulamayı çalıştırınız ve Logcat ekranını açınız. Logcat ekranının filtreleme bölümüne System yazarak filtreleme yapınız. Bu sayede System.out.println komutlarının tamamı görünür. Görsel 6.35'te olduğu gibi uygulama açılınca onCreate(), onStart() ve onResume() metotlarının çalıştığını gözlemleyiniz.

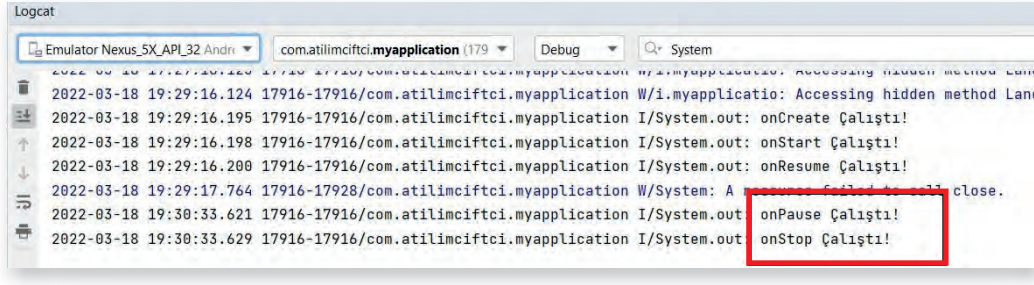


Görsel 6.35: Life Cycle Logcat ekranı

7. Adım: Uygulamayı Görsel 6.36'da olduğu gibi önce arka plana alıp sonra ana ekrana dönünüz.

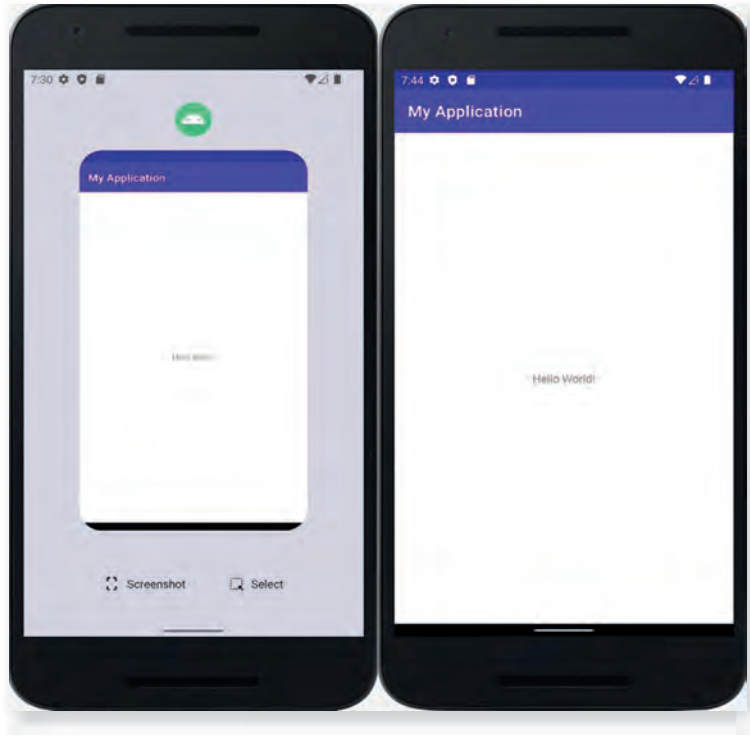
Görsel 6.36: Life Cycle uygulama ekranı

8. Adım: Ana ekrana dönüp Logcat ekranında Görsel 6.37'de olduğu gibi onPause() ve onStop() metotlarının çalıştığını gözlemleyiniz.

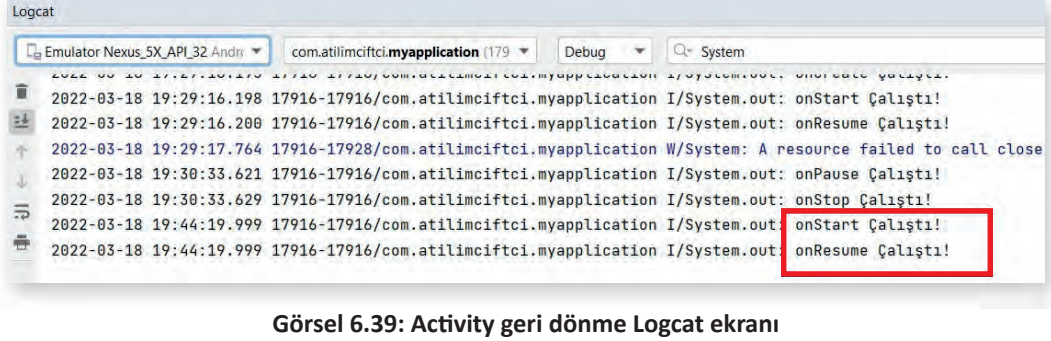


Görsel 6.37: onPause(), onStop() Life Cycle Logcat ekranı

9. Adım: Uygulama arka planında Görsel 6.38'de olduğu gibi tekrar Activity'ye dönünüz. Görsel 6.39'daki gibi Logcat ekranında onStart(), onResume() yaşam döngülerinin yeniden çalıştığını gözlemleyiniz.



Görsel 6.38: Activity'ye yeniden dönme



Görsel 6.39: Activity geri dönme Logcat ekranı

NOT: Activity'ye dönüldüğünde onStart() ve onResume metotları yeniden çalışır fakat onCreate() metodu çalışmaz. Bunun nedeni Activity kapatılmadığı sürece (Destroy) onCreate() metodunun sadece ilk Activity yüklemesinde çalışmasıdır.

10. Adım: Activity'yi arka plana alıp sonra da direkt kapatınız. Logcat ekranını kontrol ediniz. onDestroy() metodunun çalıştığını görünüz.

NOT: Uygulamada belirtilen Activity tamamen kapatıldıktan sonra yeniden çalıştırılırsa onCreate() metodu tekrar çalışır ve Activity yeniden kurulur.

6.3.2. Çoklu Aktiviteler

Mobil uygulamalar için vazgeçilemez bir yapı olan aktiviteler, bir mobil uygulamada sadece bir tane olabileceği gibi sınırsız sayıda da olabilir. Oluşturulan her Activity'nin AndroidManifest.xml dosyası içinde tanımlanması gerekir. Ayrıca ana aktivite de **"<intent-filter>"** etiketleri arasında tanımlanmalıdır. Bu intent-filter etiketi içinde ise "action" olarak "MAIN" özelliği, "category" olarak da "LAUNCHER" özelliği almalıdır.

Bir uygulama birden fazla aktivite içerirse zorunlu olan husus, aktiviteler birbirleri arasında bilgi alışverişi yapmasıdır. Örneğin kullanıcı adı ve şifresiyle uygulamaya giriş yapacak kullanıcı önce giriş ekranındaki aktivite ile ilgilenirken kullanıcı adı ve şifresi doğruysa mobil uygulama bir sonraki aktiviteye tüm bilgileri aktarmalıdır. Bu aktarım için de birden fazla yol bulunur.

6.3.2.1. Intent Yöntemiyle Activityler Arası Veri Taşıma

Intent, aktiviteler arasında veri taşımada en yaygın olarak bilinen yöntemdir. Kolay bir kullanımı olmakla birlikte Java dilinde sadece "Primitive değişken" olarak adlandırılan değişken türlerini taşıyabilir (int, short, byte, long, float, double, String, boolean, char). Object veya referans tipler için bu veri taşıma yöntemi uygulanamaz. Doğal olarak daha büyük projelerde bu veri taşıma yolu çok fazla tercih edilmez.

Intent kullanımı için birinci Activity, belirlediği değişkenleri gönderme işlemi yapmalıdır. Bunun için Intent sınıfından öncelikle bir nesne üretmelidir. Bu nesne üretilirken parametre olarak **"Package Context"** ve verinin gönderileceği sınıf (Activity ismi) referans olarak girilmelidir. Ardından bu nesneye **"putExtra"** metodu kullanılarak öncelikle gönderilecek verinin anahtar ismi sonrasında da verinin kendisi referans verilip yüklenmelidir. startActivity() metodu ile diğer Activity başlatılmalı ve oluşturulan intent nesnesi referans olarak yüklenmelidir.

```
Intent intent = new Intent(this, DigerActivity.class);
intent.putExtra("Key", data); //veri gönderiliyor
startActivity(intent);
```

NOT: Intent sınıfından üretilen intent nesnesine referans olarak gönderilen **this**, Context'i işaret eder. **Key**, datanın anahtar ismini, **data** ise verinin tutulduğu değişkeni veya veriyi ifade eder.

İkinci Activity ise gelen verileri karşılamalıdır. Bunun için açılan ikinci Activity'de Intent sınıfından bir nesne üretilmeli fakat bu nesne new ile tanımlanmamalıdır. Bu nesne, getIntent metodu kullanılarak tanımlanmalıdır. Tanımlanan nesne sıfır bir nesne değil, ilk Activity'den gelen veriyi alacak bir nesnedir. Daha sonra da getStringExtra, getIntExtra, getBooleanExtra vb. metotlarla karşılanmalıdır. Kullanılan metoda göre referans değişir. getStringExtra metodunda sadece verinin anahtar adını isterken getIntExtra metodunda hem verinin anahtar ismini hem de default değerini ister. Gelen verileri almak için Intent yönteminin kullanımı şu şekildedir:

```
Intent intent = getIntent();
String data = intent.getStringExtra("Key"); //Alınan veri String'e çevrilerek aynı tipli data değişkenine atanıyor.
```

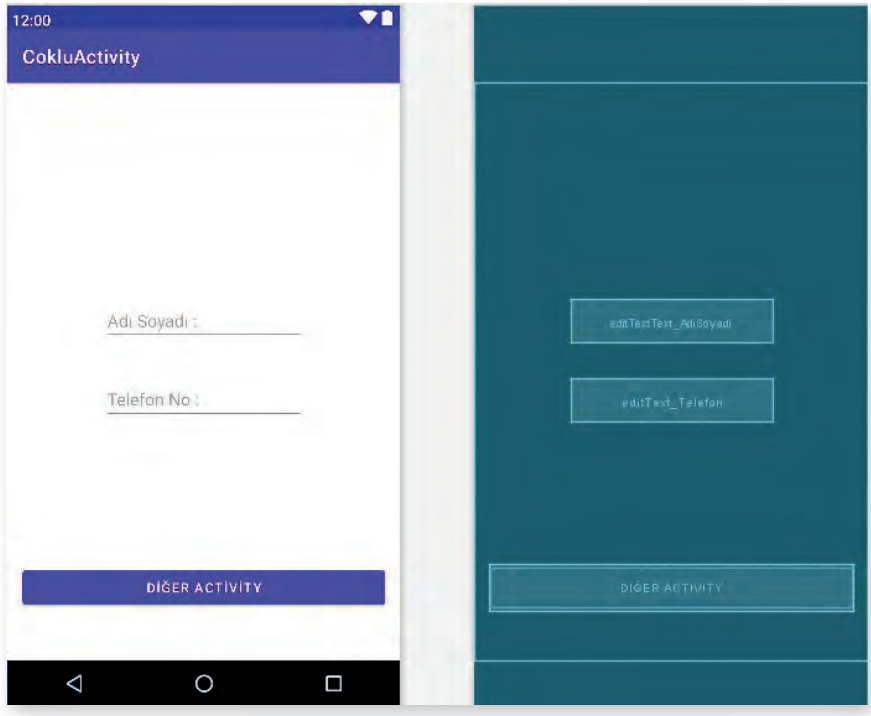
13. UYGULAMA: İşlem adımlarına göre "CokluActivity" isimli yeni bir proje açınız. "activity_main.xml" isimli layout içinde "Plain Text" olarak tanımlanmış "adiSoyadi"ve"Phone" olarak tanımlanmış "telefonNumarasi" şeklinde iki adet EditText ekleyiniz. Bir adet button tanımlayarak, buttona tıklandığında EditText içindeki verilerin, girildiği MainActivity sınıfından DigerActivity sınıfına gönderimini yapınız. "activity_diger.xml" içinde de iki adet TextView oluşturunuz. Gönderilen bilgileri bu TextViewlerde gösteren uygulamayı tasarlayınız.

1. Adım: Mobil geliştirme platformu üzerinden yeni bir proje dosyası açınız. Proje ismine "CokluActivity" ismini veriniz. Empty Activity seçerek proje dosyasının oluşturulmasını tamamlayınız.

2. Adım: Görsel 6.40'ta görüldüğü gibi bir adet "PlainText" şeklinde EditText ekleyerek id'sine "editTextText_AdiSoyadi" giriniz. Hint özelliğine "Adı Soyadı" şeklinde giriniz. Text özelliğini siliniz. Bir adet "Phone" şeklinde EditText ekleyerek id'sine "editText_Telefon" giriniz. Hint özelliğine de "Telefon No" giriniz. Bir adet de button ekleyerek id'sine "button_DigerActivity", text özelliğine de "DİĞER ACTİVİTY" şeklinde giriniz. onClick özelliğine "digerActivity" şeklinde yazınız. layout_width özelliğini "match_parent" girerek ekranı yataydan tam ekran alınız. "Infer Constraint" seçeneğine basarak ögelerin yerlerini sabitleyiniz.

3. Adım: viewBinding yöntemini kullanmak için build.gradle (Module) dosyasını açınız. "android" etiketlerinin içine şu kodu giriniz:

```
buildFeatures{
    viewBinding true
}
```

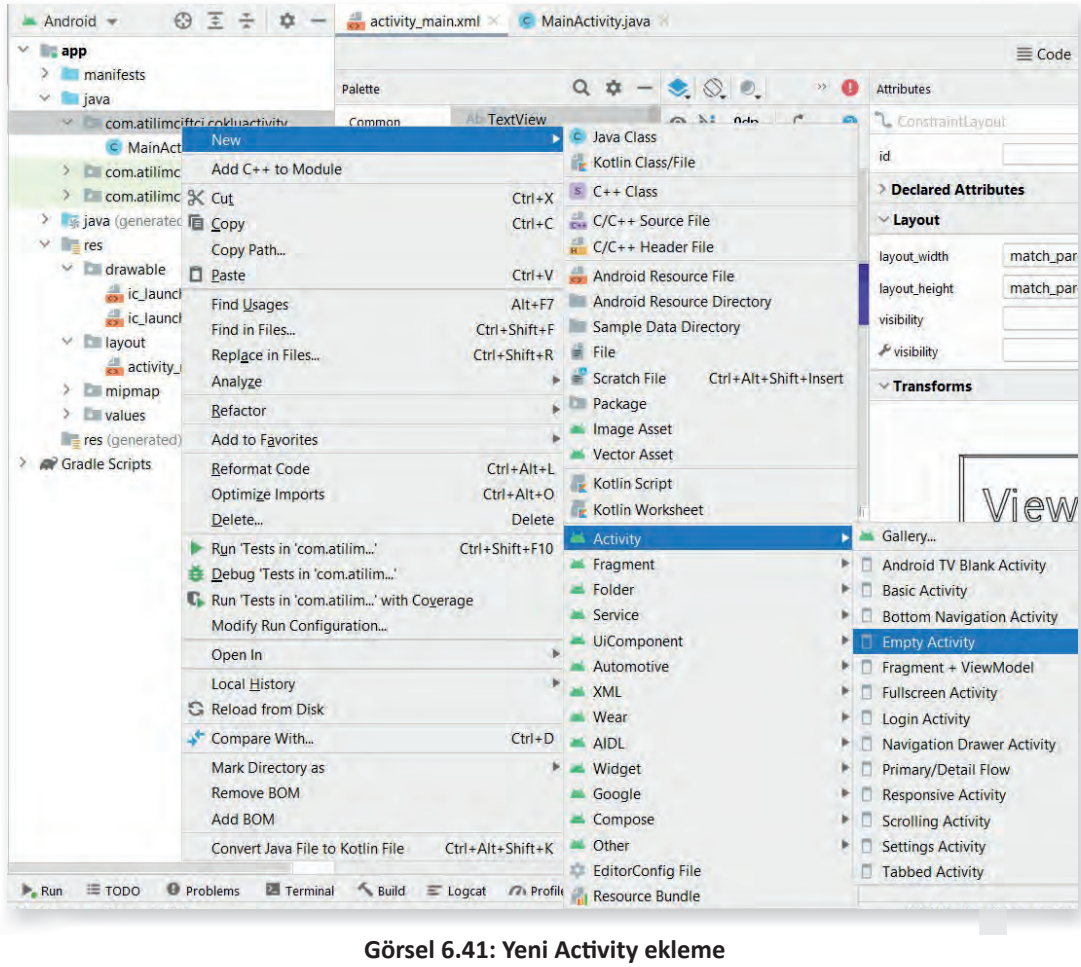


Görsel 6.40: CokluActivity uygulaması

4. Adım: MainActivity'ye dönünüz. onCreate metodunun dışında ve üstünde, MainActivity sınıfının blokları içindeyken "`private ActivityMainBinding binding;`" kodunu ekleyerek tanımlamayı yapınız. onCreate içini şu şekilde değiştiriniz:

```
private ActivityMainBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityMainBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
}
```

5. Adım: Görsel 6.41'de olduğu gibi java klasörü altında yer alan paket adına (com.atilimciftci.cokluactivity) sağ tuş ile tıklayınız. New>Activity>Empty Activity seçiniz. İsmine "BilgiActivity" giriniz.



Görsel 6.41: Yeni Activity ekleme

6. Adım: digerActivity metodunun içeriğini yazmak için MainActivity.java sınıfına giriniz. Sınıf blokları arasındayken digerActivity metodunu şu şekilde oluşturunuz:

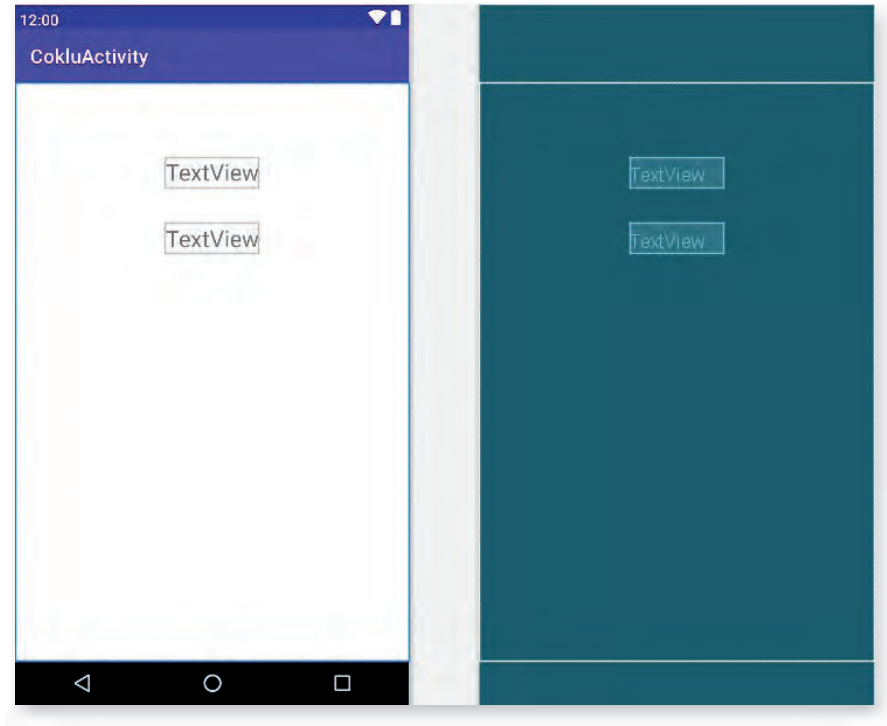
```
public void digerActivity(View view){
    String adiSoyadi = binding.editTextTextAdiSoyadi.getText().toString();
    String telefonNo = binding.editTextTelefon.getText().toString();
    Intent intent = new Intent(MainActivity.this,BilgiActivity.class);
    intent.putExtra("adiSoyadiKey",adiSoyadi);
    intent.putExtra("telefonNoKey",telefonNo);
    startActivity(intent);
}
```

MainActivity içeriğinin son şekli Görsel 6.42'de verilmiştir.

```
package com.atilimciftci.cokluactivity;

import ...

public class MainActivity extends AppCompatActivity {
    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);
    }
    public void digerActivity(View view){
        String adiSoyadi = binding.editTextTextAdiSoyadi.getText().toString();
        String telefonNo = binding.editTextTelefon.getText().toString();
        Intent intent = new Intent( packageContext: MainActivity.this,BilgiActivity.class);
        intent.putExtra( name: "adiSoyadiKey",adiSoyadi);
        intent.putExtra( name: "telefonNoKey",telefonNo);
        startActivity(intent);
    }
}
```

Görsel 6.42: MainActivity içeriği

7. Adım: "BilgiActivity" sınıfına ait olan "activity_bilgi.xml" layout dosyasını açınız. Görsel 6.43'te olduğu gibi basit bir sayfa tasarımı yapınız. Sayfanın içine sadece iki adet TextView ekleyiniz. Bunlardan birini "textView_AdiSoyadi" diğerini ise "textView_TelefonNo" id'leri ile adlandırınız. Infer Constraint'e basarak yerlerini sabitleyiniz.



Görsel 6.43: activity_bilgi.xml tasarımı

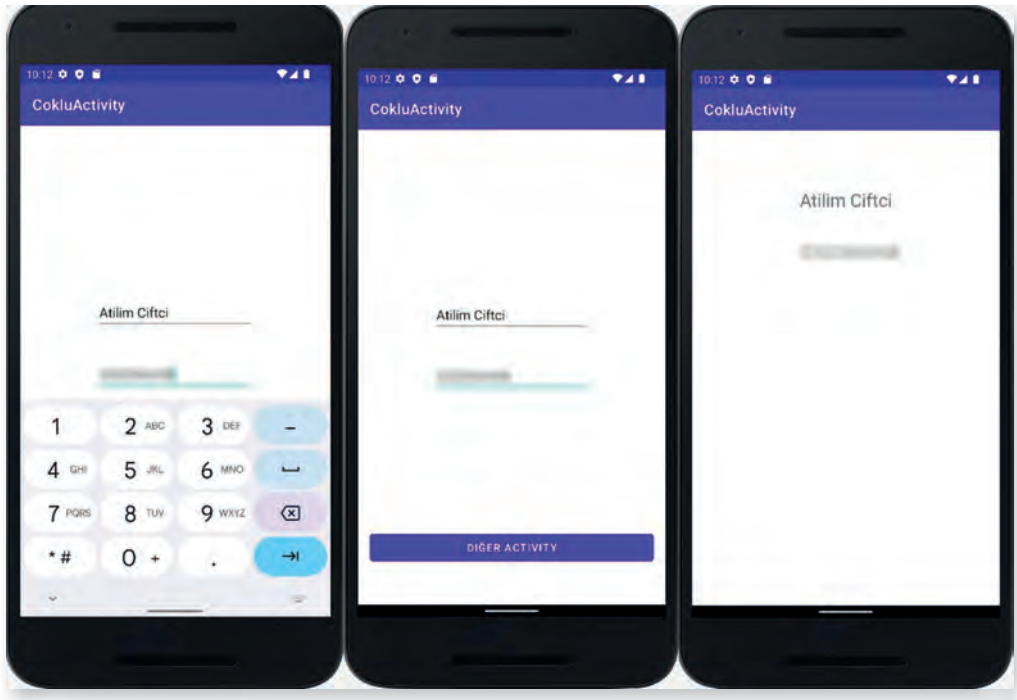
8. Adım: "BilgiActivity" sınıfını açınız. Daha önce gradle.build (Module) içine viewBinding eklendiği için tekrar gradle içine ekleme yapılmasına gerek yoktur. activity_bilgi.xml içine eklenen TextView ögelerine ulaşılabilmek adına viewBinding yönteminin sınıf içinde yapılması gereken değişikliklerini "BilgiActivity" sınıfı için gerçekleştiriniz. onCreate metodunun içeriğini şu şekilde yapınız:

```
private ActivityBilgiBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityBilgiBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
}
```

9. Adım: Veri aktarımı için onCreate içine Intent sınıfından bir nesne tanımlaması yapınız. Bu nesneyi "new" parametresi ile değil, "getIntent" ile oluşturunuz. Bu sayede önceki activityden gelen verilerin oluşturulan nesne aracılığı ile yakalanmasını sağlayınız. Gelen verileri, girilen key numaralarına göre bir değişkene yükleyiniz. Bunları da ilgili TextViewlerde gösteriniz. İçeriğe girilecek kod şu şekildedir:

```
private ActivityBilgiBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityBilgiBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
    Intent intent = getIntent();
    String bilgiAdiSoyadi = intent.getStringExtra("adiSoyadiKey");
    String bilgiTelefonNo = intent.getStringExtra("telefonNoKey");
    binding.textViewAdiSoyadi.setText(bilgiAdiSoyadi);
    binding.textViewTelefonNo.setText(bilgiTelefonNo);
}
```

Uygulama çalıştırıldığında ortaya çıkacak sonuç Görsel 6.44'te verilmiştir.



Görsel 6.44: CokluActivity uygulaması

SIRA SİZDE: Kullanıcı adı ve şifresi isteyen "MainActivity.java" dosyasında bilgiler EditTextlere girildikten sonra buttona basıldığında "DetayActivity" dosyasına kullanıcı adı ve kullanıcı şifresi bilgilerini gönderen ve bunları TextViewde gösteren uygulamayı viewBinding yöntemini kullanarak tasarlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. "MainActivity" isimli bir sınıf oluşturdu.		
2. "DetayActivity" isimli yeni bir sınıf ve bu sınıfa ait bir layout dosyası oluşturdu.		
3. activity_main içinde gereken görsel tasarımları oluşturdu.		
4. viewBindinge uygun kodları hazırladı.		
5. Button için MainActivity sınıfına uygun metodu oluşturdu.		
6. Intent oluşturarak aldığı verileri intent nesnesi üzerinden DetayActivity sınıfına gönderdi.		
7. activity_detay üzerinde gerekli görsel tasarımları oluşturdu.		
8. viewBindinge uygun kodları hazırladı.		
9. MainActivity üzerinden intent ile gönderilen verileri DetayActivity üzerinden yakaladı.		
10. activity_detay üzerinde verileri görüntüledi.		

6.3.2.2. Singleton Sınıfıyla Activityler Arası Veri Taşıma

Singleton sınıfı sadece bir objeye sahip olan sınıf yapısıdır. Contructerlar yapısı itibarıyla diğer sınıflar üzerinden erişim sağlanabilmesi için normalde public tanımlanmalıdır. Singleton sınıfında ise constructerlar private tanımlanır. Yalnızca tek obje oluşturulduğu için bu sınıf üzerinden erişim sağlanmak istendiğinde yanlışlıkla başka bir objeye erişim sağlama ihtimali bulunmaz. Singleton ile veri taşıyabilmek için öncelikle Singleton özelliğine sahip bir sınıf tasarımı yapmak gerekir. Bu işlem için şu şekilde bir sınıf oluşturulur:

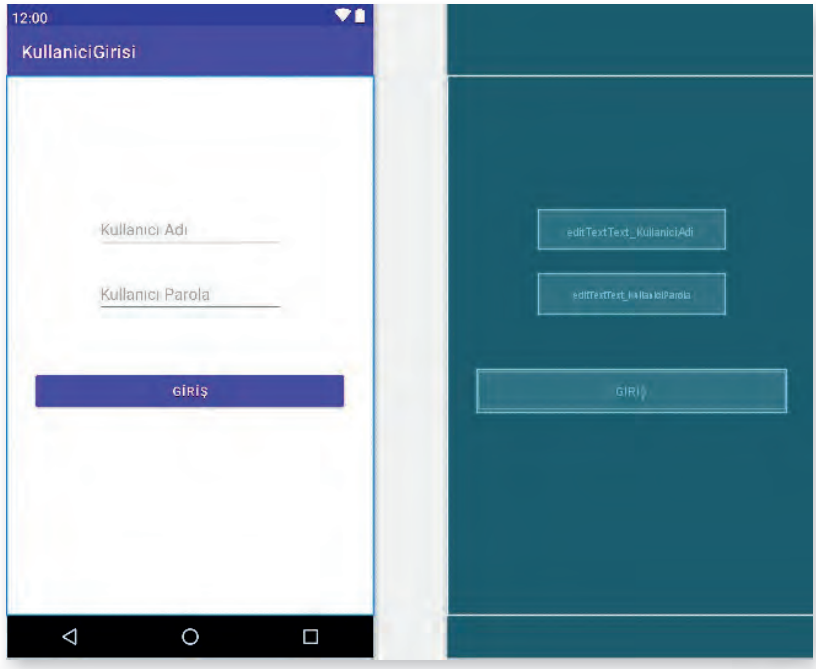
```
Public class Singleton{
       //Tanımlanacak değişkenler
       private static Singleton singleton;
       //private şeklinde bir constuctor
       //public türünde getter ve setter
       public static Singleton getInstance(){
              if (singleton==null){
                 singleton = new Singleton();
              }
       return singleton;
       }
}
```

Bu yapı sayesinde birinci Activity üzerinden bir nesne üretilir ve setter ile bu sınıfa gönderilir. Aynı nesne bir diğer Activity'ye geçildiğinde getter ile çağrılır. Tek nesne olduğu için veriler güvenli bir biçimde taşınır.

14. UYGULAMA: İşlem adımlarına göre MainActivity üzerinden "Kullanıcı Adı" ve "Kullanıcı Şifresi" istendiğinde onayla isimli buttona tıklandığı zaman bu kullanıcının Adı ve Şifre bilgilerini "Anasayfa.java" isimli sınıfa gönderip, "activity_ana_sayfa.xml" isimli layouttaki iki adet TextViewde gösteren uygulamayı viewBinding yapısı kullanarak tasarlayınız. Activity arasında veri gönderme işlemlerini Singleton metodu ile gerçekleştiriniz.

1. Adım: Yeni bir proje açınız. Empty Activity seçerek "KullaniciGirisi" ismini veriniz.

2. Adım: Görsel 6.45'teki gibi bir kullanıcı giriş ekranı oluşturunuz. Bu ekranı oluşturma esnasında iki adet "PlainText" özelliğinde EditText kullanınız. Bu EditTextlerin id'leri "editTextText_KullaniciAdi" ve "editTextText_KullaniciParola" olacaktır. Ayrıca bir adet button kullanarak id'si "button_Giris" onClick özelliği de "kullaniciGirisi" olarak eklemeleri bitiriniz. Infer Constraint'e tıklayarak yerlerini sabitleyiniz.



Görsel 6.45: KullaniciGirisi uygulaması

3. Adım: java klasöründeki paket ismine sağ tuş ile tıklayarak, "New>Class" seçeneğini seçerek bir sınıf oluşturunuz. Sınıf ismini "Singleton" olarak giriniz. Sınıfın içeriğinde private olarak String veri tipinde "kullaniciAdi" ve "kullaniciParola" ile iki adet değişken tanımlayınız. Ayrıca Singleton özelliği ekleyebilmek için private ve static özellikte Singleton sınıfının veri tipinden singleton isminde bir özellik tanımlayınız. Constructer, Getter-Setter, Singleton metot ile birlikte sınıf içeriği şu şekildedir:

```
public class Singleton
{
    private String kullaniciAdi;
    private String kullaniciParola;
    private static Singleton singleton;
    private Singleton(){
    }
    //Kullanıcı Adı için getter ve setter
    public String getKullaniciAdi(){
        return kullaniciAdi;
}
    public void setKullaniciAdi(String kullaniciAdi){
        this.kullaniciAdi = kullaniciAdi;
}
    // Kullanıcı Parolası için getter ve setter
    public String getKullaniciParola(){
        return kullaniciParola;
}
    public void setKullaniciParola(String kullaniciParola){
        this.kullaniciParola=kullaniciParola;
}
```

```
    //Singleton sınıfının nesne oluşturması
    public static Singleton getInstance(){
        if (singleton==null){
            singleton = new Singleton();
}
        return singleton;
}
}
```

4. Adım: build.gradle (Module) içeriğini açarak viewBinding için "buildFeatures{ viewBinding true}" kodunu ekleyiniz. MainActivity içine dönerek onCreate içeriğinde viewBinding için tanımlamaları bitiriniz. kullaniciGirisi isimli metodu oluşturarak içine Singleton tanımlamalarını yapınız.

```
private ActivityMainBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityMainBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
}
public void kullaniciGirisi(View view){
    String kullaniciAdi = binding.editTextTextKullaniciAdi.getText().toString();
    String parola = binding.editTextTextKullaniciParola.getText().toString();
//Singletoon tanımlama ve değer atamaları
    Singleton singleton = Singleton.getInstance();
    singleton.setKullaniciAdi(kullaniciAdi);
    singleton.setKullaniciParola(parola);
//Diğer Activity'ye geçiş
    Intent intent = new Intent(this,AnaSayfa.class);
    startActivity(intent);
}
```

5. Adım: Yeni bir Activity oluşturunuz ve "AnaSayfa" ismini veriniz. Layout içine iki adet TextView ekleyiniz. Id özelliklerini "textView_KullaniciAdi" ve "textView_KullaniciSifresi" şeklinde veriniz.

6. Adım: AnaSayfa içeriğindeki viewBinding için tanımlamaları yapınız ve Singleton içinden getter metodunu kullanarak nesne içindeki ögeleri çekiniz. Uygulamayı çalıştırınız.

```
private ActivityAnaSayfaBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityAnaSayfaBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
    String kullaniciAdi = Singleton.getInstance().getKullaniciAdi();
    String parola = Singleton.getInstance().getKullaniciParola();
    binding.textViewKullaniciAdi.setText(kullaniciAdi);
    binding.textViewKullaniciSifresi.setText(parola);
}
```

SIRA SİZDE: Sınıf arkadaşlarınızdan üç kişinin adını, soyadını, okul numarasını, telefon numarasını MainActivity.java sınıfına kaydediniz. activity_main.xml'de üç adet button oluşturunuz. Her bir button ile bir arkadaşınızın bilgilerini "Detay.java" isimli sınıfa Singleton sınıfı yöntemi ile gönderip "activity.detay.xml" layout dosyasında gösteren uygulamayı viewBinding yöntemini de kullanarak tasarlayınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

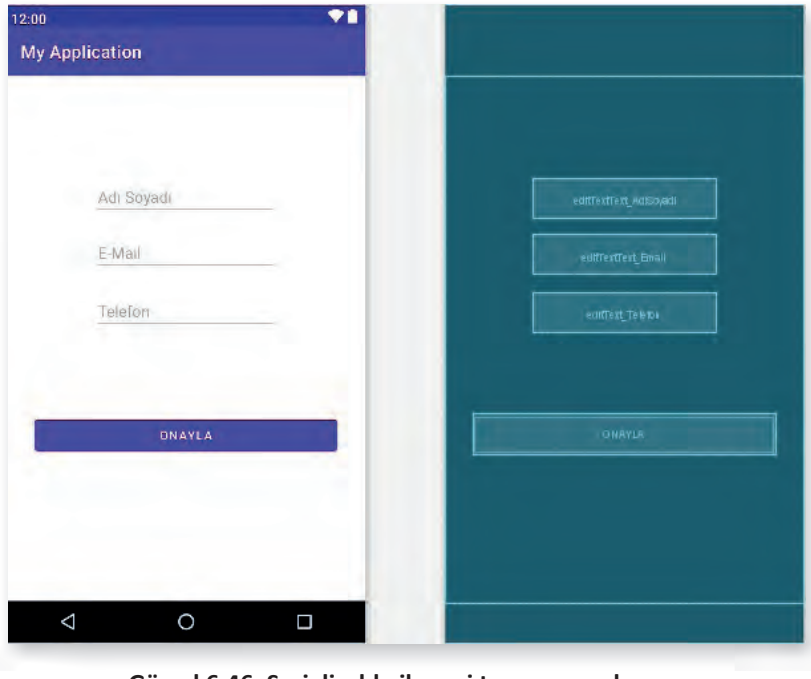
DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. "MainActivity" isimli bir sınıf oluşturdu.		
2. "DetayActivity" isimli yeni bir sınıf ve bu sınıfa ait bir layout dosyası oluşturdu.		
3. DetayActivity içinde alanlar ve Constructer yapıları oluşturdu.		
4. Singleton yapısı ile bir sınıf hazırladı.		
5. activity_main içinde gereken görsel tasarımları oluşturdu.		
6. viewBindinge uygun kodları hazırladı.		
7. MainActivity içinde buttonlara ait metotları oluşturdu.		
8. Metotlar içinde Singleton ile çalışacak intentler hazırladı.		
9. Singleton ile verileri "DetayActivity"e gönderdi.		
10. DetayActivity üzerinden verileri görüntüledi.		

6.3.2.3. Serializable Yöntemiyle Activityler Arası Veri Taşıma

"Primitive" değişkenler (int, String, bool, char vb.) nasıl ki intent ile aktiviteler arası gönderilirse object bir türde öge göndermek istendiğinde de Serializable yöntemi kullanılır. Serializable aslında bir interfacedir. Öncelikle nesnesi oluşturulan bir sınıfa implement edilmesi gerekir. Bu implement edilen interface sayesinde sınıf içindeki nesneler (intence) bir paket hâline dönüşür. Serializable yöntemi kullanılırken veriler intent ile gönderilir fakat bu kez bir paket şeklinde gönderilir. Bu paket, aktiviteler arasında gönderildiğinde ise gittiği Activity'de **"getSerializableExtra()"** metodu ile alınır ve serialize işlemi yapılmış sınıf veri türünden bir değişkene atılır. Burada dikkat edilmesi gereken husus, gelen veri Serializable ama atılan Class tipi olduğu için Casting (tip dönüşümü) işleminin gerekliliğidir. Casting işlemi uygulanır ve paket açılır. Paket içindeki veriler tek tek alınabilir duruma gelir.

15. UYGULAMA: İşlem adımlarına göre dördüncü uygulamaya benzer biçimde mobil uygulama ekranından alınan adı soyadı, e-mail ve telefon no bilgilerini buttona tıklandığı zaman Serializable yöntemi ile ikinci Activity'ye gönderen ve bunları TextView ile gösteren uygulamayı viewBinding yöntemini de kullanarak tasarlayınız.

1. Adım: Yeni bir proje açarak Empty Activity oluşturunuz. Görsel 6.46'ya benzer bir tasarım oluşturunuz. Tasarım ekranında üç adet "PlainText" özellikli EditText ve bir adet Buttona yer veriniz. Id'leri "editTextText_AdiSoyadi", "editTextText_Email", "editText_Telefon" şeklinde giriniz. Hint özelliklerini Görsel 6. 46'daki gibi gösteriniz. Buttonun id'sini "button_Onayla" şeklinde oluşturup onClick özelliğine de "digerActivity" giriniz.



Görsel 6.46: Serializable ile veri taşıma uygulaması

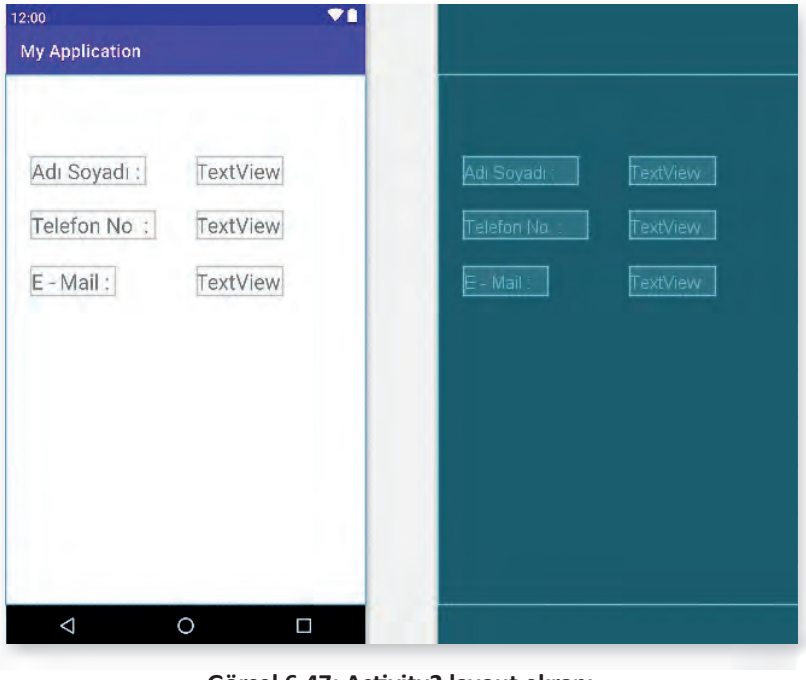
2. Adım: build.gradle (Module) dosyasında android etiketi içine buildFeatures{ viewBinding true } kodunu ekleyiniz ve senkronizasyonunu yapınız.

3. Adım: Bilgiler isimli bir sınıf oluşturunuz. Serializable interface'ini sınıfa implement ediniz. Ardından private erişime sahip String tipte adıSoyadi, telefonNo, eMail nesneleri tanımlayınız. Sınıfa ait bir constructer tanımlayınız. Her nesnenin getter metodunu oluşturunuz.

```
package com.atilimciftci.myapplication;
import java.io.Serializable;
public class Bilgiler implements Serializable
{
    private String adiSoyadi;
    private String telefonNo;
    private String eMail;
    public Bilgiler(String adiSoyadi, String telefonNo, String eMail){
        this.adiSoyadi = adiSoyadi;
        this.eMail = eMail;
        this.telefonNo = telefonNo;
    }
```

```
    public String getAdiSoyadi(){
        return adiSoyadi;
    }
    public String getTelefonNo(){
        return telefonNo;
    }
    public String geteMail(){
        return eMail;
    }
}
```

4. Adım: Activity2 isminde ikinci bir Activity oluşturunuz ve Görsel 6.47'ye benzer bir biçimde tasarımını gerçekleştiriniz. Id'ler "textView_AdiSoyadi", "textView_TelefonNo", "textView_Email" şeklinde olur.



Görsel 6.47: Activity2 layout ekranı

5. Adım: MainActivity.java sınıfına girerek viewBinding kullanımı için onCreate içindeki değişiklikleri gerçekleştiriniz. Buttona tıklanınca çalışacak digerActivity metodunu oluşturunuz. Metot içinde değişkenlere EditTextlerdeki bilgileri atayınız. Bilgiler sınıfından bir nesne üretiniz ve contructer yardımı ile girilen değerleri sınıftaki nesnelere yükleyiniz. Sınıftan üretilen nesneyi oluşturulan Activity2'ye Intent yardımıyla gönderiniz.

```
package com.atilimciftci.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import com.atilimciftci.myapplication.databinding.ActivityMainBinding;
```

```
public class MainActivity extends AppCompatActivity {
    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);
    }
    public void digerActivity(View view){
        String adiSoyadi = binding.editTextTextAdiSoyadi.getText().toString();
        String eMail= binding.editTextTextEmail.getText().toString();
        String telefonNo= binding.editTextTelefon.getText().toString();
        Bilgiler bilgilerSerializable = new Bilgiler(adiSoyadi,eMail,telefonNo);
        Intent intent = new Intent(this.getApplicationContext(),Activity2.class);
        intent.putExtra("bilgiler",bilgilerSerializable);
        startActivity(intent);
    }
}
```

6. Adım: viewBinding kullanımında Activity2 için onCreate içindeki değişimleri yapınız. Intent sınıfından gelen intentleri yakalayacak bir nesne üretiniz. Bilgiler sınıfı tipinde bir nesne oluşturunuz. Bu nesne içine intente ait nesnenin getSerializableExtra() metodu ile gelen bilgileri yakalayınız. İşlem bittiğinde uygulamayı çalıştırınız.

NOT: Serializable ile gelen bilgiler Bilgiler sınıf tipinde olduğu için Casting işlemi yapılır.

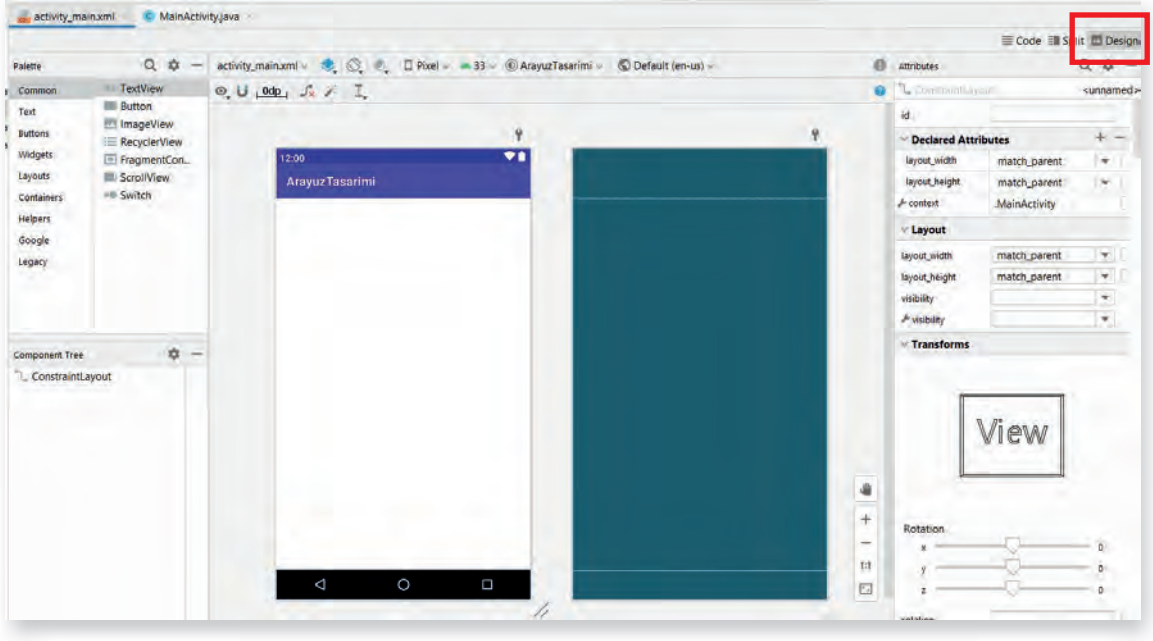
```
package com.atilimciftci.myapplication;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import com.atilimciftci.myapplication.databinding.Activity2Binding;
public class Activity2 extends AppCompatActivity {
    private Activity2Binding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = Activity2Binding.inflate(getLayoutInflater());
        View view = binding.getRoot();
        setContentView(view);
        Intent intent = getIntent();
        Bilgiler bilgiler = (Bilgiler) intent.getSerializableExtra("bilgiler");
        binding.textViewAdiSoyadi.setText(bilgiler.getAdiSoyadi());
        binding.textViewEmail.setText(bilgiler.geteMail());
        binding.textViewTelefonNo.setText(bilgiler.getTelefonNo());
    }
}
```

6.4. ARAYÜZ TASARIMI

Arayüz, kullanıcıların tasarlanan uygulamaları kullandıkları esnada gördükleri veya etkileşime girdikleri ögelerin ve bu ögelerin barındırdığı tasarımların genel ismidir. Mobil uygulama tasarım programı, tasarım esnasında birden fazla arayüz tasarım yolu sunar. Bu tasarım yolları şunlardır:

- Desing
- Code
- Split

Desing; şimdiye kadar kullanılan, sürükle bırak mantığı ile çalışan ve ögelerin bire bir görülüp özelliklerinin değiştirilebildiği tasarım ekranıdır (Görsel 6.48). Bu ekran için herhangi bir kod yazılmaz. Eklenen ögelerin özellikleri de ögelerin üzerine tıklanıp seçim yapıldıktan sonra "Attributes" penceresinden düzenlenir.

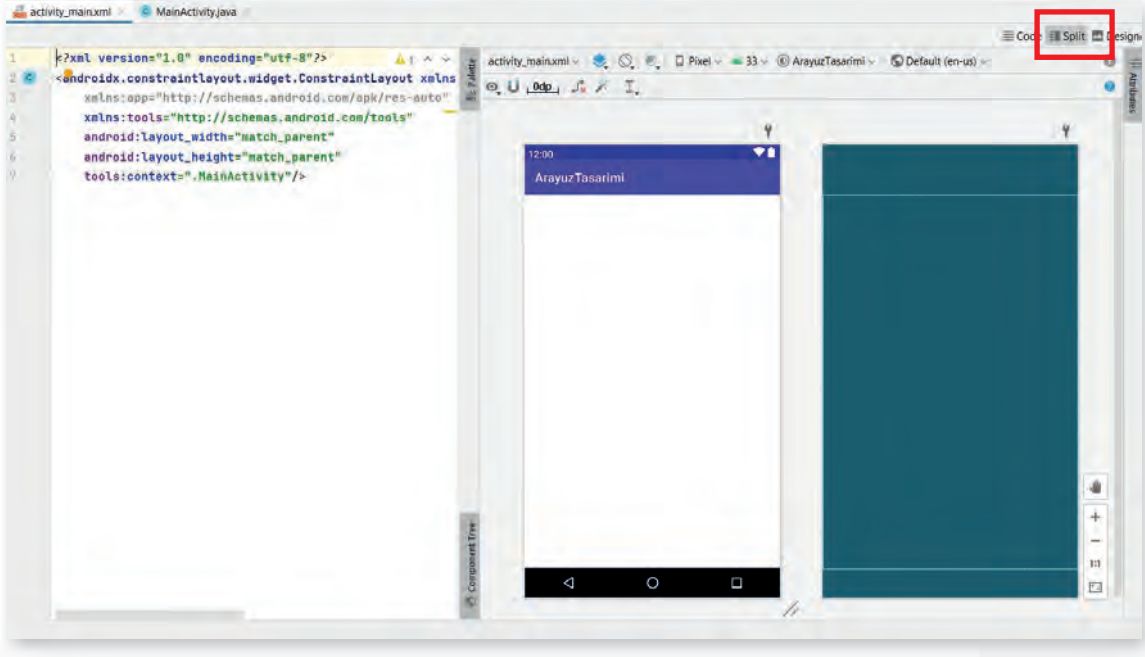


Görsel 6.48: Desing tasarım ekranı

Code tasarım ekranı, Desing ekranın tam tersine hiçbir ögeyi görsel şekilde göstermeden sadece kodlar vasıtası ile eklemeyi ve düzenlemeyi sağlar (Görsel 6.49). Kullanılan kodlar "xml" kodlarıdır. "xml" ile oluşturulan ögeler düzenlenmek istendiğinde ögelerin bloklarının arasına girilerek, özellikleri tek tek kodlanarak yazılır. Genelde Desing ekranı kullanılmış olsa da bazı işlemler Code ekranında daha kolay ve daha hızlı bir biçimde yapılabilir. Bu nedenle Code ekranına da hâkim olunması oldukça önemlidir.

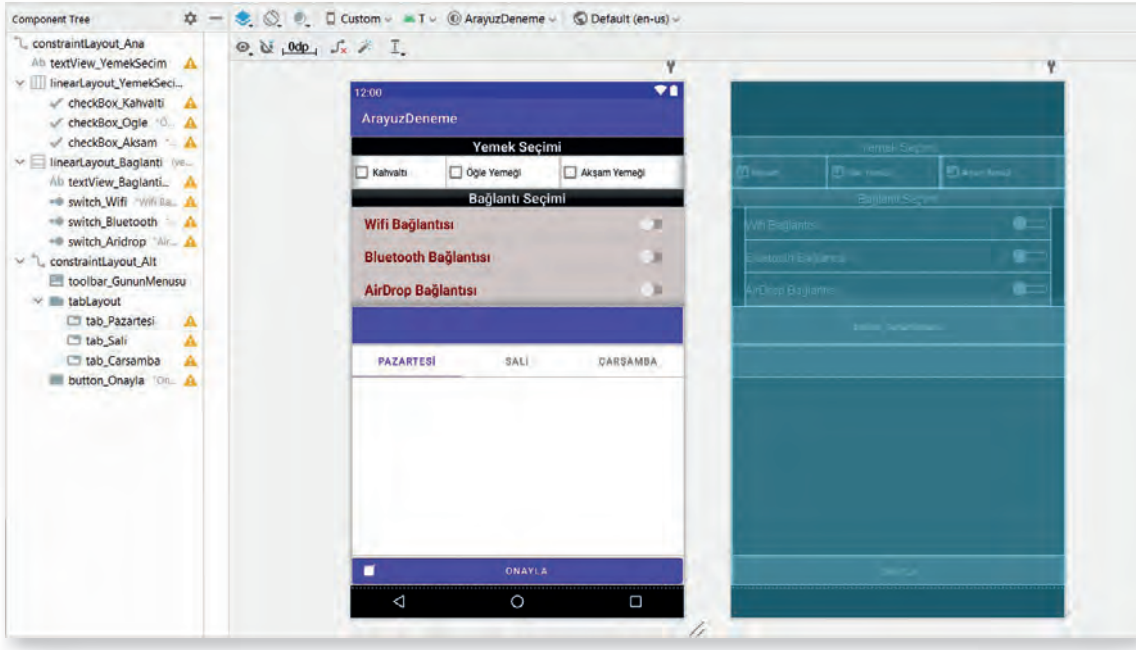
Görsel 6.49: Code tasarım ekranı

Split tasarım ekranı ise kendine ait bir özellik barındırmaz. Bu tasarım ekranı aslında Code tasarım ekranı ile Desing tasarım ekranının birleşiminden meydana gelir (Görsel 6.50). Split ekranı kullanılırken tasarım yapıldığı sırada Code ekranından kod yazılarak eklenen öge aynı anda Desing görüntüsünde de görünür, değiştirilen özellikleri aynı anda Desing ekranına da yansır. Bu durumun tersi de geçerlidir. Desing ekranında oluşturulan ögenin kod hâli, Code tasarım ekranına da eklenir ve Attributes menüsünden değiştirilen özellikleri de Code ekranına yansır.



Görsel 6.50: Split tasarım ekranı

16. UYGULAMA: İşlem adımlarına göre Görsel 6.51'de görünen tasarımı Component_tree'deki şekli ile Desing tasarım ekranında hazırlayınız.



Görsel 6.51: ArayuzDeneme tasarım ekranı

1. Adım: File>New>New Project menüsünden Empty Activity oluşturunuz. İsmini "ArayuzDeneme" şeklinde giriniz. Açılan sayfada activity_main içinde Desing bölümüne geçiniz. Constaint_Layout id'sini **constraintLayout_Ana** olarak ayarlayınız. ConstraintLayout içine öncelikle bir TextView ekleyiniz. Constraint sınırlarını ayarlamak için sağ, sol ve üst köşelerinden çekip ConstraintLayout sınırlarına götürünüz. Bu sayede layout içinde yeri sabitlenir. layout_with özelliğini match_parent yaparak tüm satırı kaplamasını sağlayınız. layout_height özelliğini de wrap_content olarak girerek içerik kadar yükseklik olmasını sağlayınız. Alt köşesine dokunmayınız. Attributes menüsü altında id değerini "`textView_YemekSecim`" şeklinde giriniz. Background özelliğini bulunuz ve arka planındaki siyahlık için menüden siyah seçiniz veya değerini "`#090909`" şeklinde ayarlayınız. Text özelliğine "`Yemek Seçimi`" giriniz. TextAligment özelliğini **center** yaparak yazıyı ortalayınız. TextSize özelliğini **20 sp**, TextSytle özelliğini de **bold** ayarlayınız.

2. Adım: ConstraintLayout içine Palette menüsünden Layouts seçeneği altından bir LinearLayout(horizontal) ekleyiniz. Bu layout, yan yana ögeler oluşturulmasını sağlar. Id değerine "`linearLayout_YemekSecimi`" giriniz. layout_width özelliğini "match_parent", layout_height özelliğini de "wrap_content" olarak ayarlayınız. Eklenen LinearLayout, ConstraintLayouta ait bir öge olması nedeniyle sınır değerleri verilerek sabitlenmelidir. LinearLayoutun sağ, sol ve üst bölgelerinde çıkan noktalardan sabitlemeyi oluşturunuz. Sabitlemeyi ConstraintLayout yerine textView_YemekSecim ögesine yapınız. Bu sayede textView_YemekSecim hareket ederse linearLayout da onunla birlikte hareket eder. Linear_Layoutun sağ noktasını textView_YemekSecim'in sağ noktasına, Linear_Layoutun sol noktasını textView_YemekSecim'in sol noktasına, Linear_Layoutun üst noktasını da textView_YemekSecim ögesinin alt noktasına sabitleyiniz.

3. Adım: Palette menüsünde yer alan Buttons menüsü altındaki CheckBox ögesini linearLayout_YemekSecimi içine sürükle bırak ile ekleyiniz. Üç defa bu işlemi tekrar ediniz.

Id değerlerini "checkBox_Kahvalti", "checkBox_Ogle", "checkBox_Aksam" şeklinde giriniz. layout_width özelliklerini wrap_content, layout_height özelliklerini de wrap_content olarak ayarlayınız. Text özelliklerini de sırasıyla "Kahvaltı", "Öğle Yemeği", "Akşam Yemeği" şeklinde giriniz.

4. Adım: linearLayout_YemekSecimi dışına çıkarak constraintLayout_Ana içine bir LinearLayout daha ekleyiniz. Ögelerin alt alta dizilebilmesi için bu LinearLayout vertical özellikli olmalıdır. Id'sini "linearLayout_Baglanti" şeklinde giriniz. layout_widht özelliğine match_parent, layout_height özelliğine de wrap_content giriniz. Background özelliğine gri renk veya "#D6CDCD" kodunu giriniz. constraintLayout_Baglanti içinde yerleşim için linearLayouta tıkladığında çıkan sol, sağ ve üst noktalarını linearLayout_YemekSecimi'nin sol, sağ ve üst noktalarına götürünüz. Bu sayede linearLayout_Baglanti da diğer ögelere bağlanır.

NOT: Yerleşimlerde problem yaşanırsa Component Tree üzerinden ögeler sürükle bırak ile birbirleri içine taşınabilir.

5. Adım: linearLayout_Baglanti içine id'si "textView_BaglantiSecim" olarak bir adet TextView ekleyiniz. layout_widht özelliğine match_parent, layout_height özelliğine de wrap_content giriniz. Background özelliğine siyah renk seçiniz veya "#090909" kodunu giriniz. Text özelliğine "Bağlantı Seçimi" yazınız. Id'si "textView_BaglantiSecim" olarak oluşturunuz. textAligment özelliğini "center" yaparak yazıyı ortalayınız. textColor özelliğini beyaz seçiniz veya "#EFEAEA" kodunu yazınız. textSize özelliğini 20 sp, textSytle özelliğini de bold ayarlayınız.

6. Adım: Palette menüsü altındaki Button içinde yer alan Switch ögesinden linearLayout_Baglanti içine üç adet ekleyiniz. Id'lerini "switch_Wifi", "switch_Bluetooth", "switch_Aridrop" şeklinde ayarlayınız. layout_width özelliklerini "match_parent", layout_height özelliğini de wrap_content olarak ayarlayınız. Margin_Left ve Margin_Right özelliklerini "20dp" olarak ayarlayıp iç boşluk veriniz. minHeight özelliğini "48dp" olarak ayarlayınız. Bu özellik ile minimum yazı yüksekliği ayarlanır. Text özelliklerine sırasıyla "Wifi Bağlantısı", "Bluetooth Bağlantısı", "AirDrop Bağlantısı" veriniz. TextColor özelliğini bordo seçiniz veya "#850408" kodunu giriniz. TextSize özelliklerini "20sp" olarak ayarlayınız ve TextStyle özelliğini bold yapınız.

7. Adım: linearLayout_Baglanti'dan çıkarak constraintLayout_Ana'ya dönünüz. İçine bir constraint_layout daha ekleyiniz. Id'sini "**constraintLayout_Alt**" olarak giriniz. layout_width özelliğini "**484dp**", layout_height özelliğini "**410dp**" olarak giriniz. Bu ölçü uygun gelmezse emülatörünüze uygun değerleri giriniz. constraintLayout_Alt'ın sabitlemesini sol, sağ ve üst noktalarından linearLayout_Baglanti'ya yapınız.

8. Adım: Palette üzerinde Containers menüsü altında yer alan Toolbar ögesini constraintLayout_Alt'ın içine sürükle bırak ile ekleyiniz. Id özelliğini "toolbar_GununMenusu" şeklinde giriniz. layout_width özelliğini "match_parent", layout_height özelliğini "wrap_content" olarak ayarlayınız. Background özelliğini mor renk olarak seçiniz veya "**?attr/colorPrimary**" olarak kodu giriniz. Sınır genişliklerini de constraintLayout_Alt'a sınırlayınız.

9. Adım: Palette üzerinde Containers menüsü altında yer alan tabLayout ögesini constraintLayout_Alt'ın içine ekleyiniz. Id değeri "tabLayout", layout_width özelliği "match_parent", layout_height özelliği "wrap_content" olacak şekilde giriniz. Sınırlarını da Toolbar nesnesine göre ayarlayınız. ComponentTree üzerinde, tabLayout altında yer alan ögeler mevcuttur. Bu ögelere tabItem adı verilir. Birinci tabItema tıklayarak text üzerinde yazan "Monday" kısmını "**Pazartesi**" olarak değiştiriniz ve id'sini "**tab_Pazartesi**" yapınız. İkinci tabItema tıklayarak text üzerinde yazan "T**uesday**" kısmını "**Salı**" olarak değiştiriniz ve id'sini "**tab_Sali**" yapınız. Üçüncü tabItema tıklayarak text üzerinde yazan "**Wednesday**" kısmını "**Çarşamba**" olarak değiştiriniz ve id'sini "**tab_Carsamba**" yapınız.

NOT: tabLayout, aynı activity içinde farklı işlemler yapılmasını sağlayan bir ögedir.

10. Adım: Son olarak yeniden constraintLayout_Alt içine bir adet Button ögesi ekleyiniz. Id özelliğini "`button_Onayla`", layout_width özelliğini "`match_parent`", layout_height özelliğini "`wrap_content`" olacak şekilde ayarlayınız. Text özelliğini "`Onayla`" yapınız ve icon olarak "`checkbox_on_background`" iconunu seçiniz. Sağ, sol ve üst sınırlarını da "`parent`" olarak ayarlayınız.

! **UYARI:** Altıncı uygulama tamamen Desing ortamında yapılmıştır. Bazı durumlarda sadece Code ekranını kullanarak tasarım oluşturmak veya tasarımda yapılması gereken değişiklikler için Code ekranını kullanmak daha uygun olabilir.

17. UYGULAMA: İşlem adımlarına göre Görsel 6.51'de gösterilip altıncı uygulamada Desing ortamında yapılan uygulamayı Code ekranında yeniden tasarlayınız.

1. Adım: File>New>New Project menüsünden Empty Activity oluşturunuz. İsmini "ArayuzDenemeCode" şeklinde giriniz. Açılan sayfada activity_main içinde Code bölümüne geçiniz. ConstraintLayout tagı arasında id yazınız ve çıkan "**android:id**" önerisine Enter tuşu ile basınız. Çıkan "**@+id**" önerisine de Enter tuşu ile basınız. Tırnak işaretlerinin içindeyken "**constraintLayout_Alt**" yazınız (Görsel 6.52).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="ht
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/constraintLayout_Ana"
    tools:context=".MainActivity">
</androidx.constraintlayout.widget.ConstraintLayout>
```

Görsel 6.52: constraintLayout_Ana id eklemesi

NOT: "</`androidx.constraintlayout.widget.ConstraintLayout`>" yazan kod constraintLayoutun bitişi anlamına gelir. Bu kodun bir üst satırında kalındığı sürece constraintLayout içinde olunduğunu gösterir.

NOT: Code bölümünde tasarım oluşturulurken öneriler sıkça kullanılır. Bu nedenle yapılması gereken iş ile ilgili kilit kelimeler unutulmalıdır. Bu kilit kelimeler sayesinde mobil uygulama geliştirme ortamının sunduğu önerilerle kodun bütünlüğü sağlanır.

2. Adım: constraintLayout_Ana içine textView oluşturunuz. Bunun için "<te" yazıldığında çıkan TextView önerisine Enter tuşu ile basınız. Width ve hight özellikleri ekrana geldiğinde width kısmına "**match_parent**", height kısmına "**wrap_content**" yazınız. İşlem bittikten sonra en son satırın sonuna "**>**" işareti veya "**/**" işareti koyunuz. İki işaret de TextView ögenizi bitirdiğinizin işaretidir. Bu işaret öncesi yapacağınız her değişiklik TextViewi etkiler. İşareti alt satıra alınız ve aralığa girerek boş satıra id yazınız. "**android:id**" önerisine Enter tuşu ile basınız. Gelen "**@+id**" önerisine de Enter tuşu ile basınız ve "textView_YemekSecim" yazınız. Boş bir yere tıklayarak "background" yazınız ve öneriye Enter tuşu ile basınız. İçeriğe "#090909" yazarak renk kodunu veriniz. Arka plan siyah olur. Alt satıra inerek "text" yazınız. Gelen öneriye Enter tuşu ile basarak "Yemek Seçimi" yazınız. Tekrar alt satıra inerek "textAlignment" yazınız ve öneriye Enter tuşu ile basarak "center" yazınız. Textcolor yazdıktan sonra öneriyi kabul ederek, "#EFEAEA" kodunu yazıp beyaz yapınız veya beyaz rengi öneriler arasında seçiniz. Code ekranında ConstraintLayout sınırlarının belirlenmesi için right, left, top gibi anahtar kelimeler kullanılıp önerilerle tamamlanır. Parent, aile anlamına gelir. Sınırları verilen öge, içinde bulunduğu constraintLayouta bağlı olduğu için o contraintLayout ailesidir gibi kabul edilir. Bu nedenle yazılan parent kodu, ögenin içinde bulunduğu ConstraintLayoutu işaret eder ve ona göre ögenin sınırlarını belirler. Bir başka ögeye sınırlamak için referans alınması istenen ögenin id'si yazılır. Oluşturulan kodlar şu şekildedir:

```
<TextView
    android:id="@+id/textView_YemekSecim"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#090909"
    android:text="Yemek Seçimi"
    android:textAlignment="center"
    android:textColor="#EFEAEA"
    android:textSize="20sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Görsel 6.53: Kapatılan TextView etiketi

3. Adım: Görsel 6.53'te olduğu gibi yazılan "TextView" etiketini kodların karışmaması için sol tarafında bulunan "**–**" işaretine basarak küçültünüz. Oluşan "**+**" işaretine bastığınızda görünüm eski hâline gelir. TextView dışına çıkıp ConstraintLayout içindeyken bir LinearLayout oluşturunuz. Bunun için ikinci adımdaki önerilere göre kodları şu şekilde yazınız:

```
<LinearLayout
    android:id="@+id/linearLayout_YemekSecimi"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="@+id/textView_YemekSecim"
    app:layout_constraintTop_toBottomOf="@+id/textView_YemekSecim">
</LinearLayout>
```

NOT: Ögenin içine girilmesi, kapatma etiketinden "/>" öncesine gidilmesi anlamı taşır.

4. Adım: Oluşturulan "**textView_YemekSecim**" layoutunun içine giriniz. **Checkbox** oluşturunuz. id'sini "checkBox_Kahvalti", genişlik ve yüksekliğini de "wrap_content" olarak ayarlayınız. Texti de "Kahvaltı" şeklinde yazınız.

```
<LinearLayout
    android:id="@+id/linearLayout_YemekSecimi"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="@+id/textView_YemekSecim"
    app:layout_constraintTop_toBottomOf="@+id/textView_YemekSecim">
</LinearLayout>
```

NOT: LinearLayout bir constraintLayout ögesi olduğu için sınırları belirlenir fakat CheckBox bir linearLayout ögesi olduğu için sınır belirtmeye gerek yoktur.

5. Adım: Diğer iki CheckBox ögesini de şu kodları yazarak ekleyiniz:

```
<LinearLayout
    android:id="@+id/linearLayout_YemekSecimi"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="@+id/textView_YemekSecim"
    app:layout_constraintTop_toBottomOf="@+id/textView_YemekSecim">
    <CheckBox
        android:id="@+id/checkBox_Kahvalti"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Kahvaltı" />
    <CheckBox
        android:id="@+id/checkBox_Ogle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Öğle Yemeği" />
    <CheckBox
        android:id="@+id/checkBox_Aksam"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Akşam Yemeği" />
</LinearLayout>
```

6. Adım: linearLayout_YemekSecimi ögesinin dışına çıkarak constraintLayout_Ana'nın içeriğine giriniz. Bir linearLayout daha ekleyiniz. Bu sefer **orientation (yerleşim)** özelliğinin **vertical (dikey)** olmasını sağlayınız.

```
<LinearLayout
    android:id="@+id/linearLayout_Baglanti"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#D6CDCD"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="@+id/linearLayout_YemekSecimi"
    app:layout_constraintStart_toStartOf="@+id/linearLayout_YemekSecimi"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout_YemekSecimi">
</LinearLayout>
```

7. Adım: Oluşturulan layout içine girerek TextView oluşturunuz.

```
<TextView
    android:id="@+id/textView_BaglantiSecim"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#090909"
    android:text="Bağlantı Seçimi"
    android:textAlignment="center"
    android:textColor="#EFEAEA"
    android:textSize="20sp"
    android:textStyle="bold" />
```

8. Adım: TextView etiketinin bitişinin alt satırına girerek devam ediniz ve üç adet Switch ögesi oluşturunuz. Oluşturulan ögelerde bir sınır bulunmaz çünkü bunlar constraintLayout ögeleri değil, linearLayouta ait ögelerdir.

```
<LinearLayout
    android:id="@+id/linearLayout_Baglanti"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#D6CDCD"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="@+id/linearLayout_YemekSecimi"
    app:layout_constraintStart_toStartOf="@+id/linearLayout_YemekSecimi"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout_YemekSecimi">
    <TextView
        android:id="@+id/textView_BaglantiSecim"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#090909"
        android:text="Bağlantı Seçimi"
        android:textAlignment="center"
        android:textColor="#EFEAEA"
        android:textSize="20sp"
```

```
        android:textStyle="bold" />
    <Switch
        android:id="@+id/switch_Wifi"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:minHeight="48dp"
        android:text="Wifi Bağlantısı"
        android:textColor="#850408"
        android:textSize="20sp"
        android:textStyle="bold" />
    <Switch
        android:id="@+id/switch_Bluetooth"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:minHeight="48dp"
        android:text="Bluetooth Bağlantısı"
        android:textColor="#850408"
        android:textSize="20sp"
        android:textStyle="bold" />
    <Switch
        android:id="@+id/switch_Aridrop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:minHeight="48dp"
        android:text="AirDrop Bağlantısı"
        android:textColor="#850408"
        android:textSize="20sp"
        android:textStyle="bold" />
</LinearLayout>
```

9. Adım: linearLayout_Baglantı'nın dışına çıkarak constraintLayout_Ana yerleşiminin içine geçiniz. constraintLayout_Ana yerleşiminin içine bir constraintLayout daha oluşturunuz ve id'sine "constraintLayout_Alt" giriniz.

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/constraintLayout_Alt"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout_Baglanti"
    app:layout_constraintLeft_toLeftOf="@+id/linearLayout_Baglanti"
    app:layout_constraintRight_toRightOf="@+id/linearLayout_Baglanti"
    android:layout_width="484dp"
    android:layout_height="410dp"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="247dp">
</androidx.constraintlayout.widget.ConstraintLayout>
```

10. Adım: Yeni oluşturulan constraintLayout_Alt içine giriniz ve Toolbar ögesi ekleyiniz. Anahtar kelime kullanarak otomatik tamamlama ve akıllı koddan yardım alınız. "toolbar_GununMenusu" ismini id olarak veriniz.

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar_GununMenusu"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"
    android:minHeight="?attr/actionBarSize"
    android:theme="?attr/actionBarTheme"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

11. Adım: constraintLayout_Alt'ın içindeyken "tabLayout" oluşturunuz. tabLayout oluşturduktan sonra sınırlılıklarını, genişlik ve yükseklik değerlerini veriniz. Ayrıca bu tabLayouta ait "tabItem"-lerin eklenmesi gerekir. tabLayout içindeyken "<tabItem" yazınız ve öneriyi Enter tuşu ile onaylayınız. Gelen kod yapısı hata verir ve öncekiler gibi height ve width özelliklerini açmaz. Hatanın üzerindeki resme tıklayarak bunları ekleyiniz veya onları da el ile yazınız.

NOT: constraintLayout etiketleri arasında direkt olarak yazılan her bir ögenin sınırlarının belirlenmesi zorunludur. Bu nedenle iç içe oluşturulan layoutlarda, kendine ait layout ailesi constraintLayout olan her bir ögede sınır belirtilmelidir.

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar_GununMenusu"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"
    android:minHeight="?attr/actionBarSize"
    android:theme="?attr/actionBarTheme"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<com.google.android.material.tabs.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/toolbar_GununMenusu">
    <com.google.android.material.tabs.TabItem
        android:id="@+id/tab_Pazartesi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pazartesi" />
    <com.google.android.material.tabs.TabItem
        android:id="@+id/tab_Sali"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Salı" />
    <com.google.android.material.tabs.TabItem
        android:id="@+id/tab_Carsamba"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Çarşamba" />
</com.google.android.material.tabs.TabLayout>
```

12. Adım: constraintLayout_Alt içine tekrar girerek bir adet Button oluşturunuz.

```
<Button
    android:id="@+id/button_Onayla"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Onayla"
    app:icon="@android:drawable/checkbox_on_background"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:ignore="TouchTargetSizeCheck" />
```

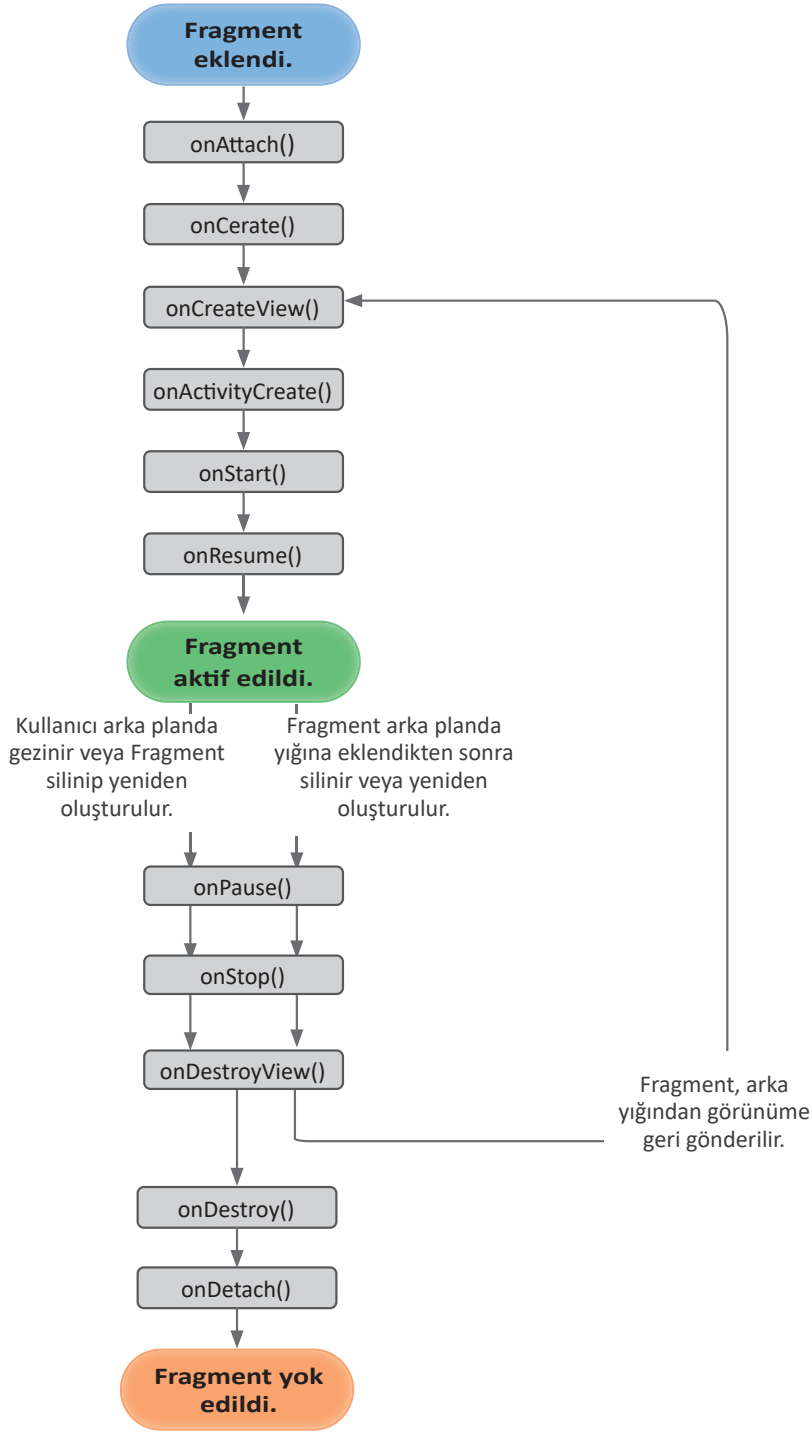
Yazılan kodlar topluca şu şekilde görülür:

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/constraintLayout_Alt"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout_Baglanti"
    app:layout_constraintLeft_toLeftOf="@+id/linearLayout_Baglanti"
    app:layout_constraintRight_toRightOf="@+id/linearLayout_Baglanti"
    android:layout_width="484dp"
    android:layout_height="410dp"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="247dp">
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar_GununMenusu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <com.google.android.material.tabs.TabLayout
        android:id="@+id/tabLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar_GununMenusu">
        <com.google.android.material.tabs.TabItem
            android:id="@+id/tab_Pazartesi"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Pazartesi" />
        <com.google.android.material.tabs.TabItem
            android:id="@+id/tab_Sali"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Salı" />
        <com.google.android.material.tabs.TabItem
            android:id="@+id/tab_Carsamba"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Çarşamba" />
    </com.google.android.material.tabs.TabLayout>
    <Button
        android:id="@+id/button_Onayla"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Onayla"
        app:icon="@android:drawable/checkbox_on_background"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="TouchTargetSizeCheck" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

6.5. FRAGMENT YAPISI

Mobil uygulama geliştirirken çoklu aktivite ile birden fazla sayfa kullanmak yerine tek Activity içinde birden fazla sayfa kullanmak mümkündür. Bu tasarımı ortaya koyan yapıya Fragment denir. Bu yapılar alt Activity gibi düşünülebilir. Fragment yapılarının da tıpkı aktiviteler gibi kendine ait bir yaşam döngüsü vardır (Görsel 6.54).



Görsel 6.54: Fragment yaşam döngüsü

Görseldeki metotlardan çoğu, Activity yaşam döngülerindeki metotlarla aynı işleve sahiptir. Farklı olan onAttach(), onCreateView(), onDestroyView() metotlarının açıklaması şu şekildedir:

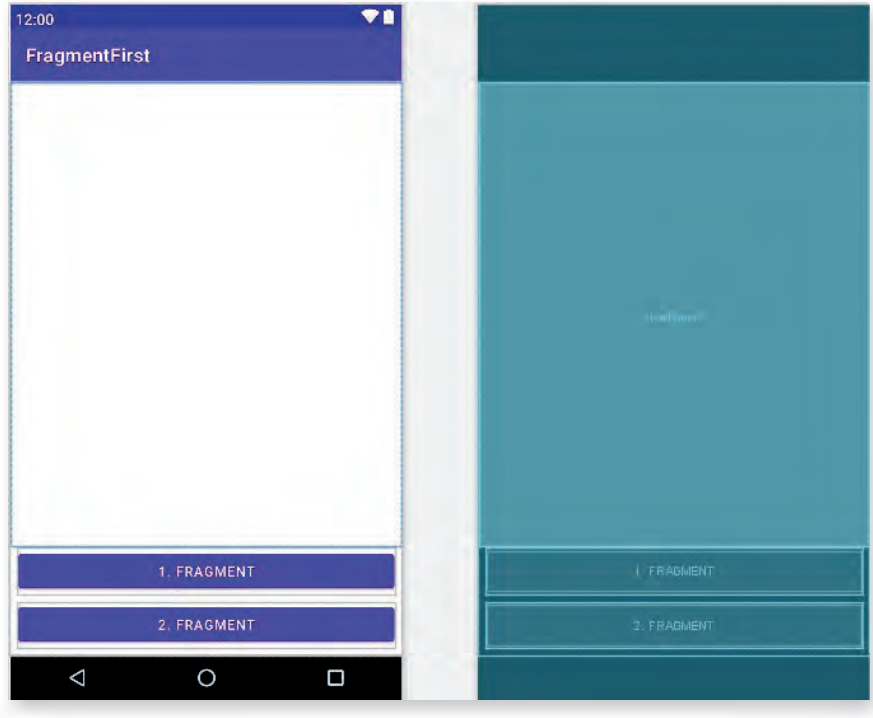
- **onAttach():** Mobil uygulama için sistem hiçbir şey yaratmadan önce bu metot çağrılır ve Fragment için yer ayrılır.
- **onCreateView():** Bir Fragment'ı oluştururken layout tanımı yapmak için bu metot kullanılır. Fragment'ı ayakta tutmak için kullanılan tek callback metodu budur. Burada view değişkenleri atanabilir.
- **onActivityCreated():** Her şey tamamlandıktan sonra en son başlatılması istenen işlemler için kullanılan metottur. onCreateView() metodundan hemen sonra çalıştırılır.

Her Fragment'in kendine özel bir sınıfı ve kendine özel de bir layout dosyası vardır. Ayrıca Fragment oluşturmanın da birden fazla yöntemi bulunur.

18. UYGULAMA: İşlem adımlarına göre bir Activity içinde bulunan iki adet buttondan birincisine tıklandığında birinci Fragment'i açıp ekranda "1.Fragment Ekranı", ikincisine tıklanınca da ikinci Fragment'i açarak "2. Fragment Ekranı" yazan uygulamayı tasarlayınız (Görsel 6.55).

1. Adım: Yeni bir proje açınız ve Empty Activity seçiniz.

2. Adım: "activity_main.xml" layout dosyasını açınız ve Code ekranına geçiniz. Gerekli tüm eklemeleri Code ekranında yapınız.



Görsel 6.55: FragmentFirst uygulaması tasarım ekranı

3. Adım: "constraintlayout" olarak devam edecek kodlama yapılacaktır. Kodlama sonucunda Görsel 6.56'daki kodlama yapısı oluşturulmalıdır. Code ekranında constraintlayout arasına girerek önce "1. Fragment" buttonunu sonra da "2. Fragment" buttonunu oluşturunuz. Her button oluşturma işleminde genişlik, yükseklik, id, onClick, constraintlayout yerleşimleri, boşluk ayarları tek tek kodla yazılır. Görsel 6.56'da açıklamaları mevcut olan ve "2. Fragment" buttonunu oluşturmak için yazılan kodlar şunlardır:

```
<Button
    android:id="@+id/button_IkinciFragment"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="8dp"
    android:onClick="fragment2Gecis"
    android:text="2. Fragment"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent">
</Button>
```

```
<Button
    android:id="@+id/button_IkinciFragment"          → Buttona id değeri atar.
    android:layout_width="match_parent"              → Buttonun genişliğini tüm satır olacak şekilde ayarlar.
    android:layout_height="wrap_content"             → Buttonun yüksekliğini text metin olacak şekilde ayarlar.
    android:layout_marginLeft="8dp"                  → Buttonun sol boşluğunu 8dp olarak ayarlar.
    android:layout_marginRight="8dp"                 → Buttonun sağ boşluğunu 8dp olarak ayarlar.
    android:layout_marginBottom="8dp"                → Buttonun alt boşluğunu 8dp olarak ayarlar.
    android:onClick="fragment2Gecis"                 → Buttona tıklanınca çalışacak metodu tanımlar.
    android:text="2. Fragment"                       → Buttonun içinde yazacak texti ayarlar.
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"      → Buttonun constraintinin alt, sol ve sağ
    app:layout_constraintRight_toRightOf="parent">     ilişkisini Activity'nin içi olarak ayarlar.
</Button>
```

Görsel 6.56: Button oluşturmak için Code ekranı

4. Adım: "1. Fragment" isimli buttonu oluşturmak için şu kodları constraintlayout içine yazınız:

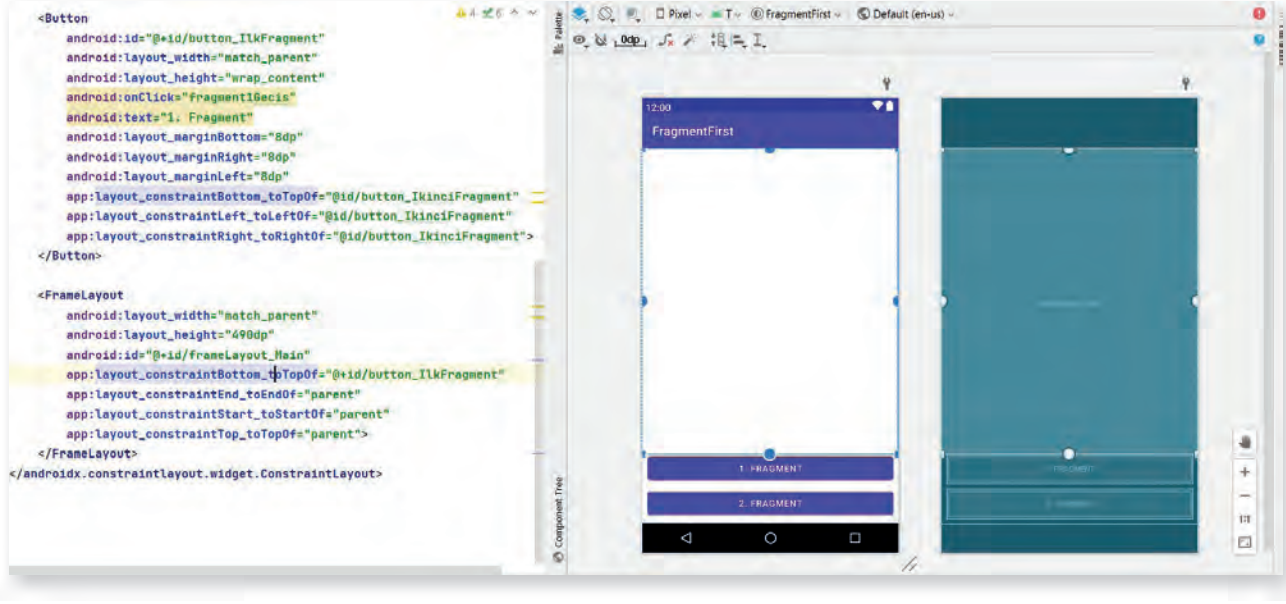
```
<Button
    android:id="@+id/button_IlkFragment"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="fragment1Gecis"
    android:text="1. Fragment"
    android:layout_marginBottom="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginLeft="8dp"
    app:layout_constraintBottom_toTopOf="@id/button_IkinciFragment"
    app:layout_constraintLeft_toLeftOf="@id/button_IkinciFragment"
    app:layout_constraintRight_toRightOf="@id/button_IkinciFragment">
</Button>
```

5. Adım: Mobil uygulama ekranının altında iki adet button oluşturulduğuna göre bu buttonlara tıklandığı zaman değişecek Fragment ekranını tasarlanan buttonlar üzerine yerleştiriniz. Uygulama içinde tek bir Activity bulunur. Bu Activity içinde de değişen Fragment ekranları vardır. Değişen Fragment ekranlarının içinde bulunduğu bir layout eklenmesi gerekir. Eklenecek layout içinde tek bir öge bulunacağı için "framelayout" tasarımı yapılır. Oluşturulması istenen framelayout için constraintlayout kodları arasına giriniz ve şu kodları yazınız:

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="490dp"
    android:id="@+id/frameLayout_Main"
    app:layout_constraintBottom_toTopOf="@+id/button_IlkFragment"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
</FrameLayout>
```

NOT:

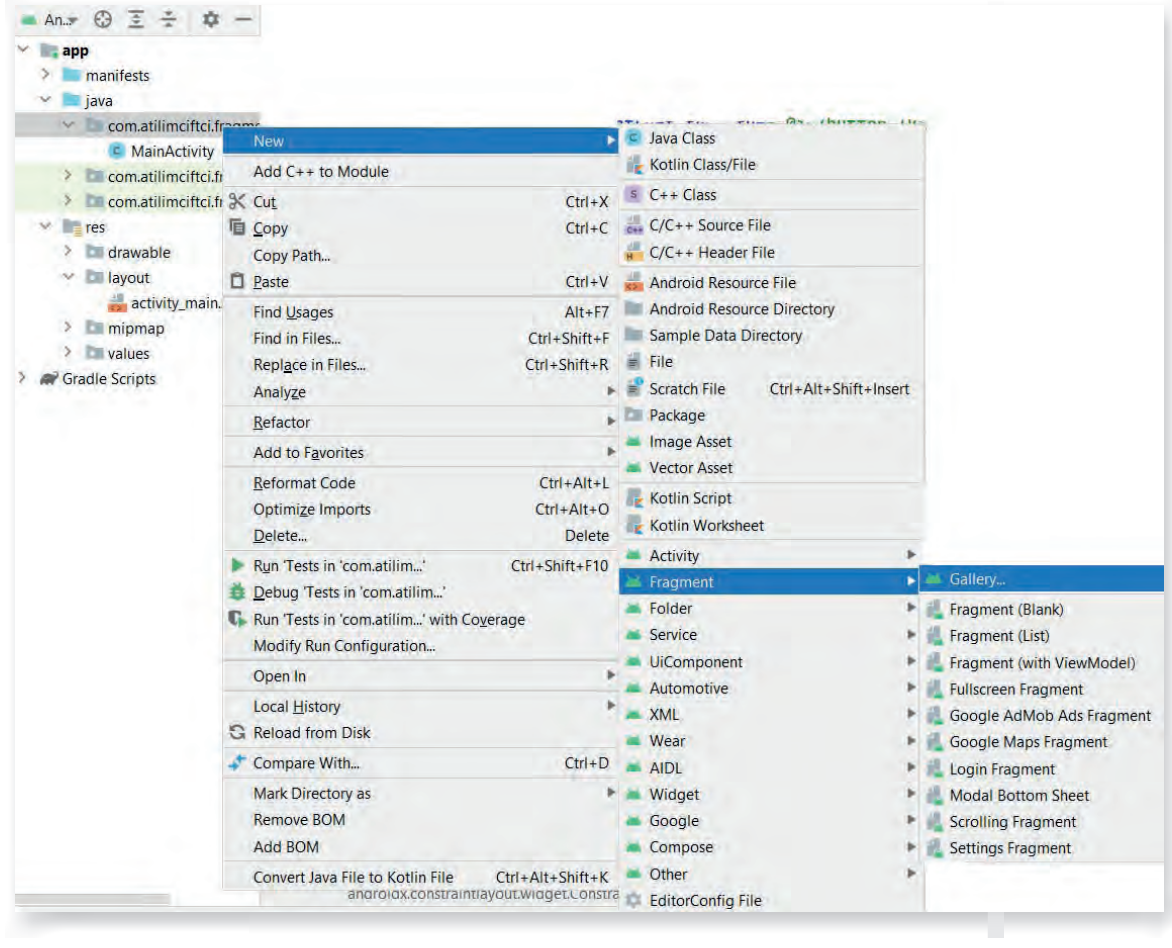
İç içe layout oluşturulabilir. Uygulamada da constraintlayout içinde bir framelayout oluşturulur. Ayrıca oluşturulan framelayoutun yüksekliği Split ekranında kontrol edilerek yazılır. Burada 490 dp verilmiştir. Framelayout yapısı, mobil ekranın başlangıcından buttonun olduğu bölmeye kadardır. Görsel 6.57'de tasarımın son hâli bulunur. Bu tasarım için kodlar şu şekildedir:



Görsel 6.57: FragmentFirst tasarım ekranı son hâli

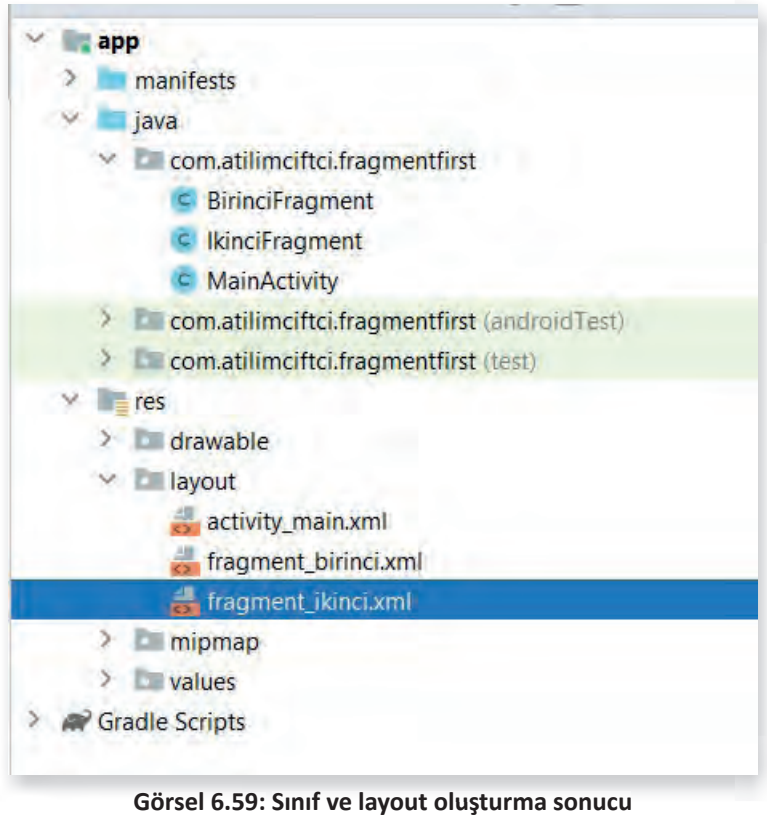
NOT:

Tasarım ekranında detaylı çalışma için hem Code hem Desing hem de Split ekranlarda çalışılır. Sekizinci uygulama bu nedenle Code ekranında tasarlanmış ve Split ekranda tasarımlar kontrol edilmiştir.



Görsel 6.58: Fragment ekleme menüsü

6. Adım: Fragment için bir sınıf bir de layout açılacaktır. Bu işlem tek tek sınıf ve layout oluşturup bağlanarak veya Görsel 6.58'de görüldüğü gibi otomatik olarak yapılabilir. Otomatik olarak Fragment oluşturmak için paket ismine mouse sağ tuşu ile tıklayınız. New sekmesinden Fragment sekmesine geliniz. Gallery seçeneğinden tüm Fragment seçenekleri görülebileceği gibi alt seçeneklerden uygun olanlardan biri de seçilebilir. Bu uygulamada Fragment'in içeriğini tam olarak anlayabilmek adına tek tek class ve layout ekleyerek sayfalar oluşturulacaktır. Bunun için de paket ismine sağ tuş ile tıklayıp **New>Java Class** seçeneğini seçiniz ve ismine de "BirinciFragment" giriniz. Aynı işlemi tekrar ederek bir sınıf daha oluşturunuz ve ismine de "IkinciFragment" giriniz. Bu sınıfların bağlanacağı layout dosyalarını oluşturmak için de layout klasörüne mouse sağ tuşu ile tıklayarak **New>Layout Resource File** seçeneğini tıklayınız ve "fragment_birinci" isminde layout dosyası oluşturunuz. Aynı işlemi tekrar ederek "fragment_ikinci" isminde de bir layout daha oluşturunuz. İşlem sonucu Görsel 6.59'da verilmiştir.



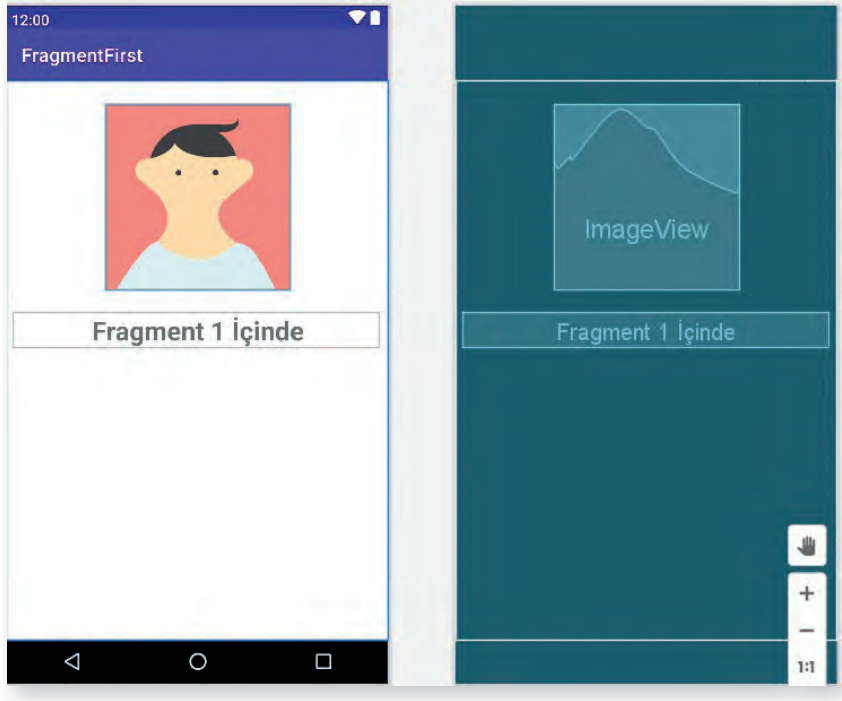
Görsel 6.59: Sınıf ve layout oluşturma sonucu

```
public class BirinciFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_birinci,container,false);
    }
}
```

7. Adım: Oluşturulan sınıfı ve layout dosyalarını birbirine bağlamak için "BirinciFragment" sınıfına girerek bunu Fragment sınıfına **extends** ediniz. Extends işlemi sonucunda onCerateView() yaşam döngüsü oluşturulup burada layouta bağlanır. Metodu oluşturmak için sınıf bloklarının içinde "onCreateView" yazınız ve önerilen koda Enter tuşu ile basınız. Burada "`return super.onCreateView(inflater, container, savedInstanceState);`" satırını siliniz. Sildiğiniz kodun yerine "`return inflater.inflate(R.layout.fragment_birinci,container,false);`" kodunu ekleyiniz. Aynı işlemi "IkinciFragment" için de tekrarlayarak "fragment_ikinci" isimli layouta bağlayınız.

8. Adım: Activitye Fragment'lerin eklenmesi için "FragmentManager" isimli bir yönetici sınıfı kullanılır. Öncelikle Fragment'leri ayırabilmek için "fragment_birinci" isimli layout dosyasına giderek Code ekranına geçiniz ve constraintlayout içinde bir adet ImageView bir adet de TextView oluşturunuz. Oluşum bittikten sonra ImageViewe sağ tuş tıklayarak "Set Sample Data" seçeneğinden bir resim ekleyiniz (Görsel 6.60).

```
<ImageView
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:id="@+id/imageViewFr1"
    android:layout_marginTop="25dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    ></ImageView>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/textViewFr1"
    app:layout_constraintLeft_toLeftOf="@id/imageViewFr1"
    app:layout_constraintRight_toRightOf="@id/imageViewFr1"
    app:layout_constraintTop_toBottomOf="@id/imageViewFr1"
    android:layout_marginTop="25dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:text="Fragment 1 İçinde"
    android:textAlignment="center"
    android:textSize="28sp"
    android:textStyle="bold"
    ></TextView>
```



Görsel 6.60: fragment_birinci layout

9. Adım: Benzer biçimde "fragment_ikinci" isimli layoutta da dosya oluşturunuz.

```
<ImageView
    android:id="@+id/imageViewFr2"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_marginTop="25dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:srcCompat="@tools:sample/avatars[1]"/>
<TextView
    android:id="@+id/textViewFr2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Fragment 2 İçinde"
    android:textAlignment="center"
    android:textSize="28sp"
    android:textStyle="bold"
    android:layout_marginTop="25dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    app:layout_constraintEnd_toEndOf="@+id/imageViewFr2"
    app:layout_constraintStart_toStartOf="@+id/imageViewFr2"
    app:layout_constraintTop_toBottomOf="@+id/imageViewFr2"/>
```

10. Adım: MainActivity sınıfına giderek buttonlar tıklandığında çalışacak metotları oluşturunuz. Fragment'leri aktivitelere bağlamak için FragmentManager'dan bir nesne oluşturunuz ve getSupportFragment ile içeriğini getiriniz. "FragmentTransaction" sınıfını kullanarak oluşturulan nesneyi başlatınız. Ardından BirinciFragment sınıfından bir nesne türetip bu Fragment nerede gösterilecekse (Uygulama için MainActivity'de FrameLayout'ta gösterilecektir.) orayı FragmentTransaction'a eklemek ve bağlantı kurmak gerekir.

```
public void fragment1Gecis(View view){
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    BirinciFragment birinciFragment = new BirinciFragment();
    fragmentTransaction.replace(R.id.frameLayout_Main,birinciFragment);
    fragmentTransaction.commit();
}
```

11. Adım: Onuncu adıma benzer durumu fragment2Gecis metodu için de yaparak ikinciFragment'i getiriniz. Daha sonra uygulamayı çalıştırarak denemesini yapınız.

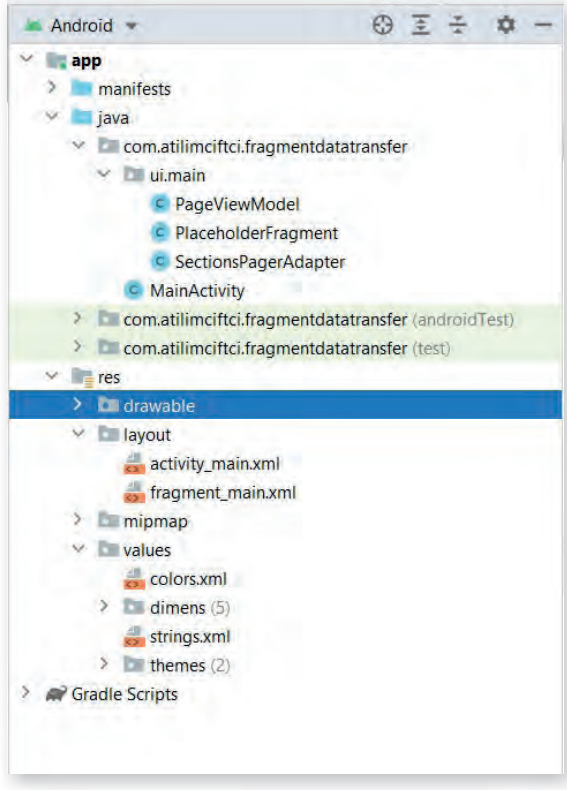
NOT: Bu uygulamada fragmentTransaction.add metodu da kullanılabilir fakat o durumda iki Fragment üst üste geçer. "replace()" metoduyla yerine yerleştirme yapılır.

```
public void fragment2Gecis(View view){
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction =fragmentManager.beginTransaction();
    IkinciFragment ikinciFragment = new IkinciFragment();
    fragmentTransaction.replace(R.id.frameLayout_Main,ikinciFragment);
    fragmentTransaction.commit();
}
```

Fragment'ler Arası Veri Paylaşımı: Aktiviteler arasındaki veri paylaşımında nasıl ki birden fazla aktivite, verilerini birbirine aktarabiliyorsa Fragment'ler de alt aktivite olmalarından ötürü içinde tuttukları bilgileri birbirleriyle paylaşabilir ve işleyebilir. Bu paylaşımı "ViewModel" denilen bir yapı ile sağlarlar. ViewModel, kendine ait bir sınıfı olan yapıdır. Bir Fragment elinde bulundurduğu bilgileri bu yapıya gönderir ve bilgiler sınıf içinde tutulur. Bu sayede de diğer Fragment bu bilgilere ulaşabilir ve kullanıcı için bu bilgileri işleyebilir. ViewModel yapısında da getter ve setter metotları ile çalışılır.

19. UYGULAMA: İşlem adımlarına göre TabbedActivity ile oluşturulan Activity içinde iki adet Fragment ile kullanıcı girişi yapan, kullanıcı adı ve şifresi editTextlere doğru girilmesi hâlinde herhangi bir buttona basmadan 2. Fragment'te "Hoşgeldiniz (KullanıcıAdı)" şeklinde belirten uygulamayı mobil uygulama tasarım programında hazırlayınız.

1. Adım: File>New>New Project seçeneğinden **TabbedActivity** seçiniz. İsmini de "FragmentDataTransfer" şeklinde oluşturunuz. Oluşan dosyanın yapısı Görsel 6.61'de verilmiştir.



Görsel 6.61: TabbedActivity ile uygulama

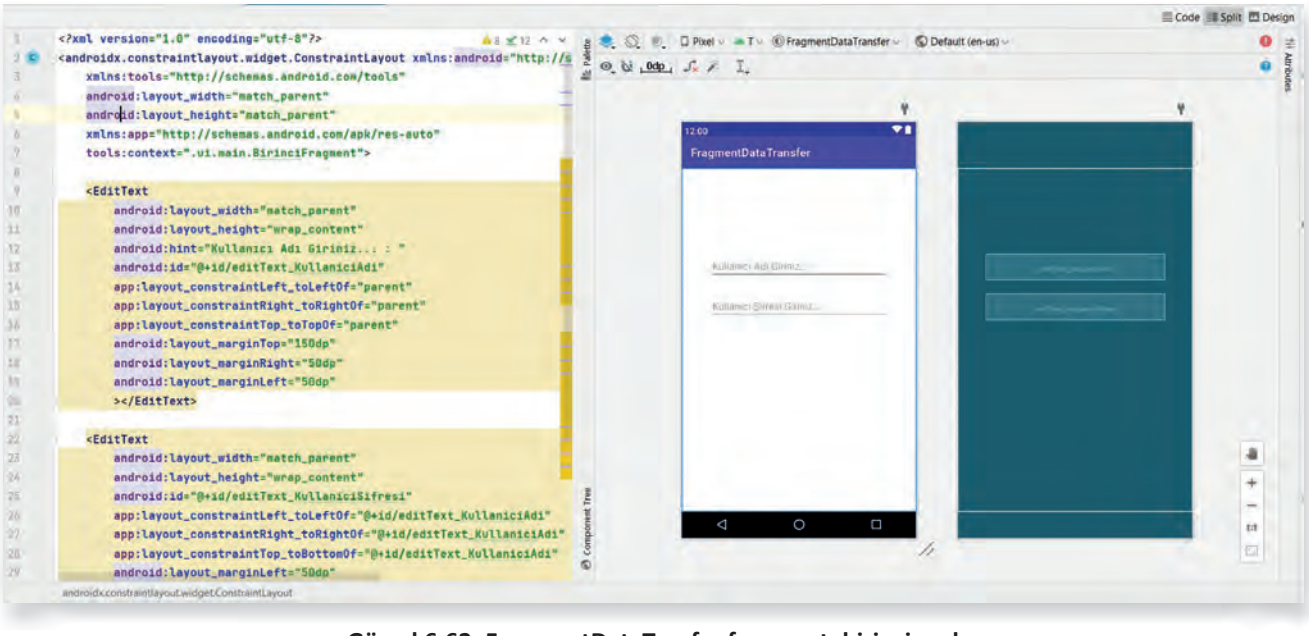
NOT:

TabbedActivity sıkça kullanılan bir Activity seçeneğidir. Uygulama ekranının üst kısmında oluşan tablardan meydana gelir ve istendiği miktarda tablar artırılabilir. Hazır bir biçimde oluşturulan PageViewModel sınıfı anlık olarak fragmentlerdeki verileri tutmak için tasarlanır. PlaceHolderFragment hazır olarak getirilmiş bir Fragment'tir. SectionPageAdapter ise TabbedActivity'ye ait verilerin tutulduğu bölümdür.

2. Adım: İki tabbed nesnesinin altında görünmesi için iki ayrı Fragment oluşturunuz. Bunun için otomatik olarak oluşturmayı seçiniz. **"ui.main"** klasörüne sağ tuş ile tıklayarak **New>Fragment>Fragment(Blank)** seçeneğini seçiniz. İsim olarak "BirinciFragment" diğerine "IkinciFragment" ismini veriniz.

3. Adım: fragment_birinci.xml içine giriniz. Split ekranını açınız. Oluşturulan FrameLayout etiketini ConstraintsLayout olarak değiştiriniz. İçerikteki TextViewi siliniz. "editText_KullaniciAdi" ve "editText_KullaniciSifresi" id'lerine sahip bir adet kullanıcı adı girişi için, bir adet de kullanıcı şifresi için EditText oluşturunuz (Görsel 6.62).

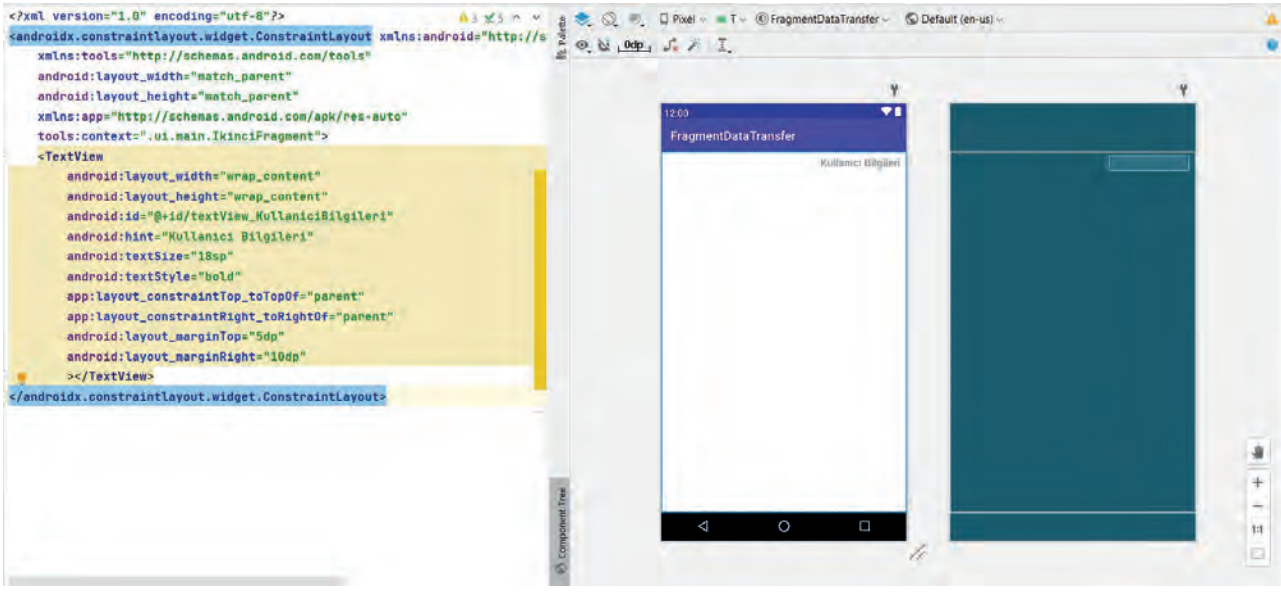
```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Kullanıcı Adı Giriniz... : "
    android:id="@+id/editText_KullaniciAdi"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:layout_marginTop="150dp"
    android:layout_marginRight="50dp"
    android:layout_marginLeft="50dp"
    ></EditText>

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText_KullaniciSifresi"
    app:layout_constraintLeft_toLeftOf="@+id/editText_KullaniciAdi"
    app:layout_constraintRight_toRightOf="@+id/editText_KullaniciAdi"
    app:layout_constraintTop_toBottomOf="@+id/editText_KullaniciAdi"
    android:layout_marginLeft="50dp"
    android:layout_marginRight="50dp"
    android:layout_marginTop="25dp"
    android:hint="Kullanıcı Şifresi Giriniz... :"
    ></EditText>
```



Görsel 6.62: FragmentDataTranfer fragment_birinci.xml

4. Adım: fragment_ikinci.xml içine giriniz. Split ekran tasarımına geçiniz. Kod bölümünden FrameLayout kısmını değiştirip ContraintLayout yapınız. KullanıcıAdını göstermesi için bir adet TextView tasarımı oluşturunuz. Id'si "textView_KullaniciBilgileri" olacaktır. TextView ögesini ekranın sağ üst köşesine konumlandırınız (Görsel 6.63).

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ui.main.IkinciFragment">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView_KullaniciBilgileri"
        android:hint="Kullanıcı Bilgileri"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="5dp"
        android:layout_marginRight="10dp"
        ></TextView>
</androidx.constraintlayout.widget.ConstraintLayout>
```



Görsel 6.63: FragmentDataTranfer fragment_ikinci.xml

5. Adım: PageViewModel sınıfının içeriğini inceleyiniz. PageViewModel sınıfının içinde "MutableLiveData" ve "LiveData" adı verilen yapılar bulunur. Bu yapılar dışında getter ve setter metotları vardır. Bu yapılar ve metotlar silinip sınıf içeriğini daha iyi anlamak adına PageViewModel sınıfı baştan yazılır. "`public class PageViewModel extends ViewModel { }`" haricinde içerideki tüm metotları siliniz. Ardından kullanıcı adı ve şifresi olarak bir sabit tanımlama yapınız. MutableLiveData oluşturarak kullaniciAdi ve kullaniciSifre için new parametresi ile açınız. kullaniciAdi ve kullaniciSifre private olarak tanımlandığı için dışarıdan gönderildiğinde içeriğine değer girecek setter metotları olan setKullaniciAdi ve setKullaniciSifre metotlarını oluşturunuz. LiveData tipinden veri döndüren bir metot tanımlanmalıdır. LiveData, canlı veri anlamına gelir. Bir başka deyişle işlemleri anlık olarak döndürür. private tanımlanan kullaniciAdi ve kullaniciSifre için dışarıdan erişim istendiğinde verileri alabilecek getter metotları olan getKullaniciAdi ve getKullaniciSifre metotlarını hazırlayınız. kullaniciSifrenin aslında fragment_ikinci.xml içinde gösterileceği için bir if kontrol yapısı ile şifre ve kullanıcı adı sorgulanır, doğru ise dönüş yapılır.

NOT: MutableLiveData, değiştirilebilen canlı veri anlamına gelir. kullaniciAdi ve kullaniciSifresi bu nedenle MutableLiveData<String> veri tipinde tanımlanır.

```
public class PageViewModel extends ViewModel {
    private static final String name = "atilimciftci";
    private static final String sifre = "12345";
    private MutableLiveData<String> kullaniciAdi = new MutableLiveData<>();
    private MutableLiveData<String> kullaniciSifre = new MutableLiveData<>();
    public void setKullaniciAdi(String kullaniciAdi) {
        this.kullaniciAdi.setValue(kullaniciAdi);
    }
    public void setKullaniciSifre(String kullaniciSifre) {
        this.kullaniciSifre.setValue(kullaniciSifre);
    }
    public LiveData<String> getKullaniciAdi() {
            return kullaniciAdi;
    }
    public LiveData<String> getKullaniciSifre(){
        return kullaniciSifre;
    }
    public static String getName() { return name; }
    public static String getSifre(){return sifre;}
}
```

6. Adım: Fragment sınıflarının düzenlenmesi gerekir. BirinciFragment sınıfını açınız. Karşınıza birçok karmaşık metot gelir. Bunları anlamak için en baştan yazılması daha uygundur. Bu nedenle içerideki tüm kodları silerek sadece "`public class BirinciFragment extends Fragment {}`" kalacak şekilde oluşumu düzenleyiniz. Sınıf tanımının olduğu bloka giriniz. Öncelikle **onCreate** sonrasında da **onCreateView** metodunu oluşturunuz. Bunun için onCreate yazarak önerilen metot yapısına Enter tuşu ile basınız. Sonrasında onCreate metodunun dışına çıkarak onCreateView yazınız ve öneriye Enter tuşu ile basınız. Yapı oluşturulduktan sonra onCreateView bloku içindeki return ile başlayan satırı silip yerine "`return inflater.inflate(R.layout.fragment_birinci,container,false);`" kodunu ekleyiniz.

```
public class BirinciFragment extends Fragment {
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_birinci,container,false);
    }
}
```

7. Adım: BirinciFragment sınıfının içinde ViewModel yapısının tanıtılıp başlatılması gerekir. Yaşam döngülerinin dışına çıkınız ve onCreate metodunun üstüne geliniz. PageViewModel sınıfından pageViewModel nesnesini oluşturunuz. Bu nesneyi de onCreate yaşam döngüsünün içinde initialize ediniz. Initialize işlemi "`pageViewModel = new ViewModelProvider(requireActivity()).get(PageViewModel.class);`" şeklinde olacaktır. Bu kodda ViewModelProvider yapısını kullanarak geçerli olan Activity üzerine PageViewModel içindeki dataların bağlanması belirtilir.

8. Adım: onCreatedView metodunu tanımlayınız. Tüm metotların dışına çıkarak onCreatedView yazınız ve çıkan öneriye Enter tuşu ile basınız. Bu metot içinde, metoda ait layout dosyasında bulunan ögelerin tanımlanması ve işleme alınması uygulamalarını gerçekleştiriniz. Bir başka deyişle fragment_birinci.xml dosyasında yer alan editText_KullaniciAdi ve editText_KullaniciSifresi'nin tanımlamalarını yapınız. Ardından bu editTextteki anlık değişimleri kontrol eden metodu uygulayınız. Bunun için de **"addTextChangedListener"** metodu kullanılır. Bu metoda parametre olarak istenen **"TextWatcher"** ögesi new ile yazılır ve **"new TextWatcher"** şeklinde gönderilir. EditText ögesine ait tanımlanan EditText nesnesi kullanılarak oluşturulan addTextChangedListener metodu üç ayrı metot döndürerek override edecektir. Bunlar; text değişmeden önce, text değişmeden sonra ve text değişirken yapılacak işlemlere ait metotlardır.

NOT: EditText sınıfından nesne üretilirken layout dosyasında bulunanEditTexte ulaşmak için öncelikle onViewCreated sınıfına View ile gönderilen parametre kullanılır. findViewById ile tanımlama yapılmadan önce **view.findViewById** ile erişim sağlanmalıdır.

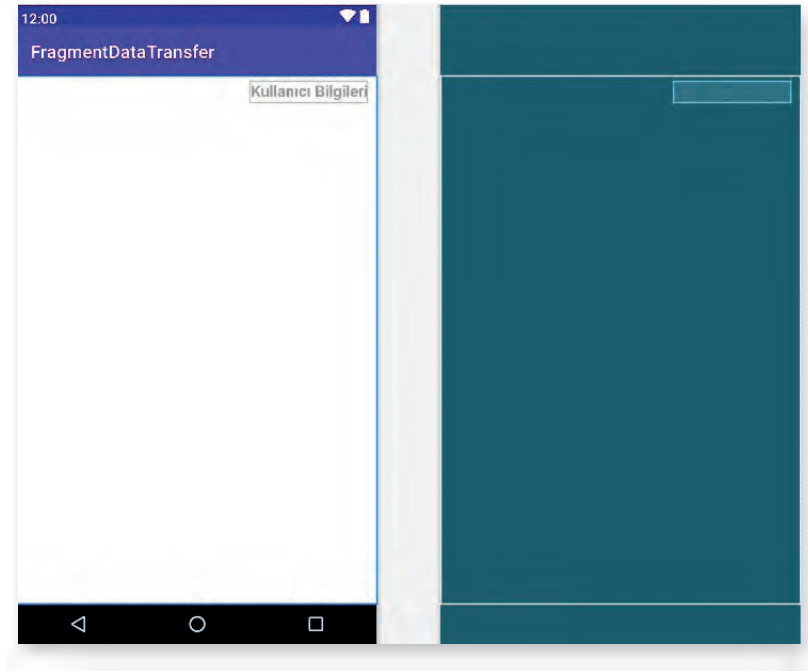
9. Adım: Bu sınıf için son olarak constructer oluşturulması gerekir. Bu constructer sayesinde SectionPagerAdapter sınıfı içinde hangi Tab bölümü seçilirse o Tab bölümüne ait Fragment çağrılır. Bu da bir başka Fragment çağırma yöntemidir. BirinciFragment içeriği şu şekildedir:

```
public class BirinciFragment extends Fragment {
    PageViewModel pageViewModel;
    public static BirinciFragment birinciFragment(){
        return new BirinciFragment();
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        pageViewModel = new ViewModelProvider(requireActivity()).get(PageView-
Model.class);
    }

        @Nullable
        @Override
        public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
            return inflater.inflate(R.layout.fragment_birinci,container,false);
        }
```

```
 @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstan-
ceState) {
        super.onViewCreated(view, savedInstanceState);
        EditText editText_KullaniciAdi = view.findViewById(R.id.editText_Kulla-
niciAdi);
        EditText editText_KullaniciSifre = view.findViewById(R.id.editText_Kul-
laniciSifresi);

        editText_KullaniciAdi.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1,
int i2) {
                pageViewModel.setKullaniciAdi(charSequence.toString());
            }
            @Override
            public void afterTextChanged(Editable editable) {
            }
        });

        editText_KullaniciSifre.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int
i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1,
int i2) {
                pageViewModel.setKullaniciSifre(charSequence.toString());
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ui.main.IkinciFragment">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView_KullaniciBilgileri"
        android:hint="Kullanıcı Bilgileri"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="5dp"
        android:layout_marginRight="10dp"
        ></TextView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

10. Adım: BirinciFragment sınıfının içeriği ve fragment_birinci.xml layoutuna benzer şekilde IkinciFragment sınıfı ve fragment_ikinci.xml oluşturulur. fragment_ikinci.xml için içerik tasarlanırken Code ekranında Frame_Layoutu, Constraint_Layout şeklinde değiştirilir. İçeriğe bir TextView eklenir (Görsel 6.64). fragment_ikinci.xml Code ekranı şu şekildedir:



Görsel 6.64: ikinci_fragment.xml içeriği

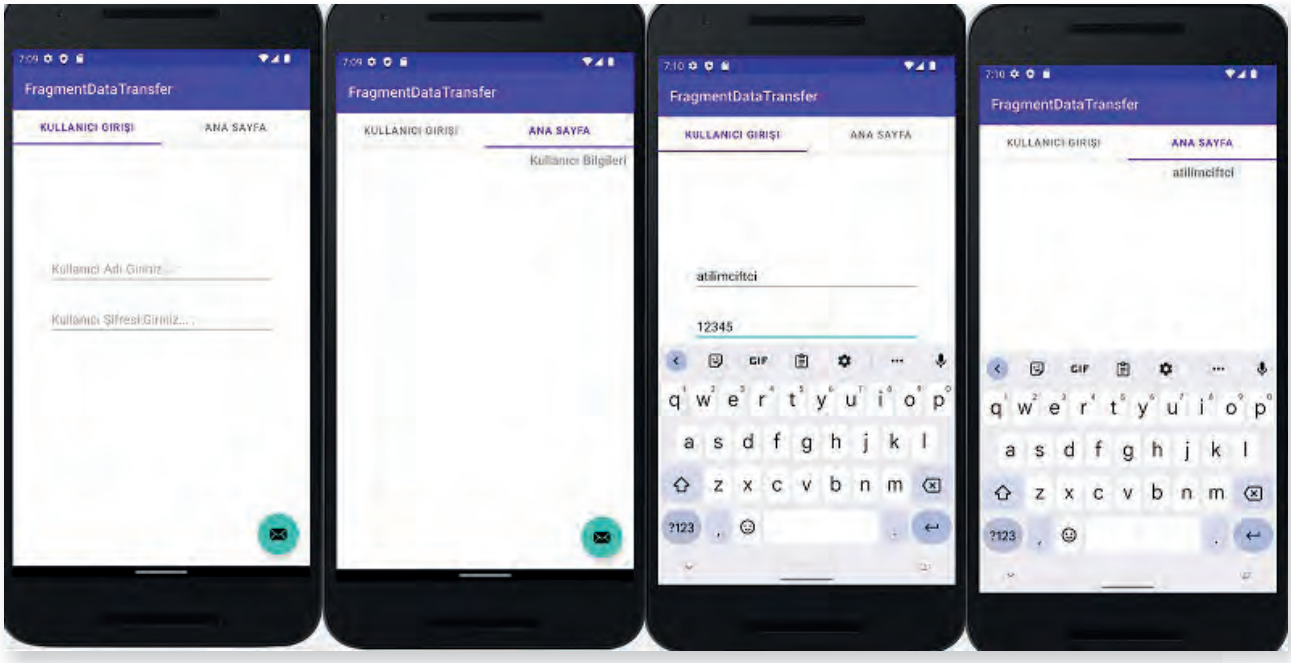
11. Adım: IkinciFragment.java içeriğini şu şekilde tasarlayınız:

```
public class IkinciFragment extends Fragment {

    PageViewModel pageViewModel;
    public static IkinciFragment ikinciFragment(){
        return new IkinciFragment();
    }
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        pageViewModel = new ViewModelProvider(requireActivity()).get(PageViewModel.class);
    }
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup contai-
ner, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_ikinci,container,false);
    }
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        TextView textView_KullaniciAdi=view.findViewById(R.id.textView_KullaniciBilgileri);
        pageViewModel.getKullaniciAdi().observe(requireActivity(), new Observer<String>()
{
            @Override
            public void onChanged(String s) {
                if(pageViewModel.getName().equals(s)){
                    pageViewModel.getKullaniciSifre().observe(requireActivity(), new Ob-
server<String>() {
                        @Override
                        public void onChanged(String t) {
                            if (pageViewModel.getSifre().toString().equals(t)){
                                textView_KullaniciAdi.setText(s);
                            }else
                            {
                                textView_KullaniciAdi.setText("Kullanici Bilgileri");
                            }
                        }
                    });
                }
                else
                {
                    textView_KullaniciAdi.setText("Kullanici Bilgileri");
                }
            }
        });
    }
}
```

NOT:

Bu kodlar arasındaki **observe(),** yeni bir metottur. LiveData türünün en önemli özelliği de buradaki observe() metodudur. Bu metot sayesinde anlık olarak diğer Fragment'teki değişim takip edilir, TextWatcher parametresi ile izlenir. BirinciFragment ögesinden dönen kullaniciAdi bilgisi doğru ise kullaniciSifresi bilgisine anlık olarak bakılır ve o da doğru olduğu takdirde IkinciFragment ögesine PageViewModel sınıfı üzerinden veriler getirilip kullaniciAdi bilgisi IkinciFragment'teki TextViewe yazılır. Bu şart gerçekleşmezse TextViewde "Kullanici Bilgileri" yazar.

12. Adım: TabbedActivity ile otomatik gelen Tab1, Tab2 isimlendirmesi **SectionsPagerAdapter** sınıfı içinde **TAB_TITLES** ile verilmiş olsa da elemanlara tıklandığında da görüleceği gibi dizinin elemanları strings.xml üzerinden gelir. İsimleri strings.xml üzerine giderek "Kullanıcı Girişi" ve "Ana Sayfa" şeklinde düzenleyiniz. Daha sonra uygulamayı çalıştırarak Görsel 6.65'tekine benzer bir görüntü alınız.



Görsel 6.65: FragmentDataTransfer

NOT:

TabbedActivity ile otomatik yüklenen **PlaceHolderFragment** sınıfı ve bu sınıfın bağlı olduğu **fragment_main** örnek üzerinde kullanılmamıştır. Bu sınıfın silinmesi bir sorun oluşturmaz. Yirmi birinci uygulamada canlı verilerle çalışılmış ve Fragment'ler arasında veri transferi yapılmıştır.

! **UYARI:** Fragment, mobil uygulamaları arasında oldukça gelişmiş ve üst düzey bir yapıdır.

6.6. MOBİL UYGULAMADA İZİNLER VE İZİN YAPISI

Mobil uygulama platformunda "Android 6.0" (API level 23) sürümüne kadar sadece mobil uygulama indirilirken kullanılacak durumlar için izinler sorulurdu. Onun dışında uygulama, kullanıcı tarafından kullanılırken izin sorulmaya gerek duyulmazdı. Bu durum, 6.0 ile birlikte değişiklik göstermiş ve tehlikeli izin olarak adlandırılan (dangerous permissions) izinlerin uygulama içinde de sorulması ve izin verilmedikçe kullanılmaması şartı getirilmiştir. Bu durum; rehbere erişim, kamera kullanımı, fotoğraf galerisine erişim, konum paylaşımı gibi olaylarda kullanıcının ekstra izin vermesi gerekliliğini ortaya çıkartmıştır. Kullanıcılar tarafından farkındalık düzeyi artırılarak sakıncalı olabilecek durumların önüne geçilmesi ve veri gizliliğinin sağlanması amaçlanmıştır.

İzinler için isimlendirilen dört seviye mevcuttur.

- **Normal**

Düşük seviyeli izinleri kapsar ve sistem tarafından otomatik olarak izin verilir. Genelde her uygulamada kullanılan ve kullanıcıya zararı olmayan izin çeşitleridir. Wi-Fi kullanımı, Bluetooth kullanımı bu seviyedeki izinlere örnektir. Bu seviyedeki izinlerin sadece uygulamayı marketten indirirken söylenmesi yeterlidir. Uygulama içinde tekrar sorulmasına gerek kalmadan kullanıma izin verilir.

- **Dangerous**

Kullanıcıya ait özel verilere erişim için gereken izin türüdür. Bu tür izin gerektiren durumlar, kullanıcı için uygulama içinde tehlikeli durum yaratabileceğinden dolayı uygulamada ayrıca sorulmadan içeriği görüntülemeye, verileri almaya veya dizine girmeye izin verilmez. Örneğin bir uygulama, tehlikeli olabilecek bir veriyi talep ettiyse bu veri etkileşimine izin verilmeden önce kullanıcıdan onay istenir. Rehbere erişim, galeriye erişim, konum kullanımı bu türdeki izinlere örnektir.

- **Signature**

Mobil uygulamada sistemin, yalnızca talep eden ve izin veren uygulamanın sertifikaları eşleşiyorsa vereceği izindir. Sertifikalar eşleşirse sistem, kullanıcı onayı olmadan otomatik şekilde izin verir.

- **SignatureOrSystem**

SignatureOrSystem izni, birden çok satıcının bir sistem görüntüsünde yerleşik uygulamalara sahip olduğu ve birlikte oluşturuldukları için belirli özellikleri açıkça paylaşmaları gerektiği özel durumlarda kullanılır. Bu izin türünün kullanılmasından mümkün mertebe kaçınmakta fayda vardır. İmza koruma düzeyine bakılmayacağı için kontrolü oldukça sorunlu olabilir.

NOT: İzin seviyeleri hakkında detaylı bilgi için https://developer.android.com/guide/topics/permissions/overview#normal-dangerous doküman sayfasını inceleyiniz.

Kullanılan mobil cihaz API level 23 ve daha üst bir sürüm ise marketten indirme aşamasında herhangi bir izin için onay kutucuğu göstermez fakat dangerious seviye olan izinler için uygulama içinde mecburi onay almak gerekir.

6.6.1. İzin Tanımlamaları Yapısı

Mobil uygulamada kullanılan yapılara ait özelliklerin izninin hangi seviyede olursa olsun uygulama içinde belirtilmesi gerekir. Bunun için de "androidManifest.xml" dosyası kullanılır. Bu dosya içinde **<manifest>** etiketi arasında **<uses-permission>** etiketi kullanılarak izin tanımlaması yapılır (Görsel 6.66).

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.atilimciftci.fragmentdatatransfer">
    <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"></uses-permission>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="FragmentDataTransfer"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.FragmentDataTransfer">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="FragmentDataTransfer"
            android:theme="@style/Theme.FragmentDataTransfer.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

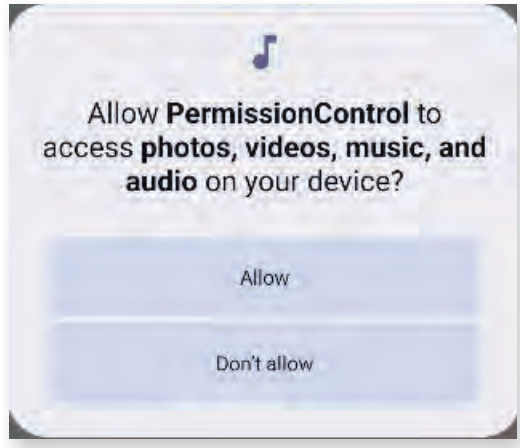
Görsel 6.66: uses-permission etiketinin kullanımı

Kullanılan izin yapısı; doküman tarafından seviye olarak normal şeklinde tanımlanmışsa, ayrıca kullanıcı gizliliğini ihlal etmezse veya cihazın çalışmasına karşı risk oluşturmazsa sistem bu izinleri otomatik olarak verir. Kullanılmak istenen izin yapısı kullanıcı verilerine erişecek (SMS kullanımı, galeri erişimi, rehber erişimi vb.) veya cihazın çalışması ile ilgili sorunlara neden olabilecek türde ise uygulama içinde kullanıcıdan ekstra izin alınması gerekir.

NOT: İzin çeşitleri hakkında daha detaylı bilgi için https://developer.android.com/reference/android/Manifest.permission doküman sayfasını inceleyiniz. Doküman içinde tüm izinlerin nasıl kullanılacağı, ne amaçla kullanılacağı ve hangi düzeyde olduğu belirtilir.

6.6.2. İzin Onayları Almak

Oluşturulan mobil uygulama içinde normal izin seviyesinde olan izinler için sadece androidManifest.xml dosyası içinde tanımlamak yeterliyken tehlikeli olarak nitelendirilen izinler için hem androidManifest.xml'de tanımlanması hem de kullanım içinde Görsel 6.67'de olduğu gibi **"İzin ver", "Reddet"** buttonlarının olduğu bir diyalog ekranı ile izin alınması gerekir.



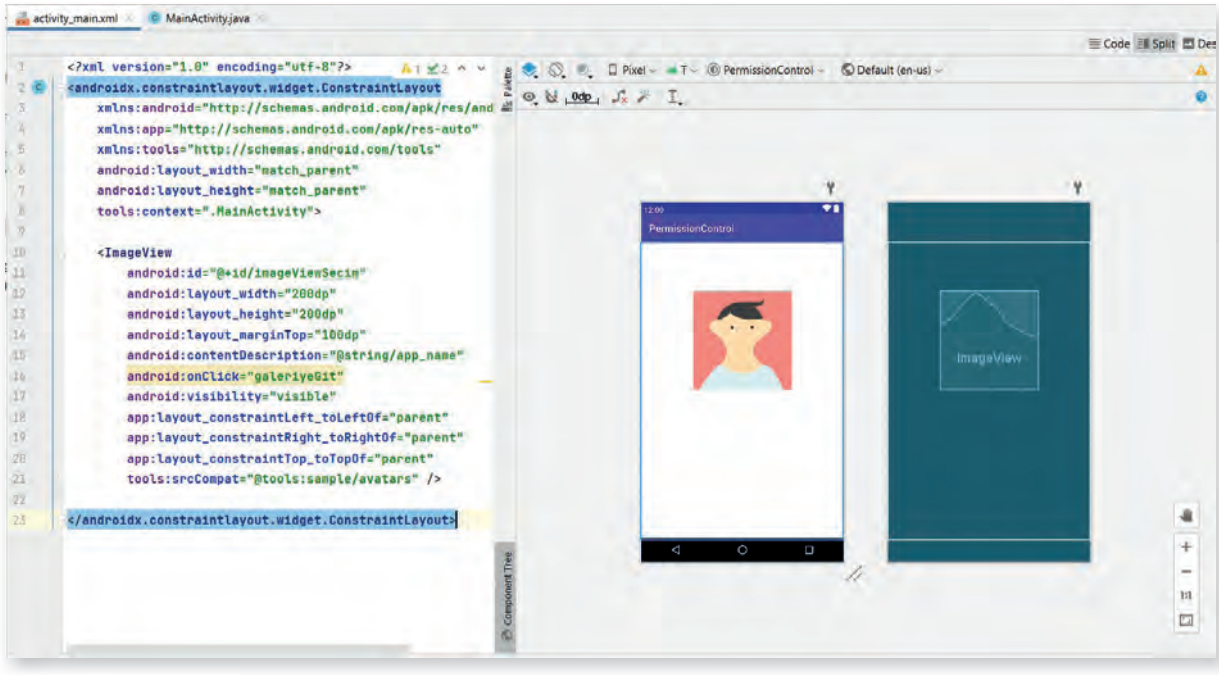
Görsel 6.67: İzin sorma ekranı

Kullanıcı, **"İzin ver" (Allow)** seçeneğini seçerse herhangi bir sıkıntı olmadan işlemlere devam edilir ve veri erişimine veya servis erişimlerine izin verilir. Kullanıcı, **"Reddet" (Don't allow)** seçeneğini seçerse erişim sağlanmak istenen bölüm izin verilmeden iptal edilir. Aynı işlem tekrar edildiğinde aynı onay kutucuğu sorulur. **"Tekrar sorma" (Never ask again)** seçeneği seçilip **Reddet** tıklanırsa bu erişim istendiğinde işlem gerçekleşmez ve tekrar soru da sorulmaz. Bu tür durumlarda iki seçenek ele alınır. Tasarlayan kişi bunun önemini bildiren actionTab tarzı bir yapı oluşturabilir veya kullanıcı, mobil cihazının ayarlarından izinlere gidip, reddettiği izni bulup verebilir.

! **UYARI:** Her kullanıcı aynı bilgi ve teknik donanım seviyesinde olmadığı için kullanıcının ayarlar üzerinden izinlere gitmesini ve izin ver seçeneğini oradan aktif etmesini beklemek doğru bir işlem değildir. Bu işlemi kullanıcıya bırakmaktansa tasarımcının uygulama içinde bu tür durumlara önlem alması ve izin vermeyen kullanıcılar için yeniden bağlantı oluşturması her zaman daha mantıklı bir seçenektir.

20. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranındaki Activity içinde bir ImageViewe tıklanınca galeriye gitmek için izin isteyen ve izin verildiğinde galeriye ulaşarak seçilen fotoğrafı ImageViewe getiren uygulamayı tasarlayınız.

1. Adım: New>Project>Empty Activity ile yeni bir proje oluşturunuz. İsmini "Permission Control" olarak giriniz. activity_main.xml içine ConstraintLayout içine bir adet ImageView ögesi ekleyiniz. Görüntü olarak herhangi bir avatar seçerek Code ekranında tasarıma devam ediniz. id değerini "imageViewSecim" olarak giriniz. onClick özelliği için de "galeriyeGit" yazınız (Görsel 6.68).



Görsel 6.68: PermissionControl uygulaması

2. Adım: AndroidManifest.xml dosyasını açınız. manifest etiketi arasına şu şekilde kodlamayı yazınız:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"></
uses-permission>
```

NOT: "READ_EXTERNAL_STORAGE" izni, mobil cihaz içinde yer alan depolama birimine erişim ve depolama biriminden veri okuma için gerekli olan izin isteğinde kullanılır. Detaylı bir liste için https://developer.android.com/reference/android/Manifest.permission adresini inceleyiniz.

3. Adım: galeriyeGit isimli imageViewe tıklandığında çalışması gereken metodu MainActivity içinde tanımlayınız. Bu metot içinde öncelikle galeriye erişim için gerekli iznin verilip verilmediği kontrol edilir. Bu kontrol işlemi, **ContextCompat.checkSelfPermission()** metodu ile yapılır. Metoda parametre olarak da context ve izin parametreleri gönderilmelidir. Bu metot, dönüş tipi olarak izin verildi **(PackageManager.PERMISSION_GRANTED)** ya da izin verilmedi **(PackageManager.PERMISSION_DENIED)** şeklinde dönüş yapar.

```
public void galeriyeGit(View view){
    if(ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTER-
NAL_STORAGE)== PackageManager.PERMISSION_DENIED){
        //İzin yok, izin istenecektir.
    }else{
        //İzin var, galeriye gidilecektir.
        Intent galeri =new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.
EXTERNAL_CONTENT_URI);
    }
}
```

Galeriye gitme adımını yapınız. Galeriye de tıpkı farklı bir Activity'ye gitmek için kullanılan Intent ile gidilir. Buradaki fark, Activity açarken parametre olarak **context** ve **sınıf** gönderilir. Galeri gibi bir lokasyona giderken ise bir **action** bir de **URI** parametresi gönderilir. Galeriye gitmek için yazılması gereken şu kodu else bölümüne ekleyiniz:

```
Intent galeri =new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_
CONTENT_URI);
```

"`Intent.ACTION_PICK`", intente git ve oradakini al anlamına gelen bir aksiyondur. "`MediaStore.Images.Media.EXTERNAL_CONTENT_URI`" ise resimlere ulaşmak için gerekli olan URI adresidir.

NOT:

URI (Uniform Resource Identifier), URL'nin (Uniform Resource Locator) detaylandırılmış hâlidir. URL, kaynağın yerine işaret eden standart bir formata uygun karakter dizisi iken URI, kaynağın tam yerine işaret eden (resim veya belge) standart formata uygun bir karakter dizisidir.

4. Adım: İzinleri istemek için gereken ve Android 11'den sonra izin konularında yapılan geliştirmelerle yeni kullanılmaya başlayan **`ActivityResultLauncher`** sınıfı kullanılır. Bu sınıftan üretilen nesne sayesinde hem izinlerin kontrolü yapılır hem de galeriye gidilip oradan veri döndürülebilir. Galeri için ayrı bir `ActivityResultLauncher`, izinler için ise ayrı bir `ActivityResultLauncher` tanımlanır. Bunları MainActivity blokları arasında, onCreate() metodunun dışında ve üst kısmında şu şekilde tanımlayınız:

```
ActivityResultLauncher<Intent> galeryResultLauncher;
ActivityResultLauncher<String> izinlerResultLauncher;
```

Burada oluşturulan "galeryResultLauncher", izin alındığı takdirde galeriye gitmesi için yazılan nesnedir. Intent ile gönderim yapılacağı için de dönüş tipi Intent'tir. "izinlerResultLauncher" nesnesi ise izin istemek için yazılır ve String tipte olmalıdır. Oluşturulan bu nesneler onCreate içinde tanımlanmalıdır fakat içerik biraz karışık olacağı için ayrı birer metotta bunları başlatıp sonrasında bu oluşturulan metotları onCreate() içinde çağırmak daha mantıklıdır. Galeriye gidebilmesi için gereken metodu ve bu metot kodlarını MainActivity içinde oluşturunuz. Nesne içeriği yazılırken öneri olarak seçenekler çıktığında Enter tuşuna basılırsa otomatik şekilde override edilir. **RESULT_OK,** galeride OK tuşuna basılarak dönüldüyse anlamını taşır ve gelen veri de boş değilse (null) veriyi URI sınıfından bir fotoVeri değişkenine kaydeder. Bu işlem için yazılması gereken kodlar şu şekildedir:

```
public void metotGaleryResultLauncher(){
    galeryResultLauncher = registerForActivityResult(new ActivityResultContra-
cts.StartActivityForResult(), new ActivityResultCallback<ActivityResult>() {
        @Override
        public void onActivityResult(ActivityResult result) {
            if(result.getResultCode()==RESULT_OK){
                Intent intentSonuc = result.getData();
                if(intentSonuc!=null){
                    Uri fotoVeri = intentSonuc.getData();
                }
            }
        }});
}
```

5. Adım: metotIzinlerResultLauncher isimli metodu oluşturunuz ve bu nesne ile izin işlemlerinin gerçekleşmesini sağlayınız. Kodlar dördüncü adımda olduğu gibi yazılırken önerilere Enter tuşu ile basıldığında çoğunluk otomatik şekilde oluşturulup override edilir. İzin verilmediğinde bir Toast mesajı ile hatırlatma yapılır.

```
public void metotIzinlerResultLauncher(){
    izinlerResultLauncher = registerForActivityResult(new ActivityResultContra-
cts.RequestPermission(), new ActivityResultCallback<Boolean>() {
        @Override
        public void onActivityResult(Boolean result) {
            if(result==true){
                //İzin var, galeriye gidilecektir.
                Intent galeri =new Intent(Intent.ACTION_PICK, MediaStore.Ima-
ges.Media.EXTERNAL_CONTENT_URI);
            }else
            {
              //İzin istenmesi gerekecektir.
                Toast.makeText(MainActivity.this, "İzin vermeniz gerekir.", To-
ast.LENGTH_SHORT).show();
            }
        }
    });
}
```

6. Adım: MainActivity sınıfında onCreate() metodunun içinde, dördüncü ve beşinci adımda oluşturulan metotları çağırınız ve kayıt işlemini yapınız.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    metotGaleryResultLauncher();
    metotIzinlerResultLauncher();
}
```

7. Adım: Intent ile galeriye erişim için "galeri" nesnesi oluşturulup henüz bu nesne kullanılmamıştır. Bu nedenle **metotIzinlerResultActivity** içinde, Intent ile nesnesi oluşturulan satırın bir alt satırında şu kodu kullanarak galeriye gidiniz:

```
galeryResultLauncher.launch(galeri);
```

Aynı kodu **galeriyeGit** metodu içinde yazılan Intent'in bir alt satırına da ekleyiniz. Bu kod ile galeriye erişiniz.

8. Adım: Önceki adımlarda izin istenecek bölümlere yorum satırı eklenmiştir. Yorum satırı eklenen galeriyeGit metodunda izin istemek için şu kodu yazınız:

```
izinlerResultLauncher.launch(Manifest.permission.READ_EXTERNAL_STORAGE);
```

9. Adım: ImageView ögesini ister binding yöntemi ister findViewById ile tanımlayınız. Burada öncelikle tek öge olduğu için findViewById ile tanımlanır. MainActivity sınıfının içine "`ImageView imageView;`" ekleyiniz. onCreate() metoduna da "`imageView = findViewById(R.id.imageViewSecim);`" şeklinde kodu ekleyerek imageView ögesini initialize ediniz.

10. Adım: Mobil uygulama geliştirme platformunun yeni güncellemesinde imageView nesnelerine öge atanmadığında ekranda göstermeme durumları oluşur. Ekranda imageView görünmediyse **drawable** klasörü içine uygun olan herhangi bir görsel ögesi koyarak onCreate içinde "`imageView.setImageResource(R.drawable.gorselsecim);`" görsel içeriğini atayınız.

11. Adım: Alınan verinin imageView içine gömülmesi gerekir. Bunun için **metotGaleryForResult()** metodu içinde fotoVeri değişkenini çektiğiniz satırın altına "`imageView.setImageURI(fotoVeri);`" kodunu ekleyerek seçilen görselin imageView içine yüklenmesini sağlayınız. Uygulamayı çalıştırdığınızda Görsel 6.69'daki gibi bir görüntü ortaya çıkar.

Görsel 6.69: PermissionControl uygulaması

Tüm MainActivity içeriği şu şekildedir:

```
package com.atilimciftci.permissioncontrol;
public class MainActivity extends AppCompatActivity {
    ActivityResultLauncher<Intent> galeryResultLauncher;
    ActivityResultLauncher<String> izinlerResultLauncher;
    ImageView imageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        metotGaleryResultLauncher();
        metotIzinlerResultLauncher();
        imageView = findViewById(R.id.imageViewSecim);
        imageView.setImageResource(R.drawable.gorselsecim);
    }
    public void galeriyeGit(View view){
        if(ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EX-
TERNAL_STORAGE)== PackageManager.PERMISSION_DENIED){
            //İzin yok, izin istenecektir.
            izinlerResultLauncher.launch(Manifest.permission.READ_EXTERNAL_STO-
RAGE);
        }else{
            //İzin var, galeriye gidilecektir.
            Intent galeri =new Intent(Intent.ACTION_PICK, MediaStore.Images.
Media.EXTERNAL_CONTENT_URI);
            galeryResultLauncher.launch(galeri);
        }
    }
    public void metotGaleryResultLauncher(){
        galeryResultLauncher = registerForActivityResult(new ActivityResult-
Contracts.StartActivityForResult(), new ActivityResultCallback<ActivityRe-
sult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if(result.getResultCode()==RESULT_OK){
                    Intent intentSonuc = result.getData();
                    if(intentSonuc!=null){
                        Uri fotoVeri = intentSonuc.getData();
                        imageView.setImageURI(fotoVeri);
                    }
                }
            }
        });
    }
    public void metotIzinlerResultLauncher(){
        izinlerResultLauncher = registerForActivityResult(new ActivityResult-
Contracts.RequestPermission(), new ActivityResultCallback<Boolean>() {
            @Override
```

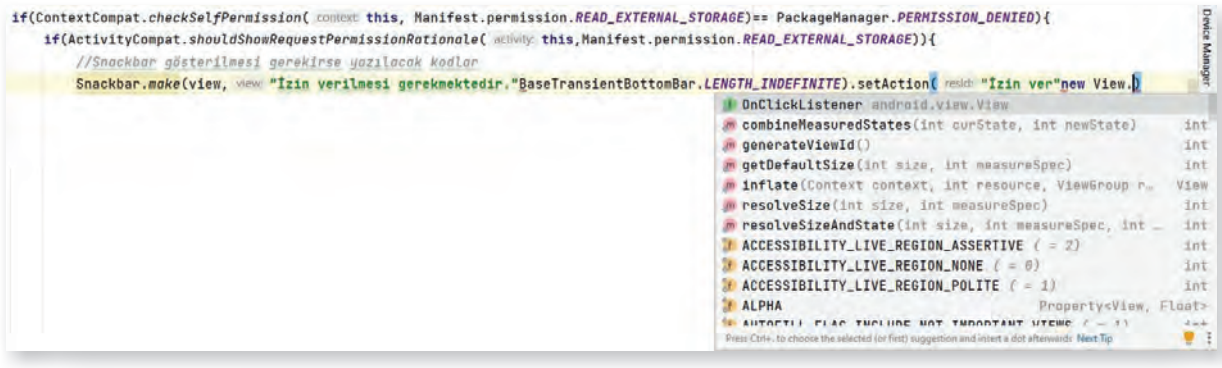
```
            public void onActivityResult(Boolean result) {
                if(result==true){
                    //İzin var, galeriye gidilecektir.
                    Intent galeri =new Intent(Intent.ACTION_PICK, MediaStore.
Images.Media.EXTERNAL_CONTENT_URI);
                    galeryResultLauncher.launch(galeri);
                }else
                {
                    //İzin istenmesi gerekecektir.
                    Toast.makeText(MainActivity.this, "İzin vermeniz gerekmek-
tedir.", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

NOT: Uygulamada kullanılan emülatör içinde normal şartlarda görsel vb. bulunmaz. Emülatörün tarayıcısı ile internette görsel aratıp, üzerine basılı tutarak "Download" seçilirse seçilen görsel de galeriye iner. Bu sayede galeriye erişim sağlandığında görseller ortaya çıkar.

! **UYARI:** İzin uygulamalarında genel olarak karşılaşılan bir sorun vardır. Kullanıcı, bir daha gösterme seçeneği çıkarsa ve bunu işaretleyip izin vermezse veya Görsel 6.67'de olduğu gibi bir seçenekte iki defa izin vermezse kullanıcıya tekrar izin sorulmaz. Kullanıcı bu durumda Ayarlar üzerinden izinleri bulup, uygulamaya ait izne gelip bu izni kabul etmelidir. Her kullanıcının bunu yapabilmesi mümkün olmadığı için de bu durumu mobil uygulama geliştiricisi düşünmelidir. Bunun için gerekli mobil uygulama geliştirme platformunun yeni güncellemesiyle bir metot geliştirilmiştir. Bu, "shouldShowRequestPermissionRationale" metodudur. İzin isteme ekranı için Snackbar yazıldığında bu Snackbarın gösterilmesinin gerekli olup olmadığını sorgular. İlgili metodun içine yazılacak bir Snackbar ile de sadece gerekli olduğu durumlarda bir menü ile aynı izin çağrılabilir. Hâlihazırda var olan bir Snackbarı ilk izin istemede göstermez. "Yeniden gösterme" seçeneği tıklandığında veya iki defa normal izin verilmeyerek kapatıldığında sonrasında izin istemek için tıklandığı zaman bu Snackbar ile izin işlemlerine tekrar dönülebilir. Bunun için onuncu uygulamaya şu eklemeler yapılmalıdır:

1. Adım: galeriyeGit() metodu içinde yer alan karar yapısında izin verilip verilmediği sorgulanır. İzin verilmediği durumda Snackbar göstergesinin gösterilmesi gerekip gerekmediğini belirten "shouldShowRequestPermissionRationale" ile karar yapısı yazınız.

```
if(ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_
STORAGE)== PackageManager.PERMISSION_DENIED)
{
        //Snackbar gösterilmesi gerekirse yazılacak kodlar
}else
{
        //Snackbar gösterilmeden izin istenecek kodlar
}
```



Görsel 6.70: Snackbar tasarımı

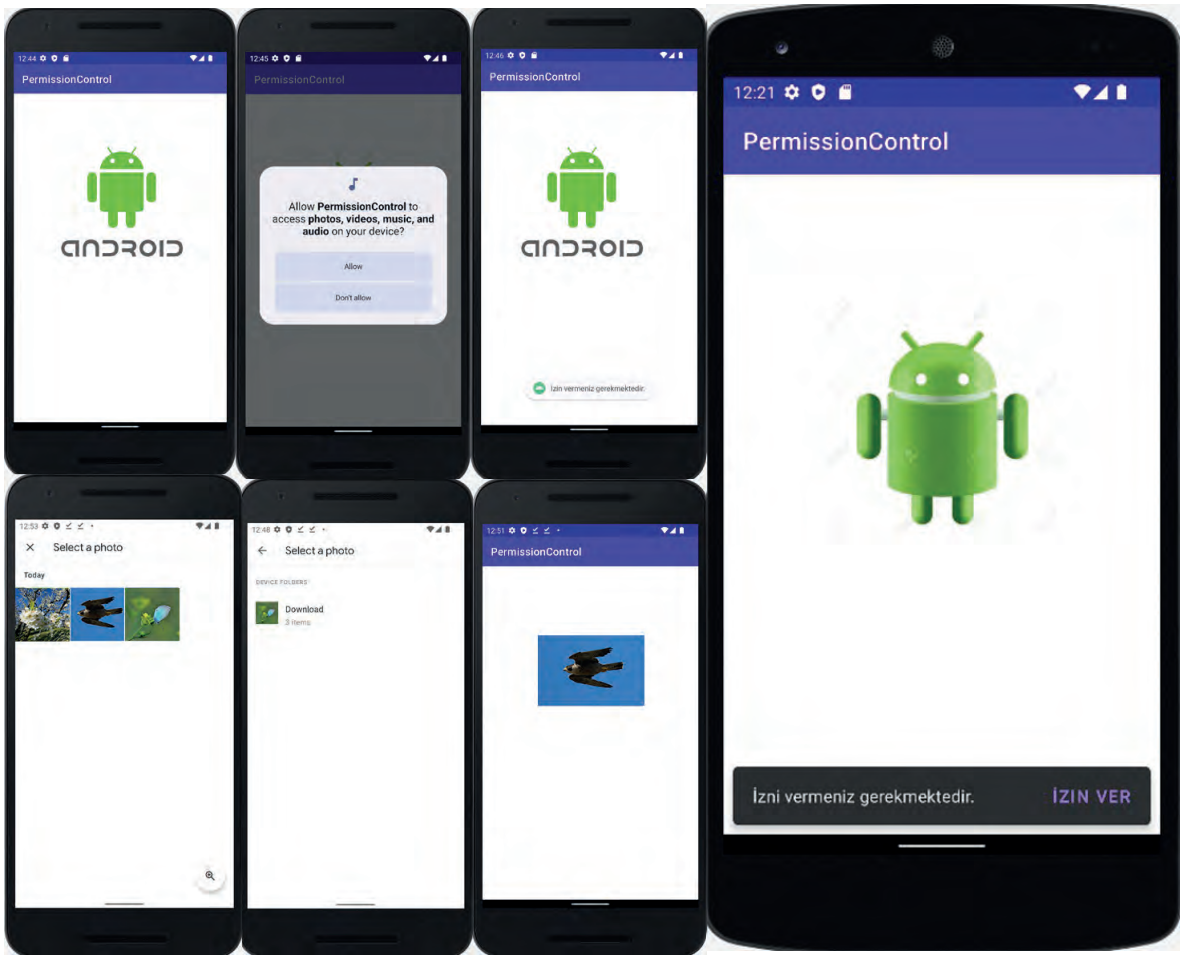
2. Adım: Karar yapısının sonucu true olursa yapılacak işlemlerin olduğu blokun içine, Snackbar gösterilmesi gereken bölüme, Snackbar ögesini oluşturunuz. Oluşturma aşamasında parametre olarak Görsel 6.62'de yer alan parametreleri gönderiniz. Son bölümde yer alan **".setAction"** metoduna gönderilen parametreler için de Görsel 6.70'te olduğu gibi **"View."** yazıldığında öneri olarak gelen **"onClickListener"** üzerindeyken Enter tuşuna basıldığında yapı override edilir. Override işlemi sonrası .show() metodunu ekleyiniz. Eklenen karar yapısının else kısmını yazarak, Snackbar gerekmeden istenen iznin yazılacağı bölümü de oluşturunuz.

```
if(ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.permission.READ_
EXTERNAL_STORAGE)){
    //Snackbar gösterilmesi gerekirse yazılacak kodlar
    Snackbar.make(view,"İzni vermeniz gerekmektedir.",Snackbar.LENGTH_INDEFINITE).setA-
ction("İzin ver", new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //Snackbar gösterilerek izin istenecektir.
        }
    }).show();
}else{
    //Snackbar gösterilmeden izin istenecek kodlar
}
```

3. Adım: onClick içinde de else içinde de aynı şekilde izin isteyiniz. "`izinlerResultLauncher.launch(Manifest.permission.READ_EXTERNAL_STORAGE);`" iznini bu bölümlere ekleyiniz. İzin verilmeyip tekrar tıklandığında Görsel 6.71'de olduğu gibi bir Snackbar ile tıklanıncaya kadar gösterir. İzin ver buttonuna tıklanırsa yeniden izin sorulur.

```
public void galeriyeGit(View view){
    if(ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTER-
NAL_STORAGE)== PackageManager.PERMISSION_DENIED){
        if(ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.
permission.READ_EXTERNAL_STORAGE)){
            //Snackbar gösterilmesi gerekirse yazılacak kodlar
            Snackbar.make(view,"İzni vermeniz gerekmektedir.",Snackbar.LENGTH_
INDEFINITE).setAction("İzin ver", new View.OnClickListener() {
                @Override
```

```
                public void onClick(View view) {
                    //İzin istenecektir.
                    izinlerResultLauncher.launch(Manifest.permission.READ_EX-
TERNAL_STORAGE);
                }
            }).show();
        }else{
            //Snackbar gösterilmeden izin istenecek kodlar
            izinlerResultLauncher.launch(Manifest.permission.READ_EXTERNAL_STO-
RAGE);
        }
    }else{
        //İzin var, galeriye gidilecektir.
        Intent galeri =new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.
EXTERNAL_CONTENT_URI);
        galeryResultLauncher.launch(galeri);
    }
}
```



Görsel 6.71: Snackbar uygulama ekranı

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) build.gradle (Module) projeye ait temel bilgilerin tutulduğu dosyadır.

2. ( ) AndroidManifest dosyasının uzantısı xml'dir.

3. ( ) Projeye ait görseller res altında yer alan drawable klasörüne atılır.

4. ( ) Mobil uygulama geliştirme platformunda sadece ConstraintLayout ile tasarım yapılır.

5. ( ) View Binding yöntemini kullanabilmek için AndroidManifest içine ekleme yapılması gerekir.

6. ( ) Code tasarım ekranı hem kod yazılan hem de aynı anda dizayn ekranının izlenebildiği tasarım ekranıdır.

7. ( ) onDestroy() metodu, Activity arka plana alındığında çalışan yaşam döngüsüdür.

8. ( ) Attributes penceresinden TextView içine yazılacak bir yazının rengi değiştirilir.

9. ( ) FrameLayout içine dikey olarak birden fazla öge eklenir.

10. ( ) API-23 ve altı sürümlerde kullanıcıdan bazı erişimler için uygulama içinde ayrıca izin istenir.

B) Aşağıdaki cümlelerde boş bırakılan yerleri kutularda verilen ifadelerle tamamlayınız.

uses-permission	ViewBinding	wrap_content
Dangerous	Intent	return
	Serializable	

11. Dönüş tipi olan metotlarda değer döndürmeye yarayan koda ..... adı verilir.

12. Mobil uygulamada izin gerektiren işlemler için Manifest dosyası içine ..... kodu yazılır.

13. Mobil uygulama kullanılırken ....., kullanıcı tarafından izin verilmesi gereken izin türüdür.

14. Farklı türdeki veriler, aktiviteler arasında paket hâlinde ..... yöntemi ile taşınabilir.

15. Kullanıcı arayüzlerini "findById()" şeklinde tek tek tanımlamadan topluca bir sınıfta toplayan yapıya ..... adı verilir.

16. Activityler arası geçiş, ... sınıfı kullanılarak yapılır.

17. ConstraintLayout içinde sınırları belirlerken ögenin sınırının tüm ConstraintLayouta yayılmasını sağlayan koda ..... adı verilir.

C) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

18. Aşağıdakilerden hangisi Activity yaşam döngülerinden biri değildir?

A) onDestroy B) onCreate() C) onResume()
D) onClick() E) onPause()

19. Aşağıdakilerden hangisi izin türleri arasında bulunmaz?

A) Normal B) Dangerous C) SignatureOrSystem
D) Signature E) Viewbinding

20. Aşağıdakilerden hangisi aktiviteler arasında veri taşıma yöntemlerinden biridir?

A) Seralizable B) viewBinding C) onClick()
D) uses-permission E) Gradle (module)

21. Aşağıdaki kodlardan hangisi layout_height içine yazılacak uygunlukta değildir?

A) wrap_content B) match_layout C) match_parent
D) 400 dp E) 0d

D) Aşağıdaki sorularda verilen boşlukları uygun şekilde doldurunuz.

22. Code ekranından bir TextView oluşturmak için yazılması gereken kodları en, boy ve id değerlerini de girerek aşağıya yazınız.

23. Mobil uygulamada internet kullanabilmek için AndroidManifest dosyasına eklenmesi gereken kodu aşağıya yazınız.

24. OgrenciBilgileri isimli Java sınıfından OgrenciDetay isimli java sınıfına öğrencinin adı, soyadı, numarası bilgilerini aktaran ve OgrenciDetay sınıfında da bu bilgileri yakalayan kodları intent yöntemi ile hazırlayınız.

7. ÖĞRENME BİRİMİ

VERİ TABANI İŞLEMLERİ

KONULAR

7.1. sharedPreferences KULLANIMI

7.2. YEREL VERİ TABANIYLA ÇALIŞMAK

7.3. UZAK VERİ TABANIYLA ÇALIŞMAK

NELER ÖĞRENECEKSİNİZ?

- sharedPreferences ile basit veri kaydetme işlemleri
- Yerel veri tabanı oluşturma
- Yerel veri tabanına veri kaydetme
- Yerel veri tabanında verileri değiştirme
- Özel bağdaştırıcı kullanma
- Uzak veri tabanı yapılandırma
- Uzak veri tabanından veri alma ve gönderme
- Uzak veri tabanının güvenlik ayarlarını yönetme

ANAHTAR KELİMELER

- Adapter
- Authetication
- Firebase
- Firestore
- HashMap
- NoSQL
- Recyclerview
- Rules
- sharedPreferences
- SQL
- SqLite

HAZIRLIK ÇALIŞMALARI

1. Uygulama geliştirilirken neden veri kaydetme ihtiyacı duyulur?
2. Günlük hayatta veri tabanları nerelerde kullanılır?
3. Uzak veri tabanlarının günlük hayata etkileri nelerdir? Uzak veri tabanları olmasaydı cep telefonları arasında veri paylaşımı nasıl yapılabilirdi? Düşüncelerinizi arkadaşlarınızla paylaşınız.
4. NoSQL veri tabanlarını araştırınız. NoSQL veri tabanlarının ilişkisel veri tabanlarına göre farklılıklarını tartışınız.

7.1. sharedPreferences KULLANIMI

Mobil uygulama geliştirme ortamında her zaman basit verileri kaydetmek gerekebilir. Örneğin kullanıcıdan alınan çeşitli bilgilerin yerel cihazda kaydedilip uygulama tekrar açıldığında kullanıcıya gösterilmesi gerekebilir. Bu tür basit veri kayıt işlemleri için **sharedPreferences** nesnesi kullanılabilir. Daha gelişmiş veri tabanları da kullanılabilir ancak basit bir veriyi kaydetmek için karmaşık bir veri tabanı kullanmaya gerek yoktur.

sharedPreferences ile **<ANAHTAR>=<DEĞER>** türünde kayıtlar yapılır. Bu durum, bir ayar paneli veya geçici olarak kaydedilmesi gereken veriler için oldukça uygundur.

Uygulama geliştirme ortamında sharedPrefrences iki şekilde kullanılır. Birden fazla dosyaya ihtiyaç varsa **getSharedPreferences()** ile çağrılır, sadece basit bir veri saklanacaksa **getPreferences()** kullanılmalıdır. Bir ayar paneli varsa ve birden fazla sekmeden oluşuyorsa her bir paneli farklı bir dosyaya kaydetmek için getSharedPreferences ile veri saklama işlemi gerçekleştirilebilir.

`sharedPreferences sharedPref=this.getSharedPreferences("menu1.ds",Context.MODE_PRIVATE);` şeklinde kullanıldığında ilk parametre olarak mutlaka bir dosya ismi girilmelidir. Bu nesne ile yapılan kayıtların hepsi belirtilen dosyaya kaydedilir.

`sharedPreferences sharedPreferences=this.getPreferences(Context.MODE_PRIVATE);` şeklinde kullanılırsa tüm kayıtlar ortak olarak bir dosyaya kaydedilir. En çok bu yöntem tercih edilir. İki şekilde oluşturulan nesnelerin türü aynı, sadece çağrılma şekilleri farklıdır. SharedPreferences nesnesi oluşturulduktan sonra bir tane de **Editor** nesnesi oluşturulmalıdır. Editor nesnesi, verilerin alınarak kaydedilmesini sağlar.

`sharedPreferences.Editor editor=sharedPreferences.edit();`
Editor nesnesi oluşturulduktan sonra nesnenin metotları ile veriler hazırlanır.

```
String veri="Herhangi bir veri";
editor.putString("anahtar",veri);
editor.apply();
```

Editor nesnesinin putString metodu ile önce bir anahtar isim sonra da saklanmak istenen veri yazılır. Bu anahtar isim, veriyi tekrar çağırmak için kullanılır. Anahtar isimler belirlenirken aynı olmamasına dikkat edilmelidir.

UYARI: Aynı anahtar ismi kullanılırsa önceki veri kaybolur. Yeni veri, eski verinin üstüne yazılır (Tablo 7.1).

Tablo 7.1: Editor Nesnesi Veri Ekleme Metotları

putBoolean	Editor nesnesine Boolean türünde veri ayarlar.
putInt	Editor nesnesine Int türünde veri ayarlar.
putLong	Editor nesnesine Long türünde veri ayarlar.
putString	Editor nesnesine String türünde veri ayarlar.
putFloat	Editor nesnesine Float türünde veri ayarlar.
clear	Tüm verileri siler.
remove	remove("anahtar") şeklinde belirtilen veriyi siler.
apply	Ayarlanan tüm verileri dosyada saklar.

Editör nesnesine veri ayarlanırken veri tipine uygun olan metot seçilerek ayarlama yapılmalıdır. Örneğin String türünde bir veri, putInt veya putLong kullanılarak ayarlanamaz. String veri için en uygunu putString kullanımıdır. String veriler dosyadan şu kod ile alınır:

```
String gelenveri=sharedPreferences.getString("anahtar","bos");
```

getString metodunun iki parametresi vardır. Bunlardan birincisi verilerin kaydedildiği anahtar kelime, ikincisi de bu veri yoksa varsayılan olarak değişkene verilecek değerdir. Varsayılan değer kontrol edilerek verinin kayıtlı olup olmadığı anlaşılır. gelenveri değişkeni "bos" değerine sahipse bu anahtar ile saklanan bir veri yoktur (Tablo 7.2).

Tablo 7.2: Editor Nesnesi Veri Okuma Metotları

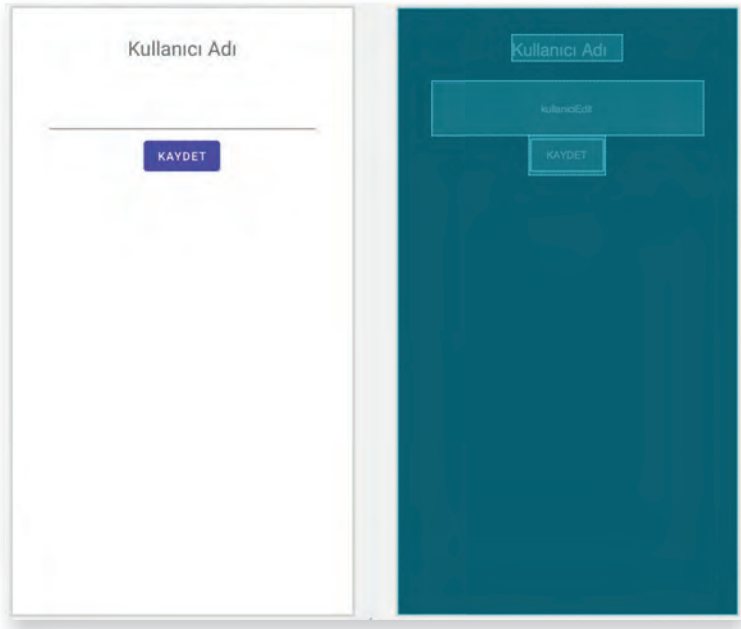
getBoolean	Saklanan verilerden Boolean verisi alır.
getInt	Saklanan verilerden Ant verisi alır.
getLong	Saklanan verilerden Long verisi alır.
getString	Saklanan verilerden String verisi alır.
getFloat	Saklanan verilerden Float verisi alır.
getAll	Tüm saklanan verileri alır.
contains	Saklanan verilerde belirtilen anahtarın olup olmadığını kontrol eder.

Veri çağrılırken de saklanan verinin türüne göre bir metot kullanılmalıdır. String veri, getInt kullanılarak tekrar çağrılamaz.

1. UYGULAMA: İşlem adımlarına göre Kullanıcı Adı yazıp Kaydet buttonuna tıkladıktan sonra veriyi dosyaya kaydetme işlemini yapınız. Daha sonra uygulamayı tekrar başlatıp, bu veriyi dosyadan okuyarak EditText kutusuna yazınız.

1. Adım: Mobil uygulama geliştirme ekranında Empty Activity seçerek yeni bir proje oluşturunuz.

2. Adım: MainActivity ekranına bir tane EditText, bir tane TextView ve bir tane Button ekleyerek Görsel 7.1'deki gibi yerleştiriniz. EditText için **kullaniciEdit** id bilgisini giriniz. Gerekli bilgiler değiştirildikten sonra **Infer Constraints** buttonuna tıklayınız ve otomatik olarak Constraints ayarlarının yapılmasını sağlayınız. Buttonun **Text** özelliğini "KAYDET", **onClick** özelliğini "btnKaydetClick" olarak ayarlayınız.



Görsel 7.1: Uygulama tasarım ekranı

3. Adım: MainActivity.java dosyasına gelip bir EditText değişkeni oluşturunuz.

4. Adım: onCreate metodu içinde EditText nesnesini **findViewById** metodu ile bağlayınız. Daha sonra bir sharedPreferences nesnesi oluşturunuz. sharedPrefrences ile kaydettiğiniz veriyi okuyup herhangi bir veri yoksa EditText nesnesindeki veriyi yazınız.

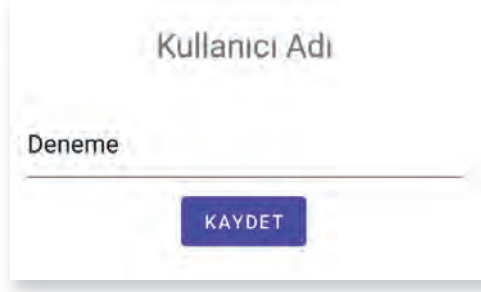
5. Adım: Button nesnesi için public void btnKaydetClick(View view) metodunu yazarak kaydetme işlemlerine başlayınız. Bir String nesnesi oluşturarak EditText nesnesindeki veriyi alınız. Daha sonra bir sharedPrefrences nesnesi oluşturarak veri yazmaya hazırlık yapınız. sharedPrefrences ile bir Editör nesnesi oluşturarak EditTextten gelen veriyi yazınız.

```
EditText kullaniciEdit;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    kullaniciEdit=findViewById(R.id.kullaniciEdit);
    SharedPreferences sharedPreferences=this.getPreferences(Context.MODE_PRIVATE);
    String gelenveri=sharedPreferences.getString("kullanici","");
    if(!gelenveri.isEmpty()){
        kullaniciEdit.setText(gelenveri);
    }
}
public void btnKaydetClick(View view){
    String veri;
    veri = kullaniciEdit.getText().toString();
    SharedPreferences sharedPreferences=this.getPreferences(Context.MODE_PRIVATE);
```

```
    SharedPreferences.Editor editor=sharedPreferences.edit();
    editor.putString("kullanici",veri);
    editor.apply();
}
```

6. Adım: Uygulamada öncelikle "kullanıcı" anahtar kelimesi ile kayıtlı herhangi bir kayıt olup olmadığına bakınız. Gelen veri varsa veriyi EditText nesnesine yazdırınız. EditText nesnesinin boş olup olmadığını String'lerin empty() metodunu çağırarak bulunuz.

7. Adım: Button nesnesine bir metot yazarak kaydetme işlemlerini yapınız. Uygulamayı emülatörde açıp, herhangi bir veriyi yazarak Kaydet buttonu ile kaydediniz. Uygulama kapatılıp tekrar açıldığında EditText içinde saklanan veri Görsel 7.2'deki gibi görünür.



Görsel 7.2: sharedPreferences ile gelen veri

2. UYGULAMA: İşlem adımlarına göre uygulamada ekran tema değişiklikleri yaparak bunu kayıt altında tutunuz.

1. Adım: Mobil uygulama geliştirme ekranında Empty Activity seçerek yeni bir proje oluşturunuz.

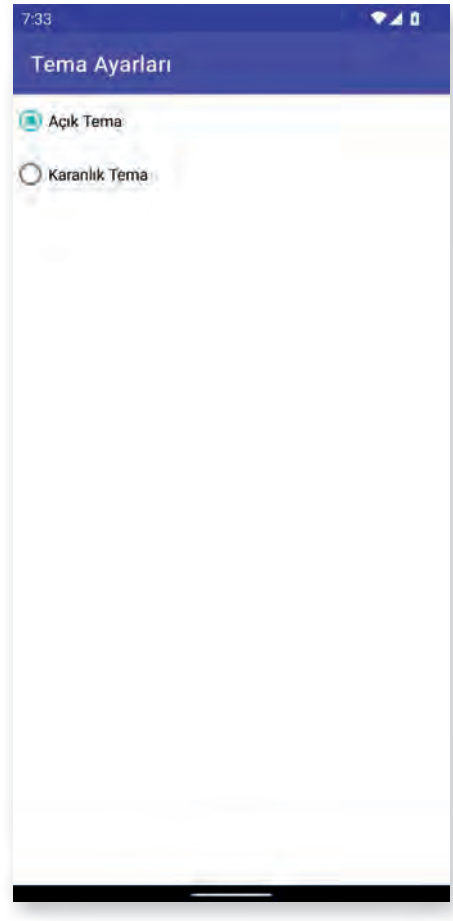
2. Adım: Ekran tasarım moduna geçmek için app>res>layout altında bulunan **main_activity.xml** dosyasını açınız. Görsel 7.3'teki gibi ana ekranınızı hazırlayınız. Bir radioGrup oluşturarak içine iki adet RadioButton yerleştiriniz. Birinci RadioButton için Text özelliğini "Açık Tema" ve id bilgisini radioAcik olarak ayarlayınız. İkinci RadioButton için Text özelliğini "Koyu Tema" ve id bilgisini radioKaranlik olarak ayarlayınız.

3. Adım: MainActivity.java dosyasını açarak sharedPrefrences ve sharedPreferences.Editor nesnelerini global olarak tanımlayınız.

4. Adım: onCreate metodu içinde iki tane RadioButton nesnesi tanımlayınız. Bu nesneleri findViewById metodu ile bağlayınız.

5. Adım: sharedPreferences ve Editor nesnelerini oluşturunuz.

6. Adım: Editor nesnesi ile "tema" isimli bir kayıt olup olmadığına bakınız. Herhangi bir değer yoksa varsayılan olarak AppcompatDelegate.MODE_NIGHT_NO değerini ayarlayınız.



Görsel 7.3: Açık tema

7. Adım: Gelen veri MODE_NIGHT_NO ise radioAcik nesnesini seçili hâle getiriniz. Gelen veri MODE_NIGHT_NO değilse radioKaranlik nesnesini şu şekilde aktifleştiriniz:

```
if(veri==AppCompatDelegate.MODE_NIGHT_NO)
    radioacik.setChecked(true);
else
    radioKaranlik.setChecked(true);
```

8. Adım: RadioButtonların davranışlarını kontrol etmek için onRadioClicked isimli metodu şu şekilde oluşturunuz:

```
public void onRadioClicked(View view){
    boolean checked=((RadioButton)view).isChecked();
    switch (view.getId()){
        case R.id.radioAcik:
            if(checked){
                AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_
NIGHT_NO);
                editor.putInt("tema",AppCompatDelegate.MODE_NIGHT_NO);
                editor.apply();
            }
            break;
        case R.id.radioKaranlik:
            if(checked){
                AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_
NIGHT_YES);
                editor.putInt("tema",AppCompatDelegate.MODE_NIGHT_YES);
                editor.apply();
            }
            break;
    }
}
```

9. Adım: sharedPreferences nesnesini onDestroy metodunda yok ediniz.

```
protected void onDestroy() {
    sharedPreferences=null;
    super.onDestroy();
}
```

10. Adım: Uygulamayı çalıştırınız. Koyu temayı seçtiğinizde uygulamanın Görsel 7.4'teki gibi görünüp görünmediğini kontrol ediniz.

Görsel 7.4: Koyu tema

SIRA SİZDE: Yeni bir Empty Activity ile bir proje oluşturunuz. Projeye iki EditText ve bir Button ekleyerek adınızı ve soyadınızı bir sharedPreferences nesnesi ile kaydediniz. Uygulama her açılıp kapandığında verileri okuyarak EditText nesnelerine yazınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Ekran tasarımını yaptı.		
3. Görünüm nesnelerine id bilgilerini verdi.		
4. sharedPreferences nesnesi tanımladı.		
5. sharedPreferences nesnesinden veri gelip gelmediğini kontrol etti.		
6. sharedPreferences ile gelen veri varsa EditText nesnelerine yazdı.		
7. sharedPreferences nesnesi ile veri kayıt işlemini yaptı.		

7.2. YEREL VERİ TABANIYLA ÇALIŞMAK

Her ne kadar bazı veriler sharedPrefrences kullanılarak kaydedilebilse de bazı durumlarda sharedPreferences tek başına yeterli olmaz. Daha karmaşık verilerin kaydedilmesi gerekirse daha fazlası kullanmalıdır. Bunlardan bir tanesi, yerel bir veri tabanı kullanılmasıdır. Mobil işletim sisteminde yerel veri tabanı olarak SQLite kullanılır. SQLite kurmak için herhangi bir işlem yapılmasına gerek yoktur. SQLite, işletim sisteminde kurulu olarak gelir.

Yerel veri tabanları birçok veriyi kaydetmek için oldukça elverişlidir ancak yerel veri tabanlarının da dezavantajları vardır. Örneğin uygulama yanlışlıkla kaldırılırsa tüm veriler kaybolur veya yetkisiz bir erişimle başka bir uygulama verilerin silinmesine neden olabilir. Yerel veri tabanları, mobil uygulama geliştiriciler tarafından daha çok bir tür yedekleme işlemi yapmak için kullanılır. Çok önemli veriler bir sunucu veya uzak veri tabanında tutulmalıdır. Bunun da sıkıntıları vardır. Örneğin internet bağlantısı yoksa uzaktaki sunucudan veri alınamaz. Bu tür durumlarda yerel veri tabanları imdada yetişir. Sunucudaki veriler yedeklenerek yerel veri tabanına kaydedilir. Kullanıcının interneti olmasa bile uygulama yine de kullanılır hâlde tutulur.

Mobil işletim sisteminde varsayılan olarak SQLite veri tabanı kullanılır. SQLite veri tabanı tamamen SQL sorgu komutları ile işlem yapar. Mobil uygulama geliştirme ortamı, bu komutları kendi derleyicisinden geçirmez. Veri tabanı için yazılan sorgulama komutları düşük seviye bir sisteme aktarılır. Burada çalıştırılır ve sonucu döndürülür. Bundan dolayı sorgulama komutlarında herhangi bir hata yapılsa bile derleyici hata vermez. Bu durum, yerel veri tabanının en dezavantajlı hâlidir. Sorgulama komutları yazılırken çok dikkat edilmelidir. Uygulama çalıştırıldıktan sonra herhangi bir sonuç alınmazsa sorgulama komutlarından birinde hata yapılmıştır.

7.2.1. Sorgulama Komutları

Veri tabanı sorgu komutları OLUŞTURMA, OKUMA, GÜNCELLEME, SİLME (CREATE, READ, UPDATE, DELETE) olmak üzere dört temel unsur üzerine çalışır (Tablo 7.3).

Tablo 7.3: Sorgu Komutları Türleri

Unsur	İşlem	SQL Karşılığı
CREATE	Veri ekleme	INSERT
READ	Veri okuma	SELECT
UPDATE	Veri güncelleme	UPDATE
DELETE	Veri silme	DELETE

Yerel veri tabanları tüm verileri bir tabloya kaydeder. Veri kayıt işleminden önce tablo oluşturulması gereklidir (Tablo 7.4).

Tablo 7.4: Ürünler Tablosu

id	urunadi	fiyat	adet
1	Televizyon	12500.00	12
2	Monitör	3500.00	25
3	Bilgisayar	10000.00	16
4	Klavye	125.50	134

Okulda veya sınıfta bulunan öğrenciler bir veri tabanına kaydedilmek istenirse Tablo 7.4'te görüldüğü gibi bir tablo oluşturulmalıdır. Tablonun üst tarafındaki alanlar, veri sütunlarıdır. Tablo oluşturulurken verilerin türleri önemlidir. Örneğin bir sütunun veri türü INTEGER olarak ayarlanmışsa buraya kişinin ad ve soyadı kaydedilemez. Yerel veri tabanında kullanılabilecek veri türleri Tablo 7.5'te verilmiştir.

Tablo 7.5: Yerel Veri Tabanı Veri Türleri

Veri Türü	Açıklama	Geliştirme Ortamı Karşılığı
INTEGER	Tam sayılar ve Boolean türünde veri kaydetmek için kullanılır.	int, long, byte …
REAL	Ondalık sayı türünden verileri kaydetmek için kullanılır.	float, double
TEXT	Metin türünden (String) verileri kaydetmek için kullanılır.	String
BLOB	Resim, dosya gibi verileri kaydetmek için kullanılır.	byte[]

Verilerin kaydedileceği tablolar oluşturulurken birincil anahtar (Primary Key) belirlemeye dikkat edilmelidir. Bir tablodaki boş olmayan herhangi bir sütun, birincil anahtar olabilir. Birincil anahtar olarak belirlenecek alandaki kayıtların hepsi birbirinden farklı olmalıdır. Örneğin ad soyad alanı birincil anahtar yapılırsa aynı isimli birden fazla öğrenci ile karşılaşılabilir. Bu da çok büyük problemlere neden olabilir. Ülkemizde yaşayan her vatandaşın bir T.C. Kimlik Numarası vardır. Aynı isimli birden fazla vatandaş olsa bile T.C. Kimlik Numarası ile vatandaşlar birbirinden ayırt edilebilir. Veri tabanlarındaki birincil anahtar da bu amaçla kullanılır.

7.2.1.1. Yerel Veri Tabanında Tablo Oluşturmak

Sorgu dilinde bir tablo oluşturmak için CREATE TABLE komutu kullanılır. Sorgu komutlarını denemek için on-line birçok internet sitesi bulunur. Bunlardan herhangi biri ile sorgular test edilebilir. CREATE TABLE komutunun kullanımı şu şekildedir:

```
CREATE TABLE [IF NOT EXISTS] <tablo ismi> ( <sütun adı> <veri türü> , <diğer sütunlar>)
```

NOT: IF NOT EXISTS, tablo oluşturulmamış ise tabloyu oluştur anlamına gelir. Kullanma zorunluluğu yoktur.

ÖRNEK

```
CREATE TABLE IF NOT EXISTS urunler(id INTEGER PRIMARY KEY,
                                   urunadi TEXT,
                                   fiyat DOUBLE,
                                   adet INTEGER)
```

NOT: Sorgu komutunda Türkçe karakter kullanmamaya dikkat edilir. Veri olarak Türkçe karakter kullanılabilir ancak tablo adı ve sütun isimlerinde Türkçe karakter kullanılmamalıdır.

NOT: Sorgu komutu yazılırken temel komutlar büyük harflerle yazılmalıdır.

7.2.1.2. Yerel Veri Tabanına Veri Eklemek

Yerel veri tabanına veri eklemek için INSERT ifadesi kullanılır. INSERT komutunda önce tablo adı ve veri eklenecek alanlar belirtilir. Daha sonra VALUES anahtar kelimesi ile yazılması gereken veriler eklenir. Örnek bir veri ekleme komutu şu şekilde yazılır:

```
INSERT INTO <tablo ismi> ( <sütun adı>, <diğer sütunlar>…) values(<değerler>)
```

ÖRNEK

INSERT INTO urunler(urunadi,fiyat,adet) VALUES(“Televizyon”,12500.00,12)

7.2.1.3. Yerel Veri Tabanında Veri Sorgulamak

SQL komutları içinde en kapsamlı olanı veri sorgulama komutlarıdır. Sadece SELECT komutu ile yapılmasına rağmen çok farklı şekillerde sorgular yazılabilir. SELECT komutunun kullanımı şu şekildedir:

```
SELECT <sütun adları> FROM <tablo adı> WHERE <kısıtlamalar> ORDER BY <sıralama türü>
```

SELECT komutunun en çok tercih edilen kullanımları Tablo 7.6’da verilmiştir.

Tablo 7.6: SELECT Komutunun En Sık Kullanım Şekilleri

Sorgu	Sorgunun Amacı
SELECT * FROM urunler	Tablodaki tüm kayıtları listeler.
SELECT urunadi FROM urunler	Tüm ürün isimlerini listeler.
SELECT urunadi FROM urunler WHERE adet=12	Adet bilgisi 12 olan ürünleri listeler.
SELECT urunadi FROM urunler WHERE adet>=120	Adet bilgisi 120’den büyük kayıtları listeler.
SELECT adet FROM urunler WHERE id>=2 AND adet>120	id bilgisi 2 veya 2’den büyük, numarası 120’den büyük kayıtları listeler.
SELECT urunadi FROM urunler WHERE urunadi LIKE “B%”	Ürünadı “B” harfi ile başlayan kayıtları listeler.
SELECT urunadi FROM urunler WHERE urunadi LIKE “%D%”	ÜrünAdı alanında içinde “D” harfi olan kayıtları listeler.
SELECT urunadi FROM urunler WHERE urunadi LIKE “%r”	Sonu “r” ile biten kayıtları listeler.
SELECT urunadi FROM urunler WHERE adet BETWEEN 10 and 50	Adet bilgisi 10 ile 50 arasında olan kayıtları listeler.
SELECT urunadi FROM urunler ORDER BY urunadi ASC	Tüm kayıtları ürün ismine göre küçükten büyüğe doğru sıralayarak listeler.
SELECT urunadi FROM urunler ORDER BY urunadi DESC	Tüm kayıtları ürün ismine göre büyükten küçüğe doğru sıralayarak listeler.

7.2.3. Mobil Uygulama Geliştirme Ortamında Yerel Veri Tabanı Kullanmak

Mobil uygulama geliştirme ortamı, SQLite yerel veri tabanını destekler. Mobil uygulama geliştirme ortamı, yerel veri tabanı kullanmak için **SQLiteDatabase** nesnesine sahiptir. Bu nesne ile tüm veri tabanı işlemleri yapılabilir.

Mobil uygulama geliştirme ortamında veri tabanı örneği şu şekilde oluşturabilir:

```
SQLiteDatabase database;
```

Bu kod sadece bir SQLiteDatabase nesnesi verir. Yerel veri tabanını kullanabilmek için gerekli düzenlemeler yapılmalıdır. Nesne oluşturulduktan sonra ayarlamalar şu şekilde yapılır:

```
database = this.openOrCreateDatabase(<Veri Tabanı Adı>, MODE_PRIVATE, null);
```

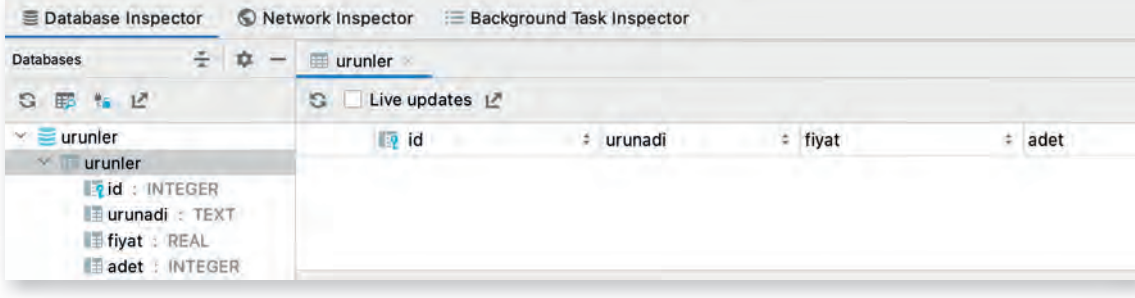
Veri tabanı nesnesi **Context** aracılığı ile oluşturulur. Context nesnesinin openOrCreateDatabase metodu ile gerekli parametreler verilerek nesne elde edilir.

- İlk parametre ile yerel veri tabanında kullanılacak dosya ismi belirtilir. Burada verilen isimle başka bir activity içinde yerel veri tabanına ulaşılabilir.
- İkinci parametre ile veri tabanının başka uygulamalara açılıp açılmayacağı belirlenir. Genellikle her uygulamanın özel bir veri tabanı olduğu için bu parametre **MODE_PRIVATE** olarak ayarlanır.
- Üçüncü parametre gelen bir imleç verisi varsa bunu almak için gerekli ayarlamalar yapılır. Böyle bir durum yoksa bu değer **null** olarak ayarlanır.

ÖRNEK

```
String TABLO="CREATE TABLE IF NOT EXISTS urunler(id INTEGER PRIMARY KEY,";
TABLO+="urunadi VARCHAR,";
TABLO+="fiyat DOUBLE,";
TABLO+="adet INTEGER)";
database.execSQL(TABLO);
```

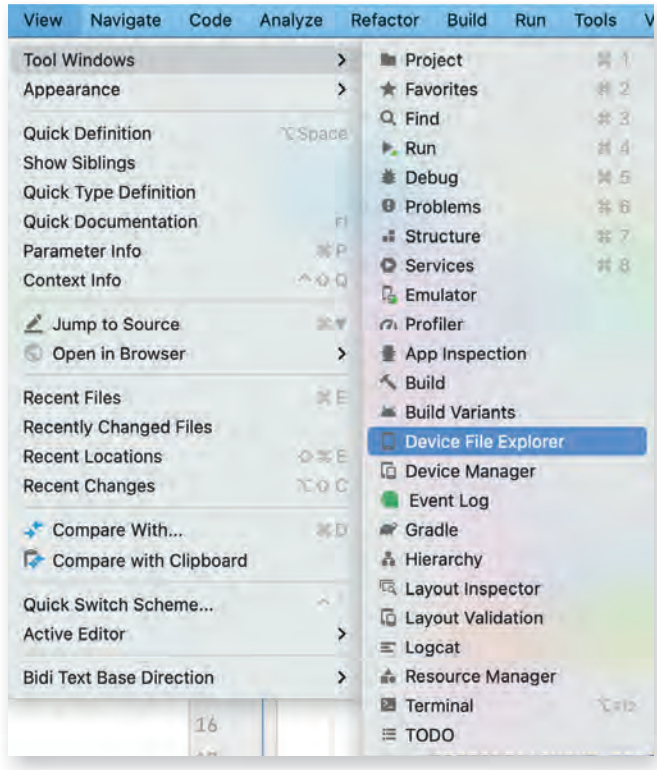
Kodlar yazılırken kullanıcıya yardımcı olması açısından TABLO isimli bir String değişken kullanılır. TABLO değişkeni ile sorgulama komutu hazırlanır. Sorgulama komutu hazırlanırken çok dikkat edilmelidir. Uygulama çalıştırıldıktan sonra sorgulama kodlarında herhangi bir hata olsa bile uyarı alınmaz. Tablonun oluşup oluşmadığını anlamak için **App Inspection** penceresi açılır (Görsel 7.5).



Görsel 7.5: App Inspection penceresi

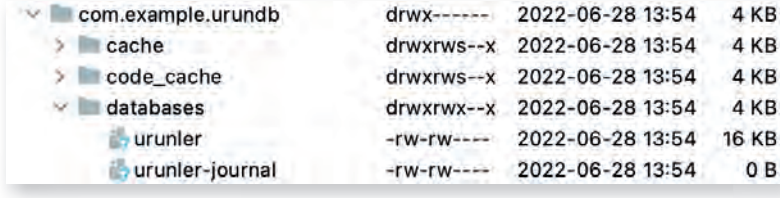
App Inspection penceresindeki **Database Inspector** bölümünde tablo görünürse kodlar sorunsuz çalışmıştır.

Tablo oluşturulduktan sonra dosyanın nerede bulunduğuna bakılır. Bunun için View>Tool Windows>Device File Explorer seçeneği seçilir (Görsel 7.6).



Görsel 7.6: Device File Explorer menüsü

Açılan pencereden **Data>Data** klasörüne gidilir. Data klasörü içinde Görsel 7.7'deki gibi uygulama için verilen paket ismi bulunur. Paket ismi ile başlayan klasörün içindeki **databases** klasöründe yerel veri tabanı dosyaları vardır. Herhangi bir sorun çıktığında buradaki dosya silinebilir veya uygulama tamamen silinip mobil uygulama geliştirme ortamında yeniden çalıştırılabilir.



Görsel 7.7: Device File Explorer ürünler tablosu dosyası

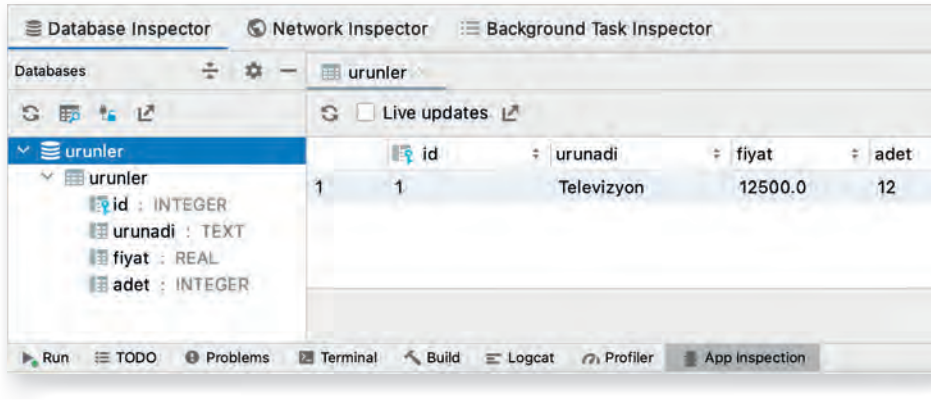
7.2.4. Mobil Uygulama Geliştirme Ortamında Kayıt Eklemek

Yerel veri tabanına bir kayıt eklemek için bir sorgu komutu oluşturulup gerekli parametreler bir SQLStatement nesnesi ile sorgu komutuna eklenmelidir. Daha sonra SQLStatement nesnesinin execute metodu çağrılır. Kayıt eklendikten sonra **id** numarası otomatik verilir. Mobil uygulama geliştirme ortamında herhangi bir kayıt şu şekilde eklenir:

```
String SORGU="INSERT INTO urunler(urunadi,fiyat,adet) VALUES(?,?,?)";
SQLiteStatement durumlar=database.compileStatement(SORGU);
durumlar.bindString(1,"Televizyon");
durumlar.bindString(2,12500.00);
durumlar.bindLong(3,12);
durumlar.execute();
```

SQLStatement nesnesi oluşturulmadan önce sorgu komutuna dışarıdan gelecek veriler için birer soru işareti (?) işareti kullanılır. Soru işareti (?) sayesinde SQLStatement nesnesi ile dışarıdan gelen veriler sorgu komutuna eklenir.

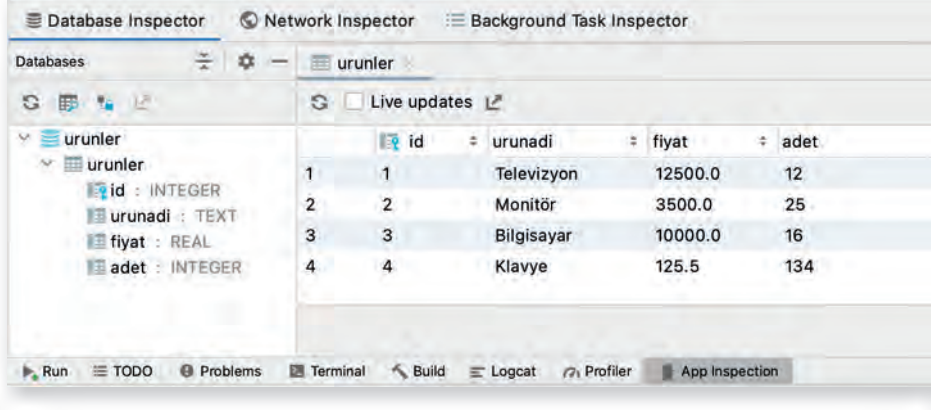
Kodlar yazılıp çalıştırıldıktan sonra ekranda herhangi bir işlem olmaz ancak App Inspection penceresi açılarak veri tabanına kayıt yapılıp yapılmadığı kontrol edilebilir (Görsel 7.8).



Görsel 7.8: App Inspection eklenen kayıt

7.2.5. Mobil Uygulama Geliştirme Ortamında Kayıt Silmek

Veri silmek için sorgu komutu hazırlanıp gerekli parametreler SQLStatement ile sorguya eklenir (Görsel 7.9).

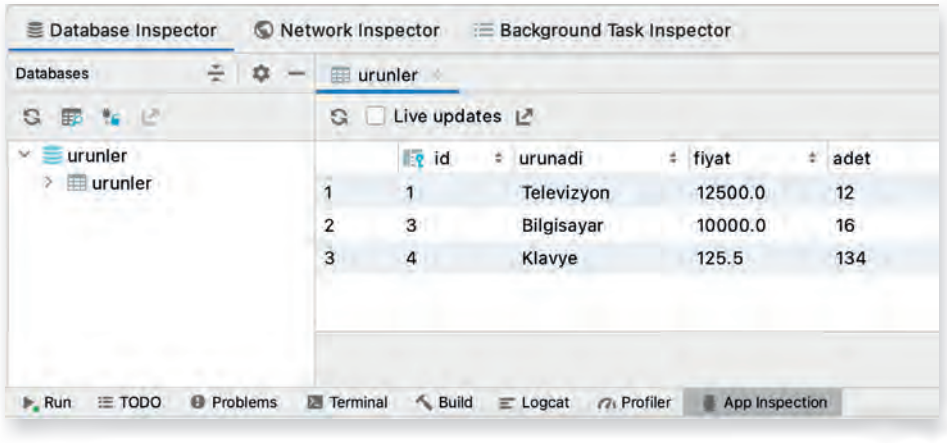


Görsel 7.9: App Inspection eklenmiş kayıtlar

Bu kayıtlardan id numarası 2 olanı silmek için şu kodlar kullanılır:

```
String SORGU="DELETE FROM urunler WHERE id=?";
SQLiteStatement durumlar=database.compileStatement(SORGU);
durumlar.bindLong(1,2);
durumlar.execute();
```

Diğer alanlarda aynı veriler olabilir ancak birincil anahtarda kesinlikle aynı veri olamaz. Kayıt silme işleminden sonra App Inspection penceresi Görsel 7.10'daki gibi olur.



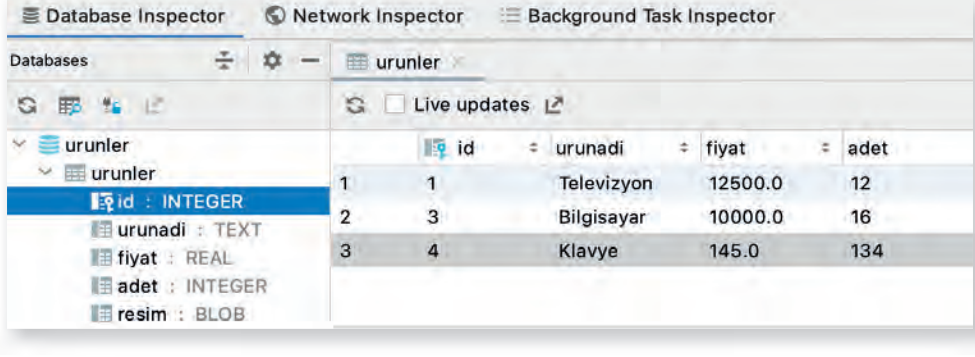
Görsel 7.10: App Inspection silinen veri

7.2.6. Mobil Uygulama Geliştirme Ortamında Kayıt Güncellemek

Güncelleme yapılması için bir sorgu komutu hazırlanıp SQLStatement nesnesi ile veriler sorguya dâhil edilir. Görsel 7.10'daki verilerden id numarası 3 olan kaydın fiyat bilgisi 145 olarak şu şekilde değiştirilir:

```
String SORGU="UPDATE ürünler SET fiyat=? WHERE id=?";
SQLiteStatement durumlar=database.compileStatement(SORGU);
durumlar.bindString(1,145);
durumlar.bindLong(2,3);
durumlar.execute();
```

Kodlar hazırlandıktan sonra uygulama çalıştırıldığında App Inspection penceresinden değişiklikler takip edilebilir (Görsel 7.11).



Görsel 7.11: App Inspection güncellenen veri

7.2.7. Mobil Uygulama Geliştirme Ortamında Tüm Kayıtları Listelemek

Kayıt listelemek için bir sorgu hazırlanıp Cursor nesnesi ile verilere ulaşılır. Cursor nesnesi, tabloda bulunan tüm kayıtlar üzerinde gezinmeyi sağlar. Cursor nesnesi bir satır veri aldıktan sonra hangi kolona ait verinin okunması istenirse bunu belirtmek yeterlidir. Tüm verileri okumak için şu şekilde bir düzenleme yapılmalıdır:

```
database=this.openOrCreateDatabase("urunler",MODE_PRIVATE,null);
Cursor cursor=database.rawQuery("SELECT urunadi,fiyat,adet FROM urunler",null);
String veriler="";
int kolonUrunadi=cursor.getColumnIndex("urunadi");
int kolonFiyat= cursor.getColumnIndex("fiyat");
int kolonAdet=cursor.getColumnIndex("adet");
while (cursor.moveToNext()){
    String urunadi=cursor.getString(kolonUrunadi);
    double fiyat=cursor.getDouble(kolonFiyat);
    int adet=cursor.getInt(kolonAdet);
    veriler+=urunadi+" "+fiyat+" "+adet+"\n";
}
cursor.close();
Toast.makeText(this, veriler, Toast.LENGTH_SHORT).show();
```

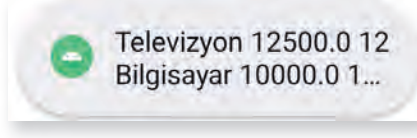
Cursor nesnesi, rawQuery metodu ile Database nesnesinden oluşturulur. Veri sorgulamak için rawQuery metodu kullanılır. rawQuery metodu, Cursor türünde bir nesne geri döndürür.

Bir sonraki adımda kolonların indeks numaraları bulunur. Cursor nesnesinin getColumnIndex metodu, kolonun indeks numarasını verir. Tablonun kolonlarının indeks numaraları en soldan itibaren 0'dan başlar (Tablo 7.7).

Tablo 7.7: Tablo Kolonlarının İndeks Numaraları

0	1	2	3
id	urunadi	fiyat	adet
1	Televizyon	12000	12

`String urunadi=cursor.getString(kolonUrunadi)` ile `String urunadi=cursor.getString(1)` aynı sonucu verir. Tüm veriler okunurken bir döngü kurulup Cursor nesnesinin son kayda kadar gitmesi sağlanır. **cursor.moveNext** metodu ile bir sonraki kayda geçilir. Kayıt bitmişse **false** değeri gönderilerek döngü bitirilir. Tüm kayıtlar okunduktan sonra Cursor nesnesi mutlaka kapatılmalıdır. Kodlarda tüm veriler bir String içine eklenerek bir Toast mesajı ile ekranda gösterilir (Görsel 7.12).



Görsel 7.12: Tüm kayıtların listelenmesi

7.2.8. Özel Adaptör Kullanmak

Adaptörler, veri kaynağındaki verileri okur ve View nesnelerine bağlar. ListView nesnesi, hangi verinin hangi View nesnesine bağlanacağını bilemez. Mobil uygulama geliştiriciler adaptör kullanarak verileri View nesnelerine bağlar. Özel adaptör kullanabilmek için öncelikle bir veri modeli oluşturmak gereklidir.

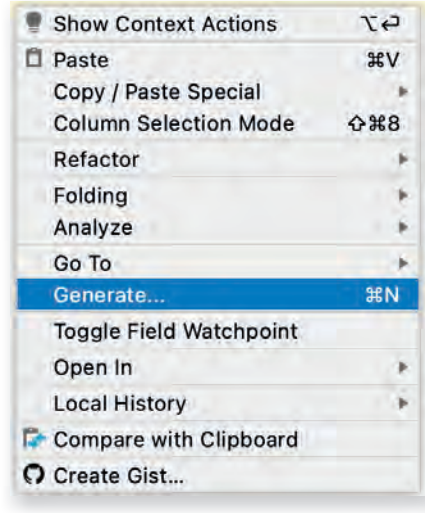
7.2.9. Model Oluşturmak

Tabloda bulunan her veri bir sınıf ile temsil edilebilir. Bu sınıfa **model** denir. Oluşturulacak model, tablodaki tüm veriler ile uygun olmalıdır. Örneğin tabloda bir sütun int türünden ise modelin bu özelliği de int türünden olmalıdır. Üçüncü uygulamada kullanılan tablo için örnek bir model şu şekilde oluşturulur:

```
public class Urun {
    private int id;
    private String urunadi;
    private double fiyat;
    private int adet;
    private Bitmap resim;
}
```

Modelin özellikleri tanımlandıktan sonra model bir java data sınıfı hâline getirilmelidir. Bu işlemler için ekstra kod yazılmasına gerek yoktur.

Kodlar üzerinde boş bir yere tıklanır ve çıkan seçeneklerden **Generate** seçeneği seçilir (Görsel 7.13).



Görsel 7.13: Sağ tıklama menüsü

Generate menüsü açıldıktan sonra **Constructor** seçeneği Görsel 7.14'teki gibi seçilip model için bir yapılandırıcı metot hazırlanır.

Constructor metoduna hangi alanlar eklenecekse Görsel 7.15'teki gibi seçilir.

```
public Urun(int id,String urunadi, double fiyat, int adet, Bitmap resim) {
    this.id=id;
    this.urunadi = urunadi;
    this.fiyat = fiyat;
    this.adet = adet;
    this.resim = resim;
}
```

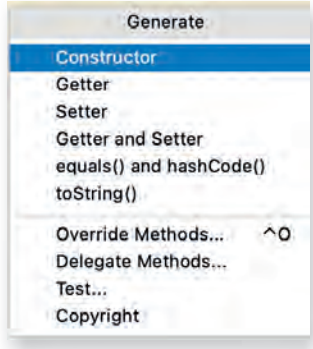
Yapılandırıcı metot kullanılarak bir ürün nesnesi şu şekilde oluşturulur:

```
Urun urun=new Urun(1,"Televizyon",12500,12);
```

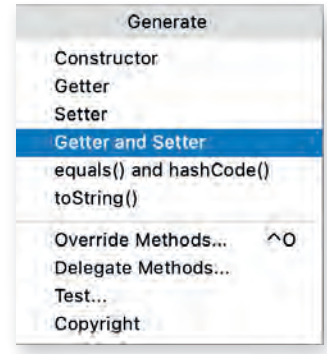
Data sınıfı için Getter ve Setter metotları oluşturulur. Görsel 7.16'daki Generated menüsünden **Getter and Setter** seçeneği seçilir.

Görsel 7.17'deki pencereden alanların tümü seçilip OK buttonuna tıklanır. Mobil uygulama geliştirme ortamı için tüm Getter ve Setter metotları otomatik olarak oluşturulur.

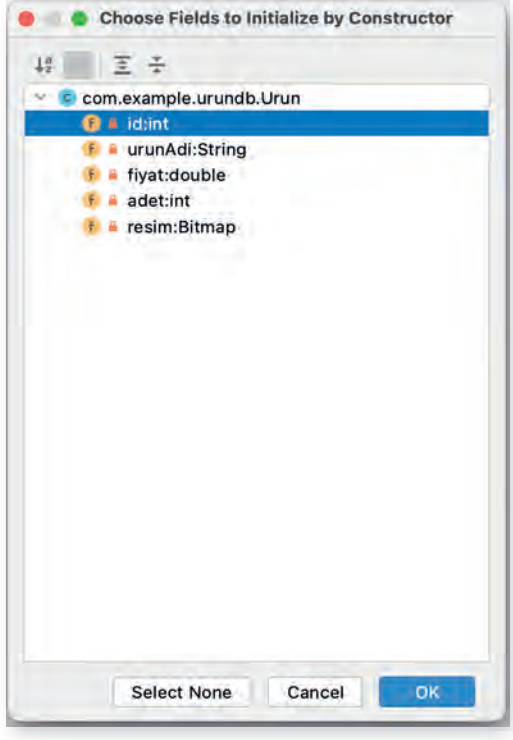
Model ile oluşturulan bir nesnenin urunadi özelliğine erişim sağlamak için **urun.getadSoyad()** metodunu çağırmak yeterlidir. urunadi bilgisini değiştirmek için ise **urun.setadSoyad("Televizyon")** metodu çağrılır. Metot parametresi olarak değiştirilmek istenen değer verilir. **get** ile başlayan metotlar veriyi almak için, **set** ile başlayan metotlar ise veriyi değiştirmek için kullanılır.



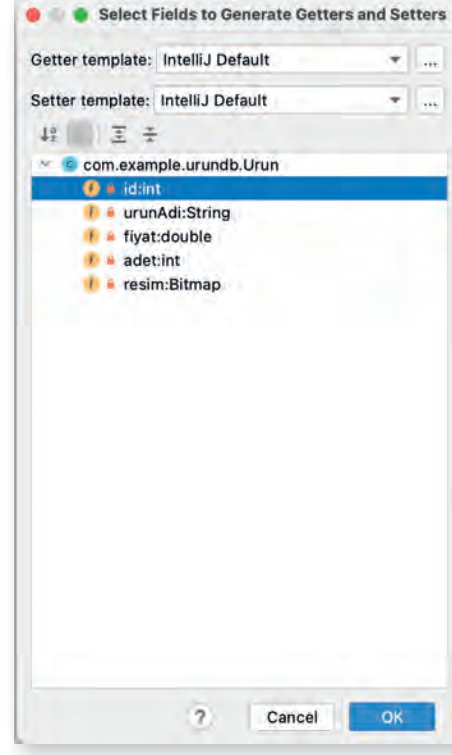
Görsel 7.14 Constructor menüsü



Görsel 7.16 Getter and Setter menüsü



Görsel 7.15: Constructor penceresi

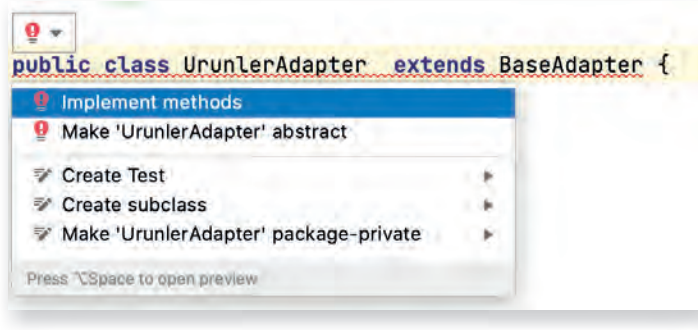


Görsel 7.17: Getter and Setter penceresi

Özel adaptör oluşturmak için yeni bir java sınıfı daha projeye eklenir. Özel adaptörler **BaseAdapter** sınıfından türetimelidir. Bir özel adaptör sınıfı şu şekilde oluşturulabilir:

```
public class UrunAdaptor extends BaseAdapter
```

Özel adaptör sınıfı tanımladıktan sonra mobil uygulama geliştirme ortamından bir hata mesajı alınır (Görsel 7.18). Kalıtım yolu ile bir sınıf yazıldığı için BaseAdapter sınıfının metotları oluşturulmalıdır.



Görsel 7.18: Kalıtım hatası

Kırmızı ampül simgesine basıldığında mobil uygulama geliştirme ortamı hatanın ortadan kalkması için kullanıcıya çeşitli seçenekler sunar. Açılan menüden Görsel 7.18'deki **Implements method** seçeneği seçilir.

```
public class UrunlerAdapter extends BaseAdapter {
    @Override
    public int getCount() {
        return 0;
    }
    @Override
    public Object getItem(int i) {
        return null;
    }
    @Override
    public long getItemId(int i) {
        return 0;
    }
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        return null;
    }
}
```

Özel adaptöre listelenmek istenen veriler dışarıdan verilmelidir. Ayrıca XML dosyasından gerekli Viewları bulmak için Context nesnesine ihtiyaç duyulur. Bir java sınıfı içinde Contexte ulaşmak kolay değildir. Bu yüzden Context nesnesi de dışarıdan alınır. Bunun için özel adaptör sınıfına şu iki özellik eklenir:

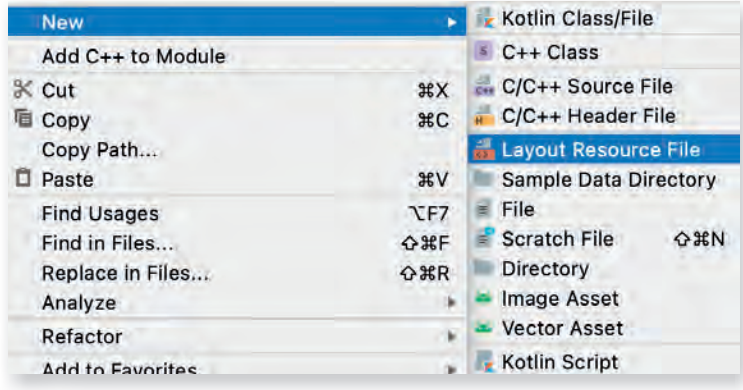
```
Context context;
ArrayList<Urun> liste;

public UrunlerAdapter(Context context, ArrayList<Urun> liste) {
    this.context = context;
    this.liste = liste;
}
```

Özellik olarak eklenen nesnelerin dışarıdan kolay alınabilmesi için bir yapılandırıcı metot tanımlanır. BaseAdapter sınıfından alınan metotlar Tablo 7.8'de verilmiştir.

Tablo 7.8: BaseAdapter Metotları

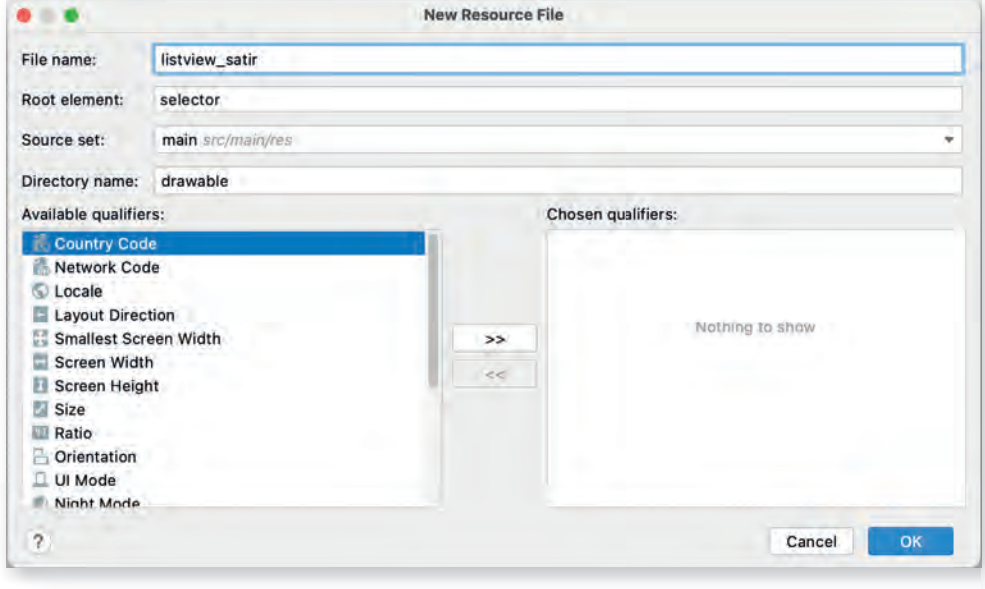
int getCount()	Veri listesinde kaç tane veri olduğunu geri gönderir.
object getItem(int i)	Veri listesinde indeks numarası ile veri elde etmeyi sağlar. İndeksi girilen veriyi geri gönderir.
long getItemId(int i)	Verinin indeks numarasını gönderir.
View getView(int i, View view, ViewGroup viewGroup)	Viewlara verilerin yazıldığı kısımdır.

Özel adaptörde her bir kaydın görüntüleneceği satır önceden tanımlanmalıdır. Mobil uygulama geliştirme ortamında app>res>layout üzerine sağ tıklanarak New>Layout Resource File seçeneği seçilir (Görsel 7.19).



Görsel 7.19: Layout Resource File seçim menüsü

Görsel 7.20'deki pencereden layout dosyasına bir isim verilir. Bu isim önemlidir. Bu isim özel adaptörde getView metodunda kullanılır. Oluşturulan layout dosyasında veri eklenmesi istenen nesnelere bir id verilmesi unutulmamalıdır.



Görsel 7.20: New Resource File penceresi

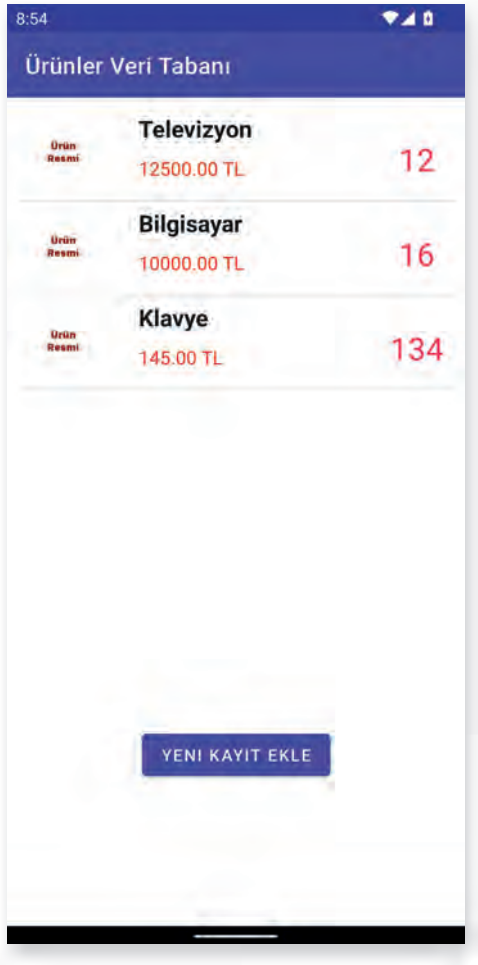
3. UYGULAMA: İşlem adımlarına göre Görsel 7.21'deki gibi bir veri tabanı uygulaması geliştiriniz. Geliştirdiğiniz uygulamayı Tablo 7.4'teki verilere uygun olarak tasarlayınız.

1. Adım: Mobil uygulama geliştirme ortamında yeni bir proje açıp Empty Activity seçiniz.

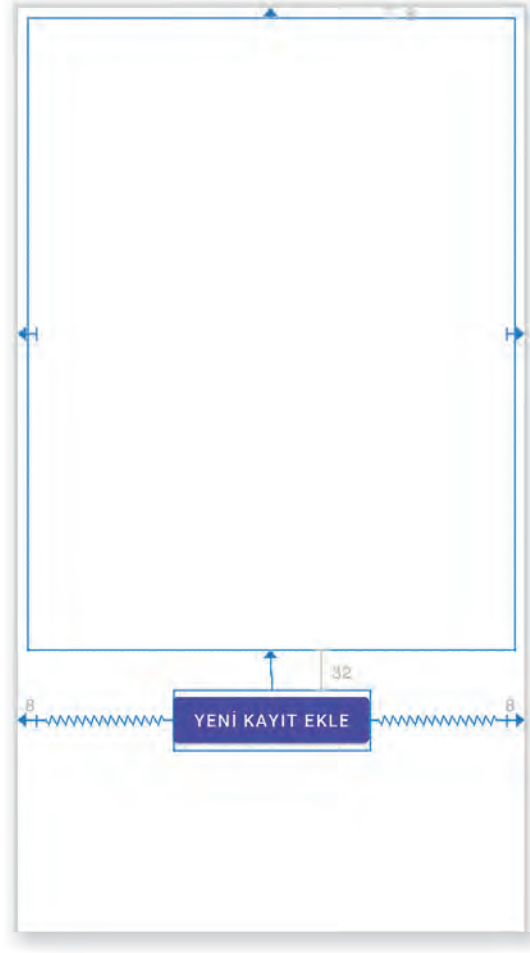
2. Adım: app>res>values>strings.xml dosyasını açarak şu şekilde değiştiriniz:

```
<resources>
    <string name="app_name">Ürünler Veri Tabanı</string>
</resources>
```

3. Adım: app>res>layout>activity_main.xml dosyasını açarak ana activitynin görünümünü Görsel 7.22'deki gibi tasarlayınız.



Görsel 7.21: Ürünler veri tabanı uygulaması ekranları



Görsel 7.22: Ürünler veri tabanı uygulaması MainActivity tasarım ekranı

4. Adım: Tasarım ekranına bir tane ListView, bir tane de button ekleyiniz.

5. Adım: ListView için id bilgisi urunListe, button için btnYeniKayitEkle değerlerini veriniz.

6. Adım: Yeni bir java sınıfı oluşturunuz ve adını Urun.java veriniz.

7. Adım: Urun.java dosyasını şu şekilde kodlayınız:

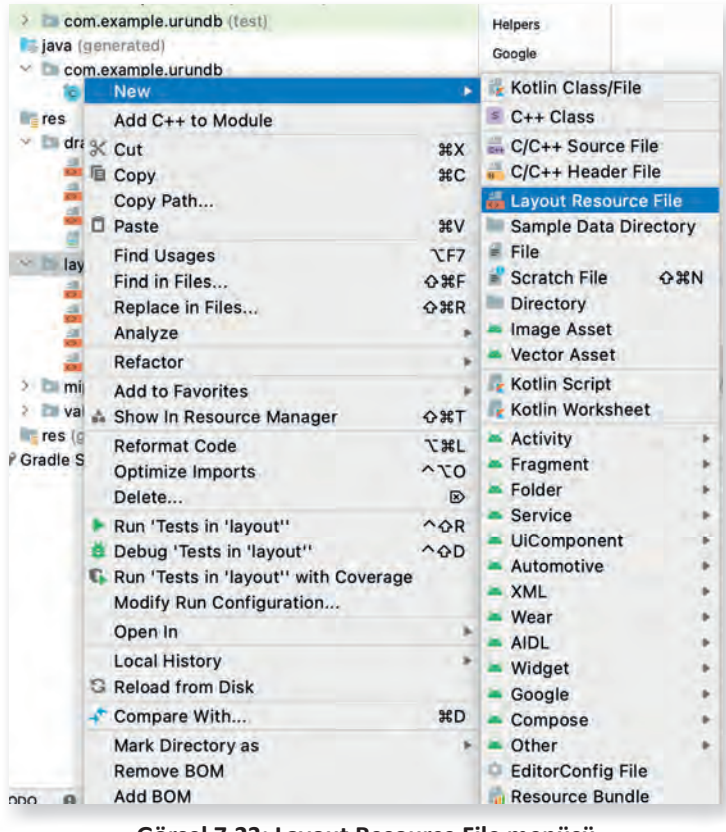
```
public class Urun {
    private int id;
    private String urunadi;
    private double fiyat;
    private int adet;
    private Bitmap resim;
    public Urun(int id,String urunadi, double fiyat, int adet, Bitmap resim) {
        this.id=id;
        this.urunadi = urunadi;
        this.fiyat = fiyat;
        this.adet = adet;
        this.resim = resim;
    }
    public int getId(){
        return this.id;
    }
public void setId(int id){
        this.id=id;
    }
    public String getUrunadi() {
        return urunadi;
    }
    public void setUrunadi(String urunadi) {
        this.urunadi = urunadi;
    }
    public double getFiyat() {
        return fiyat;
    }
    public void setFiyat(double fiyat) {
        this.fiyat = fiyat;
    }
    public int getAdet() {
        return adet;
    }
    public void setAdet(int adet) {
        this.adet = adet;
    }
    public Bitmap getResim() {
        return resim;
    }
    public void setResim(Bitmap resim) {
        this.resim = resim;
    }
}
```

8. Adım: UrunlerAdapter isimli yeni bir sınıf oluşturunuz.

9. Adım: UrunlerAdapter sınıfını şu şekilde kodlayınız:

```
public class UrunlerAdapter  extends BaseAdapter {
    ArrayList<Urun> liste;
    Context context;
    public UrunlerAdapter(ArrayList<Urun> liste, Context context) {
        this.liste = liste;
        this.context = context;
    }
    @Override
    public int getCount() {
        return liste.size();
    }
    @Override
    public Object getItem(int i) {
        return liste.get(i);
    }
@Override
    public long getItemId(int i) {
        return i;
    }
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        TextView satirUrunadi;
        TextView satirUrunFiyat;
        TextView satirUrunAdet;
        ImageView satirUrunResmi;
        Urun urun=liste.get(i);
        if(view==null){
            view= LayoutInflater.from(context).inflate(R.layout.listview_sa-
tir,viewGroup,false);
        }
        satirUrunadi=view.findViewById(R.id.satirUrunadi);
        satirUrunFiyat=view.findViewById(R.id.satirUrunFiyat);
        satirUrunAdet=view.findViewById(R.id.satirUrunAdet);
        satirUrunResmi=view.findViewById(R.id.satirUrunResmi);
        satirUrunadi.setText(urun.getUrunadi());
        satirUrunFiyat.setText(String.format("%.02f",urun.getFiyat())+ " TL");
        satirUrunAdet.setText(urun.getAdet()+"");
        if(urun.getResim()!=null)
         satirUrunResmi.setImageBitmap(urun.getResim());
        return view;
    }
}
```

10. Adım: res>layout dizinine gelerek dizin adına sağ tıklayınız. New menüsünden Layout Resource File seçeneğini seçiniz. Dosyayı listview_satir.xml yapınız (Görsel 7.23).



Görsel 7.23: Layout Resource File menüsü

11. Adım: listview_satir.xml dosyasını şu şekilde kodlayınız:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/satirUrunResmi"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:scaleType="fitCenter"
        android:src="@drawable/urun_resmi"/>
    <LinearLayout
        android:layout_width="230dp"
        android:layout_height="wrap_content"
        android:paddingLeft="20dp"
        android:orientation="vertical">
```

```
        <TextView
            android:id="@+id/satirUrunadi"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="#212121"
            android:text="Ürün Adı"
            android:padding="5dp"
            android:textStyle="bold"
            android:textSize="20sp"/>
        <TextView
            android:id="@+id/satirUrunFiyat"
            android:textColor="#dd2c00"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Ürün Fiyatı"
            android:padding="5dp"
            android:textSize="16sp"/>
    </LinearLayout>
    <TextView
        android:id="@+id/satirUrunAdet"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="999"
        android:textColor="#f50057"
        android:textSize="27sp"
        android:padding="5dp"
        android:gravity="center"
        android:layout_marginTop="10dp"/>
</LinearLayout>
```

12. Adım: Uygulamaya yeni bir Empty Activity ekleyiniz ve adını UrunKayit veriniz.

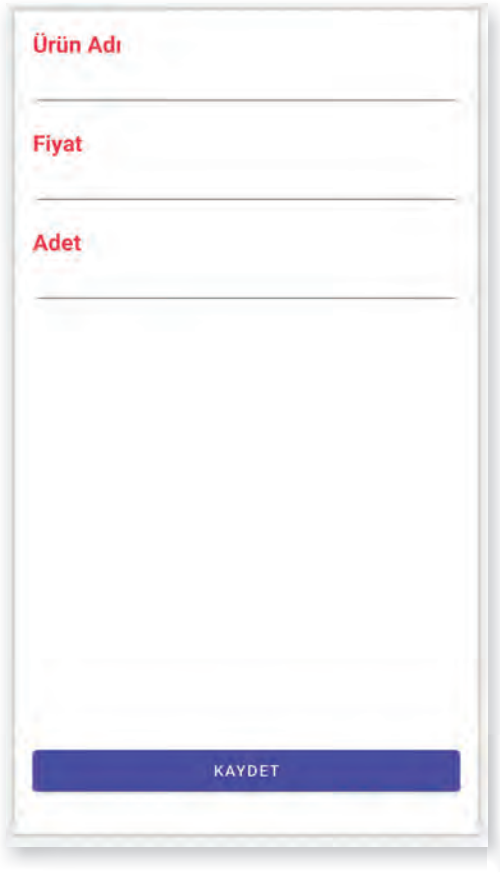
13. Adım: Uygulamaya yeni bir Empty Activity ekleyiniz ve adını UrunDetay veriniz.

14. Adım: Uygulamayı çalıştırınız ve App Inspection penceresinden veri ekleyerek uygulamayı test ediniz.

4. UYGULAMA: İşlem adımlarına göre üçüncü uygulamadaki projeye veri kayıt bölümü ekleyiniz.

1. Adım: Üçüncü uygulamada yaptığınız projeyi açınız.

2. Adım: activity_urun_kayit.xml dosyasını açınız. Activity'yi Görsel 7.24'teki gibi tasarlayınız.



Görsel 7.24: UrunEkle tasarım ekranı

3. Adım: Üç tane EditText ve bir tane button ekleyiniz. EditTextlere editKayitUrunadi, editKayitFiyat ve editKayitAdet id bilgilerini veriniz.

4. Adım: Button id bilgisini btnKayit veriniz.

5. Adım: UrunKayit.java dosyasını açıp şu şekilde kodlayınız:

```
public class UrunKayit extends AppCompatActivity {
    SQLiteDatabase database;
    EditText urunadi;
    EditText urunFiyat;
    EditText urunAdet;
    Button btnKaydet;
    int id;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_urun_kayit);
        urunadi=findViewById(R.id.editKayitUrunadi);
        urunFiyat=findViewById(R.id.editKayitFiyat);
        urunAdet=findViewById(R.id.editKayitAdet);
        btnKaydet=findViewById(R.id.btnKayit);
        Intent intent=getIntent();
```

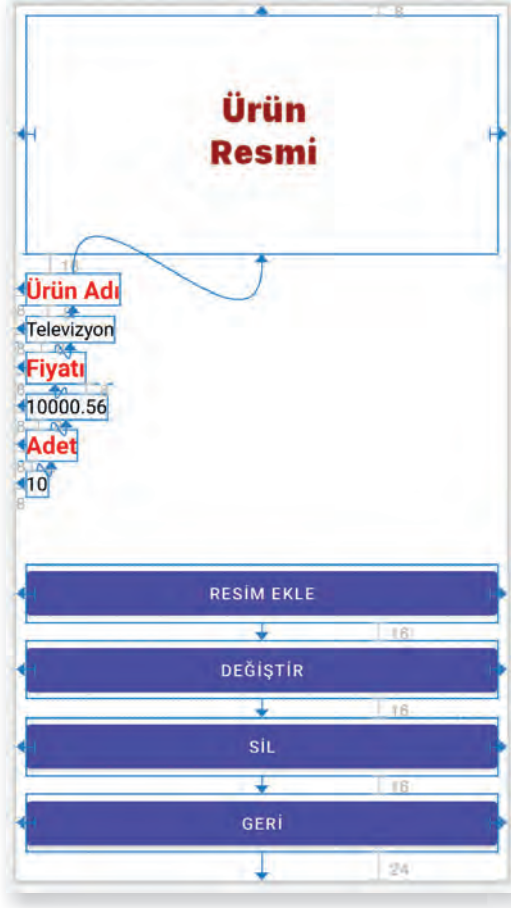
```
        id=intent.getIntExtra("id",0);
        String mod=intent.getStringExtra("mod");
        database=this.openOrCreateDatabase("urunler",MODE_PRIVATE,null);
        if(mod.equals("degistir")){
            try {
                Cursor cursor=database.rawQuery("SELECT urunadi,fiyat,adet FROM
urunler WHERE id=?",
                        new String[]{String.valueOf(id)});
                int kolonUrunadi=cursor.getColumnIndex("urunadi");
                int kolonFiyat=cursor.getColumnIndex("fiyat");
                int kolonAdet=cursor.getColumnIndex("adet");
                while (cursor.moveToNext()){
                    urunadi.setText(cursor.getString(kolonUrunadi));
                    urunFiyat.setText(cursor.getString(kolonFiyat)+"");
                    urunAdet.setText(cursor.getInt(kolonAdet)+"");
                }
                cursor.close();
            }catch (Exception e)
            {
                e.printStackTrace();
            }
        }
        btnKaydet.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(mod.equals("degistir")){
                    String SORGU="UPDATE urunler SET urunadi=?,fiyat=?,adet=?
WHERE id=?";
                    SQLiteStatement durumlar=database.compileStatement(SORGU);
                    durumlar.bindString(1,
                            urunadi.getText().toString());
                      durumlar.bindDouble(2,
                            Double.parseDouble(urunFiyat.getText().toS-
tring()));
                    durumlar.bindLong(3,
                            Integer.parseInt(urunAdet.getText().toString()));
                    durumlar.bindLong(4,id);
                    durumlar.execute();
                }else{
         String SORGU="INSERT INTO urunler(urunadi,fiyat,adet) VALUES(?,?,?)";
                    SQLiteStatement durumlar=database.compileStatement(SORGU);
                    durumlar.bindString(1,urunadi.getText().toString());
                    durumlar.bindDouble(2,
                            Double.parseDouble(urunFiyat.getText().toS-
tring()));
                    durumlar.bindLong(3,
                            Integer.parseInt(urunAdet.getText().toString()));
durumlar.execute();
                }
```

```
  Intent intent=new Intent(UrunKayit.this, MainActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

6. Adım: Uygulamayı çalıştırınız ve yeni kayıtlar ekleyiniz.

5. UYGULAMA: İşlem adımlarına göre dördüncü uygulama ile yaptığınız projeyi açınız. Dördüncü uygulamaya veri güncelleme, silme ve ürünlere görsel ekleme özelliklerini geliştiriniz.

1. Adım: activity_urun_detay.xml dosyasını açınız. Görsel 7.25'teki gibi ekran tasarımını yapınız.



Görsel 7.25: Ürünler veri tabanı uygulaması UrunDetay ekranı tasarımı

2. Adım: Bir tane ImageView, dört tane button ekleyiniz.

3. Adım: ImageView için urunResim id bilgisini, buttonlar için btnResimEkle, btnDegistir, btnSil ve btnGeri id bilgilerini veriniz.

4. Adım: UrunDetay.java dosyasını açıp şu şekilde kodlayınız:

```
public class UrunDetay extends AppCompatActivity {
    SQLiteDatabase database;
    TextView urunadi;
    TextView urunFiyat;
    TextView urunAdet;
    Button btnDegistir;
    Button btnGeri;
    Button btnSil;
    Button btnResimEkle;
    ImageView urunResim;
    int id;
    ActivityResultLauncher<Intent> galleryLauncher;
    ActivityResultLauncher<String> galleryPermisson;
    Bitmap bitmap;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_urun_detay);
        urunadi=findViewById(R.id.txtUrunadi);
        urunFiyat=findViewById(R.id.txtUrunFiyati);
        urunAdet=findViewById(R.id.txtUrunAdet);
        btnDegistir=findViewById(R.id.btnDegistir);
        btnGeri=findViewById(R.id.btnGeri);
        btnSil=findViewById(R.id.btnSil);
        btnResimEkle=findViewById(R.id.btnResimEkle);
        urunResim=findViewById(R.id.urunResim);
        Intent intent=getIntent();
        id=intent.getIntExtra("id",0);
        try {
            database=this.openOrCreateDatabase("urunler",MODE_PRIVATE,null);
            Cursor cursor=database.rawQuery("SELECT * FROM urunler WHERE id=?",
                    new String[]{String.valueOf(id)});
            int kolonUrunadi=cursor.getColumnIndex("urunadi");
            int kolonFiyat=cursor.getColumnIndex("fiyat");
            int kolonAdet=cursor.getColumnIndex("adet");
            int kolonResim=cursor.getColumnIndex("resim");
            while (cursor.moveToNext()){
                urunadi.setText(cursor.getString(kolonUrunadi));
                urunFiyat.setText(cursor.getString(kolonFiyat)+"");
                urunAdet.setText(cursor.getInt(kolonAdet)+"");
                byte[] bytes=cursor.getBlob(kolonResim);
                Bitmap bitmap= BitmapFactory.decodeByteArray(bytes,0,bytes.length);
                urunResim.setImageBitmap(bitmap);
            }
            cursor.close();
        }catch (Exception e)
        {
```

```
            e.printStackTrace();
        }
        Log.d(„SERVİS", "onCreate: "+ContextCompat.checkSelfPermission(this,Manifest.
permission.READ_EXTERNAL_STORAGE));
        registerLauncher();
        if(ContextCompat.checkSelfPermission(this,Manifest.permission.READ_EXTER-
NAL_STORAGE)
                !=
                PackageManager.PERMISSION_GRANTED){
                if(ActivityCompat.shouldShowRequestPermissionRationale(this,
                        Manifest.permission.READ_EXTERNAL_STORAGE)){
                    Toast.makeText(this, "İzin gerekli", Toast.LENGTH_SHORT).show();
                }
                else
                {
                    galleryPermisson.launch(Manifest.permission.READ_EXTERNAL_STO-
RAGE);
                }
        }
        btnResimEkle.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intentToGallery=new Intent(Intent.ACTION_PICK,
                        MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                galleryLauncher.launch(intentToGallery);
            }
        });
        btnDegistir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(getApplicationContext(),UrunKayit.class);
                intent.putExtra("mod","degistir");
                intent.putExtra("id",id);
                startActivity(intent);
                finish();
            }
        });
        btnSil.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                try {
                    String SORGU="DELETE FROM urunler WHERE id=?";
                    SQLiteStatement durumlar=database.compileStatement(SORGU);
                    durumlar.bindLong(1,id);
                    durumlar.execute();
                }catch (Exception e){
                    e.printStackTrace();
```

```
                }
                finish();
            }
        });
        btnGeri.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent1=new Intent(UrunDetay.this,MainActivity.class);
                startActivity(intent1);
                finish();
            }
        });
    }
    public Bitmap resimKucultucu(Bitmap b,int buyukluk){
        double oran=(double) b.getWidth()/(double) b.getHeight();
        double uzunluk=(double)buyukluk/oran;
        return bitmap.createScaledBitmap(bitmap,buyukluk,
                (int)uzunluk, true);
    }
    public void Kaydet(){
        Bitmap kucukResim = resimKucultucu(bitmap, 250);
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        kucukResim.compress(Bitmap.CompressFormat.PNG, 50, byteArrayOutputStream);
        byte[] bytes = byteArrayOutputStream.toByteArray();
        try {
            ContentValues contentValues=new ContentValues();
            contentValues.put("resim",bytes);
            database.update("urunler",contentValues,"id="+id,null);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void registerLauncher(){
        galleryLauncher=registerForActivityResult(new ActivityResultContracts.StartActivity-
ForResult(),
                new ActivityResultCallback<ActivityResult>() {
                    @Override
                    public void onActivityResult(ActivityResult result) {
                        if(result.getResultCode()==RESULT_OK){
                            Intent intentResult=result.getData();
                            if(intentResult!=null){
                                Uri galleryUri=intentResult.getData();
                                try {
                                    if(Build.VERSION.SDK_INT>=28){
                                        ImageDecoder.Source source=ImageDecoder.create-
Source(getContentResolver(),galleryUri);
                                        bitmap=ImageDecoder.decodeBitmap(source);
```

```
                                    urunResim.setImageBitmap(bitmap);
                                    Kaydet();
                                }else{
                                    bitmap= MediaStore.Images.Media.getBitmap(getCon-
tentResolver(),galleryUri);
                                    urunResim.setImageBitmap(bitmap);
                                    Kaydet();
                                }
                            }catch (Exception e){
                                e.printStackTrace();
                            }
                        }
                    }
                });

        galleryPermisson=registerForActivityResult(new ActivityResultContracts.RequestPer-
mission(),
                new ActivityResultCallback<Boolean>() {
                    @Override
                    public void onActivityResult(Boolean result) {
                        if(result){
                            Intent intentToGallery=new Intent(Intent.ACTION_PICK, MediaS-
tore.Images.Media.EXTERNAL_CONTENT_URI);
                            galleryLauncher.launch(intentToGallery);
                        }else{
                            Toast.makeText(UrunDetay.this,"İzin vermeniz gerekli!",Toast.
LENGTH_LONG).show();
                        }
                    }
                });
    }
}
```

5. Adım: Uygulamayı çalıştırınız ve veri tabanına veri ekleyiniz.

6. Adım: Eklenen kayıtlara galeriden görsel seçerek kayıtları güncelleyiniz.

7.2.10. ArrayAdapter Sınıfıyla Özel Adaptör Oluşturmak

Özel adaptör olarak BaseAdapter sınıfından türetilen bir sınıf kullanılabileceği gibi ArrayAdapter sınıfından da türetilebilir. İki sınıfın da yöntemleri benzerdir ve en önemli işleri iki sınıf da getView metodu ile yapar. ListView ile çok fazla veri göstermek uygulamanın hızını düşürebilir. Mobil uygulama geliştirme ortamında tasarım kısmının XML olması sistem kaynaklarını tüketir. Adaptör ile yapılan her bir satırlık işlemde mobil uygulama geliştirme ortamı, XML dosyasını tarayarak Viewları bulur. XML dosyalarını tarama işlemi zaman alır. Listelenmesi gereken çok sayıda veri varsa uygulama yavaşlar. Yavaşlamaya neden olan durumu ortadan kaldırmak için ViewHolder nesnesi veya mobil uygulama geliştirme ortamının sunduğu RecyclerView kullanılır.

6. UYGULAMA: İşlem adımlarına göre ArrayAdapter sınıfından türemiş bir özel adaptör yazınız.

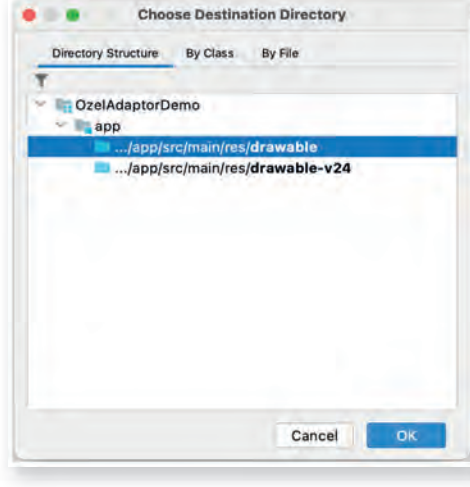
1. Adım: İnternet üzerinden telif hakkı olmayan ülke bayrakları ile ilgili resimleri temin ediniz.

2. Adım: Empty Activity ile yeni bir proje oluşturunuz.

3. Adım: app>res>values bölümünde strings.xml dosyasını açıp düzenleyiniz.

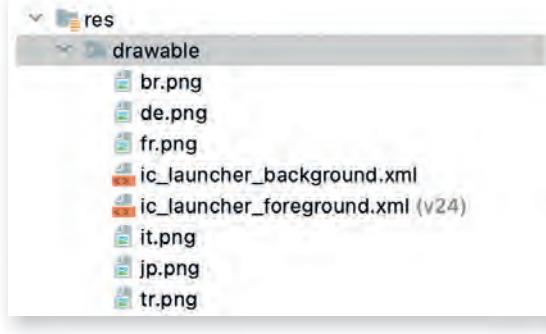
```
<resources>
    <string name="app_name">Ülke Bayrakları</string>
</resources>
```

4. Adım: Bayrak görüntülerini kopyalayıp mobil uygulama geliştirme ortamında Görsel 7.26'daki gibi app>res>drawable bölümüne yapıştırınız. Sadece drawable yazan seçeneği seçip OK butonuna basınız.



Görsel 7.26: Drawable klasörüne resim yapıştırma

5. Adım: Görsel 7.27'deki drawable klasöründe resimlerin göründüğünü kontrol ediniz.



Görsel 7.27: drawable klasörü

6. Adım: app>res>layout bölümüne gelip sağ tıklayarak yeni bir Layout Resource File ekleyiniz. Dosyanın adını listview_satir olarak veriniz. Daha sonra listview_satir.xml dosyasını düzenleyiniz.

```
<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <ImageView
            android:id="@+id/listeViewSatirBayrak"
            android:layout_width="80dp"
            android:layout_height="50dp"
            android:layout_marginLeft="3dp"
            android:layout_marginTop="7dp"
            app:srcCompat="@drawable/tr" />
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical">
           <TextView
               android:id="@+id/listViewAd"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:layout_margin="3dp"
               android:text="Türkiye"
               android:textSize="24sp"
               android:textStyle="bold" />
           <LinearLayout
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:orientation="horizontal">
              <TextView
                  android:id="@+id/listViewParaBirimi"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_margin="3dp"
                  android:text="Para birimi:Lira"
                  android:textSize="14sp" />
              <TextView
                  android:id="@+id/listViewNufus"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_margin="3dp"
                  android:text="Nüfus:87000000"
                  android:textSize="14sp" />
           </LinearLayout>
       </LinearLayout>
   </LinearLayout>
```

7. Adım: Modeli tasarlamak için Ulke.java isimli bir sınıf oluşturunuz. Modelin özelliklerini düzenleyip Generate menüsünden bir data sınıfına dönüştürünüz.

```
public class Ulke {
    int bayrak;
    String ad;
    int nufus;
    String paraBirimi;
}
```

8. Adım: app>res>layout bölümünde activity_main.xml dosyasını açıp düzenleyiniz.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginStart="1dp"
        android:layout_marginTop="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="1dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.5" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

9. Adım: Yeni bir java sınıf dosyası oluşturup adını OzelAdaptor.java olarak değiştiriniz. OzelAdaptor.java dosyasını düzenleyiniz.

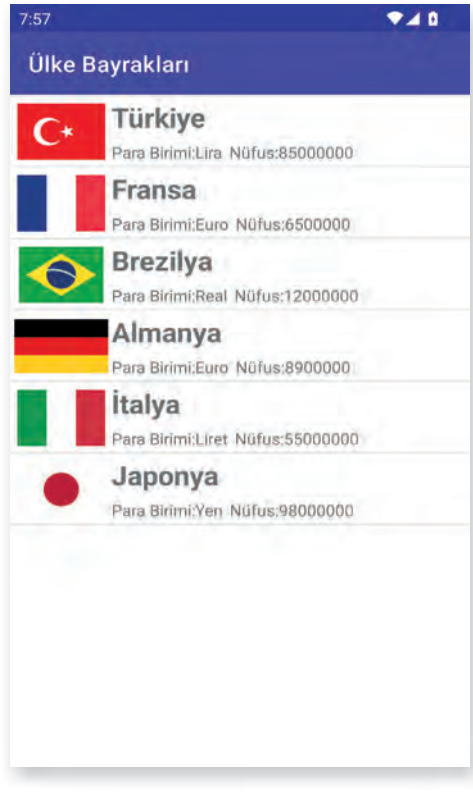
```
public class OzelAdaptor extends ArrayAdapter<Ulke> {
    private static class ViewHolder{
        TextView satir_ad;
        TextView satir_nufus;
        TextView satir_parabirimi;
        ImageView satir_bayrak;
    }
    public OzelAdaptor(Context context, ArrayList<Ulke> liste) {
        super(context,R.layout.listeview_satir,liste);
    }
```

```
@Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        Ulke ulke=getItem(i);
        ViewHolder viewHolder;
        if(view==null){
            viewHolder=new ViewHolder();
            LayoutInflater inflater = LayoutInflater.from(getContext());
            view=inflater.inflate(R.layout.listeview_satir,
                    viewGroup,false);
            viewHolder.satir_ad=view.findViewById(R.id.listViewAd);
            viewHolder.satir_nufus= view.findViewById(R.id.listViewNufus);
            viewHolder.satir_parabirimi= view.findViewById(R.id.listViewParaBirimi);
            viewHolder.satir_bayrak=view.findViewById(R.id.listeViewSatirBayrak);
            view.setTag(viewHolder);
        }
        else{
            viewHolder=(ViewHolder) view.getTag();
        }
        viewHolder.satir_bayrak.setImageResource(ulke.getBayrak());
        viewHolder.satir_ad.setText(ulke.getAd());
        viewHolder.satir_nufus.setText("Nüfus:"+ulke.getNufus());
        viewHolder.satir_parabirimi.setText("Para Birimi:"+ulke.getParaBirimi());

        return view;
    }
}
```

10. Adım: MainActivity.java dosyasını açıp düzenleyiniz.

```
public class MainActivity extends AppCompatActivity {
    ArrayList<Ulke> ulkeler;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView listView=(ListView) this.findViewById(R.id.listView);
        ulkeler=new ArrayList<Ulke>(){{
            add(new Ulke(R.drawable.tr,"Türkiye",85000000,"Lira"));
            add(new Ulke(R.drawable.fr,"Fransa",6500000,"Euro"));
            add(new Ulke(R.drawable.br,"Brezilya",12000000,"Real"));
            add(new Ulke(R.drawable.de,"Almanya",8900000,"Euro"));
            add(new Ulke(R.drawable.it,"İtalya",55000000,"Liret"));
            add(new Ulke(R.drawable.jp,"Japonya",98000000,"Yen"));
        }};
        OzelAdaptor adaptor=new OzelAdaptor(this,ulkeler);
        listView.setAdapter(adaptor);
    }
}
```

11. Adım: Uygulamayı çalıştırınız. Görsel 7.28'deki gibi görünüp görünmediğini kontrol ediniz.



Görsel 7.28: Ülke bayrakları uygulaması

SIRA SİZDE: Yeni bir Empty Activity ile bir proje oluşturunuz. Projede bir günlük ödevlerinizi kaydetmenizi sağlayacak bir model oluşturunuz. Modele uygun bir kayıt ekranı tasarlayarak verilerinizi yerel veri tabanına kaydeden bir uygulama geliştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Model oluşturdu.		
3. Modele uygun ekran tasarımları yaptı.		
4. Yerel veri tabanı oluşturdu.		
5. Yerel veri tabanına kayıt işlemlerini yaptı.		
6. Verilerin silme ve güncelleme işlemlerini tanımladı.		
7. Özel adaptör oluşturarak verileri gösterdi.		

7.3. UZAK VERİ TABANIYLA ÇALIŞMAK

Uygulama marketinde en popüler uygulamalar genellikle paylaşımın çok kolay olduğu uygulamalardır. Mobil cihazlarda herhangi resim, yazı veya videoyu paylaşabilmenin en kolay yolu bunları bir sunucuya göndermek ve buradan diğer kullanıcıların almasını sağlamaktır.

Mobil uygulama geliştirme ortamı uzak veri tabanları ile çalışmaya olanak verir. Mobil uygulama geliştirme ortamında birçok uzak veri tabanı kullanılabilir ancak Firebase veri tabanına tam destek verilir. İlişkisel veri tabanında tablolar bulunur ve veri bu tablolara yazılır. Firebase veri tabanı ise **NoSQL** adı verilen bir veri tabanı türüdür. NoSQL veri tabanlarına doküman veri tabanı adı da verilir. Bu veri tabanının en önemli özelliği çok hızlı olmasıdır. İlişkisel veri tabanlarında tablolar arasında ilişkilere bakılıp, tüm veriler birleştirilerek geri gönderilir. NoSQL veri tabanında ise böyle bir işlem yapılmaz, veri olduğu gibi geri gönderilir. İlişkisel veri tabanları tüm veri işlemlerini SQL sorgusu ile yapar. Örneğin veri almak için "SELECT" ile başlayan bir SQL sorgusu yazılmalıdır. NoSQL veri tabanlarında SQL sorguları yoktur. Sunucudan bir veri almak için koleksiyon adını ve verinin uid numarasını bilmek yeterlidir. Gelen veri düzenlenir ve aynı uid numarası ile gönderilirse sunucu bunu "UPDATE" işlemi olarak kabul eder. Gönderilen verinin uid numarası yoksa bu işlem de "INSERT" olarak kabul edilir ve koleksiyona yeni bir kayıt eklenir.

NoSQL veri tabanlarında veriler Javascript Nesne Gösterimi (JavaScript Object Notation) **JSON** formatına göre kaydedilir. Herhangi bir veri **"anahtar":"değer"** şeklinde gönderilirse NoSQL veri tabanlarına kaydedilir.

7.3.1. JSON Veri Düzeni

JSON günümüzde en sık kullanılan dosya kaydetme ve taşıma formatıdır. Tamamen metin tabanlı olarak çalıştığı için kullanımı çok kolay ve hızlıdır. Tüm veriler "anahtar":"değer" şeklindedir. Örnek bir JSON verisi şu şekildedir:

```
{
    "urunadi":"Bilgisayar",
    "fiyat": 12500.50,
    "adet":12
}
```

Bu şekilde oluşturulan herhangi bir JSON dokümanı bir NoSQL veri tabanına gönderilirse olduğu gibi kaydedilir. Tablo 7.4'te bulunan ürünler tablosundaki veriler bir JSON belgesine şu şekilde dönüştürülür:

```
"urunler":[
{ "id":1,"urunadi":"Televizyon","fiyat":12500.00,"adet":12},
{ "id":2,"urunadi":"Monitör","fiyat":3500.00,"adet":25},
{ "id":3,"urunadi":"Bilgisayar","fiyat":10000.00,"adet":16},
{ "id":4,"urunadi":"Klavye","fiyat":125.50,"adet":134}]
```

JSON belgesi tamamen metinlerden oluşturulabilir ancak burada oluşturulan "anahtar":"değer" çiftleri veri tipine göre düzenlenmelidir. Ürünler tablosunda bulunan veriler, veri türüne göre JSON belgesine dönüştürülmüştür. Tüm metin içeren alanlar " " işaretleri arasına yerleştirilmiş ve bunun bir metin olduğu verinin gönderilecek veri tabanına bildirilmiştir. Sayı içeren alanlarda ise veri olduğu gibi yazılmıştır. JSON belgesinin başında "urunler" ifadesi bulunur.

NoSQL veri tabanlarında tablolar bulunmadığı için tablo yerine koleksiyonlar kullanılır. Bir koleksiyon, köşeli ayraç "[ ]" işaretleri arasına yerleştirilir. Ürünler tablosu NoSQL veri tabanına taşındığında "urunler" koleksiyonu kullanılır. Tabloda bulunan bir satırlık kayda ise NoSQL veri tabanında doküman adı verilir. Bir başka deyişle NoSQL veri tabanında "urunler" koleksiyonunun dokümanları üzerinde işlemler yapılır.

Ürünler tablosunda ürünlerin bir de kategorileri kaydedilmek istenirse bu işlemi aynı tabloda yapmak mümkün değildir. İkinci bir kategori tablosu oluşturulup kategoriler bu tabloya kaydedilmelidir. JSON dosyalarında bu işlemi yapmak çok kolaydır. JSON dosyalarında bir koleksiyonun içinde başka bir koleksiyon tanımlanabilir. Ürünler koleksiyonuna kategoriler isimli bir koleksiyon şu şekilde yerleştirilir:

```
{
        "id":1,
        "urunadi":"Televizyon",
        "fiyat":12500.00,
        "adet":12,
        "kategoriler":["Elektronik","Görüntü Sistemleri","Ev/Sinema"]
}
```

JSON dosyaları; Sayı (Number), Metin (String), Koleksiyon (Array), Doğru/Yanlış (Boolean), Boş (Null) olmak üzere beş temel veri türüne sahiptir. NoSQL veri tabanlarında da bu veri türleri kullanılır. Veri örnekleri Tablo 7.9'da verilmiştir.

Tablo 7.9: JSON Veri Türleri

JSON Veri Türü	Java'daki Karşılığı	Örnek	Açıklama
Number	int, long, double, float	"deger" : 1 "oran" : 0,5	Sayısal değerler mutlaka bir sayı ile başlamalıdır. Ondalık sayılar 0 ile başlasa bile sayının başına 0 yazılmalıdır.
String	String	"urun":"klavye"	Metinler " " işaretleri arasına yazılmalıdır.
Boolean	boolean	"durum":true "durum":false	Doğru için **true,** yanlış için **false** yazılır.
Array	Array, List, ArrayList	"dongu":["bas":8, "bit":5]	Koleksiyon verisi diğer JSON veri türleri ile uyumlu olarak istenen her veri yazılabilir.
Null	Null	"sondeger":null	Bir alan boş bırakılmak istenirse mutlaka null yazılmalıdır.

JSON dosyalarında tür bağımlılığı yoktur. Örneğin { "deger" : "mobil" } gibi bir JSON dokümanı tanımlandığında bu veri daha sonra { "deger" : null } yapılabilir. Hatta { "deger" : 0,5 } gibi çok değişik bir tür ataması bile yapılabilir. Veri hangi şekilde NoSQL veri tabanına gönderilirse gönderilsin son hâli ile kaydedilir.

7.3.2. Mobil Uygulama Geliştirme Ortamında Uzak Sunucu Yapılandırması

Firebase veri tabanını kullanmak için mutlaka bir Google hesabına ihtiyaç vardır. Veri tabanı hizmetinin verildiği https://firebase.google.com/ internet sitesine gidilerek Google hesabı ile oturum açılmalıdır. Oturum açıldıktan sonra veri tabanı yönetim konsoluna gidilerek bir proje oluşturulur. Proje için gerekli ayarlamaların yapıldığı bir ayar dosyası indirilerek mobil uygulama geliştirme ortamında doğru yere yüklenmelidir. İndirilen dosyanın isminde veya uzantısında herhangi bir değişiklik yapılmamalıdır.

7. UYGULAMA: İşlem adımlarına göre Firebase veri tabanını bir Android projesine dâhil ediniz.

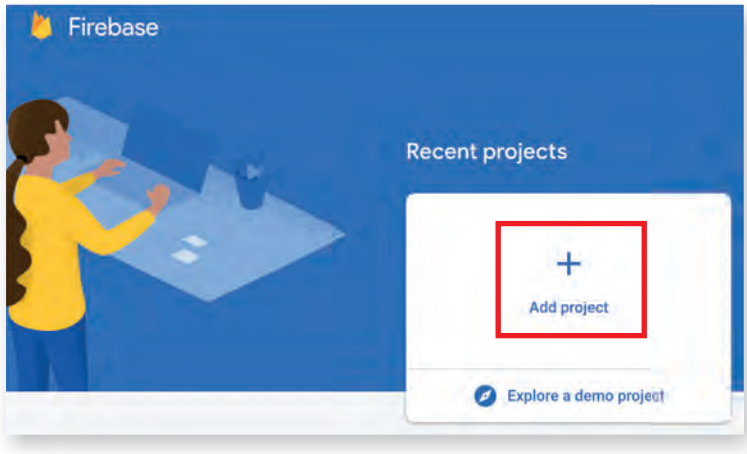
1. Adım: Mobil uygulama geliştirme ortamında yeni bir proje oluşturup adını "urunlerfirebase" veriniz.

2. Adım: Herhangi bir internet tarayıcısından https://firebase.google.com/ internet sitesini kullanarak Google hesabı ile oturum açınız. Hesap açtıktan sonra Görsel 7.29'daki Go to Console linkine tıklayınız.

Görsel 7.29: Go to console linki

3. Adım: Yeni bir proje oluşturmak için Görsel 7.30'daki **Add project** linkine tıklayınız.

4. Adım: Gelen sayfada Firebase projenize Görsel 7.31'deki gibi "urunlerfirebase" adını veriniz. Daha sonra **Continue** buttonuna basınız.



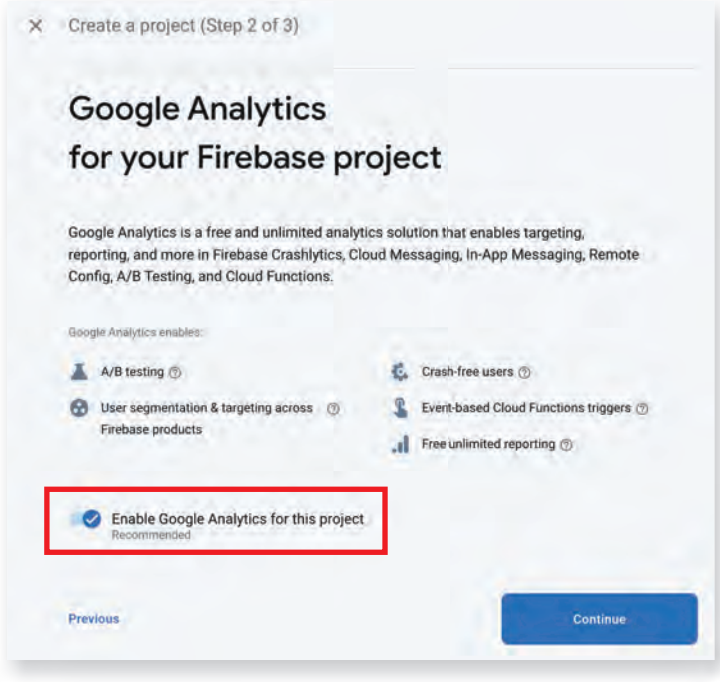
Görsel 7.30: Add project buttonu

Görsel 7.31: Firebase proje isminin belirlenmesi

NOT:

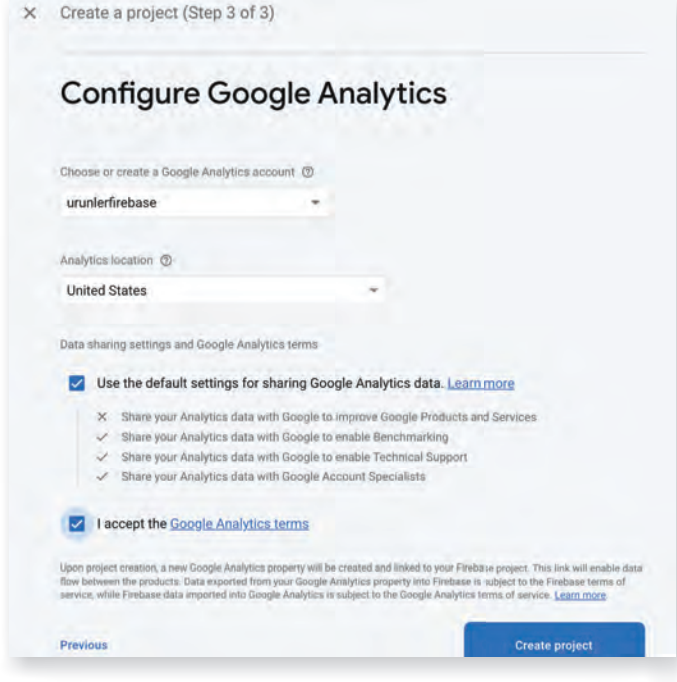
Firebase proje isimleri sadece konsolda görünen isimleri belirler. Mobil uygulama geliştirme ortamında herhangi bir önemi yoktur.

5. Adım: Firebase projenize "Google Analytics" hizmeti eklemek isterseniz Görsel 7.32'deki gibi spinner kontrolünü aktifleştiriniz. Daha sonra Continue buttonuna basınız.



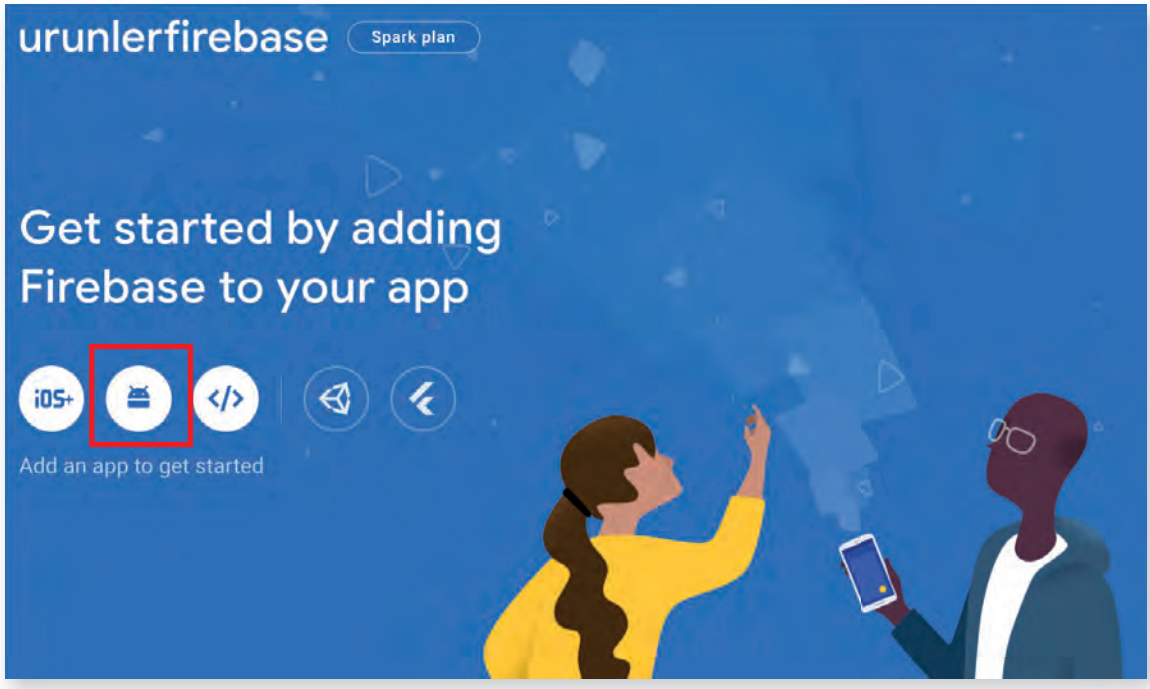
Görsel 7.32: Google Analytics aktifleştirme ekranı

6. Adım: "Google Analytics" projenizde aktifleştirdiğiniz için bir Analytics hesabı açınız. Bu işlem için urunlerfirebase ismini vererek ve diğer ayarları da Görsel 7.33'teki gibi bırakarak **Create project** buttonuna basınız.



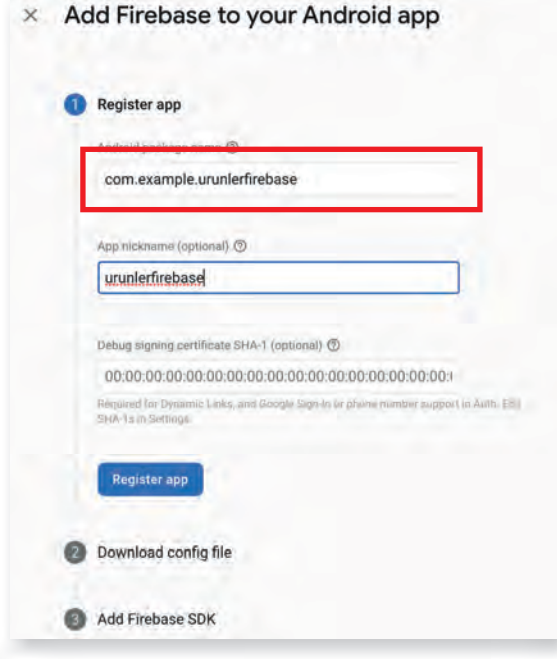
Görsel 7.33: Google Analytics hesabının açılması

7. Adım: Projenizi oluşturduktan sonra proje ana sayfasına yönlendirileceksiniz. Mobil uygulama geliştirme ortamında ayarların yapılması için Görsel 7.34'te bulunan Android simgesine basınız.



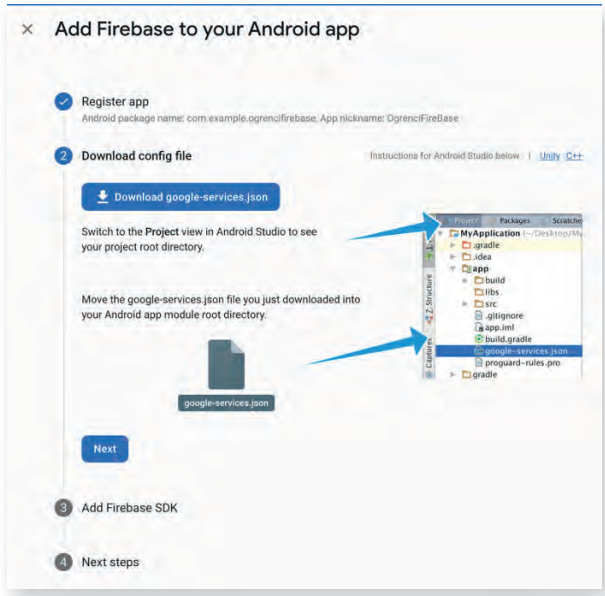
Görsel 7.34: urunlerfirebase projesi ana ekranı

8. Adım: Mobil geliştirme ortamında oluşturduğunuz projeyi açınız. MainActivity.java dosyasını açınız. En üstte bulunan paket adını kopyalayıp Görsel 7.35'te bulunan **Android packege name** bölümüne yapıştırınız. Diğer ayarları da Görsel 7.35'teki gibi yapıp **Register app** buttonuna basınız.



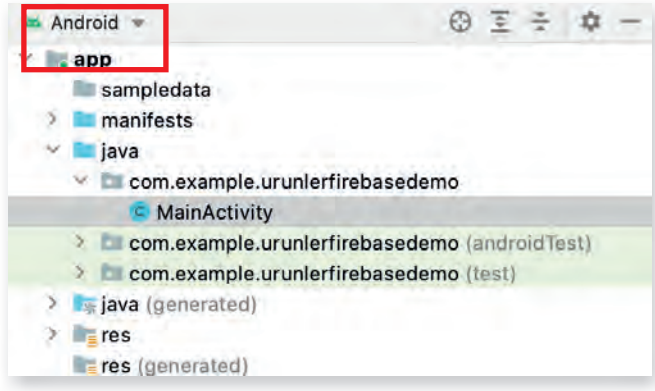
Görsel 7.35: Firebase projesi kayıt ekranı

9. Adım: Görsel 7.36'da bulunan **Download google-service.json** buttonuna basınız. Sunucudan bir ayar dosyası inecektir. **google-service.json** dosyasının adını kesinlikle değiştirmeyiniz.



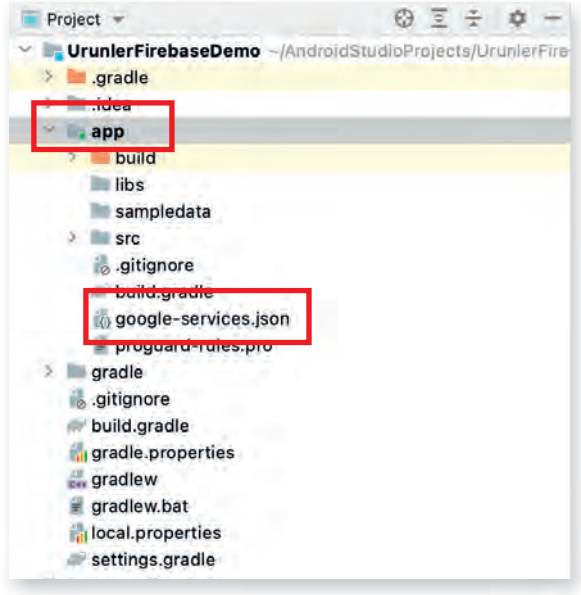
Görsel 7.36: Konfigürasyon dosyasının elde edilmesi

10. Adım: google-service.json dosyasını indirildiği yerde bulunuz. Dosyanın üzerine sağ tıklayıp **Kopyala** seçeneğini seçiniz. Tekrar mobil uygulama geliştirme ortamını açınız. Proje dosyaları bölümünde Görsel 7.37'deki işaretlenmiş kısımdan görünüm modunu Project'e geçiriniz.



Görsel 7.37: Proje dosyalarının Android görünümü

11.Adım: Project görünümünde **app** klasörünü bulunuz. Kopyaladığınız json dosyasını buraya yapıştırınız (Görsel 7.38). Bu işlemi yaptıktan sonra proje dosyaları modunu Android olarak ayarlayınız. Mobil uygulama geliştirme ortamında işlemleriniz bittikten sonra Firebase internet sitesine gelip Next buttonuna basınız.



Görsel 7.38: Proje dosyalarının Project görünümü

12. Adım: Firebase kütüphanelerini projenize ekleyiniz. İnternet sitesinde çıkan yönergeleri yapınız. Mobil uygulama geliştirme ortamında **bulid.gradle (Project)** dosyasını açınız. build.gradle dosyasının en üstüne şu kodları ekleyiniz:

```
buildscript {
    repositories {
        google()
    }
    dependencies {
        classpath 'com.google.gms:google-services:4.3.10'
    }
}
```

Mobil uygulama geliştirme ortamında **bulid.gradle (Module)** dosyasını açıp **dependencies** bölümüne şu kodları yazınız:

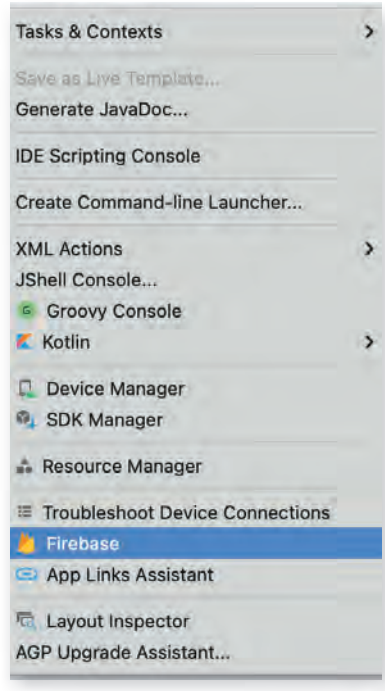
```
implementation platform('com.google.firebase:firebase-bom:30.1.0')
implementation 'com.google.firebase:firebase-analytics'
```

dependencies bölümünün önüne ise şu kodları ekleyiniz:

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
```

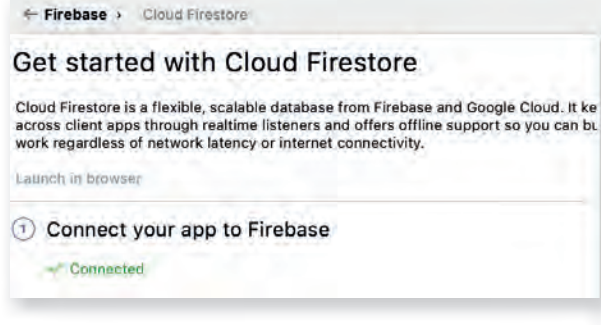
Tüm kodları yazdıktan sonra projeyi kaydedip ekranın sağ üst köşesinde bulunan **Sync now** butonuna basınız. Bütün kütüphanelerin indirilmesini ve Gradle işleminin başarılı tamamlanmasını bekleyiniz.

13. Adım: Gradle işlemi bittikten sonra uygulamanın Firebase sunucusuna bağlanıp bağlanmadığını test ediniz. Görsel 7.39'daki gibi Tools menüsünden Firebase aracını seçiniz.



Görsel 7.39: Firebase araçları

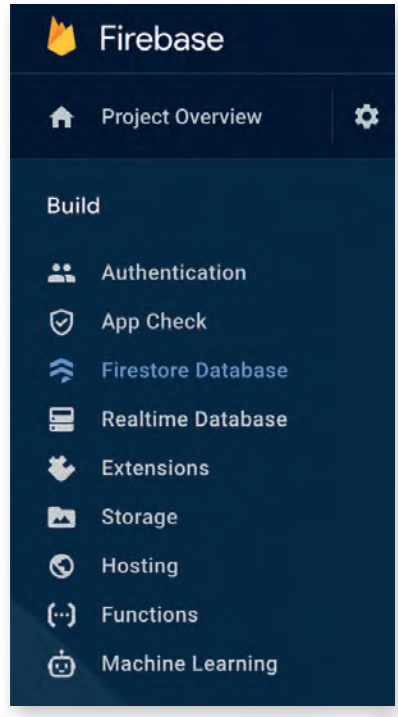
14. Adım: Açılan menüden **Colud Firestore** seçeneğini bulunuz. Bu bölümde **Get started with cloud Firestore** linkine basınız. Görsel 7.40'taki gibi **Connected** yazısı görünürse uygulama, Firebase veri tabanı ile iletişim hâlindedir.



"Görsel 7.40: Coloud Firestore aracı

7.3.3. Uzak Veri Tabanı Yönetimi

Uzak veri tabanını kullanmak için konsolda çeşitli ayarlamalar yapılmalıdır. Yönetim konsolu Görsel 7.41'de görülen seçeneklerden oluşur.



Görsel 7.41: Uzak veri tabanı yönetim menüsü

- **Authentication**

  Uzak veri tabanları internette çalıştığı için veri tabanı herkese açıktır. Veri tabanına herkesin erişmesi istenmeyen bir durumdur. Sadece yetkili kişilerin veri tabanına ulaşması için Authentication bölümü kullanılır. Authentication bölümünde yetkilendirme türünün belirlendiği ve kullanıcıların kayıtlı olduğu bir veri tabanı bulunur.

- **App Check**

  Uzak veri tabanlarında en önemli sorunlardan biri yetkilendirme sahtekârlığıdır. Zararlı kullanıcılar yetki alabilmek türlü usulsüzler yaparak veri tabanına sızmaya çalışabilir. App Check bölümünde bu tür yetkisiz girişler tespit edilerek önlem alınması sağlanır.

- **Firestore Database**

  Firestore veri tabanı, Google tarafından geliştirilen en güncel veri tabanıdır. Firestore veri tabanını kullanmak için gerekli hazır fonksiyonlar önceki veri tabanlarından tamamen farklıdır.

- **Realtime Database**

  Realtime veri tabanı, Firestore veri tabanından daha önce kullanılan uzak veri tabanı sistemidir. Geriye uyumluluk için tüm versiyonlar hâlâ desteklenir. Realtime Database fonksiyonları Firestore veri tabanından tamamen farklıdır.

- **Extensions**

  Uzak veri tabanına bağlanmak için çeşitli hazır fonksiyonlar bulunur. Bu fonksiyonların yetersiz geldiği durumlarda Extensions bölümü kullanılır.

- **Storage**

  Storage, uzak veri tabanına kullanıcıların resim veya video gibi dosyalarını kaydetmelerini sağlar. Firestore ile beraber gelen bir özelliktir. Firestore veri tabanı aktif olduğunda Storage bölümü de aktifleşir.

- **Hosting**

  Hosting, veri tabanı haricinde web sayfaları veya bir veri servisini internette paylaşmak için kullanılır. Hosting sistemi, isteklerin coğrafi konumuna göre yeniden düzenleyerek sunumu yapılacak web sayfalarının veya servislerinin daha hızlı açılmasını sağlar.

- **Functions**

  Function bölümü, Typescript veya Javascript ile yazılmış kodların tetikleyici kullanarak Firebase hizmetlerini çalıştırmayı sağlar. Mobil uygulama geliştime ortamında kullanılmaz.

- **Machine Learning**

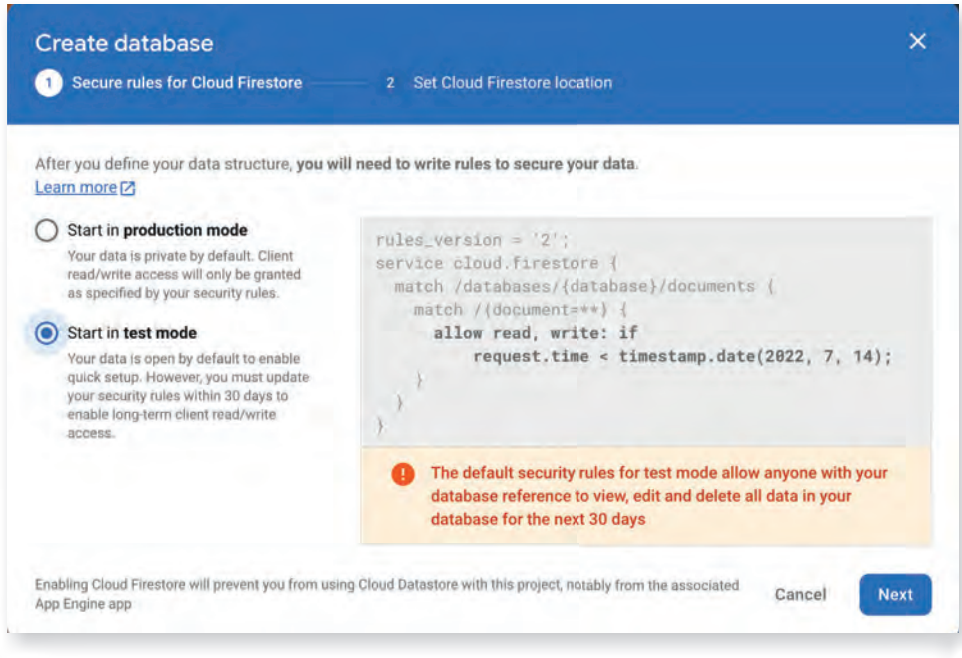
  Machine Learning, uzak veri tabanına gelen veriler üzerinde yapay zekâ teknikleri kullanarak çeşitli analizler yapılmasını sağlar.

7.3.4. Firestore Veri Tabanı Oluşturmak

Uzak veri tabanı kullanabilmek için öncelikle bir Firestore veri tabanı oluşturulmalıdır. Firestore Database bölümü açıldığında herhangi bir veri tabanı oluşturulmamışsa Görsel 7.42'deki ekran görülür.

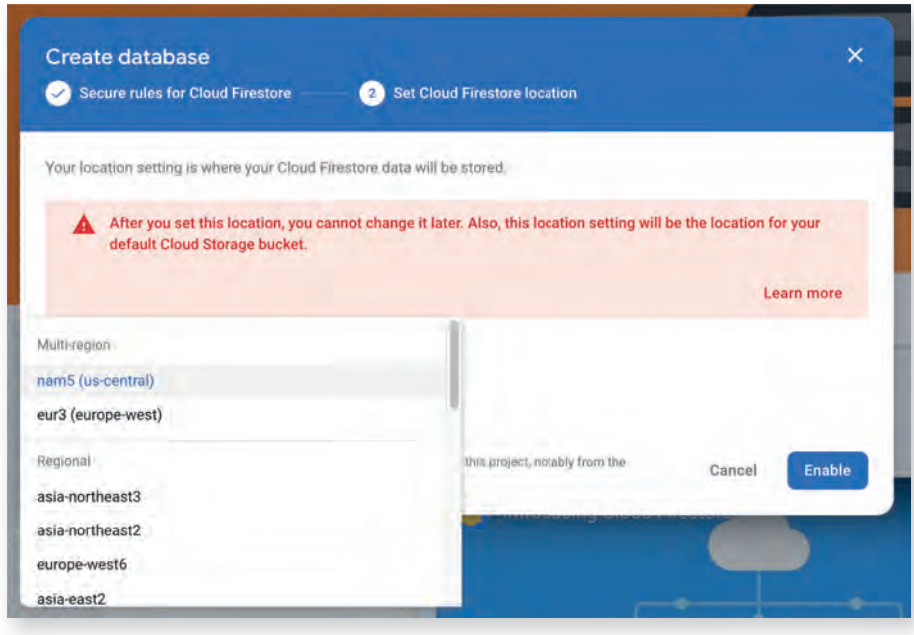
Görsel 7.42: Firestore veri tabanı oluşturma

Yeni bir veri tabanı oluşturmak için Görsel 7.42'deki **Create database** buttonuna basılır. Yeni bir veri tabanı oluşturulduktan sonra Görsel 7.43'teki gibi projenin hangi modda çalışacağı belirlenir.



Görsel 7.43: Firestore güvenlik kuralları belirleme ekranı

Sadece geliştirme amaçlı olarak **Start in test mode** seçeneği seçilmelidir. Diğer mod seçilirse detaylı olarak güvenlik ayarları yazılmalıdır. Geliştirme amaçlı olarak Start in test mode ihtiyaçlar için yeterlidir. Next buttonuna basılarak sonraki ekrana geçildiğinde Görsel 7.44'teki ekranla karşılaşılır.



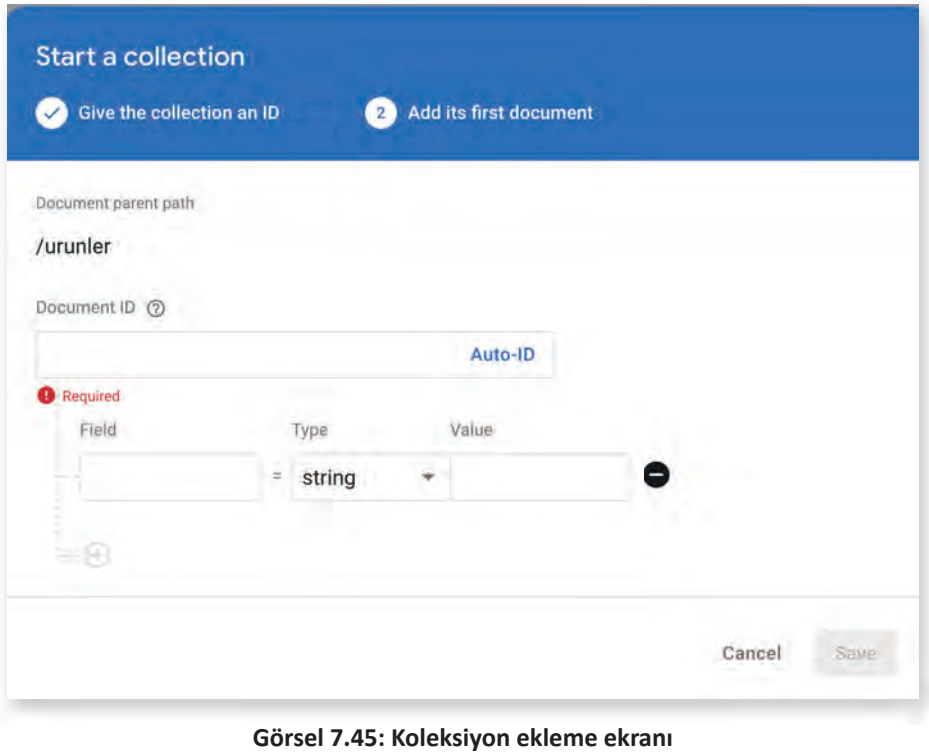
Görsel 7.44: Coğrafi bölge seçim ekranı

Görsel 7.44'te veri tabanının oluşturulacağı yer seçilir. Bu seçeneklerden coğrafi olarak en yakın

bölge seçilebilir. Coğrafi bölge seçiminden sonra Enable buttonuna basılarak veri tabanı oluşturulur.

7.3.4.1. Uzak Veri Tabanında Koleksiyon ve Doküman Oluşturmak

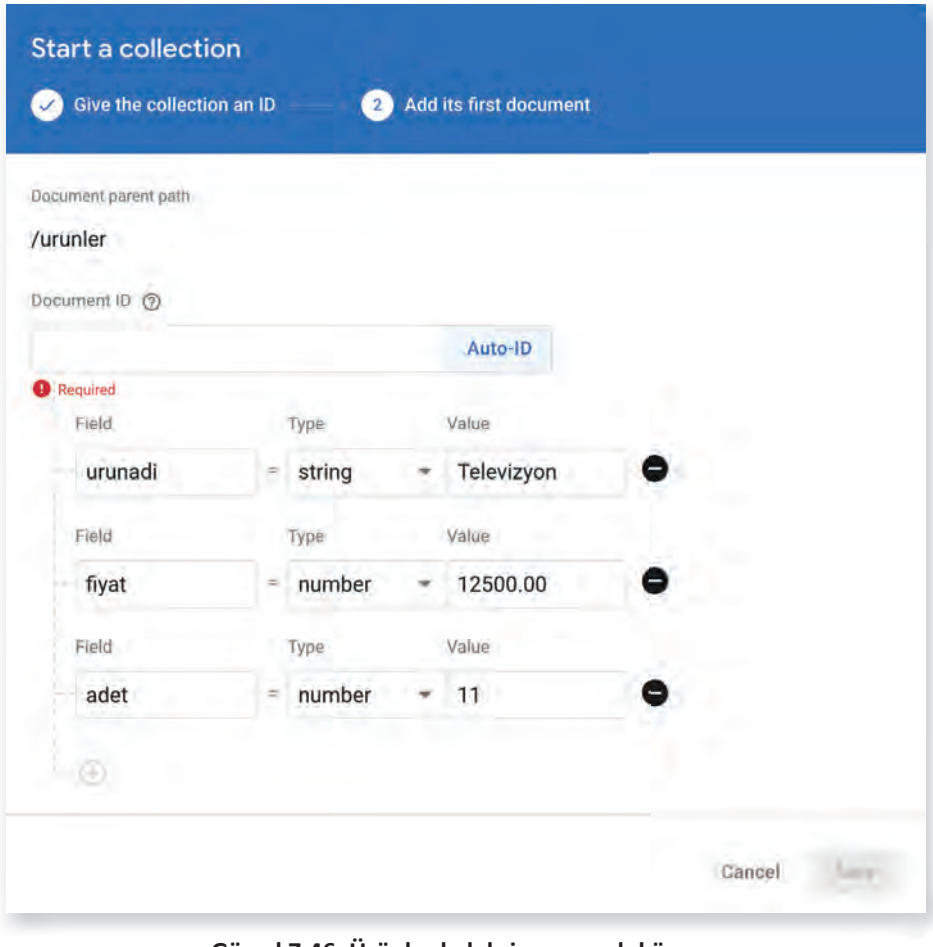
Uzak veri tabanı sisteminde koleksiyon ve dokümanlar kod kullanılarak oluşturulabilir ancak uzak veri tabanı sisteminde de koleksiyon ve dokümanların nasıl oluşturulduğu bilinmelidir. Örneğin mobil uygulama geliştirme ortamında koleksiyon silmek için herhangi bir metot yoktur. Koleksiyonların yönetimi mecburen veri tabanı yöneticisi tarafından el ile yapılmalıdır. **Start collection** buttonuna basılarak yeni bir koleksiyon oluşturulur. Koleksiyona bir isim verildikten sonra Next buttonuna basılır. Bir sonraki adımda doküman oluşturma ekranı ile karşılaşılır (Görsel 7.45).



Görsel 7.45: Koleksiyon ekleme ekranı

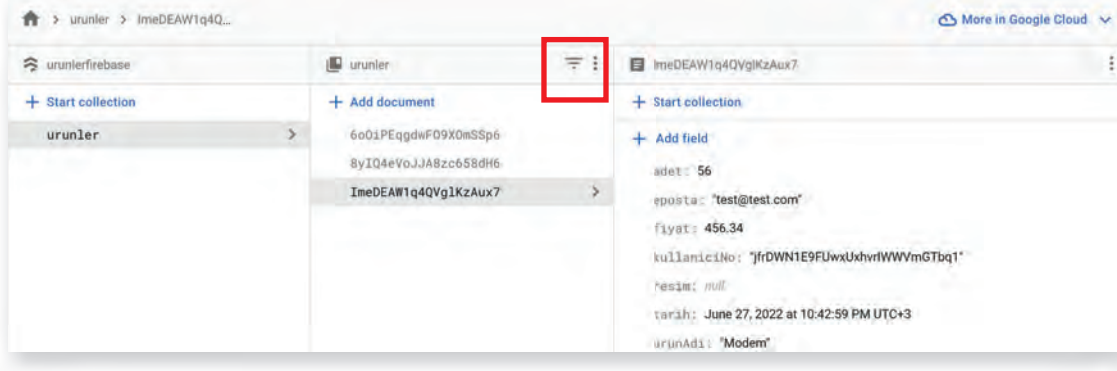
Doküman ekleme ekranında Document ID bölümü bulunur. NoSQL veri tabanlarında her bir doküman için bir ID numarası vermek zorunlu değildir ancak Firestore veri tabanında her doküman için bir ID numarası otomatik oluşturulur. Doküman ID numaraları ile çalışmak mobil uygulama geliştiricilerin işlerini oldukça kolaylaştırır. Sistem üzerinde bir doküman oluşturulduğunda Doküman ID numarası **Auto-ID** buttonuna basılarak otomatik olarak elde edilir. Kod ile bir doküman eklendiğinde ise Doküman ID numarası kayıt yapılırken otomatik verilir.

Doküman ekleme ekranında istendiği kadar alan oluşturulabilir. Alan ismi **Field** bölümüne yazıldıktan sonra **Type** bölümünden veri tipi seçilir. Görsel 7.46'daki gibi basit bir ürün veri tabanı oluşturulabilir. Tüm form doldurulduktan sonra **Save** buttonuna basılarak koleksiyon ve doküman oluşturulur.



Görsel 7.46: Ürünler koleksiyonu ve dokümanı

Görsel 7.47'deki Add Document buttonuna basılarak sınırsız sayıda doküman koleksiyona eklenebilir.



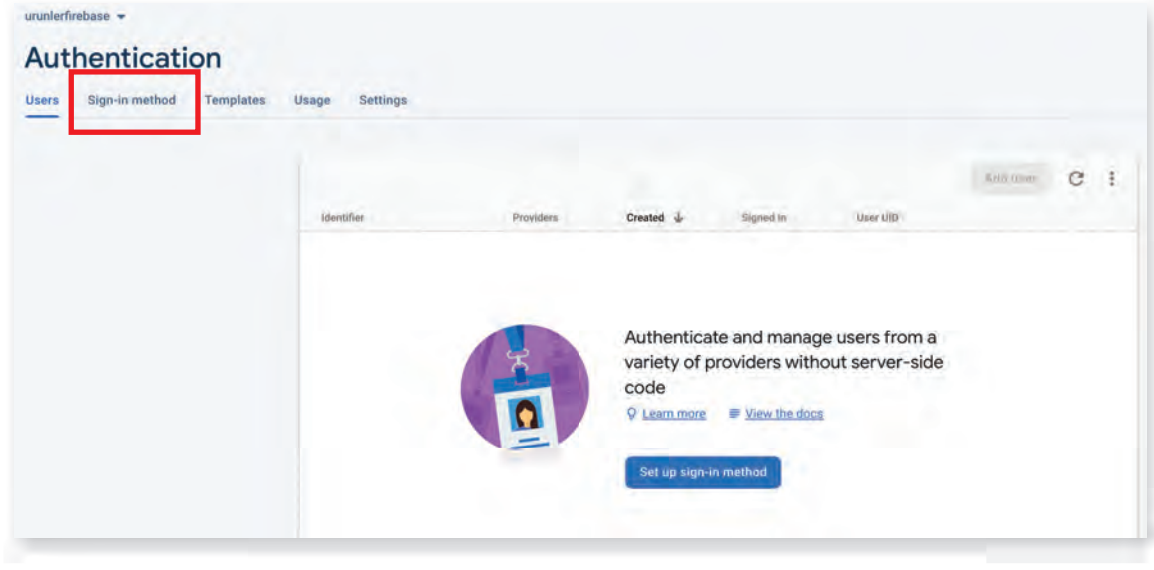
Görsel 7.47: Koleksiyonlar

7.3.4.2. Uzak Veri Tabanında Koleksiyon ve Doküman Silmek

Uzak veri tabanından koleksiyon silmek için Görsel 7.47'deki işaretli menü açılıp **Delete collection** seçeneği seçilir. Doküman silmek için ise aynı şekilde dokümanların üst tarafında bulunan menü simgesi açılarak açık olan doküman silinir.

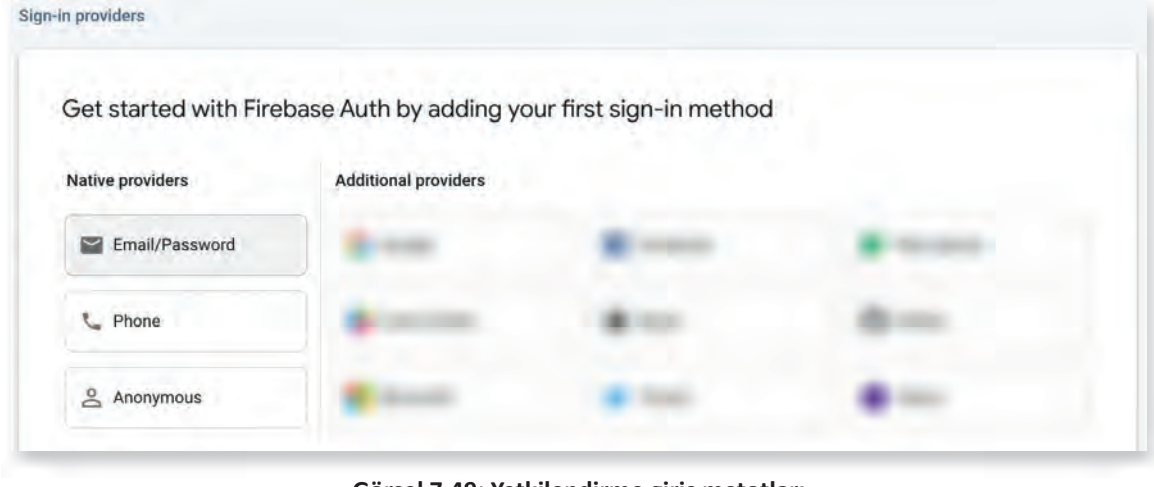
7.3.5. Uzak Veri Tabanında Yetkilendirme İşlemleri

Uzak veri tabanları internet üzerinde çalıştığı için internete bağlı her cihaz kullanabilir. Veri tabanı oluşturulduktan sonra internete bağlı her kullanıcının veri tabanına müdahale etmesi istenmez. Sadece yetkili kişilerin veri tabanına ulaşması için Authetication bölümü kullanılır· Herhangi bir yetkilendirme işlemi yapılmamışsa Görsel 7.48'deki ekranla karşılaşılır.



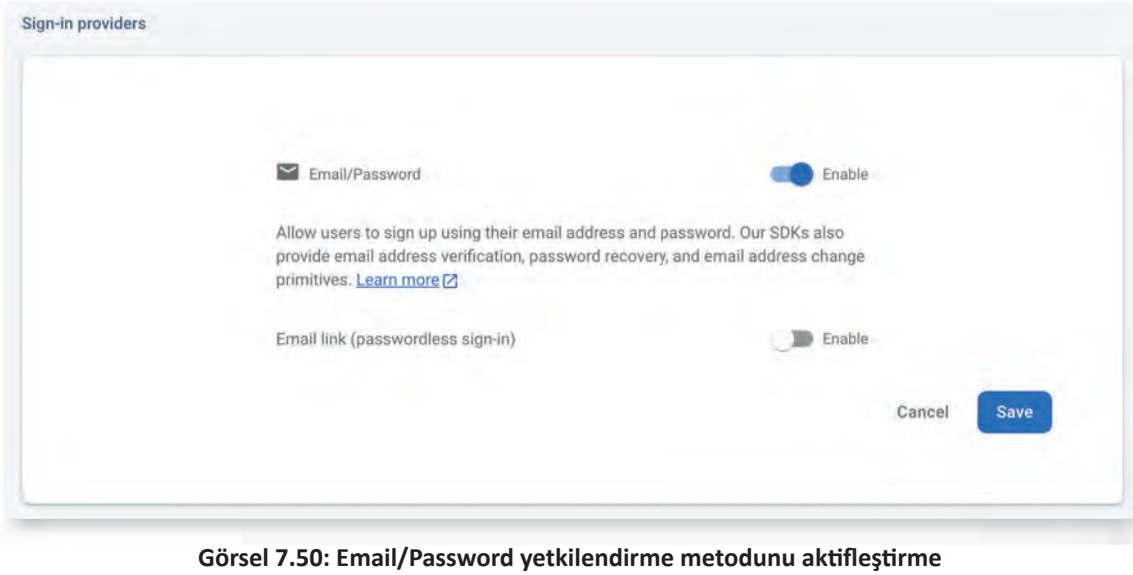
Görsel 7.48: Authentication giriş ekranı

Yetkilendirme işlemleri için öncelikle giriş metodu seçilmelidir. Görsel 7.48'deki seçili buttonlara basılarak Görsel 7.49'da görülen yetkilendirme giriş metodu ekranı açılır.



Görsel 7.49: Yetkilendirme giriş metotları

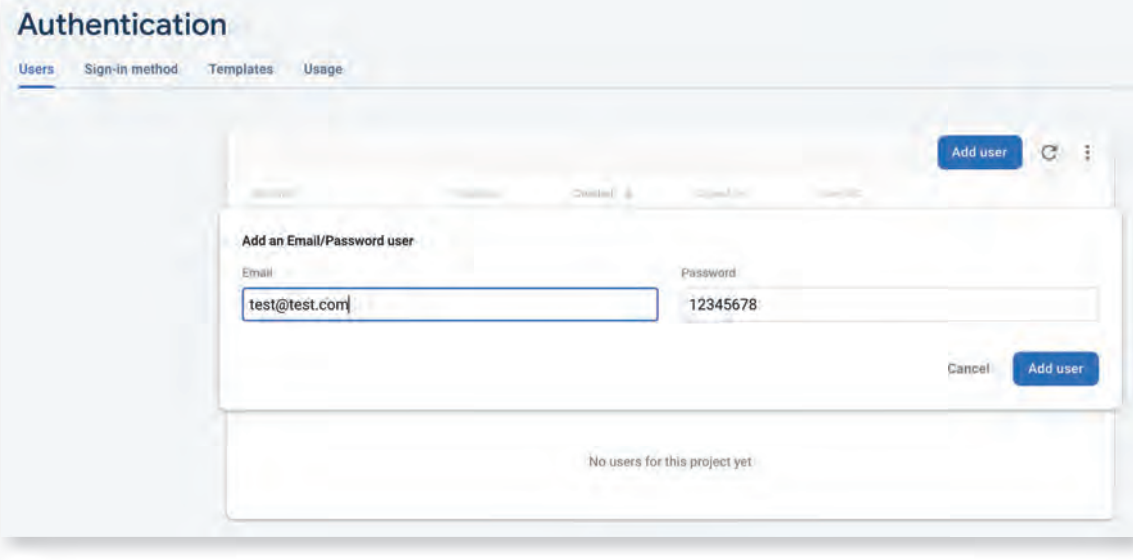
Yetkilendirme işlemlerinde giriş metotlarından biri seçilir. Sosyal medya hesaplarından Google hesaplarına istenen her hesap ile yetkilendirme yapılabilir. Email/Password yetkilendirme metodu seçildikten sonra Görsel 7.50'deki ekranda giriş metodunun aktif edilmesi sağlanır. Save buttonuna basıldıktan sonra giriş metodu ayarlanır.



Görsel 7.50: Email/Password yetkilendirme metodunu aktifleştirme

7.3.5.1. Uzak Veri Tabanına Kullanıcı Eklemek ve Kullanıcı İşlemleri

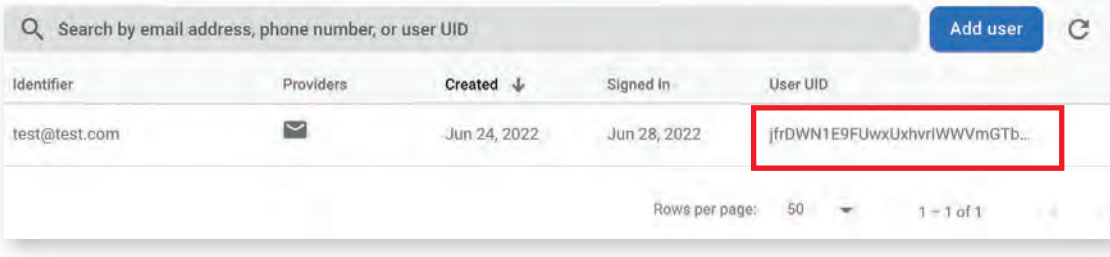
Firestore veri tabanında özellikle bir kullanıcı hakkındaki işlemler Users bölümünde yapılır. Görsel 7.51'deki Users bölümünde kullanıcılar oluşturulabilir.



Görsel 7.51: Kullanıcı oluşturma ekranı

Bir kullanıcı oluşturulduktan sonra Görsel 7.52'deki gibi sisteme kaydedilir. Kaydedilen her kullanıcı için bir User UID numarası verilir. Kullanıcılar için verilen User UID numaraları çok önemlidir. Ortak bir veri tabanı kullanıldığında hangi kullanıcının ne işlem yaptığı User UID bilgisinden belli olur.

Mobil uygulama geliştirme ortamında uzak veri tabanına gönderilen her veriye User UID bilgisi eklenir.



Görsel 7.52: User UID bilgisi

7.3.5.2. Uzak Veri Tabanında Doğrulama E-Postalarını Yönetmek

Uzak veri tabanı sisteminde e-posta ve şifre ile yetkilendirme yapmak, iyi niyetli olmayan kişilerin sorumsuzca sistemi kullanmalarına neden olabilir. İyi niyetli olmayan kullanıcılar sahte bir e-posta adresi ile sisteme giriş yapabilir. Firebase veri tabanı sisteminde yetkisiz e-posta adreslerini engellemek için e-posta doğrulama servisi vardır. Kullanıcı, sisteme kaydolduktan sonra mobil uygulama geliştirici tarafından kullanıcıya doğrulama e-postası gönderilir. Bu arada mobil uygulama geliştirici isterse doğrulanmamış e-postaların sisteme girişini de engelleyebilir.

Kullanıcılara gönderilen herhangi bir doğrulama e-postası şu şekildedir:

Hello,

Follow this link to verify your email address.

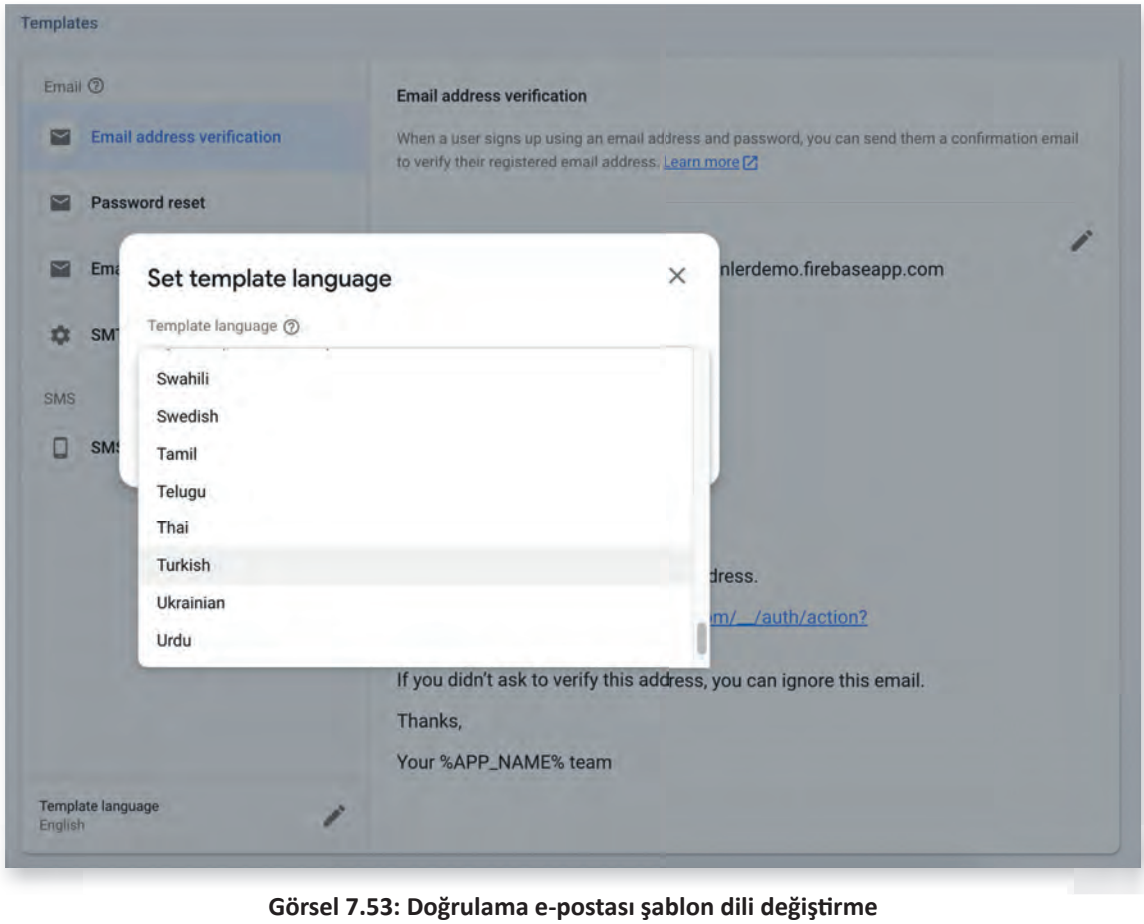
https://fir-urunler.firebaseapp.com/__/auth/action?...

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your project-156168565411 team

Doğrulama e-postası varsayılan olarak İngilizce gönderilir. Authentication modülünün **Templates** bölümü açılarak doğrulama e-postalarının şablonları görüntülenir. Alt kısımda bulunan **Template Language** seçeceği ile Görsel 7.53'teki gibi istenirse doğrulama e-postası Türkçe olarak da ayarlanabilir.



Görsel 7.53: Doğrulama e-postası şablon dili değiştirme

Şablon dili Türkçe yapıldıktan sonra doğrulama e-postaları şu şekilde gönderilir:

Sayın Kullanıcımız,

E-posta adresinizi doğrulamak için bu bağlantıyı tıklayın.

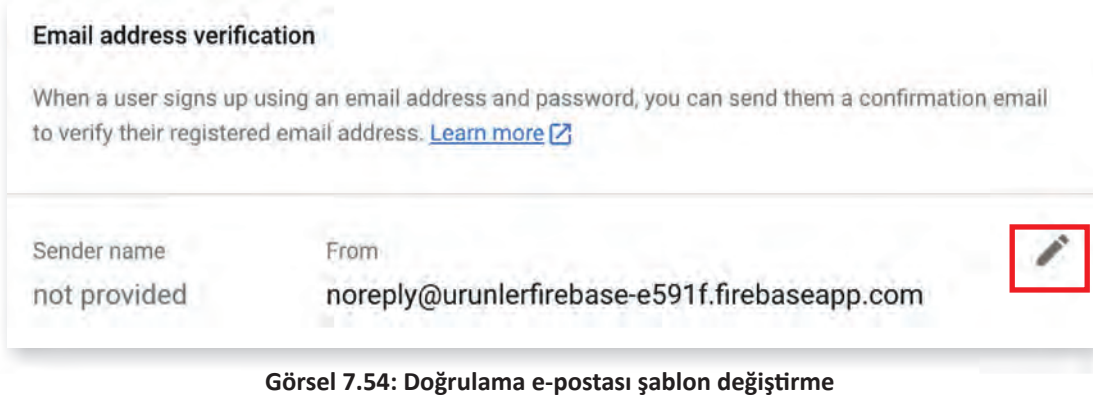
https://fir-urunler.firebaseapp.com/__/auth/action?...

Bu adresi doğrulamayı siz talep etmediyseniz bu e-postayı yoksayabilirsiniz.

Teşekkürler,

project-156168565411 destek ekibi

Doğrulama e-postasının şablonu değiştirilebilir. Görsel 7.54'teki düzenleme buttonuna basılarak şablon tamamen özelleştirilebilir.



Görsel 7.54: Doğrulama e-postası şablon değiştirme

Gerekli değişikler yapıldıktan sonra Save buttonuna basılarak şablon düzenlenir. Şifre sıfırlama ve e-posta değiştirme şablonları da aynı şekilde değiştirilir.

7.3.6. Mobil Uygulama Geliştirme Ortamında Uzak Veri Tabanıyla Çalışmak

Mobil uygulama geliştirme ortamında Firebase veri tabanını kullanmak için çeşitli Firebase kütüphanelerinin yüklenmesi gereklidir. Firebase kütüphanelerinin tüm listesine https://firebase.google.com/docs/android/setup#available-libraries adresinden ulaşılabilir. Mobil uygulama geliştiriciler bu listeden ihtiyacı olan kütüphaneyi uygulamaya ekleyerek kullanabilir.

Temel bir Firebase veri tabanını kullanmak için mobil uygulamaya Cloud Firestore, Cloud Storage ve Authentication kütüphanelerinin eklenmesi yeterlidir. Mobil uygulama geliştirici, ihtiyacına göre istediği kütüphaneyi uygulamaya ekleyebilir. Gerekli kütüphaneler şu kodlar kullanılarak build.gradle dosyasına eklenir:

```
implementation 'com.google.firebase:firebase-analytics'
implementation platform('com.google.firebase:firebase-bom:30.1.0')
implementation 'com.google.firebase:firebase-auth'
implementation 'com.google.firebase:firebase-firestore'
implementation 'com.google.firebase:firebase-storage'
```

7.3.6.1. Mobil Uygulama Geliştirme Ortamında Kullanıcı İşlemleri

Mobil uygulama geliştirme ortamında Firebase veri tabanı kullanıcı işlemlerinin tamamı **FirebaseAuth** nesnesi ile yapılır.

Firebase nesnelerinin tümü asenkron olarak çalışır. Mobil uygulama geliştirme ortamında ise yazılan tüm kodlar senkron olarak çalışır. Bir başka deyişle satır sırasına göre önce yazılan komut bitmeden sonraki komut çalışmaz. Herhangi bir kod asenkron çalıştığında ise diğer kodların beklemesine gerek yoktur. Sonuç ne zaman gelirse ondan sonra veri kullanılabilir. Uzak veri tabanları internette çalıştığı için verinin sunucudan tekrar cihaza getirilmesi zaman alır. Veri getirme işlemleri senkron olarak yapılırsa sonuç daha gelmeden bir sonraki kod çalışmaz ve sonuç görünmez veya bir hatanın çıkmasına yol açar.

7.3.6.2. Mobil Uygulama Ortamında Kullanıcı Kaydetmek

FirebaseAuth nesnesi ile yeni bir kullanıcı **createUserWithEmailAndPassword** metodu kullanılarak oluşturulur. Metot, e-posta adresi ve şifre olmak üzere iki parola alır. İşlem sonucu asenkron çalışacağı için işlemin başarılı olması durumunda **onSuccess** olayı çalışır, işlemin başarısız olması durumunda ise **onFailure** olayı aktif olur. Örneğin "test@test.com" ve "123456" şifresine sahip bir kullanıcı kaydı şu şekilde yapılır:

```
FirebaseAuth.getInstance()
        .createUserWithEmailAndPassword("test@test.com","123456")
        .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
            @Override
            public void onSuccess(AuthResult authResult) {
                Toast.makeText(MainActivity.this, "Kullanıcı oluşturuldu",
                   Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText(MainActivity.this, "Kullanıcı oluşturulamadı",
         Toast.LENGTH_SHORT).show();
    }
});
```

7.3.6.3. Mobil Uygulama Geliştirme Ortamında Kullanıcı Giriş İşlemleri

FirebaseAuth nesnesi ile kullanıcı giriş işlemi **signInWithEmailAndPassword** metodu kullanılarak yapılır. Metot, e-posta adresi ve şifre olmak üzere iki tane parametre alır. İşlem sonucu asenkron olarak gelir. Sonuç başarılı olursa onSuccess olayı çalışır, sonuç başarısız olursa onFailure olayı çalışır. "test@test.com" e-posta adresi ve "123456" şifresine sahip bir hesabın giriş işlemleri şu şekilde yapılır:

```
FirebaseAuth.getInstance()
        .signInWithEmailAndPassword("test@test.com","123456")
        .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
            @Override
            public void onSuccess(AuthResult authResult) {
                Toast.makeText(MainActivity.this, "Giriş başarılı",
                 Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText(MainActivity.this, "Giriş başarısız",
         Toast.LENGTH_SHORT).show();
    }
});
```

7.3.6.3. Mobil Uygulama Geliştirme Ortamında Oturum Kapatmak

Mobil uygulama geliştirme ortamında oturum kapatmak için FirebaseAuth nesnesi şu şekilde kullanılır:

```
FirebaseAuth.getInstance().signOut();
```

7.3.6.4. Mobil Uygulama Geliştirme Ortamında Doğrulama İletisi Göndermek

Mobil uygulama geliştirme ortamında doğrulama e-postası gönderebilmek için kullanıcının oturum açması gerekir. FirebaseAuth nesnesinin getCurrentuser metodu, oturum açmış kullanıcıyı verir. Oturum açmış kullanıcıya doğrulama e-postası şu şekilde gönderilir:

```
FirebaseAuth.getInstance()
        .getCurrentUser()
        .sendEmailVerification()
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                Toast.makeText(MainActivity.this, "Doğrulama e-postası gönderildi",
                 Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
                Toast.makeText(MainActivity.this, "Doğrulama e-postası gönderilemedi",
                 Toast.LENGTH_SHORT).show();
    }
});
```

7.3.6.5. Mobil Uygulama Geliştirme Ortamında Şifre Sıfırlamak

Kullanıcılar belli bir süre sonra şifresini unutabilir. Uygulama içinde kullanıcıya şifre hatırlatması için birtakım seçenekler mutlaka verilmelidir. Mobil uygulama geliştirme ortamında Firebase şifre sıfırlama işlemleri şu şekilde yapılır:

```
FirebaseAuth.getInstance()
        .sendPasswordResetEmail("test@test.com")
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                Toast.makeText(MainActivity.this,
                  "Şifre değiştirme e-postası gönderildi",
                  Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
```

```
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText(MainActivity.this,
         "Şifre değiştirme e-postası gönderilemedi",
         Toast.LENGTH_SHORT).show();
    }
});
```

7.3.6.6. Mobil Uygulama Geliştirme Ortamında Şifre Değiştirmek

Mobil uygulama geliştirme ortamında kullanıcıların şifrelerinin değiştirilmesi işlemi, FirebaseAuth nesnesinin **updatePassword** metodu ile yapılır. Mobil uygulama geliştirme ortamında kullanıcılar şifrelerini şu şekilde değiştirebilir:

```
FirebaseAuth.getInstance()
        .getCurrentUser()
        .updatePassword("Gizlibilgi")
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                // İşlem başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // İşlem başarısız
    }
});
```

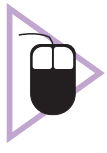
7.3.6.7. Mobil Uygulama Geliştirme Ortamında Kullanıcı Silmek

Mobil uygulama geliştirme ortamında kullanıcı silmek için FirebaseAuth nesnesinin **delete** metodu kullanılır. Bir kullanıcının silinebilmesi için yakın zamanda oturum açmış olması gerekir. Kullanıcı yakın zamanda oturum açmamışsa silme işlemi başarısız olur. Kullanıcı silme işlemi şu şekilde yapılır:

```
FirebaseAuth.getInstance()
        .getCurrentUser()
        .delete()
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                // İşlem başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // İşlem başarısız
    }
});
```

Hata alınırsa kullanıcının yeniden oturum açması sağlanmalıdır. Örneğin "test@test.com" e-postasına sahip ve şifresi "123456" olan kullanıcının yeniden oturum açma işlemi şu şekilde yapılır:

```
AuthCredential authCredential= EmailAuthProvider
        .getCredential("test@test.com","123456");
FirebaseAuth.getInstance().getCurrentUser()
        .reauthenticate(authCredential)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                // İşlem başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // İşlem başarısız
    }
});
```



8. UYGULAMA: İşlem adımlarına göre yedinci uygulama ile bağlantı kurulan projeyi açarak bir kullanıcı giriş formu oluşturunuz. Kullanıcı kaydı yaparak giriş işlemi sağlayınız. Giriş işlemi başarılı ise sonraki Activity'nin açılmasını sağlayan uygulamayı hazırlayınız.

1. Adım: Yedinci uygulama ile yapılan projeyi açınız. Kaydedilmiş bir uygulama yoksa tüm adımları tekrarlayarak yedinci uygulamayı gerçekleştiriniz.

2. Adım: build.gradle dosyasını açarak şu kodları **dependencies** bölümüne ekleyiniz:

```
implementation 'com.google.firebase:firebase-analytics'
implementation platform('com.google.firebase:firebase-bom:30.1.0')
implementation 'com.google.firebase:firebase-auth'
implementation 'com.google.firebase:firebase-firestore'
implementation 'com.google.firebase:firebase-storage'
implementation 'com.squareup.picasso:picasso:2.8'
```

3. Adım: viewBinding özelliğini aktifleştirmek için build.gradle dosyasına şu kodları ekleyiniz:

```
buildFeatures{
    viewBinding true
}
```

4. Adım: res>values>strings.xml dosyasını açarak şu şekilde değiştiriniz:

```
<resources>
    <string name="app_name">Ürünler uygulaması</string>
</resources>
```

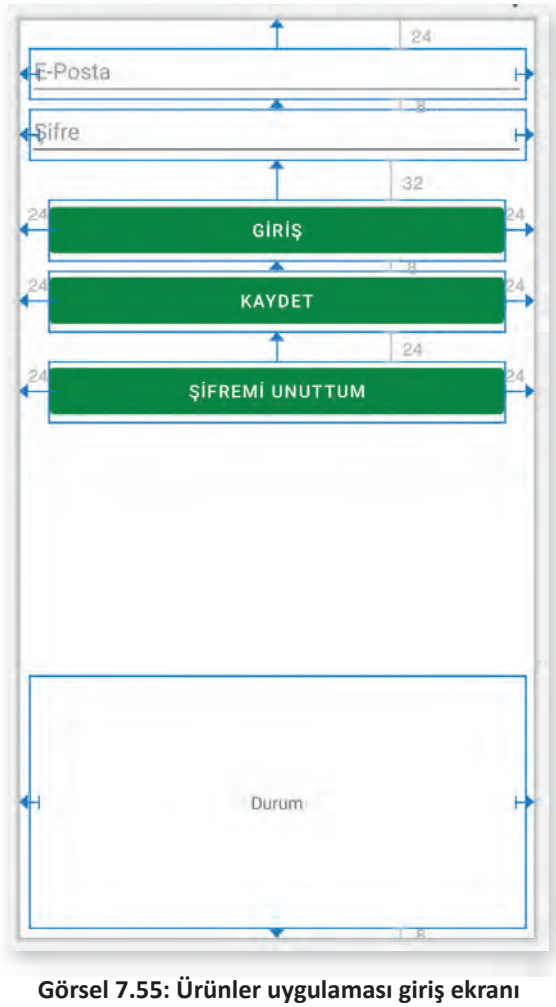
5. Adım: res>values>colors.xml dosyasını açarak şu şekilde değiştiriniz:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="lime">#d4e157</color>
    <color name="darkgreen">#2e7d32</color>
    <color name="lightgreen">#64dd17</color>
    <color name="orange">#ff8f00</color>
    <color name="lightred">#f50057</color>
</resources>
```

6. Adım: res>values>themes>themes.xml dosyasını açarak şu şekilde değiştiriniz:

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.Urunlerfirebase" parent="Theme.MaterialComponents.Day-
Night.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/darkgreen</item>

<item name="colorPrimaryVariant">@color/lime</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPri-
maryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

7. Adım: res>layout>activity_maim.xml dosyasını açarak Görsel 7.55'teki gibi tasarlayınız.



Görsel 7.55: Ürünler uygulaması giriş ekranı

8. Adım: EditTextlerin id bilgilerini editEposta ve editSifre veriniz. Buttonlar için btnGiris, btnKayit ve btnSifremiUnuttum id bilgilerini veriniz. Alt tarafta bulunan TextView için ise txtDurum id bilgisini veriniz.

9. Adım: MainActivity.java dosyasını açarak viewBinding özelliğini şu şekilde aktifleştiriniz:

```
ActivityMainBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding=ActivityMainBinding.inflate(getLayoutInflater());
    View view=binding.getRoot();
    setContentView(view);
}
```

UYARI: View nesnesini android.view.View; kütüphanesinden import ediniz.

10. Adım: btnKayit buttonuna bir onClickListener ekleyip onClick olayını şu şekilde yazınız:

```
public void onClick(View view) {
    String eposta=binding.editEposta.getText().toString().trim();
    String sifre=binding.editSifre.getText().toString().trim();
    FirebaseAuth.getInstance()
            .createUserWithEmailAndPassword(eposta,sifre)
            .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                @Override
                public void onSuccess(AuthResult authResult) {
                    binding.txtDurum.setTextColor(Color.parseColor("#2e7d32"));
                    binding.txtDurum.setText("Kullanıcı oluşturuldu");
                }
            }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            binding.txtDurum.setTextColor(Color.parseColor("#ff1744"));
            binding.txtDurum.setText("Kullanıcı oluşturulamadı\n"+
                    e.getLocalizedMessage());
        }
    });
}
```

11. Adım: btnGiris buttonuna bir onClickListener ekleyip onClick olayını şu şekilde yazınız:

```
public void onClick(View view) {
    FirebaseAuth firebaseAuth=FirebaseAuth.getInstance();
    String eposta=binding.editEposta.getText().toString().trim();
    String sifre=binding.editSifre.getText().toString().trim();
    if(eposta.equals("")||sifre.equals("")){
        binding.txtDurum.setTextColor(Color.parseColor("#ff1744"));
        binding.txtDurum.setText("E-posta veya Şifre boş olamaz");
    }else{
        firebaseAuth
                .signInWithEmailAndPassword(eposta,sifre)
                .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                    @Override
                    public void onSuccess(AuthResult authResult) {
                      binding.txtDurum.setTextColor(Color.parseColor("#2e7d32"));
                      binding.txtDurum.setText("Giriş başarılı");
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                binding.txtDurum.setTextColor(Color.parseColor("#ff1744"));
                binding.txtDurum.setText("Giriş başarısız\n"+
                        e.getLocalizedMessage());
            }
    });
  }
}
```

12. Adım: btnSifremiUnuttum buttonuna bir onClickListener ekleyip onClick olayını şu şekilde yazınız:

```
public void onClick(View view) {
    String eposta=binding.editEposta.getText().toString().trim();
    FirebaseAuth.getInstance()
            .sendPasswordResetEmail(eposta)
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    binding.txtDurum.setTextColor(Color.parseColor("#2e7d32"));
                    binding.txtDurum.setText("Hatırlatma e-postası gönderildi");
                }
            }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            binding.txtDurum.setTextColor(Color.parseColor("#ff1744"));
            binding.txtDurum.setText("Hatırlatma e-postası gönderilemedi\n"+
                    e.getLocalizedMessage());
        }
    });
}
```

13. Adım: Uygulamayı çalıştırınız ve geçerli bir e-posta hesabı ile kayıt yapınız.

14. Adım: Uygulama çalışırken Firebase konsolunu açınız ve kullanıcı kaydının yapılıp yapılmadığını kontrol ediniz.

15. Adım: Uygulamayı kapatıp tekrar açarak kullanıcı adını, şifresini yanlış veya eksik yazarak gelen hata mesajlarını Görsel 7.56'daki gibi kontrol ediniz.

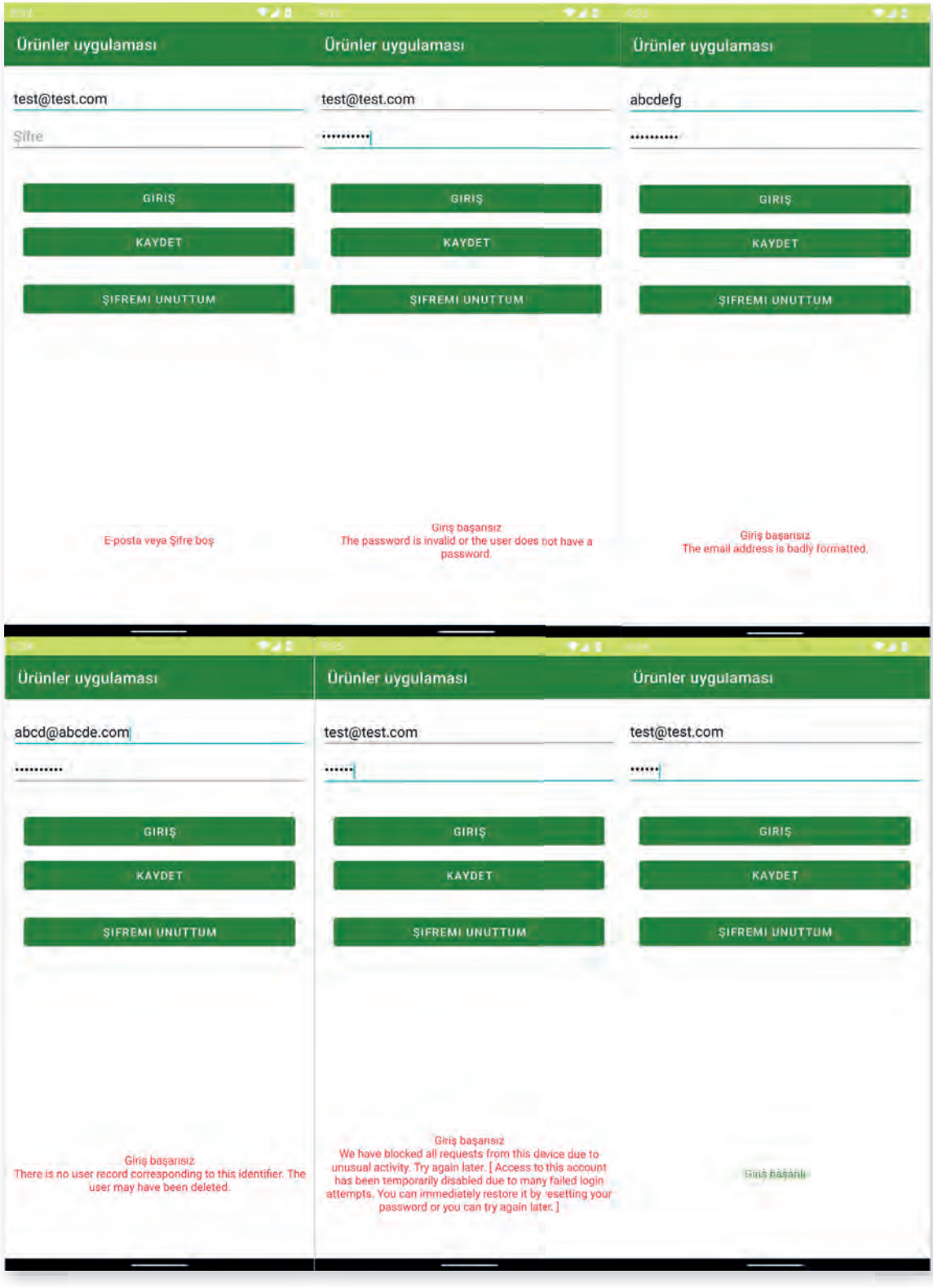
16. Adım: Uygulamayı tekrar açınız. Yeni bir Empty Activity ekleyiniz. Activity'nin adını UrunListele veriniz.

17. Adım: UrunListele.java dosyasını açıp viewBinding işlemleri için şu kodları yazınız:

```
ActivityUrunListeleBinding binding =
        ActivityUrunListeleBinding.
        inflate(getLayoutInflater());
       View view=binding.getRoot();
        setContentView(view);
```

18. Adım: MainActivity.java dosyasını açıp onCreate olayına şu kodları ekleyiniz:

```
if(FirebaseAuth.getInstance().getCurrentUser()!=null){
    Intent intent=new Intent(this,UrunListele.class);
    startActivity(intent);
}
```



Görsel 7.56: Yetkilendirme giriş hata mesajları

18. Adım: MainActivity.java dosyasını açıp onCreate olayına şu kodları ekleyiniz:

```
if(FirebaseAuth.getInstance().getCurrentUser()!=null){
    Intent intent=new Intent(this,UrunListele.class);
    startActivity(intent);
}
```

On sekizinci adım ile eklenen kod sayesinde kullanıcı bir defa oturum açtıktan sonra uygulamayı kapatıp açtığında otomatik olarak giriş yapması sağlanır. Uygulama her açıldığında kullanıcı adının ve şifresinin istenmesi tercih edilmeyen bir durumdur.

19. Adım: MainActivity.java dosyasında sisteme giriş yapılan kodları bulunuz. onSucces olayını şu şekilde değiştiriniz:

```
public void onSuccess(AuthResult authResult) {
    Intent intent=new Intent(MainActivity.this,UrunListele.class);
    startActivity(intent);
}
```

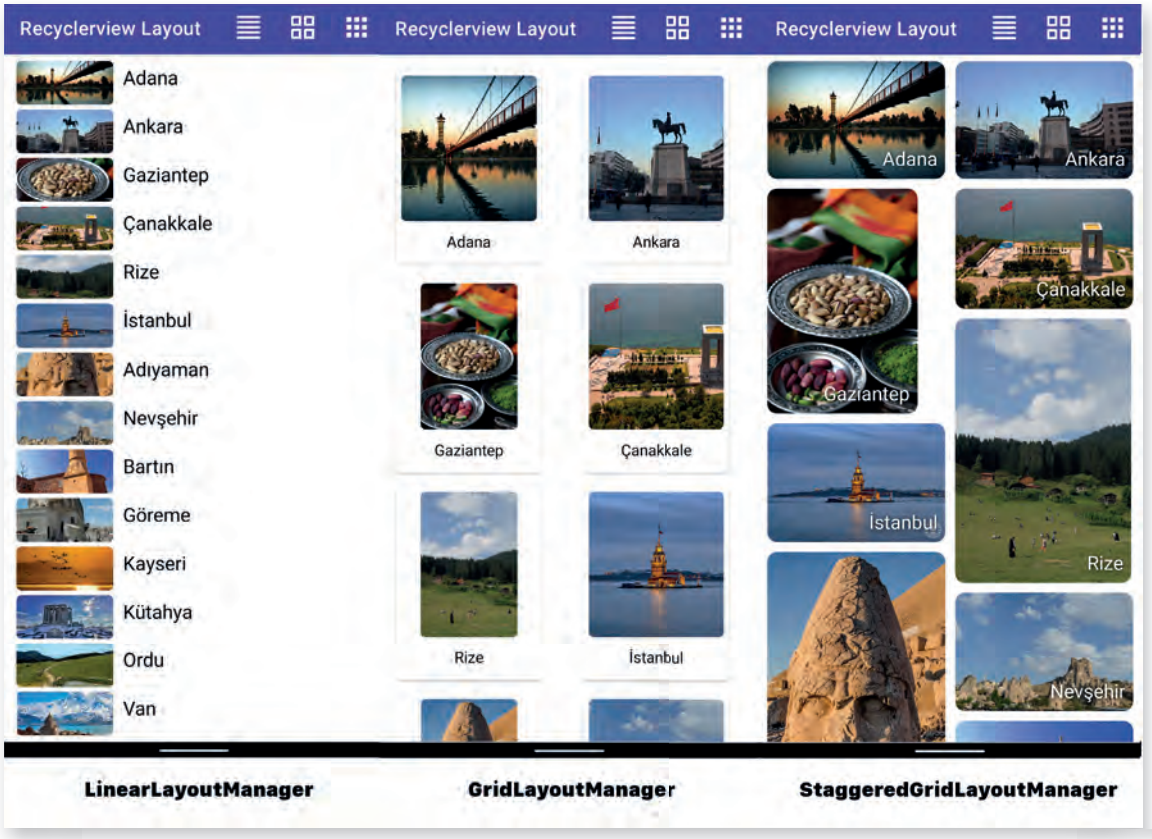
7.3.7. Mobil Uygulama Geliştirme Ortamında RecyclerView Nesnesiyle Çalışmak

RecyclerView, mobil uygulama geliştirme ortamında bulunan en gelişmiş listeleme nesnesidir. Daha önce kullanılan ListView nesnesinden çok daha hızlı çalışır. RecyclerView veri bağdaştırıcısına ne kadar veri gönderilirse gönderilsin sadece ekrana sığacak kadar veri gösterilir. Örneğin cihazın çözünürlüğüne ve ekran boyutuna bağlı olarak RecyclerView on satır veri gösterebilirse listeden sadece sıradaki on kayıt alınıp gösterilir. Bundan dolayı hem bellek tasarrufu sağlanır hem de çok büyük verilerde herhangi bir yavaşlama olmaz.

RecyclerView içinde veriler gösterilirken bir bağdaştırıcı kullanmak gereklidir. Nesne grubu bir XML dosyasında tanımlanarak bağdaştırıcı sayesinde RecyclerView içinde gösterilir. ListView nesnesinden farklı RecyclerView için bir LayoutManager tanımlanmalıdır. LayoutManager nesnesine göre veriler istenirse satır satır veya bir ızgara şeklinde gösterilir. RecyclerView için tanımlanabilecek LayoutManager nesneleri şunlardır:

- **LinearLayoutManager:** Verilerin satır satır görüntülenmesini sağlar.
- **GridLayoutManager:** Verilerin iki sütunlu bir ızgara şeklinde görüntülenmesini sağlar. Tüm satır ve sütunlar birbirine eşit olacak şekilde listelenir.
- **StaggeredGridLayoutManager:** GridLayoutManager ile benzer çalışır. Burada satır ve sütunların eşit olması zorunluluğu yoktur.

Aynı verilere sahip bir RecyclerView nesnesiyle LayoutManager nesneleri arasındaki farklar Görsel 7.57'de verilmiştir.



Görsel 7.57: Recyclerview LayoutManager görünümleri

7.3.7.1. Mobil Uygulama Geliştirme Ortamında RecyclerView Nesnesi Tanımlamak

Mobil uygulama geliştirme ortamında RecyclerView nesnesiyle verileri göstermek için her bir LayoutManager nesnesine ekran tasarlamak gereklidir. res>layout bölümünde her bir LayoutManager için ayrı ayrı ekran tasarlanır. Daha sonra Recyclerview.Adapter sınıfından türemiş bir nesne ile verilerin tasarlanan ekranlardaki View nesnelerine bağlanması sağlanır.

Temel olarak LinearLayoutManager nesnesini kullanan bir RecyclerView nesnesi şu şekilde tanımlanır:

```
adapter=new DataAdapter(Liste,UrunDetay.this,this);
binding.recyclerView.setLayoutManager(new LinearLayoutManager(this));
binding.recyclerView.setAdapter(adapter);
```

GridLayoutManager nesnesi için ikinci parametre olarak sütun sayısı belirtilir. GridLayoutManager nesnesi şu şekilde tanımlanır:

```
binding.recyclerView.setLayoutManager(new GridLayoutManager(this,2));
```

StaggeredGridLayoutManager nesnesi için iki parametre verilir. Birinci parametre sütun sayısıdır. İkinci parametre ile görsellerin dikey veya yatay olarak görünmesi ayarlanır. StaggeredGridLayoutManager nesnesi şu şekilde tanımlanır:

```
binding.recyclerView.setLayoutManager(new StaggeredGridLayoutManager( 2,
                                        StaggeredGridLayoutManager.VERTICAL));
```

7.3.7.2. RecyclerView Adapter Tanımlamak

Recyclerview.Adapter kullanımı, BaseAdapter ve ArrayAdapter kullanımından daha kolaydır. Recyclerview.Adapter metotları şunlardır:

- **onCreateViewHolder():** Bağdaştırıcı bir liste ögesi oluşturacağı zaman ilk olarak bu metot çalışır. onCreateViewHolder metodu yeni bir ViewHolder nesnesi oluşturur. Bağdaştırıcı ile birlikte Inner Class olarak da bir ViewHolder nesnesi oluşturulur. ViewHolder nesnesinde liste ögesinin bağlanacağı tüm View nesneleri tanımlanır.
- **onBindViewHolder():** onCreateViewHolder metodu çalıştıktan sonra oluşturulan ViewHolder nesnesi onBindViewHolder metoduna gönderilir. Bu metotta listedeki veriler View nesnelerine bağlanır.
- **getItemCount():** Veri listesinde kaç tane kayıt olduğu bu metotla tespit edilir.

Temel bir Recyclerview.Adapter şu şekilde tanımlanır:

```
public class DataAdapter extends RecyclerView.Adapter<DataAdapter.ViewHolder> {
    private String[] veriler;
    /*
      Her bir liste ögesinin bağlanacağı View nesnesi ViewHolder içinde
       tanımlanır. Örnekte sadece String türünden veri olduğu için
       sadece bir tane TextView tanımlanmıştır.
     */
    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView textView;
        public ViewHolder(View view) {
            super(view);
            // View nesnesine click olayı yazılacaksa burada yazılabilir.
            textView = (TextView) view.findViewById(R.id.textView);
        }
           // View nesnesine ulaşmak için bir metot tanımlanır.
        public TextView getTextView() {
            return textView;
        }
    }
    /**
      Yapılandırıcı metot ile DataAdapter tanımlandığından Activityden veri
       alınması sağlanır.
     */
    public CustomAdapter(String[] data) {
        veriler = data;
    }
```

```
    // ViewHolder nesnesi oluşturulur.
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        // Daha önceden tanımlanan View nesnesi oluşturulur.
        View view = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.text_row_item, viewGroup, false);
        return new ViewHolder(view);
    }
    // View nesnesine veriler bağlanır.
    @Override
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {
        viewHolder.getTextView().setText(veriler s[position]);
    }
    // Veri listesinin büyüklüğü bulunur.
    @Override
    public int getItemCount() {
        return veriler.length;
    }
}
```

7.3.8. Uzak Veri Tabanından Veri Getirmek

Firestore veri tabanında tek seferlik veya canlı olmak üzere iki türlü veri okunabilir. Canlı veri okuma, dokümanlarda herhangi bir değişiklik olduğunda tüm verileri getirir. Bundan dolayı uzak veri tabanı ile çok fazla iletişim kurulmasını gerektirir. Firebase veri tabanının verdiği ücretsiz kullanım kotasının çok hızlı tükenmesine neden olur. Tek seferlik veri okuma da ise veriler gerektiği zaman uzak veri tabanından getirilir.

7.3.8.1. get() Metoduyla Veri Getirmek

Tek seferlik veri okuma şu şekilde yapılır:

```
FirebaseFirestore.getInstance()
        .collection("urunler")
        .get()
        .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
            @Override
            public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                List<DocumentSnapshot>
                  snapshotList=queryDocumentSnapshots.getDocuments();
                for (DocumentSnapshot document:snapshotList){
                    Urun urun=new Urun(document.getString("user"),
                            document.getString("urunadi"),
                            document.getDouble("fiyat"),
                            document.getLong("adet"),
                            document.getId(),
```

```
                        null);
                Liste.add(urun);
            }
            adapter.notifyDataSetChanged();
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(UrunDetay.this, "Veriler Alınamadı\n"
                    + e.getLocalizedMessage(), Toast.LENGTH_LONG).show();
        }
    });
```

7.3.8.2. addSnapshotLisnener() Metoduyla Veri Getirmek

Canlı veri işlemi, addSnapshotListener metodunun tanımlanması ile şu şekilde gerçekleştirilir:

```
FirebaseFirestore.getInstance()
        .collection("urunler")
        .addSnapshotListener(new EventListener<QuerySnapshot>() {
            @Override
            public void onEvent(@Nullable QuerySnapshot value, @Nullable
              FirebaseFirestoreException error) {
                for (DocumentSnapshot document:value.getDocuments()) {
                    Urun urun=new Urun(document.getString("user"),
                            document.getString("urunadi"),
                            document.getDouble("fiyat"),
                            document.getLong("adet"),
                            document.getId(),
                            null);
                    Liste.add(urun);
                }
                adapter.notifyDataSetChanged();
            }
        })
         .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(UrunDetay.this, "Veriler Alınamadı\n"
                        + e.getLocalizedMessage(), Toast.LENGTH_LONG).show();
            }
        });
```

7.3.8.3. Verileri Sıralamak

Firestore veri tabanında veriler orderBy metodu ile sıralanır. Herhangi bir sıralama şekli belirtilmezse varsayılan olarak küçükten büyüğe doğru veriler sıralanır. Veriler **"urunadi"** alanına göre küçükten büyüğe şu şekilde sıralanır:

```
FirebaseFirestore.getInstance()
        .collection("urunler")
        .orderBy("urunadi")
        .get()
         ...
```

Sıralama yönü belirtilirse veriler şu şekilde sıralanır:

```
FirebaseFirestore.getInstance()
        .collection("urunler")
        .orderBy("urunadi", Query.Direction.ASCENDING)
        .get()
         ...
```

Büyükten küçüğe doğru veri sıralama şu şekilde yapılır:

```
FirebaseFirestore.getInstance()
        .collection("urunler")
        .orderBy("urunadi", Query.Direction.DESCENDING)
        .get()
         ...
```

Birden fazla alanda veri sıralaması şu şekilde yapılır:

```
FirebaseFirestore.getInstance()
        .collection("urunler")
        .orderBy("urunadi")
        .orderBy("fiyat")
        .get()
         ...
```

7.3.8.4. Verileri Limitlemek

Firestore veri tabanında verileri limitlemek için limit() metodu kullanılır. Firestore veri tabanındaki ilk üç kaydın alınması şu şekilde yapılır:

```
FirebaseFirestore.getInstance()
        .collection("urunler")
         .limit(3)
        .orderBy("urunadi", Query.Direction.ASCENDING)
        .get()
         ...
```

7.3.8.5. Verileri Filtrelemek

Verileri sunucu üzerinde filtrelemek için where ile başlayan sorgulama metotları kullanılır (Tablo 7.10).

Tablo 7.10: Firestore Sorgulama Metotları

Firestore Metodu	Metodun Matematik Karşılığı	Açıklama
whereEqualTo	=	Verilen değere eşit olan kayıtlar getirilir.
whereNotEqualTo	!=	Verilen değere eşit olmayan kayıtlar getirilir.
whereGreaterThan	>	Verilen değerden büyük kayıtlar getirilir.
whereGreaterThanOrEqualTo	>=	Verilen değere eşit veya büyük kayıtlar getirilir.
whereLessThan	<	Verilen değerden küçük kayıtlat getirilir.
whereLessThanOrEqualTo	<=	Verilen değere eşit veya küçük kayıtlar getirilir.
whereArrayContains		Dizi içinde bir kayıtla eşleşen sadece bir kayıt getirilir.
whereIn		Dizi içindeki herhangi bir kayıtla eşleşen kayıtlar getirilir.
whereNotIn		Dizi içindeki herhangi bir kayıtla eşleşmeyen kayıtlar getirilir.

Kullanıcının kendi eklediği ürünler uzak veri tabanından şu şekilde getirilir:

```
FirebaseUser user=FirebaseAuth.getInstance().getCurrentUser();
FirebaseFirestore.getInstance()
        .collection("urunler")
        .whereEqualTo("user",user.getUid())
        .get()
         . . .
```

Fiyatı 100'den küçük olan ürünler uzak veri tabanından şu şekilde getirilir:

```
FirebaseUser user=FirebaseAuth.getInstance().getCurrentUser();
FirebaseFirestore.getInstance()
        .collection("urunler")
        .whereLessThan("fiyat",100)
        .get()
         . . .
```

Bilgisayar olmayan ürünler uzak veri tabanından şu şekilde getirilir:

```
FirebaseUser user=FirebaseAuth.getInstance().getCurrentUser();
FirebaseFirestore.getInstance()
        .collection("urunler")
        .whereNotIn("urunadi",Arrays.asList("Bilgisayar"))
        .get()
         . . .
```

Sadece stokta bulunan ürünler uzak veri tabanından şu şekilde getirilir:

```
FirebaseUser user=FirebaseAuth.getInstance().getCurrentUser();
FirebaseFirestore.getInstance()
        .collection("urunler")
        .whereNotEqualTo("adet",0)
        .get()
         . . .
```

Sorgular bileşik olarak da kullanılabilir. Bileşik sorgu sınırlamaları şunlardır:

- Dizi kullanılarak yapılan whereIn, whereNotIn gibi ifadeler en fazla 10 zincir olacak şekilde yazılabilir.
- <, <=, >, >= sorguları sadece kendi aralarında zincir olarak kullanılabilir.
- = ve != sorguları da kendi aralarında zincir olarak kullanılır.

Sadece stokta bulunan ürünlerden fiyatı 1.000'den küçük olanlar uzak veri tabanından şu şekilde getirilir:

```
FirebaseUser user=FirebaseAuth.getInstance().getCurrentUser();
FirebaseFirestore.getInstance()
        .collection("urunler")
        .whereGreaterThan("adet",0)
        .whereLessThan("fiyat",1000)
        .get()
         . . .
```

7.3.8.6. SearchView Nesnesi Kullanarak Filtreleme Yapmak

Her ne kadar sunucu üzerinde filtreleme yapmak çok büyük kolaylık sağlasa da uygulama sonuçta internet trafiğini kullanır. Cihaza alınmış veriler üzerinde filtreleme işlemi yapmak, internet trafiğini olumlu yönde etkiler ve daha hızlı sonuç verir. SearchView metin kutusu kullanılarak basit bir filtreleme işlemi şu şekilde yapılır:

```
binding.editArama.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        return false;
    }
```

```
    @Override
    public boolean onQueryTextChange(String newText) {
        ArrayList<Urun> yedek=new ArrayList<>();
        for(Urun urun:Liste){
            if(urun.getUrunadi().toLowerCase().contains(newText.toLowerCase()))
{
                yedek.add(urun);
            }
            adapter.setData(yedek);
        }
        return true;
    }
});
```

DataAdapter sınıfına şu kodlar eklenerek işlem tamamlanır:

```
public void setData(ArrayList<Urun> liste){
    Liste=liste;
    notifyDataSetChanged();
}
```

9. UYGULAMA: İşlem adımlarına göre sekizinci uygulama ile giriş işlemleri tamamlanmış uygulamaya kayıt listeleme işlemlerini ekleyiniz.

1. Adım: Sekizinci uygulama ile yapılan projeyi açınız. Kaydedilmiş bir uygulama yoksa tüm adımları tekrarlayarak sekizinci uygulamayı gerçekleştiriniz.

2. Adım: Uygulamaya Urun isimli bir sınıf ekleyerek veri modelinizi şu şekilde tanımlayınız:

```
public class Urun implements Serializable {
    public String kullaniciNo;
    public String eposta;
    public String urunadi;
    public long adet;
    public double fiyat;
    public String resim;
    public String documentId;
    public Urun(String kullaniciNo, String eposta,
                String urunadi, long adet,
                double fiyat, String resim) {
        this.kullaniciNo = kullaniciNo;
        this.eposta = eposta;
        this.urunadi = urunadi;
        this.adet = adet;
        this.fiyat = fiyat;
        this.resim=resim;
```

```
    }
    public Urun(){}
    public String getKullaniciNo() {
        return kullaniciNo;
    }
    public String getEposta() {
        return eposta;
    }
    public String getUrunadi() {
        return urunadi;
    }
    public long getAdet() {
        return adet;
    }
    public double getFiyat() {
        return fiyat;
    }
    public String getResim() {
        return resim;
    }
}
```

3. Adım: Yeni bir sınıf oluşturunuz ve sınıfın adını DataAdaptor.class veriniz.

4. Adım: DataAdaptor sınıfını şu şekilde tanımlayınız:

```
public class DataAdaptor extends RecyclerView.Adapter<DataAdaptor.ViewHolder> {
    ArrayList<Urun> liste;
    Context context;
    public DataAdaptor(Context context,ArrayList<Urun> liste) {
        this.context=context;
        this.liste = liste;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
                                                        int viewType) {
        RecyclerviewSatirBinding binding =
         RecyclerviewSatirBinding.
         inflate(LayoutInflater.from(parent.getContext()),
                parent,false);
        return new ViewHolder(binding);
    }
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Urun urun=liste.get(position);
        if(urun.getResim()==null){
```

```
            holder.binding.satirResim.setImageResource(R.drawable.bos_resim);
        }else{
            Picasso.get().load(urun.resim).into(holder.binding.satirResim);
        }
        holder.binding.satirUrunadi.setText(urun.getUrunadi());
        holder.binding.satirAdet.setText(urun.getAdet()+"");
        holder.binding.satirFiyat.setText(urun.getFiyat()+"");
        holder.binding.satirEposta.setText(urun.getEposta());
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(context,UrunDetay.class);
                intent.putExtra("urun",urun);
                context.startActivity(intent);
            }
        });
    }
    @Override
    public int getItemCount() {
        return liste.size();
    }
    public class ViewHolder extends RecyclerView.ViewHolder {
        RecyclerviewSatirBinding binding;
        public ViewHolder(@NonNull RecyclerviewSatirBinding binding) {
            super(binding.getRoot());
            this.binding=binding;
        }
    }
    public void setData(ArrayList<Urun> yliste){
        liste=yliste;
        notifyDataSetChanged();
    }
}
```

5. Adım: Projeye yeni bir Empty Activity ekleyiniz ve Activity adını **UrunDetay** veriniz.

6. Adım: Projeye yeni bir Empty Activity ekleyiniz ve Activity adını **UrunEkle** veriniz.

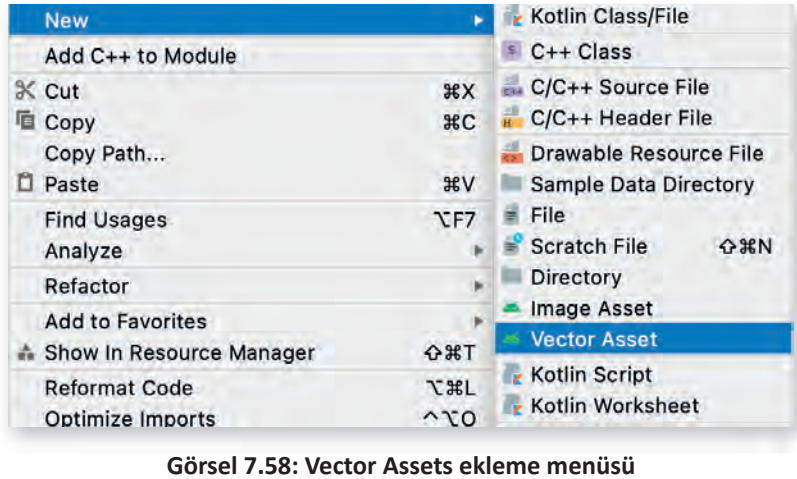
7. Adım: Projeye yeni bir Empty Activity ekleyiniz ve Activity adını **Ayarlar** veriniz.

8. Adım: Manifests dosyasını açıp dosyada şu değişiklikleri yapınız:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<activity
    android:name=".UrunDetay"
    android:parentActivityName=".UrunListele"
    android:exported="false" />
```

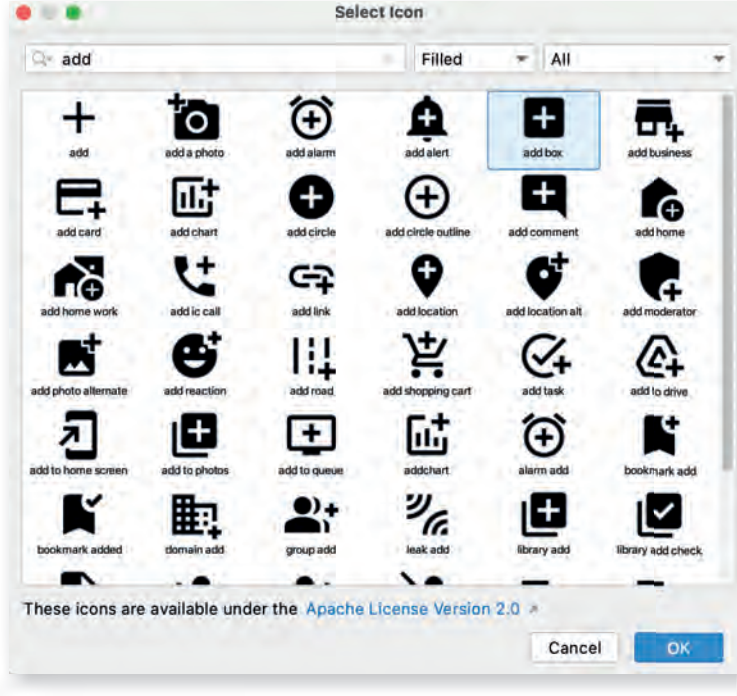
```
<activity
android:name=".UrunEkle"
    android:parentActivityName=".UrunListele"
    android:exported="false" />
<activity
    android:name=".Ayarlar"
    android:parentActivityName=".UrunListele"
    android:exported="false" />
```

9. Adım: res>drawable klasörüne gelerek Görsel 7.58'deki gibi menüyü açınız.



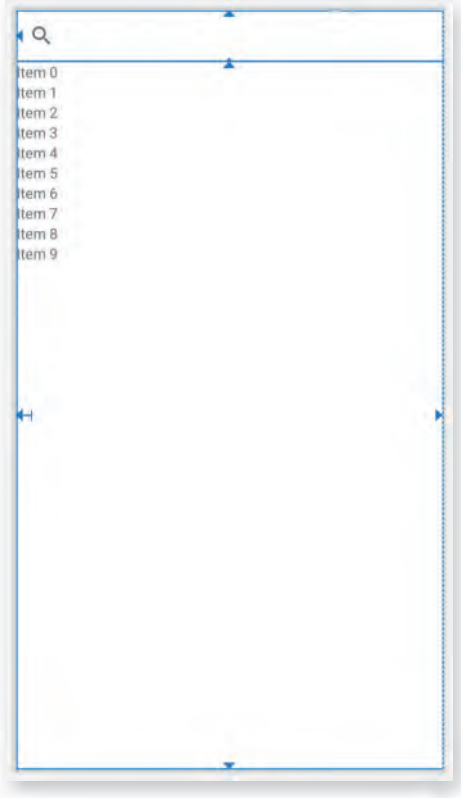
Görsel 7.58: Vector Assets ekleme menüsü

10. Adım: Açılan pencereden Android simgesine tıklayıp, Select Icon penceresinden "Add Box"-yazarak çıkan Görsel 7.59'daki simgeyi seçiniz. Simgenin rengini beyaz olarak ayarlayınız.



Görsel 7.59: Simge seçme penceresi

11. Adım: Simgeyi seçtikten sonra simgenin boyutunu 48x48 olarak ayarlayınız ve dosya ismini urun_ekle_simge olarak kaydediniz (Görsel 7.60).



Görsel 7.60: urun_ekle_xml tasarım ekranı

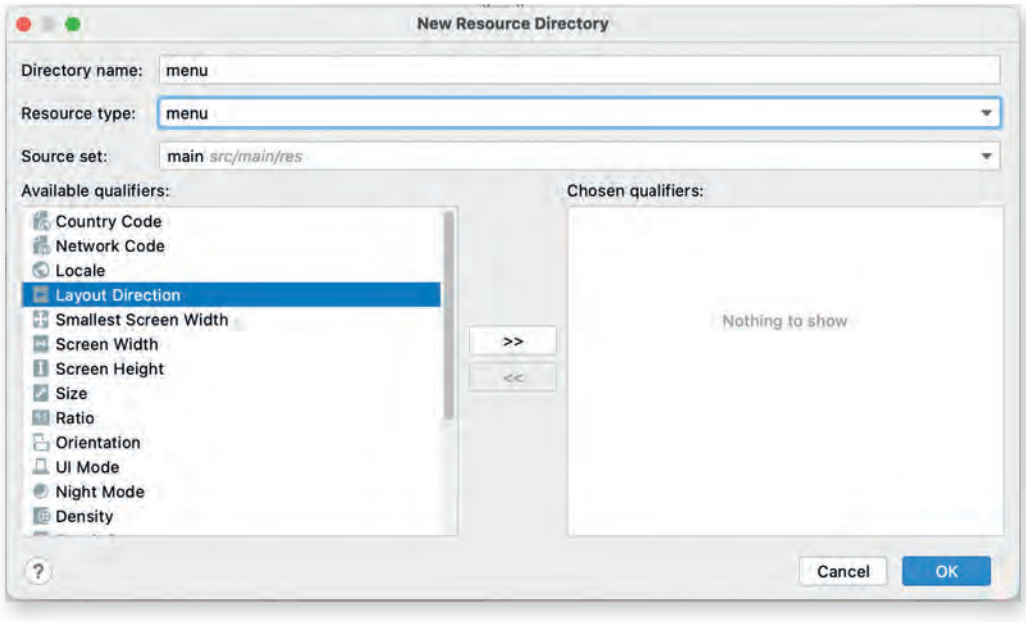
12. Adım: Vector Assets menüsünden Asset Studio'yu açarak simge seçme ekranına geçiniz.

13. Adım: Açılan pencereden **image** yazarak boş resim için kullanılacak simgeyi seçiniz.

14. Adım: Resmi seçtikten sonra resmin rengini Görsel 7.61'deki gibi düzenleyip, **bos_resim** ismini vererek kaydediniz.

Görsel 7.61: Boş resim simgesi

16. Adım: urun_listele.xml dosyasını açarak Görsel 7.62'deki gibi SearchView ve RecyclerView nesnelerini ekleyiniz.



Görsel 7.62: Android Resource Directory penceresi

17. Adım: SearchView id bilgisini **editArama**, RecyclerView nesnesinin id bilgisini **recyclerView** veriniz.

18. Adım: res klasörü üstünde sağ tıklayarak New menüsünden **Android Resource Directory** seçeneğini seçiniz.

19. Adım: Açılan pencereden Görsel 7.62'deki gibi **menu** seçeneğini seçiniz.

20. Adım: menu dizini geldikten sonra bu dizine sağ tıklayarak New menüsünden **Menu Resource File** seçeneğini seçiniz.

21. Adım: Açılan pencereden dosyaya **urunlistemenu** adını veriniz. Dosyayı şu şekilde tanımlayınız:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/menuUrunEkle"
        android:icon="@drawable/urun_ekle_simge"
        app:showAsAction="ifRoom"
        android:title="Ürün Ekle">Ürün Ekle
    </item>
    <item
        android:id="@+id/menuOturumKapat"
        app:showAsAction="never"
        android:title="Oturum kapat"/>
    <item
        android:id="@+id/menuAyarlar"
        app:showAsAction="never"
        android:title="Kullanıcı Ayarları"/>
</menu>
```

22. Adım: UrunListele.java dosyasını açınız ve dosyayı şu şekilde düzenleyiniz:

```
public class UrunListele extends AppCompatActivity {
    ActivityUrunListeleBinding binding;
    ArrayList<Urun> Liste;
    DataAdaptor dataAdaptor;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding=ActivityUrunListeleBinding.inflate(getLayoutInflater());
        View view=binding.getRoot();
        setContentView(view);
        Liste=new ArrayList<>();
        dataAdaptor=new DataAdaptor(this,Liste);
        binding.recyclerView.setLayoutManager(new LinearLayoutManager(this));
        binding.recyclerView.setAdapter(dataAdaptor);
        FirebaseFirestore.getInstance().collection("urunler")
                .get()
                .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
                    @Override
                    public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                        for(DocumentSnapshot document:queryDocumentSnapshots){
                            Urun urun=document.toObject(Urun.class);
                            urun.documentId=document.getId();
                            Liste.add(urun);
                        }
                    dataAdaptor.notifyDataSetChanged();
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(UrunListele.this, "Veriler alınamıyor", Toast.LENGTH_
SHORT).show();
            }
        });
        binding.editArama.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String s) {
                return false;
            }
            @Override
            public boolean onQueryTextChange(String s) {
                ArrayList<Urun> yedek=new ArrayList<>();
                for(Urun urun:Liste){
                    if(urun.getUrunadi().toLowerCase().contains(s.toLowerCase())){
                        yedek.add(urun);
                    }
```

```
                    dataAdaptor.setData(yedek);
                }
                return true;
            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.urunlistemenu,menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if(item.getItemId()==R.id.menuOturumKapat){
            FirebaseAuth.getInstance().signOut();
            Intent intent=new Intent(UrunListele.this,MainActivity.class);
            startActivity(intent);
        }
        if(item.getItemId()==R.id.menuAyarlar){
            Intent intent=new Intent(UrunListele.this,Ayarlar.class);
            startActivity(intent);
        }
        if(item.getItemId()==R.id.menuUrunEkle){
            Intent intent=new Intent(UrunListele.this,UrunEkle.class);
            startActivity(intent);
        }
        return super.onOptionsItemSelected(item);
    }
}
```

23. Adım: Uygulamayı çalıştırınız.

10. UYGULAMA: İşlem adımlarına göre dokuzuncu uygulama ile tamamlanan UrunListele Activity uygulamasının Kullanıcı Ayarları bölümünü yapınız.

1. Adım: Dokuzuncu uygulama ile yapılan projeyi açınız. Kaydedilmiş bir uygulama yoksa tüm adımları tekrarlayarak dokuzuncu uygulamayı gerçekleştiriniz.

2. Adım: activity_ayarlar.xml dosyasını açınız. Kullanıcı Ayarları bölümünün ekranını Görsel 7.63'teki gibi tasarlayınız.

3. Adım: İki tane editText ve iki tane button ekleyiniz. editTextlerin adlarını **editAyarlarSifre** ve **editAyarlarYeniSifre** veriniz.

4. Adım: Buttonların adlarını **btnAyarlarSifreDegistir** ve **btnAyarlarDogrulamaEposta** veriniz.

5. Adım: Buttonların id bilgilerini **btnAyarlarSifreDegistir** ve **btnAyarlarDogrulamaEposta** veriniz.

6. Adım: Ayarlar.java dosyasını açarak dosyayı şu şekilde tanımlayınız:

```
public class Ayarlar extends AppCompatActivity {
    ActivityAyarlarBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding=ActivityAyarlarBinding.inflate(getLayoutInflater());
        View view=binding.getRoot();
        setContentView(view);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        binding.btnAyarlarDogrulamaEposta.setOnClickListener(
              new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                FirebaseAuth.getInstance().getCurrentUser()
                        .sendEmailVerification()
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void unused) {
                                Toast.makeText(Ayarlar.this,
                                    "Doğrulama E-postası gönderildi",
                                Toast.LENGTH_SHORT).show();
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(Ayarlar.this,
                           "Doğrulama E-postası gönderilemedi",
                        Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
        binding.btnAyarlarSifreDegistir.setOnClickListener(
              new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                FirebaseUser user=FirebaseAuth.getInstance().getCurrentUser();
                AuthCredential authCredential= EmailAuthProvider
                        .getCredential(user.getEmail(),
                            binding.editAyarlarSifre.getText().toString());
                user.reauthenticate(authCredential).addOnSuccessListener(
                  new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void unused) {
                    user.updatePassword(
```

```
                        binding.editAyarlarYeniSifre.getText()
                                .toString())
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                                @Override
                                public void onSuccess(Void unused) {
                                    Toast.makeText(Ayarlar.this,
                                            "Kullanıcı şifresi değiştirildi",
                                    Toast.LENGTH_SHORT).show();
                                }
                            }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(Ayarlar.this,
                             "Kullanıcı şifresi değiştirilemedi",
                            Toast.LENGTH_SHORT).show();
                        }
                    });
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(Ayarlar.this,
                     "Tekrar giriş başarısız",
                    Toast.LENGTH_SHORT).show();
                }
            });
        }
    });
 }
}
```

6. Adım: Uygulamayı çalıştırınız. UrunListele bölümündeyken menüden Ayarları açınız ve şifrenizi değiştiriniz.

7. Adım: Menüden Oturumu Kapat seçeneğini seçip uygulamaya yeniden giriş yapınız.

7.3.9. Mobil Uygulama Geliştirme Ortamında Uzak Veri Tabanına Veri Eklemek

Firestore veri tabanına **add** ve **set** metotları kullanılarak veri eklenebilir. Add metodu her ne olursa olsun veriyi dokümana ekler. Set metodu ise veri dokümanda yoksa ekleme yapar, veri dokümanda varsa var olan veriyi değiştirir. Kaydedilmek istenen veriler bir HashMap veya nesne modeli kullanılarak dokümana eklenir.

7.3.9.1. Nesne Modelini Kullanarak Veri Eklemek

Set metodu kullanarak veri eklemek için bir nesne modeli gereklidir. Nesne modelinin boş bir yapılandırıcı metodu mutlaka olmalıdır.

Sehir modeli şu şekilde oluşturulur:

```
public class Sehir {
    public String ad;
    public long nufus;
    public Sehir() {}
    public Sehir(String ad, long nufus) {
        this.ad=ad;
         this.nufus=nufus;
    }
}
```

Sehir modeline göre bir dokümana şu şekilde veri eklenir:

```
FirebaseFirestore.getInstance().collection("sehirler")
        .add(new Sehir("Ankara",3500000))
        .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
                // İşlem Başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
               // İşlem Başarısız
    }
});
```

7.3.9.2. HashMap Kullanarak Veri Eklemek

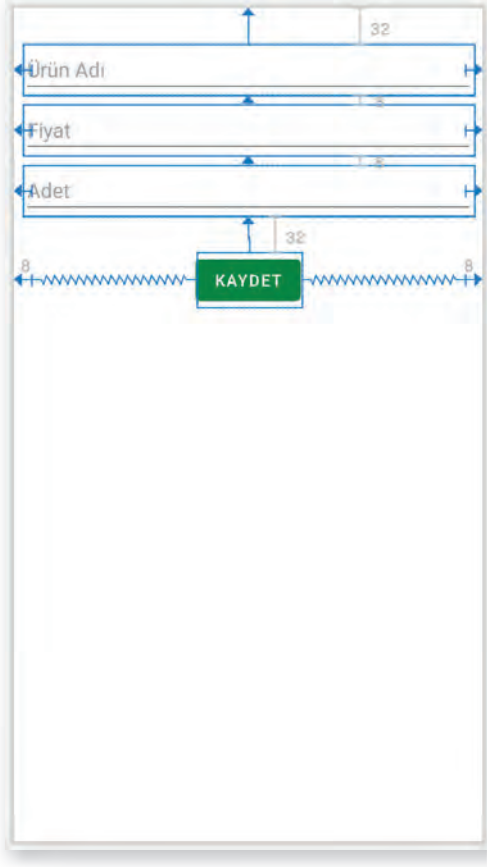
HashMap kullanılarak dokümana şu şekilde veri eklenir:

```
HashMap<String,Object> hashMap=new HashMap<>();
hashMap.put("ad","Ankara");
hashMap.put("Rufus",3500000);
FirebaseFirestore.getInstance().collection("sehirler")
        .add(hashMap)
        .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
                // İşlem Başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
public void onFailure(@NonNull Exception e) {
                // İşlem Başarısız
    }
});
```

11. UYGULAMA: İşlem adımlarına göre onuncu uygulama ile tamamlanan uygulamada veri ekleme bölümünü oluşturunuz.

1. Adım: Onuncu uygulama ile yapılan projeyi açınız. Kaydedilmiş bir uygulama yoksa tüm adımları tekrarlayarak onuncu uygulamayı gerçekleştiriniz.

2. Adım: activity_urun_ekle.xml dosyasını açınız. Activity'nin görünümünü Görsel 7.63'teki gibi tasarlayınız.



Görsel 7.63: UrunEkle Activity tasarım ekranı

3. Adım: Üç adet EditText ve bir button ekleyiniz. EditTextlerin id bilgilerini **editUrunadi, editFiyat** ve **editAdet** veriniz.

4. Adım: Eklenen buttonun adını **btnUrunEkle** veriniz.

5. Adım: UrunEkle.java dosyasını şu şekilde kodlayınız:

```
public class UrunEkle extends AppCompatActivity {
    ActivityUrunEkleBinding binding;
    Urun urun;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding=ActivityUrunEkleBinding.inflate(getLayoutInflater());
```

```
 View view=binding.getRoot();
setContentView(view);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
urun=(Urun) getIntent().getSerializableExtra("urun");
if(urun!=null){
binding.editUrunadi.setText(urun.getUrunadi());
binding.editFiyat.setText(urun.getFiyat()+"");
binding.editAdet.setText(urun.getAdet()+"");
 }
 binding.btnUrunEkle.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
  String urunadi=binding.editUrunadi.getText().toString();
  double fiyat=Double.parseDouble(binding.editFiyat.getText().toString());
  long adet=Long.parseLong(binding.editAdet.getText().toString());
  FirebaseUser kullanici=FirebaseAuth.getInstance().getCurrentUser();
     HashMap<String,Object> hashMap=new HashMap<>();
     hashMap.put("kullaniciNo",kullanici.getUid());
     hashMap.put("eposta",kullanici.getEmail());
     hashMap.put("urunadi",urunadi);
     hashMap.put("fiyat",fiyat);
     hashMap.put("adet",adet);
     hashMap.put("tarih",FieldValue.serverTimestamp());
     hashMap.put("resim",null);
        if(urun!=null){
          FirebaseFirestore.getInstance().collection("urunler")
                  .document(urun.documentId)
                  .update(hashMap)
                  .addOnSuccessListener(new OnSuccessListener<Void>()   {
                      @Override
                      public void onSuccess(Void unused) {
                          Intent intent = new
                               Intent(UrunEkle.this,UrunListele.class);
                          startActivity(intent);
                      }
                   }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
               Toast.makeText(UrunEkle.this,
                  "Güncelleme başarısız",
                  Toast.LENGTH_SHORT).show();
          }
        });
     }else{
        FirebaseFirestore.getInstance().collection("urunler")
                .add(hashMap)
                .addOnSuccessListener(new
```

```
                    OnSuccessListener<DocumentReference>() {
                          @Override
                          public void onSuccess(DocumentReference
                              documentReference) {
                                 Intent intent=new
                                   Intent(UrunEkle.this,UrunListele.class);
                                 startActivity(intent);
                          }
                       }).addOnFailureListener(new OnFailureListener() {
                  @Override
                  public void onFailure(@NonNull Exception e) {
                       Toast.makeText(UrunEkle.this,
                             "Kayıt işlemi başarısız",
                             Toast.LENGTH_LONG).show();
                   }
                      });
                  }
              }
          });
    }
}
```

7.3.10. Uzak Veri Tabanından Veri Silmek

Firestore veri tabanından veri silme işlemi, aslında doküman silme işlemidir. Silme işlemleri de asenkron olarak çalışır. Uzak veri tabanından veri silme işlemi şu şekilde yapılır:

```
FirebaseFirestore.getInstance()
         .collection("urunler")
        .document("ADF45353FGFDFGFD89")
        .delete()
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
               // İşlem Başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // İşlem Başarısız
    }
});
```

Bir doküman seçilip, delete metodu kullanılarak silme işlemi yapılır. Herhangi bir alanı silmek için

FieldValue nesnesi kullanılır. Sadece bir alanı silmek için şu kod kullanılır:

```
FirebaseFirestore.getInstance()
         .collection("urunler")
        .document("ADF45353FGFDFGFD89")
        .update("islem",FieldValue.Delete())
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
               // İşlem Başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // İşlem Başarısız
    }
});
```

7.3.11. Uzak Veri Tabanında Veri Güncellemek

Firestore veri tabanında güncelleme işlemleri update metodu ile yapılır. Update metoduna verilen HashMap türünde veriler dokümanda varsa değiştirilir, HashMap nesnesinden gelen veriler uzak veri tabanında yoksa olmayan veriler eklenir. Uzak veri tabanında güncelleme işlemi şu şekilde yapılır:

```
HashMap<String,Object> hashMap=new HashMap<>();
hashMap.put("kull","test");
hashMap.put("eposta","test@test.com");
FirebaseFirestore.getInstance().collection("urunler")
        .document("ADF45353FGFDFGFD89")
        .update(hashMap)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                // İşlem Başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // İşlem Başarısız
    }
});
```

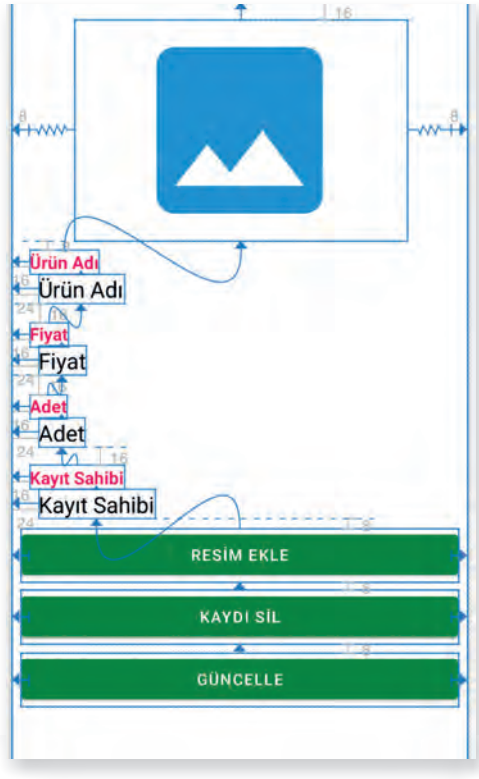
Sadece bir alan güncellenmek istenirse şu şekilde yapılır:

```
FirebaseFirestore.getInstance().collection("urunler")
        .document("ADF45353FGFDFGFD89")
        .update("kull","deneme")
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                // İşlem Başarılı
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // İşlem Başarısız
    }
});
```

12. UYGULAMA: İşlem adımlarına göre on birinci uygulama ile tamamlanan uygulamaya veri detay gösterme ve güncelleme bölümlerini ekleyiniz.

1. Adım: On birinci uygulama ile yapılan projeyi açınız. Kaydedilmiş bir uygulama yoksa tüm adımları tekrarlayarak on birinci uygulamayı gerçekleştiriniz.

2. Adım: urun_detay_xml dosyasını açarak güncelleme işlemi detay ekranını Görsel 7.64'teki gibi tasarlayınız.



Görsel 7.64: UrunDetay tasarım ekranı

3. Adım: ImageView için **detayUrunResim i**d bilgisini veriniz. **srcCompat** özelliğini **@drawable/bos_resim** yapınız.

4. Adım: Sekiz tane TextView ekleyiniz. Verilerin yazılacağı TextView nesnelerinin id bilgilerini **txtDetayUrunadi, txtDetayFiyat, txtDetayAdet, txtDetayUrunSahibi** veriniz.

5. Adım: Üç tane button ekleyiniz. Buttonların id bilgilerini **btnDetayResimEkle, btnDetayKayit-Sil, btnDetayGuncelle** yapınız.

6. Adım: UrunDetay.java dosyasını açıp dosyayı şu şekilde kodlayınız:

```
public class UrunDetay extends AppCompatActivity {
    ActivityUrunDetayBinding binding;
    ActivityResultLauncher<String> activityResultLauncher;
    Urun urun;
    Bitmap bitmap;
    private int YETKI_KODU=1;
    FirebaseStorage firebaseStorage;
    StorageReference storageReference;
    FirebaseFirestore firestore;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding=ActivityUrunDetayBinding.inflate(getLayoutInflater());
        View view=binding.getRoot();
        setContentView(view);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        firebaseStorage=FirebaseStorage.getInstance();
        storageReference=firebaseStorage.getReference();
        firestore=FirebaseFirestore.getInstance();
        activityResultLauncher=registerForActivityResult(new
                ActivityResultContracts.GetContent(),
                new ActivityResultCallback<Uri>() {
                    @Override
                    public void onActivityResult(Uri result) {
                        try {
                            if(result!=null){
                                if(Build.VERSION.SDK_INT>=28){
                                    ImageDecoder.Source source =
                                        ImageDecoder
                                        .createSource(UrunDetay.this
                                        .getContentResolver(),
                                     result);
                                     bitmap = ImageDecoder.decodeBitmap(source);
                                     binding.detayUrunResim
                                        .setImageBitmap(bitmap);
                                     ResimKaydet(result);
                                }else{
```

```
                                    bitmap = MediaStore.Images
                                                .Media
                                            .getBitmap(UrunDetay.
                                                this.getContentResolver(),
                                                result);
                                    binding.detayUrunResim.
                                                setImageBitmap(bitmap);
                                    ResimKaydet(result);
                                }
                            }
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                });
        if(getIntent().getSerializableExtra("urun")!=null){
            urun= (Urun) getIntent().getSerializableExtra("urun");
            binding.txtDetayUrunadi.setText(urun.getUrunadi());
            binding.txtDetayAdet.setText(urun.getAdet()+"");
            binding.txtDetayFiyat.setText(urun.getFiyat()+"");
            binding.txtDetayUrunSahibi.setText(urun.getEposta());
            if(urun.resim!=null){
                Picasso.get().load(urun.resim).into(binding.detayUrunResim);
            }
        }
        binding.btnDetayResimEkle.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                if(ContextCompat.checkSelfPermission(UrunDetay.this,
                        Manifest.permission.READ_EXTERNAL_STORAGE) ==
                        PackageManager.PERMISSION_GRANTED){
                    activityResultLauncher.launch("image/*");
                }
                else
                {
                   if(ActivityCompat
                        .shouldShowRequestPermissionRationale(UrunDetay.this,
                            Manifest.permission.READ_EXTERNAL_STORAGE)){
                            Snackbar.make(view,
                               "Uygulamayı kullanabilmeniz için izin gerekli!",
                            Snackbar.LENGTH_INDEFINITE)
                           .setAction("İzin ver",
                                new View.OnClickListener() {
                                    @Override
                                    public void onClick(View view) {
                                        ActivityCompat
```

```
                                        .requestPermissions(UrunDetay.this,
                                    new String[]
                                        {Manifest.permission
                                        .READ_EXTERNAL_STORAGE},
                                    YETKI_KODU);
                                }
                            }).show();
                    }else{
                        ActivityCompat.requestPermissions(UrunDetay.this,
                                new String[]{Manifest
                                   .permission.READ_EXTERNAL_STORAGE},
                                YETKI_KODU);
                    }
                }
            }
        });
        binding.btnDetayGuncelle.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent(UrunDetay.this,UrunEkle.class);
                intent.putExtra("urun",urun);
                startActivity(intent);
            }
        });
        binding.btnDetayKayitSil.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                FirebaseFirestore.getInstance().collection("urunler")
                        .document(urun.documentId)
                        .delete()
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void unused) {
                               Intent intent = new
                                    Intent(UrunDetay.this,UrunListele.class);
                               startActivity(intent);
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(UrunDetay.this,
                           "Veri silinemedi", Toast.LENGTH_SHORT).show();
                    }
                });
    }
        });
    }
```

```
    public void ResimKaydet(Uri bitmapUri){
        UUID uuid=UUID.randomUUID();
        final String dosyadi="resimler/"+uuid+".jpg";
        storageReference.child(dosyadi)
           .putFile(bitmapUri)
           .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                        FirebaseStorage.getInstance()
                                .getReference(dosyadi)
                                .getDownloadUrl()
                                .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                    @Override
                                    public void onSuccess(Uri uri) {
                                        String resimUrl=uri.toString();
                                        FirebaseFirestore.getInstance()
                                                .collection("urunler")
                                                .document(urun.documentId)
                                                .update("resim",resimUrl)
                                                .addOnSuccessListener(new OnSuc-
cessListener<Void>() {
                                                    @Override
                                                    public void onSuccess(Void un-
used) {
                                                        Intent intent=new Intent(U-
runDetay.this,UrunListele.class);
                                                        startActivity(intent);
                                                    }
                                                });
                                    }
                                });
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(UrunDetay.this,
                        "Dosya gönderimi başarısız\n"+e.getLocalizedMessage(),
                        Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

7.3.12. Firestore Veri Tabanında Kurallarla Çalışmak

Bu bölüme kadar bütün işlemler bir test veri tabanında yapılmıştır. Test veri tabanları internete açıktır. Veri tabanında bulunan kayıtlara sadece yetkili kullanıcılar erişebilmelidir. Firestore veri tabanı Firebase internet sitesinden açılarak **Rules** bölümünde kısıtlama işlemleri yapılabilir.

Rules bölümü varsayılan olarak şu şekilde gelir:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
          request.time < timestamp.date(2022, 7, 14);
    }
  }
}
```

Kuralların bütünü **match** ile belli bir eşleşmeye göre yazılır. match /{document=**} ifadesi, koleksiyon altındaki bütün doküman ve alt koleksiyonları kapsar. Sadece belli bir koleksiyon için kural şu şekilde yazılır:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /urunler/{urun} {
      allow read, write: if true;
    }
  }
}
```

Yazılan ifade ile urunler koleksiyonunun tüm dokümanları ifade edilir. Urunler koleksiyonunda birden fazla doküman olduğu için kurallar da { } arasına yazılır.

Kurallarda bir işleme izin vermek için if ifadesinden sonra true yazmak yeterlidir. Herhangi bir işlem sonucu da true olursa izin verilmiş sayılır. Temel olarak tüm okuma işlemleri için read izni, yazma işlemleri için write izni verilir.

Verilen izinler şunlardır:

- read
  - ► get
  - ► list
- write
  - ► update
  - ► delete
  - ► create

! **UYARI:** Kurallarda write ile izin verilmişse altta kalan diğer izinlerin hiçbir hükmü yoktur.

Örnek kuralda write ile yazma izni verilmiş ancak update izni kapatılmıştır. Verilen örnekte write izni olduğu için update ile izin verilmemesinin hiçbir hükmü yoktur.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /urunler/{urun} {
      allow write: if true;
       allow update: if false;
    }
  }
}
```

Kurallarda kullanıcıdan gelen veriler **request** nesnesiyle, sunucuda bulunan değerler ise **response** nesnesiyle elde edilir.

Sadece tanımlı kullanıcıların işlem yapmasını sağlayan kurallar şu şekilde yazılır:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /urunler/{urun} {
      allow read, write: if request.auth!=null;
    }
  }
}
```

Tanımlı kullanıcıların veri okumasını ve sadece kayıt yapmasını sağlayan kurallar şu şekilde yazılır:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /urunler/{urun} {
      allow read, create: if request.auth!=null;
    }
  }
}
```

Tanımlı kullanıcıların sadece kendi oluşturdukları kayıtları silmesini, başka kullanıcıların eklediği kayıtları silmemesini sağlayan kurallar şu şekilde yazılır:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /urunler/{urun} {
        allow read, create: if request.auth != null;
      allow delete : if request.auth!=null
                      &&
                      resource.data.user == request.auth.uid;
    }
  }
}
```

Tanımlı kullanıcıların sadece kendi oluşturdukları kayıtları düzenleyebilmesini sağlayan, başka kullanıcıların eklediği kayıtları değiştirmesini engelleyen kurallar şu şekilde yazılır:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /urunler/{urun} {
      allow read, create: if request.auth != null;
      allow delete, update : if request.auth!=null &&
                                  resource.data.user == request.auth.uid;
    }
  }
}
```

13. UYGULAMA: İşlem adımlarına göre on ikinci uygulamada tamamlanan uygulama için Firestore veri tabanında kurallar yazarak veri tabanını güvenli hâle getiriniz.

1. Adım: Firebase internet sitesini açarak konsola gidiniz.

2. Adım: Urunlerfirebase veri tabanını açınız.

3. Adım: Firestore veri tabanını açıp Rules bölümüne geçiniz.

4. Adım: Rules bölümünü şu şekilde yazınız:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /urunler/{urun} {
      allow read, create: if request.auth != null;
      allow delete, update : if request.auth!=null &&
                                  resource.data.user == request.auth.uid;
    }
  }
}
```

5. Adım: Uygulamayı çalıştırınız. Farklı farklı kullanımlar ile sisteme girerek veri silme ve güncelleme işlemleri yapınız.

6. Adım: Silme ve güncelleme işlemlerinden dolayı çıkan hata mesajlarını kontrol ediniz.

SIRA SİZDE: Yeni bir Empty Activity ile bir proje oluşturunuz. Projede bir günlük ödevlerinizi uzak veri tabanına kaydetmenizi ve resim eklemenizi sağlayacak bir model oluşturunuz. Modele uygun bir kayıt ekranı tasarlayarak verilerinizi uzak veri tabanına kaydeden bir uygulama geliştiriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Model oluşturdu.		
3. Modele uygun ekran tasarımları yaptı.		
4. Uzak veri tabanı oluşturdu.		
5. Uzak veri tabanında yetkilendirme ekranı tasarladı.		
6. Uzak veri tabanına veri kayıt ve düzenleme ekranlarını tasarladı.		
7. Galeriden resim alma işlemlerini yaptı.		
8. Uzak veri tabanında güvenlik için gerekli kodları yazdı.		

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) HashMap, anahtar-değer türü saklayan bir nesnedir.

2. ( ) sharedPreferences ile tablolar kaydedilir.

3. ( ) Yerel veri tabanları asla silinemez.

4. ( ) SqLite veri tabanı kullanmak için herhangi bir kurulum yapmaya gerek yoktur.

5. ( ) Cursor nesnesi ile veri güncelleme işlemleri yapılır.

6. ( ) Firestore veri tabanı eş zamanlı veri aktarımını desteklemez.

7. ( ) Yetkilendirme işlemleri FirebaseAuth nesnesi ile yönetilir.

8. ( ) Uzak veri tabanında zincir sorgular istendiği gibi yazılamaz.

9. ( ) Uzak veri tabanlarında sadece on kayıt limitlenebilir.

10. ( ) Kurallarda write izni alan bir kullanıcı tüm veri değiştirme işlemlerini yapar.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

Cursor	openOrCreateDatabase	response
sharedPreferences	put	BaseAdapter
getInstance	update	getCurrentUser
	execSQL	

11. Mobil uygulamalarda basit veriler ..... ile kaydedilir.

12. Veriler, sharedPreferences nesnesi kullanılarak ..... metodu ile alınır.

13. Yerel bir veri tabanı kullanmak için ..... metodu ile veri tabanı oluşturulur.

14. Veriler, veri tabanından ..... nesnesi ile getirilir.

15. SQL sorguları, ... metodu ile yerel veri tabanında çalıştırılır.

16. Özel adaptör sınıfları ..... sınıfından türetilir.

17. Firestore veri tabanı nesnesi ..... metodundan oluşturulur.

18. Bir kullanıcının giriş yapıp yapmadığı ..... metodu ile anlaşılır.

19. Firestore nesnesinin ..... metodu ile güncelleme işlemi yapılır.

20. Firestore kuralları yazılırken request ile gelen verilere, ... ile de sunucudaki verilere erişilir.

C) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

21. Aşağıdakilerden hangisi bir anahtar-değer nesnesidir?

A) Firestore B) FirebaseAuth C) Database
D) Date E) HashMap

22. Aşağıdakilerden hangisi sharedPreferences nesnesi ile veri kaydetmeyi sağlar?

A) Cursor B) Editor C) Tablo
D) Time E) TimeStamp

23. putFloat ile kaydedilen bir veri aşağıdaki metotlardan hangisi ile geri getirilir?

A) getInt B) getString C) getFloat
D) getBoolean E) getLong

24. Aşağıdaki metotlardan hangisi birden fazla satır veri getirir?

A) rawQuery B) execSQL C) SQL
D) getColumnIndex E) getInt

25. Yerel veri tabanına resim kaydetmek için aşağıdaki veri tiplerinden hangisi seçilmelidir?

A) VARCHAR B) INTEGER C) DOUBLE
D) BLOB E) CURSOR

26. Aşağıdakilerden hangisi bir SQL ifadesi değildir?

A) INSERT B) CREATE C) DELETE
D) UPDATE E) ADD

27. Firestore veri tabanında aşağıdaki verilerinden hangisi kaydedilmez?

A) Sayı B) Ondalık sayı C) Resim
D) Tarih E) Yazı

28. Aşağıdakilerden hangisi Firestore veri tabanında tablo yerine kullanılır?

A) Doküman B) Koleksiyon C) Alt koleksiyon
D) Alt doküman E) Anahtar-değer çiftleri

29. Aşağıdaki metotlardan hangisi Firestore veri tabanına veri ekler?

A) add B) insert C) ekle
D) map E) update

30. Aşağıdakilerden hangisi kullanıcıya sadece veri getirme izni verir?

A) update B) delete C) create
D) get E) write

D) Aşağıdaki cümlelerde verilen işlemleri gerçekleştiriniz.

31. “isim” adında bir dosyayı kaydedecek sharedPreferences nesnesini oluşturan kodları yazınız.

32. Daha önceden oluşturulan sharedPreferences isimli nesneyi kullanarak adınızı kaydediniz.

33. Sadece ad alanı olan bir yerel veri tabanından tüm verileri almayı sağlayan kodları yazınız.

34. Uzak veri tabanı uygulamasında yetki kontrolü yapınız.

35. Kullanıcıların sadece kendi kayıtlarını görmelerini sağlayan Firestore sorgusunu yazınız.

8. ÖĞRENME BİRİMİ

GELİŞMİŞ UYGULAMA TASARLAMA

KONULAR

8.1. YAYIN ALGILAYICILARLA ÇALIŞMAK

8.2. SMS VE İLETİ GÖNDERMEK

8.3. BİLDİRİMLERLE ÇALIŞMAK

8.4. BAŞKA UYGULAMALARLA ETKİLEŞİM KURMAK

8.5. SERVİSLERLE ÇALIŞMAK

8.6. WORKMANAGERLE ÇALIŞMAK

8.7. SENSÖRLERLE ÇALIŞMAK

NELER ÖĞRENECEKSİNİZ?

- Yayın alıcılarla sistem mesajlarını dinleme
- SMS gönderme ve okuma işlemleri
- Uygulama içinden e-posta gönderme işlemleri
- Başka uygulamalarla iletişim kurma
- Servis kullanarak arka planda işlemler yapma
- Bildirimler oluşturup gösterme
- Planlanmış görevler oluşturma
- Sensörlerle çalışma

ANAHTAR KELİMELER

- BoundService
- BroadcastReceiver
- Channel
- IntentFilter
- IntentService
- Notification
- PendingIntent
- Sensor
- Service
- SmsManager
- WorkManager

HAZIRLIK ÇALIŞMALARI

1. Uygulama geliştirilirken işletim sistemine ait bilgilere neden ihtiyaç duyulur? Arkadaşlarınızla tartışınız.
2. Uygulamanın arka planında hangi işlemler yapılmalıdır? Arka planda işlemler yapmanın sağladığı avantajlar nelerdir? Arkadaşlarınızla değerlendiriniz.
3. Planlanmış görevler neden oluşturulur? Açıklayınız.
4. Sensör kullanılarak ne tür uygulamalar yazılabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.

8.1. YAYIN ALGILAYICILARLA ÇALIŞMAK

Android işletim sistemi; cihazın uçak moduna alınması, şarj edilmeye başlanması veya pil durumunun kritik seviyeye gelmesi gibi durumlarda tüm sisteme mesajlar gönderir. Bu mesajlara Broadcast Mesaj denir. Broadcast, yayın anlamına gelir. Uygulamalar bu mesajları alarak duruma göre davranışlarını değiştirebilir. Örneğin pil durumu düşük olan bir cihazda arka plan işlemlerine başlanmaz.

Mesajları alabilmek için manifest dosyasına kaydolmak gereklidir. Bazı mesajlar, kayda rağmen alınmayabilir. Bu tür mesajlar için izin alınmalıdır. Özellikle tehlikeli (dangerous) olarak işaretlenmiş izinler kullanıcıdan izin alınarak dinlenebilir.

8.1.1. BroadcastReceiver Sınıfı Oluşturmak

Mesaj kaydı, manifest dosyasında tanımlanır. Örneğin pil seviyesi ile ilgili mesajları almak için manifest dosyasına şu eklemeler yapılır:

```
<receiver android:name=".PilSeviyesiAlgilayici" android:exported=»true»>
    <intent-filter>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
    </intent-filter>
</receiver>
```

Kayıt işleminde BroadcastReceiver sınıfından türemiş bir sınıf şu şekilde tanımlanır:

```
public class PilSeviyesiAlgilayici extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "Pil seviyesi düşük", Toast.LENGTH_LONG).show();
    }
}
```

BroadcastReceiver sınıfı sadece onReceive metodunu oluşturur. onRecevie metodu, Context ve Intent nesnelerini parametre olarak alır. Mesaj alınır alınmaz onReceive metodu çalışır. Tehlikeli izin gerektiren özelliklerin dinlenmesi için şu şekilde izin alınmalıdır:

```
ActivityCompat.requestPermissions(
                this,new String[]{Manifest.permission.BATTERY_STATS}
                ,PackageManager.PERMISSION_GRANTED);
```

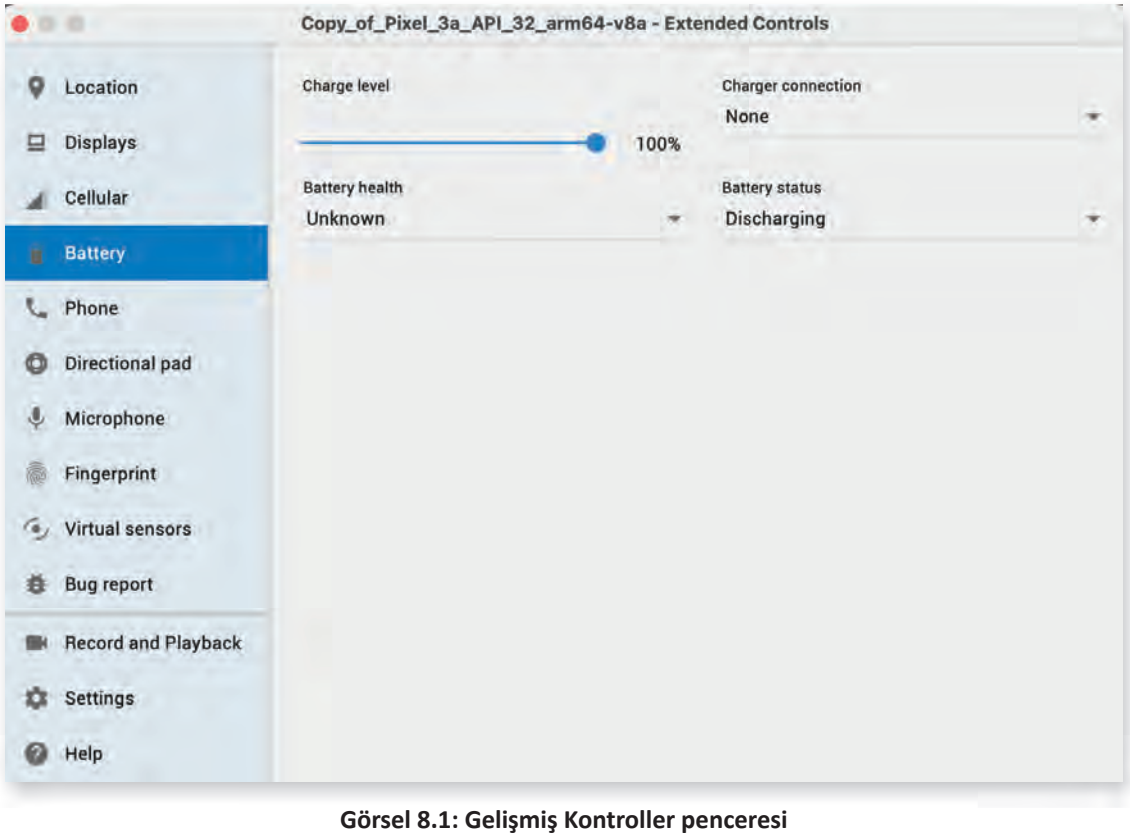
Sadece manifest dosyasına kayıt yapılarak mesajlar alınmayabilir. Android 9 ve üzeri sistemlerde sistem mesajlarının çoğunun kod kullanılarak kaydedilmesi istenir. Activitynin onResume veya onStart olaylarında kayıt işlemi yapılır, onStop olayında ise kayıt işlemi iptal edilir. Kayıt işlemi için registerReceiver metodu kullanılır. registerReceiver metodu kullanılmadan önce IntentFilter nesnesi hazırlanır. IntentFilter nesnesinde hangi yayının dinleneceği addAction metodu ile belirtilir (Intent action daha fazla bilgi için https://developer.android.com/reference/android/content/Intent sitesine bakılabilir.). Kayıt işlemi şu şekilde yapılır:

```
@Override
protected void onResume() {
    super.onResume();
    IntentFilter intentFilter=new IntentFilter();
    intentFilter.addAction(Intent.ACTION_BATTERY_LOW);
    registerReceiver(pilSeviyesiAlgilayici,intentFilter);
}
```

Kayıt işlemini iptal etmek için unregisterReceiver metodu kullanılır. onStop olayının kullanımı şu şekildedir:

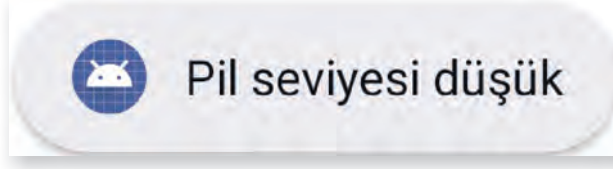
```
@Override
protected void onStop() {
    super.onStop();
    unregisterReceiver(pilSeviyesiAlgilayici);
}
```

Uygulama test edilirken emülatörde Gelişmiş Kontroller açılır. Gelişmiş Kontroller penceresinde Battery sekmesi açılır. Charger connection ve Battery Status ayarları Görsel 8.1'deki gibi olmalıdır.



Görsel 8.1: Gelişmiş Kontroller penceresi

Charge level sürgü kontrolü %15 ve altına çekildiğinde emülatörde Görsel 8.2'deki mesaj alınır.



Görsel 8.2: Yayın alıcı mesajı

1. UYGULAMA: İşlem adımlarına göre bir yayın alıcı sınıf tanımlayıp pil seviyesi düşük olduğunda kullanıcıyı Toast mesajı ile uyarınız.

1. Adım: Empty Activity seçerek yeni bir proje oluşturunuz. Projenin adını "PilSeviyesiAlgilayici-App" veriniz.

2. Adım: Yeni bir java dosyası oluşturunuz, adını "PilSeviyesiAlgilayici.java" olarak veriniz.

3. Adım: PilSeviyesiAlgilayici.java dosyasını açıp BroadcastReceiver sınıfından türetiniz ve gerekli metotları oluşturunuz.

4. Adım: PilSeviyesiAlgilayici.java dosyasında onReceive metoduyla Toast mesajı tanımlayınız.

5. Adım: Manifest dosyasını açarak yayın algılayıcı sınıfını kaydediniz.

6. Adım: MainActivity dosyasını açınız. Yayın algılayıcı sınıfından şu şekilde bir nesne oluşturunuz:

```
PilSeviyesiAlgilayici pilSeviyesiAlgilayici;
```

7. Adım: onCreate metodunda pilSeviyesiAlgilayici nesnesini oluşturunuz.

8. Adım: onCreate metodunda gerekli izinleri alınız.

9. Adım: onResume metodunda yayın alıcıyı kaydediniz.

10. Adım: onStop metodunda yayın alıcının kaydını iptal ediniz.

11. Adım: Uygulamayı çalıştırıp emülatörü açınız ve Gelişmiş Kontroller panelini kullanarak gerekli gözlemleri yapınız.

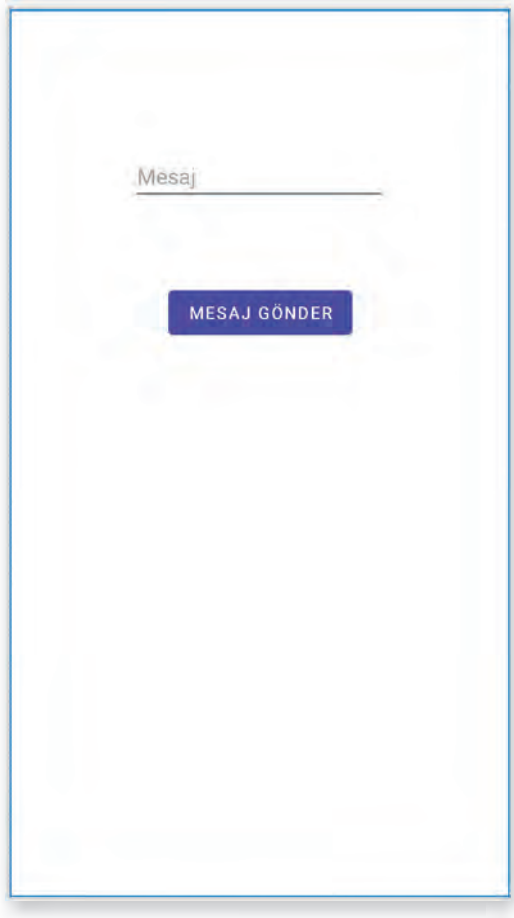
8.1.2. Kodla Yayın Algılayıcıları Tetiklemek

Yayın algılayıcılar gerektiği zaman harekete geçirilebilir. sendBroadcast metoduna bir Intent verilerek yayın alıcının çalışması sağlanır. Intent nesnesine herhangi bir veri eklenerek sendBroadcast metodu ile yayın alıcıya iletilebilir. onReceive metodu parametre olarak Context ve Intent nesnelerini alır.

2. UYGULAMA: İşlem adımlarına göre yeni bir sınıf oluşturarak BroadcastReceiver 'dan türetiniz. Uygulamanızdan bir EditText kutusuna mesaj yazıp oluşturduğunuz sınıfa gönderiniz.

1. Adım: Empty Activity seçerek yeni bir proje oluşturunuz. Projenin adını "BroadcastGonder" veriniz.

2. Adım: activity_main.xml dosyasını açarak Görsel 8.3'teki gibi bir EditText, bir de Button ekleyiniz.



Görsel 8.3: BroadcastGonder uygulaması ekran tasarımı

3. Adım: EditText nesnesine "editMesaj", Button nesnesine "btnGonder" adını veriniz.

4. Adım: gradle.build dosyasını açarak viewBinding özelliğini aktif ediniz.

5. Adım: MesajAlgilayici isimli bir sınıf oluşturunuz ve bu sınıfı BroadcastReceiver sınıfından türetiniz.

6. Adım: MesajAlgilayici.java dosyasında onReceive metodunu şu şekilde düzenleyiniz:

```
public class MesajAlgilayici extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String mesaj=intent.getStringExtra("mesaj");
        Toast.makeText(context, mesaj, Toast.LENGTH_LONG).show();
    }
}
```

7. Adım: Manifest dosyasını açarak yayın alıcıyı şu şekilde kaydediniz:

```
<receiver android:name=".MesajAlgilayici"/>
```

8. Adım: MainActivity dosyasını açıp viewBinding özelliğini şu şekilde ekleyiniz:

```
ActivityMainBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding=ActivityMainBinding.inflate(getLayoutInflater());
    View view=binding.getRoot();
    setContentView(view);
```

9. Adım: Button nesnesine şu şekilde bir onClickListener yazınız:

```
binding.btnGonder.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent(MainActivity.this,MesajAlgilayici.class);
        intent.putExtra("mesaj",binding.editMesaj.getText().toString());
        sendBroadcast(intent);
    }
});
```

10. Adım: Uygulamayı çalıştırınız. EditText kutusuna bir mesaj yazıp MESAJ GÖNDER butonuna basınız.

SIRA SİZDE: ACTION_AIRPLANE_MODE_CHANGED mesajını kullanarak cihazın uçak modu bilgisini dinleyen bir BroadcastReceiver yazıp test ediniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Manifest dosyasını düzenledi.		
3. BroadcastReceiver sınıfını yazdı.		

8.2. SMS VE İLETİ GÖNDERMEK

SMS gönderme işlemleri SmsManager sınıfı ile yapılır. SMS gönderilirken tehlikeli izinler için kullanıcıdan mutlaka izin alınmalıdır. SmsManager kullanılarak şu şekilde SMS gönderilir:

```
SmsManager sm = SmsManager.getDefault();
sm.sendTextMessage(<NUMARA>, null,
        <MESAJ>.toString(), null, null);
```

UYARI: Android işletim sisteminde varsayılan olarak SMS gönderme işlemleri SmsManager ile yapılır. Ancak bazı cihaz üreticileri müşterilerinin mağdur olmaması için SmsManager ile SMS gönderilmesini engellemiştir. SMS ile ilgili kodlar emülatörde çalışır ancak gerçek bir cihazda çalışmayabilir.

3. UYGULAMA: İşlem adımlarına göre yeni bir uygulama oluşturarak uygulama ekranına iki adet editText nesnesi ve bir adet Button nesnesi yerleştiriniz. Button nesnesine tıkladığınızda istediğiniz numaraya SMS gönderen uygulamayı yazınız.

1. Adım: Empty Activity seçerek yeni bir proje oluşturunuz. Projenin adını "SmsGonder" veriniz.

2. Adım: activity_main.xml dosyasını açarak Görsel 8.4'teki gibi tasarlayınız. Numara yazılan EditTexte editTextNumara, mesaj yazılan EditTexte editTextMesaj adını veriniz.

3. Adım: Button nesnesinin id bilgisine btnGonder adını veriniz.

4. Adım: gradle.build dosyasını açarak viewBinding özelliğini aktif ediniz.

5. Adım: Manifest dosyasını açarak şu şekilde SMS izinleri alınız:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
```

6. Adım: MainActivity.java dosyasını açarak viewBinding nesnesi oluşturunuz. Gerekli ayarlamaları yapınız.

7. Adım: SMS göndermek için kullanıcıdan şu şekilde izin alınız:

```
ActivityCompat.requestPermissions(this,new String[]{
    Manifest.permission.SEND_SMS,
    Manifest.permission.READ_SMS},
    PackageManager.PERMISSION_GRANTED);
```

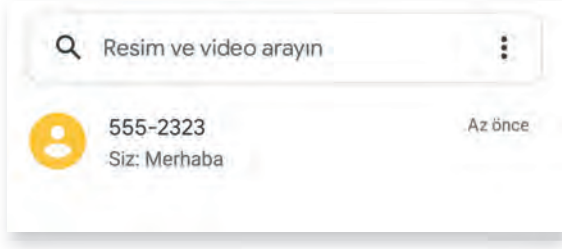
Görsel 8.4: SMS Gönder uygulaması tasarımı

8. Adım: SMS göndermek için btnGonder butonuna şu şekilde bir onClickListener oluşturunuz:

```
binding.btnGonder.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(ContextCompat.checkSelfPermission(MainActivity.this,
                Manifest.permission.SEND_SMS)==PackageManager.PERMISSION_GRANTED) {
            SmsManager sm = SmsManager.getDefault();
            sm.sendTextMessage(binding.editTextNumara.getText().toString(), null,
            binding.editTextMesaj.getText().toString(), null, null);
        }
        else
        {
            Toast.makeText(MainActivity.this, «İzin gerekli»,
                                Toast.LENGTH_LONG).show();
        }
    }
});
```

9. Adım: Uygulamanızı çalıştırınız. Emülatörden herhangi bir numara ve mesaj yazıp MESAJ GÖNDER butonuna basınız.

10. Adım: MESAJ GÖNDER butonuna bastıktan sonra ekranda herhangi bir bildirim görünmez. Uygulamayı kapatıp mesajlar uygulamasını açınız. Mesajlar uygulamasında Görsel 8.5'teki gibi mesajın gönderildiğini kontrol ediniz.



Görsel 8.5: Emülatör Mesajlar uygulaması

8.2.1. SMS Okumak

SMS okuma işlemi, yayın alıcılar tam olarak kullanılmadan yapılamaz. Cihaza SMS gönderildiği zaman işletim sistemi SMS geldiğine dair bir SMS_RECEIVED Broadcast mesajı yayınlar. Uygulamada yapılması gereken husus, SMS_RECEIVED mesajını dinlemek ve mesaj geldiğinde Intent ile SMS'leri okumaktır. Broadcast sınıfı ile SMS okuyup bir View ile göstermek çok kolay değildir. Broadcast sınıfında MainActivity dosyasında bulunan nesnelere doğrudan ulaşılamaz. SMS mesajlarını okuyup MainActivity dosyasında göstermek için Broadcast sınıfı MainActivity içinde şu şekilde tanımlanmalıdır:

```
public class MainActivity extends AppCompatActivity{
                              .
                              .

       public static class SmsAlici extends BroadcastReceiver {
                                    .
                                    .
       }

}
```

MainActivity dosyasına Broadcast sınıfı içinden ulaşılmak istendiğinde hata alınır. MainActivity dosyasının bir static nesne ile Broadcast sınıfına verilmesi gerekir. MainActivity static olarak şu şekilde çağrılır:

```
static MainActivity mainActivity;
public static MainActivity getInstance(){
    return mainActivity;
}
```

MainActivity Broadcast sınıfı içinde şu şekilde çağrılır:

```
MainActivity.getInstance()
```

Sisteme gelen SMS mesajları Intent nesnesine eklenir. Intent nesnesinde veriler bir Bundle nesnesine yerleştirilerek gelir. Bundle, "pdus" anahtarı ile bir pdus dizisi verir. pdus verileri SmsMessage nesnelerine dönüştürülür. SmsMessage nesnesinden istenen veriler alınarak gerekli işlemler yapılır.

4. UYGULAMA: İşlem adımlarına göre yeni bir uygulama oluşturarak iki adet editText nesnesi yerleştiriniz. SMS mesajlarını yakalayan bir sınıf yazarak gelen mesajları editText nesnelerinde gösteren uygulamayı yazınız.

1. Adım: Empty Activity seçerek yeni bir proje oluşturunuz. Projenin adını "SmsOkuma" veriniz

2. Adım: activity_main.xml dosyasını açarak Görsel 8.6'daki gibi tasarlayınız. EditText nesnesinin id bilgisine editTextGelenNumara adını veriniz. Mesaj EditText nesnesinin id bilgisini editTextSmsMesaj olarak veriniz.

Görsel 8.6: SMS Okuma Uygulaması Tasarım Ekranı

3. Adım: activity_main.xml dosyasında numara yazılan EditTexte editTextNumara, mesaj yazılan EditTexte editTextMesaj adını veriniz.

4. Adım: gradle.build dosyasını açarak viewBinding özelliğini aktif ediniz.

5. Adım: Manifest dosyasını açarak gerekli izinleri şu şekilde isteyiniz:

```
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
```

6. Adım: MainActivity.java dosyasını açarak viewBinding nesnesi oluşturunuz. Gerekli ayarlamaları yapınız.

7. Adım: SMS göndermek için kullanıcıdan şu şekilde izin alınız:

```
ActivityCompat.requestPermissions(this,new String[]
                        { Manifest.permission.READ_SMS,
                            Manifest.permission.RECEIVE_SMS},
                            PackageManager.PERMISSION_GRANTED);
```

8. Adım: static MainActivity nesnesi tanımlayınız. Tanımladığınız nesneyi gönderen getInstance isimli bir metot oluşturunuz.

9. Adım: MainActivity sınıfı içinde bir tane BroadcastReceiver sınıfından türemiş bir sınıf oluşturup sınıfın adını "SmsAlici" veriniz.

10. Adım: Manifest dosyasını açınız ve Broadcast sınıfını şu şekilde kaydediniz:

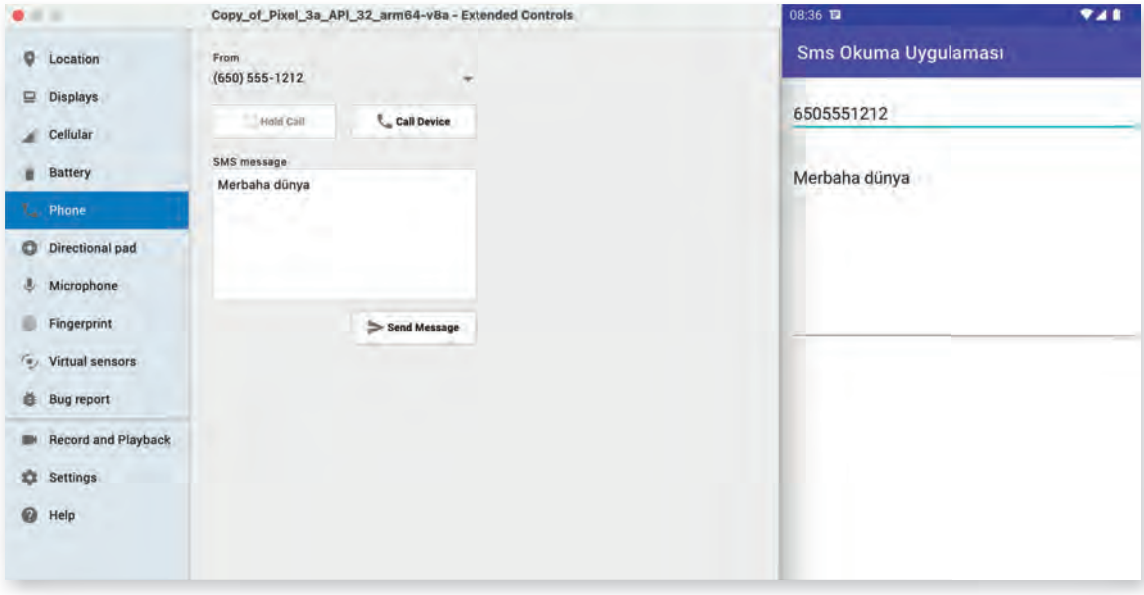
```
<receiver android:name=".MainActivity$SmsAlici" android:exported="true"
    tools:ignore="Instantiatable">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

11. Adım: SmsAlici sınıfının onReceive metodunu açıp şu şekilde yazınız:

```
public void onReceive(Context context, Intent intent) {
    if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")){
        Bundle bundle=intent.getExtras();
        SmsMessage[] mesajlar;
        String gonderen_numara;
        String mesaj;
        if(bundle!=null){
            Object[] pdus=(Object[])bundle.get("pdus");
            mesajlar=new SmsMessage[pdus.length];
            for(int i=0;i<pdus.length;i++){
                mesajlar[i] = SmsMessage.createFromPdu ((byte[])pdus[i] ,
                            bundle.getString("format"));
                gonderen_numara=mesajlar[i].getDisplayOriginatingAddress();
                mesaj=mesajlar[i].getMessageBody();
                MainActivity.getInstance().
                    binding.editTextSmsMesaj.setText(mesaj);
                MainActivity.getInstance().
                    binding.editTextGelenNumara.setText(gonderen_numara);
            }
        }
    }
}
```

12. Adım: Uygulamayı çalıştırınız. Emülatörden Gelişmiş Kontroller penceresini açınız.

13. Adım: Görsel 8.7'deki Gelişmiş Kontroller penceresinde Phone sekmesini açınız ve SMS message kutusuna bir mesaj yazarak Send Message butonuna basınız.



Görsel 8.7: Gelişmiş Kontroller penceresi Phone sekmesi

14. Adım: Uygulamanızı kontrol ediniz.

- **MainActivity.java**

```
public class MainActivity extends AppCompatActivity{
    ActivityMainBinding binding;
    static MainActivity mainActivity;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding=ActivityMainBinding.inflate(getLayoutInflater());
        View view=binding.getRoot();
        setContentView(view);
        mainActivity=this;
        ActivityCompat.requestPermissions(this,new String[]
                        { Manifest.permission.READ_SMS,
                          Manifest.permission.RECEIVE_SMS},
                          PackageManager.PERMISSION_GRANTED);
    }
    public static MainActivity getInstance(){
        return mainActivity;
    }
    public static class SmsAlici extends BroadcastReceiver {
       @Override
        public void onReceive(Context context, Intent intent) {
```

```
    if(intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")){
        Bundle bundle=intent.getExtras();
        SmsMessage[] mesajlar;
        String gonderen_numara;
        String mesaj;
        if(bundle!=null){
            Object[] pdus=(Object[])bundle.get("pdus");
            mesajlar=new SmsMessage[pdus.length];
            for(int i=0;i<pdus.length;i++){
                mesajlar[i] = SmsMessage.createFromPdu ((byte[])pdus[i] ,
                            bundle.getString("format"));
                gonderen_numara=mesajlar[i].getDisplayOriginatingAddress();
                mesaj=mesajlar[i].getMessageBody();
                MainActivity.getInstance().
                    binding.editTextSmsMesaj.setText(mesaj);
                MainActivity.getInstance().
                    binding.editTextGelenNumara.setText(gonderen_numara);
            }
        }
    }
}
```

8.2.2. İleti Göndermek

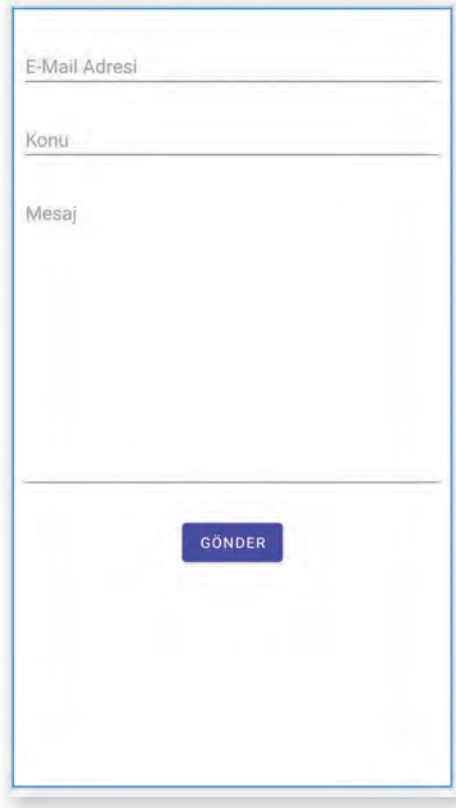
Mobil uygulama geliştirme ortamında SMTP (e-posta sunucusu) kullanarak e-posta göndermek için herhangi bir sınıf veya komut yoktur. Mutlaka e-posta göndermek gerekirse çeşitli kütüphaneler vardır. Mobil uygulama geliştirme ortamında e-posta göndermek için Gmail servisleri kullanılır. Bir Intent nesnesi hazırlanarak gönderilecek e-posta adresi, e-postanın başlığı ve mesajın içeriği nesneye eklenir. Intent için ACTION_SEND IntentFilter nesnesi seçilir. setType özelliği olarak "message/rfc822" seçilir. İleti gönderme işlemi şu şekilde yapılır:

```
Intent intent=new Intent(Intent.ACTION_SEND);
String email=binding.editTextEmail.getText().toString();
String konu=binding.editTextKonu.getText().toString();
String mesaj=binding.editTextMesaj.getText().toString();
intent.putExtra(Intent.EXTRA_EMAIL,new String[]{email});
intent.putExtra(Intent.EXTRA_SUBJECT,konu);
intent.putExtra(Intent.EXTRA_TEXT,mesaj);
intent.setType("message/rfc822");
startActivity(intent);
```

5. UYGULAMA: İşlem adımlarına göre e-posta gönderebileceğiniz bir ekran tasarlayarak herhangi bir kullanıcıya ileti gönderen uygulamayı yazınız.

1. Adım: Empty Activity seçerek yeni bir proje oluşturunuz. Projenin adını "MailGonderApp" veriniz.

2. Adım: activity_main.xml dosyasını açarak Görsel 8.8'deki gibi tasarlayınız. Mail adresi EditText nesnesinin id bilgisine editTextEmail adını veriniz. Konu EditText nesnesinin id bilgisini editText-Konu yapınız. Mesaj EditText nesnesinin id bilgisini editMesaj olarak veriniz.



Görsel 8.8: MailGönder uygulaması tasarım ekranı

3. Adım: Tasarım ekranına bir tane Button ekleyip id bilgisini btnGonder yapınız.

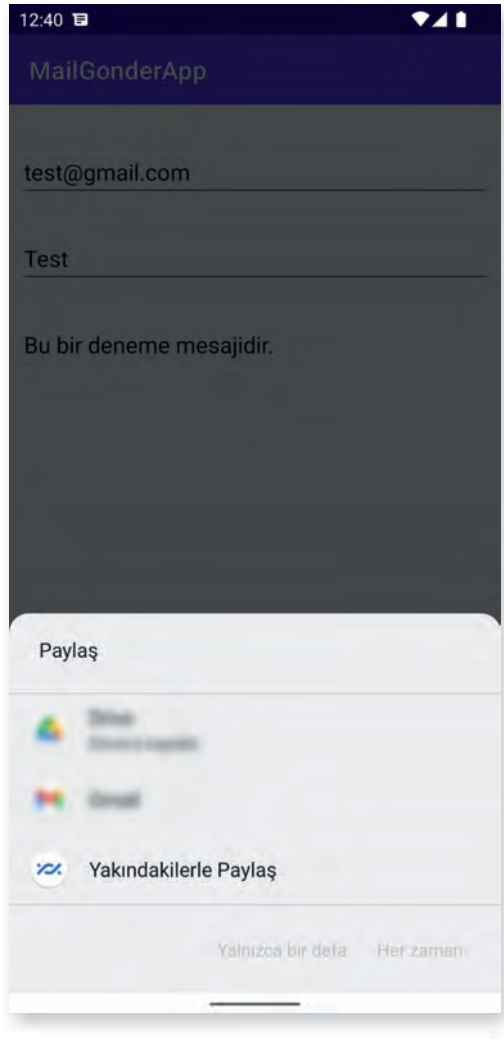
4. Adım: gradle.build dosyasını açarak viewBinding özelliğini aktif ediniz.

5. Adım: MainActivity.java dosyasını açınız ve viewBinding nesnelerini tanımlayıp gerekli ayarlamaları yapınız.

6. Adım: btnGonder butonu için onClickListener metodunu şu şekilde yazınız:

```
public void onClick(View view) {
    Intent intent=new Intent(Intent.ACTION_SEND);
    String email=binding.editTextEmail.getText().toString();
    String konu=binding.editTextKonu.getText().toString();
    String mesaj=binding.editTextMesaj.getText().toString();
    intent.putExtra(Intent.EXTRA_EMAIL,new String[]{email});
    intent.putExtra(Intent.EXTRA_SUBJECT,konu);
    intent.putExtra(Intent.EXTRA_TEXT,mesaj);
    intent.setType("message/rfc822");
    startActivity(intent);
}
```

7. Adım: Uygulamayı çalıştırıp GÖNDER butonuna tıklayınız. Görsel 8.9'daki gibi bir bildirim alıp almadığınızı kontrol ediniz.

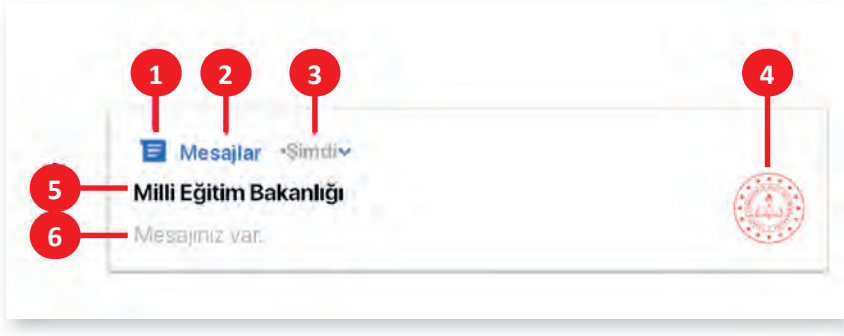


Görsel 8.9: MailGonder uygulaması seçim bildirim ekranı

8.3. BİLDİRİMLERLE ÇALIŞMAK

Bildirimler, kullanıcı ile etkileşime geçmenin en kısa yoludur. Bildirimler, kullanıcılar açısından çok büyük bir kolaylık sağladığı için hemen her uygulamada kullanılır. Kullanıcı, uygulamayı açmadan da uygulamadan bildirimler alabilir.

Bildirim ekranları mobil uygulama geliştirme ortamında NotificationManager sınıfıyla kontrol edilir. Notification nesnesiyle özellikleri belirlenen bildirim ekranı NotificationManager ile gösterilir. Android API 26 ve üzeri sürümlerde NotificationChannel kullanmak gereklidir. Channel ile bildirimin önem derecesi belirlenir. API 26 ile birlikte bildirim ekranlarının yönetimi kullanıcılara verildiği için API 26 ve üstü sürümlerde NotificationChannel kullanılmalıdır. Kullanıcı istemediği bildirimleri kanal adı seçerek kapatabilir. Görsel 8.10'da standart bir bildirim ekranının bölümleri verilmiştir.



Görsel 8.10: Bildirim yapısı

1: Bildirim ekranının küçük simgesidir. Bir bildirim ekranı tanımlanırken mutlaka belirtilmelidir. Notification nesnesinde setSmallIcon icon ile ayarlanır. Küçük simge, bildirim geldiğinde açılmasa bile ekranın en üstünde sadece simge olarak görünür.

2: Bildirimi gönderen uygulamanın adıdır.

3: Zaman damgası etiketi bildirimin ne zaman çıktığını gösterir. Notification nesnesinin setWhen metodu ile kapatılabilir veya setShowWhen ile gizlenebilir.

4: Bildirim ekranının büyük simgesidir. İsteğe bağlı olarak ayarlanmayabilir. Notification nesnesinin setLargeIcon metodu ile ayarlanır.

5: Bildirimin başlığıdır. İsteğe bağlı olarak kullanılmayabilir. Notification nesnesinin setContentTitle metodu ile ayarlanır.

6: Bildirimin mesaj kısmıdır. İsteğe bağlı olarak kullanılmayabilir. Notification nesnesinin setContentText metodu ile ayarlanır.

En basit hâli ile bir bildirim şu şekilde yapılır:

```
KANAL_ID="Bildirim App";
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    String kanal = KANAL_ID;
    String aciklama = "Bildirim yapılması sağlanacak";
    int bildirimOnemi = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel BildirimKanali = new NotificationChannel(KANAL_ID,
                        kanal, bildirimOnemi);
    BildirimKanali.setDescription(aciklama);
    NotificationManager notificationManager=getSystemService(NotificationManager.
class);
    notificationManager.createNotificationChannel(BildirimKanali);
}
NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.
this, KANAL_ID)
        .setSmallIcon(R.drawable.ic_android)
        .setContentTitle("Teset")
        .setContentText("İçerik buraya gelecek")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

```
NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(MainActivity.this);
notificationManager.notify(1000,builder.build());
```

Öncelikle API 26 ve üstü kullanılıp kullanılmadığı belirlenir. Kullanılan sistem, API 26 ve üstü ise Channel nesnesi kullanılmalıdır. Channel nesnesi ile bildirimin önem seviyesi belirtilir. Uygulama, emülatörde veya cihazda kullanılıyorsa ayarlarda Bildirim bölümü açılarak bildirim yapılan uygulamalar incelenebilir. Channel nesnesi ile verilen bilgiler burada görünür. Channel nesnesi, NotificationManager sınıfı ile kaydedilir. Kullanılan işletim sistemi API 26'dan önceki bir sürüm ise bir Channel nesnesi oluşmaz ve uygulama, if blokunun dışından devam eder.

Bildirim mesajını oluşturmak için NotificationCompat veya Notification nesneleri kullanılır. NotificationCompat, tüm Android sürümlerine uyumlu bildirim oluşturur. Notification nesnesi ise en son Android sürümü ile gelir. Builder nesnesi ile bildirim mesajı oluşturulur. Builder nesnesinin setSmallIcon özelliği ile küçük simge, setContentTitle özelliği ile mesaj başlığı ve setContentText özelliği ile mesajın içeriği hazırlanır. Builder nesnesi ile bildirimin önem derecesi ayarlanabilir ancak Channel nesnesi tanımlanmışsa öncelik değerinin bir önemi yoktur. Bildirim, NotificationManager sınıfının notify metodu ile gösterilir. notify metodunun ilk parametresi sayısal bir değerdir. Bu değer, bildirimin id numarasıdır. Bildirim yok edilecekse bu numara kullanılır. Bir bildirim şu şekilde yok edilir:

```
NotificationManager.cancel(1000);
```

6. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranına bir buton ekleyip uygulama oluşturunuz. Butona basıldığında basit bir bildirim yayınlayacak kodları yazınız.

1. Adım: Empty Activity seçerek yeni bir proje oluşturunuz. Projenin adını "BildirimBasit" veriniz.

2. Adım: activity_main.xml dosyasını açarak ekrana sadece bir tane Button yerleştiriniz. Button nesnesinin id bilgisini btnGonder olarak değiştiriniz.

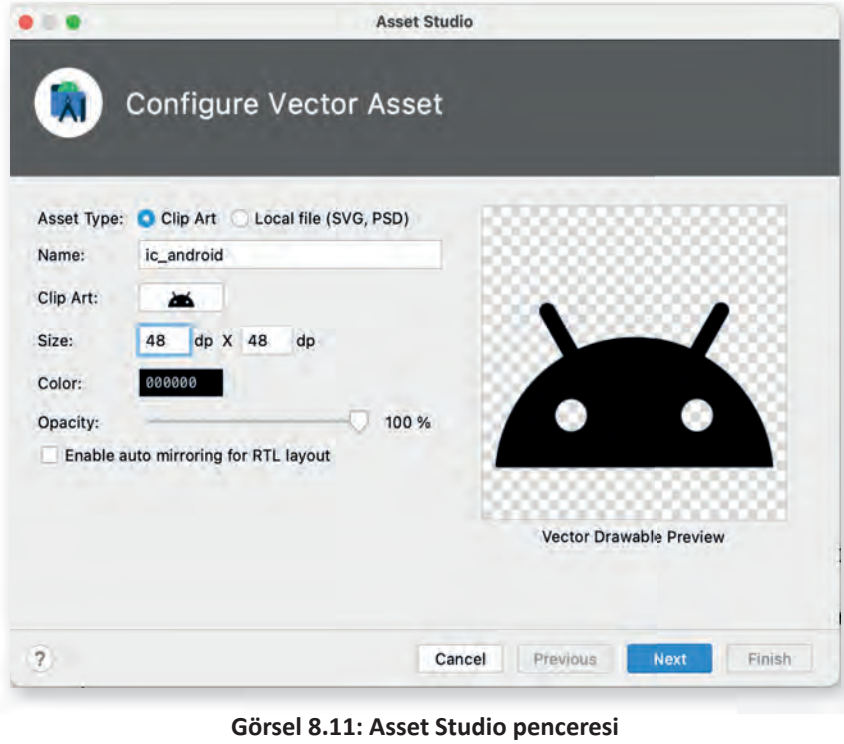
3. Adım: app>values>strings.xml dosyasını açıp uygulamanın adını "Basit Bildirim Uygulaması" yapınız.

4. Adım: gradle.build dosyasını açarak viewBinding özelliğini aktif ediniz.

5. Adım: drawable klasörüne sağ tıklayıp menüden New>Vector Assets seçeneğini seçiniz.

6. Adım: Açılan pencereden Android simgesi seçiliyken diğer ayarları Görsel 8.11'deki gibi yapınız.

7. Adım: MainActivity.java dosyasını açınız. viewBinding nesnesi tanımlayıp gerekli ayarlamaları yapınız.

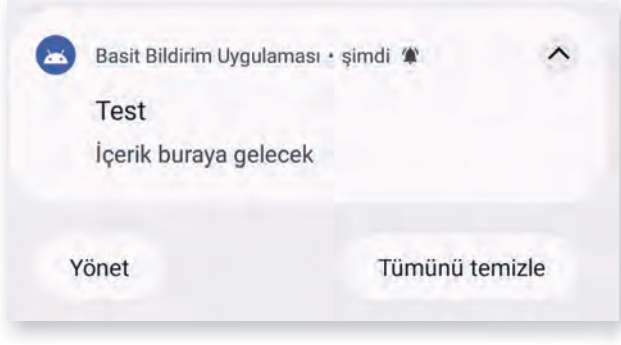


Görsel 8.11: Asset Studio penceresi

8. Adım: btnGonder Buttonu için onClickListener oluşturup onClick olayına şu kodları yazınız:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    String kanal = KANAL_ID;
    String aciklama = "Bildirim yapılması sağlanacak";
    int bildirimOnemi = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel BildirimKanali = new NotificationChannel(KANAL_ID, kanal,
bildirimOnemi);
    BildirimKanali.setDescription(aciklama);
    NotificationManager notificationManager = getSystemService(NotificationMan-
ager.class);
    notificationManager.createNotificationChannel(BildirimKanali);
}

NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.
this, KANAL_ID)
        .setSmallIcon(R.drawable.ic_android)
        .setContentTitle("Teset")
        .setContentText("İçerik buraya gelecek")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);

NotificationManagerCompat notificationManager = NotificationManagerCompat.from(-
MainActivity.this);
notificationManager.notify(1000,builder.build());
```

9. Adım: Uygulamayı çalıştırıp butona bastığınızda bildirim ekranının Görsel 8.12'deki gibi göründüğünü kontrol ediniz.



Görsel 8.12: Basit Bildirim Uygulaması ekranı

8.3.1. Bildirimlerden Veri Almak

Bildirimlerden veriler intent nesneleri kullanılarak alınabilir. Bu nesneler genellikle aktiviteler arasında dolaşmak veya servisleri başlatmak için kullanılır. Android sistemi, işi biter bitmez intent nesnelerini yok eder. Hemen yok olan bir nesneden veri almak mümkün değildir. PendingIntent nesneleri, intent nesnelerinin aksine hemen yok olmaz. Bu nesneler, kullanıcı bir işlem yapıncaya kadar bekler. Bundan dolayı bildirim mesajlarında bu nesneler kullanılır. Bildirim mesajı, kullanıcıdan bir eylem gelinceye kadar bildirim ekranında kalır.

PendingIntent ve intent nesneleri şu şekilde tanımlanır:

```
Intent intent = new Intent(MainActivity.this, MainActivity.class);
intent.putExtra("mesaj","Bu mesaj bildirim ile geldi");
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,
        0, intent, PendingIntent.FLAG_MUTABLE);
```

Intent nesnesine bir mesaj eklenir ve intent nesnesi, PendingIntent nesnesine parametre olarak verilir. Asıl veriyi taşıyan intent nesnesidir. PendingIntent, Notification nesnesine şu şekilde eklenir:

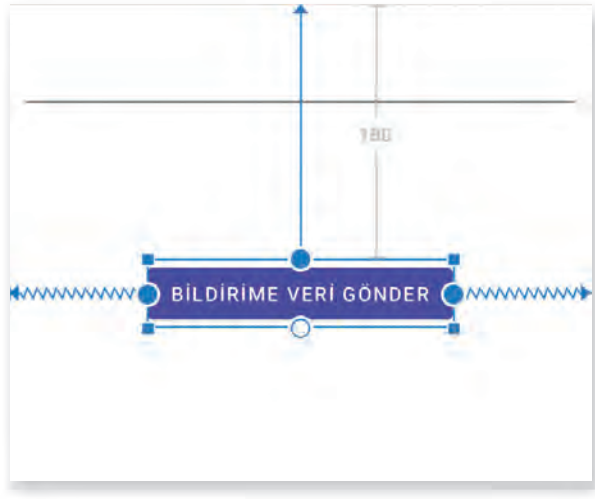
```
NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.this,
KANAL_ID)
        .setSmallIcon(R.drawable.ic_android)
        .setContentTitle("Bildirimler Pending")
        .setContentText("Bu bildirimden ana activity'ye mesaj taşıyacağız.")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(pendingIntent)
        .setAutoCancel(true);
```

PendingIntent nesnesi Notification.Builder sınıfına setContentIntent metodu ile eklenir. PendingIntent kullanıldığı için Builder sınıfında setAutoCancel metodu ile bildirim mesajının otomatik kapanması sağlanır. Kullanıcı, bildirimi okuyunca mesaj kapanır. Intent ile gelen veri hedef olarak belirtilen aktiviteye aktarılır. Bu sayede kullanıcının mesajı okuyup okumadığı anlaşılabilir. setCancelAuto, true olarak ayarlanmazsa bildirim mesajı kod ile kapatılır.

7. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranına bir EditText ve Button nesnesi ekleyerek bildirim alanına gizli bir mesaj gönderiniz. Kullanıcı, mesajı tıklayınca mesajdaki gizli bilgiyi yakalayan uygulamayı yazınız.

1. Adım: Empty Activity seçerek yeni bir proje oluşturunuz. Projenin adını "BildirimPending" veriniz.

2. Adım: activity_main.xml dosyasını açarak ekrana sadece bir EditText ve Button yerleştiriniz (Görsel 8.13).



Görsel 8.13: Bildirim Pending uygulaması tasarımı

3. Adım: EditText nesnesinin id bilgisini "editMesaj" olarak ayarlayınız. Button nesnesinin id bilgisini "button" olarak ayarlayınız.

4. Adım: gradle.build dosyasını açarak viewBinding özelliğini aktif ediniz.

5. Adım: drawable klasörüne sağ tıklayıp menüden New>Vector Assets seçeneğini seçiniz.

6. Adım: Açılan pencereden Android simgesini seçip diğer ayarları Görsel 8.11'deki gibi yapınız.

7. Adım: MainActivity.java dosyasını açınız. viewBinding nesnesi tanımlayıp gerekli ayarlamaları yapınız.

8. Adım: Button nesnesine bir onClickListener tanımlayınız.

9. Adım: Button nesnesinin onClick olayını şu şekilde yazınız:

```
String KANAL_ID ="BildirimPending App";
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    String kanal = KANAL_ID;
    String aciklama = "Bildirim yapılması sağlanacak";
    int bildirimOnemi = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel BildirimKanali = new NotificationChannel(KANAL_ID, kanal,
bildirimOnemi);
    BildirimKanali.setDescription(aciklama);
    NotificationManager notificationManager = getSystemService(NotificationMan-
ager.class);
    notificationManager.createNotificationChannel(BildirimKanali);
}
Intent intent = new Intent(MainActivity.this, MainActivity.class);
intent.putExtra("mesaj","Bu mesaj bildirim ile geldi");
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,
        0, intent, PendingIntent.FLAG_MUTABLE);

NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.
this, KANAL_ID)
        .setSmallIcon(R.drawable.ic_android)
        .setContentTitle("Bildirimler Pending")
        .setContentText("Bu bildirimden ana activity'ye mesaj taşıyacağız.")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(pendingIntent)
        .setAutoCancel(true);

NotificationManagerCompat notificationManager = NotificationManagerCompat.from(-
MainActivity.this);
notificationManager.notify(1000,builder.build());
```

10. Adım: MainActivity sınıfında **gelenVer**i isimli bir metodu şu şekilde oluşturunuz:

```
private void gelenVeri() {
    Intent intent=getIntent();
    String mesaj=intent.getStringExtra("mesaj");
    if(mesaj!=null)
    {
        binding.editMesaj.setText(mesaj);
    }
}
```

11. Adım: gelenVeri metodunu MainActivity sınıfının onCreate metodunda şu şekilde yazınız:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding=ActivityMainBinding.inflate(getLayoutInflater());
    View view=binding.getRoot();
    setContentView(view);
    gelenVeri();
```

12. Adım: Uygulamayı çalıştırınız ve bildirime tıkladığınızda Intent içine yazılan mesajın editText-te görünüp görünmediğini kontrol ediniz.

8. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranında çeşitli bildirimler göstermeyi sağlayınız.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "Bildirimler" yapınız.

2. Adım: app>drawable klasörüne sağ tıklayıp New>Vector Asset seçeneğini seçiniz.

3. Adım: Asset Studio penceresinde bir mail simgesi bulup ic_base_email ismiyle kaydediniz.

4. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

5. Adım: activity_main.xml dosyasını açarak ekrana bir tane buton yerleştiriniz.

6. Adım: Butonun id bilgisini button olarak değiştiriniz.

7. Adım: MainActivity.java dosyasını açınız ve setKanal() isimli bir metot oluşturup şu şekilde kodlarını yazınız:

```
private void Kanal () {
String KANAL_ID="Bildirimler";
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        String kanal = KANAL_ID;
        String aciklama = "Bildirim yapılması sağlanacak";
        int bildirimOnemi = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel BildirimKanali = new NotificationChannel(KANAL_ID,
kanal, bildirimOnemi);
        BildirimKanali.setDescription(aciklama);
        NotificationManager notificationManager = getSystemService(Notification-
Manager.class);
        notificationManager.createNotificationChannel(BildirimKanali);
    }
}
```

8. Adım: MainActivity.java dosyasını açınız ve viewBinding nesnesi tanımlayıp gerekli ayarlamaları yapınız.

9. Adım: MainActivity.java dosyasındaki onCreate olayında setKanal metodunu şu şekilde yazınız:

```
Kanal();
```

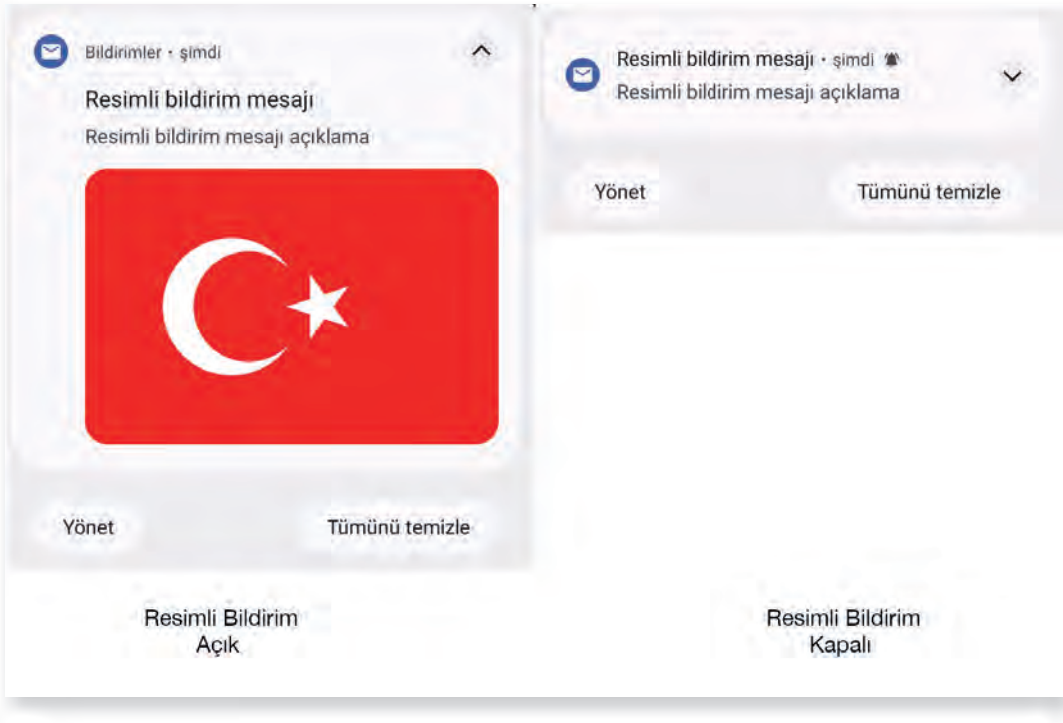
10. Adım: Telif hakkı olmayan bir bayrak görselini drawable klasörüne kopyalayınız. Bitmap nesnesi oluştururken görsel adına dikkat ediniz.

11. Adım: Button için onClickListener yazınız.

12. Adım: onClick olayını şu şekilde yazınız:

```
String KANAL_ID="Bildirimler";
Notification notification;
Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.tr);
notification = new NotificationCompat.Builder(MainActivity.this, KANAL_ID)
        .setSmallIcon(R.drawable.ic_baseline_email)
        .setContentTitle("Resimli bildirim mesajı»)
        .setContentText("Resimli bildirim mesajı açıklama")
        .setStyle(new NotificationCompat.BigPictureStyle()
                .bigPicture(bitmap))
        .build();
NotificationManagerCompat notificationManager = NotificationManagerCompat.
from(this);
notificationManager.notify(1, notification);
```

13. Adım: Uygulamayı çalıştırıp butona basınız. Uygulamanın Görsel 8.14'teki gibi görünüp görünmediğini kontrol ediniz.

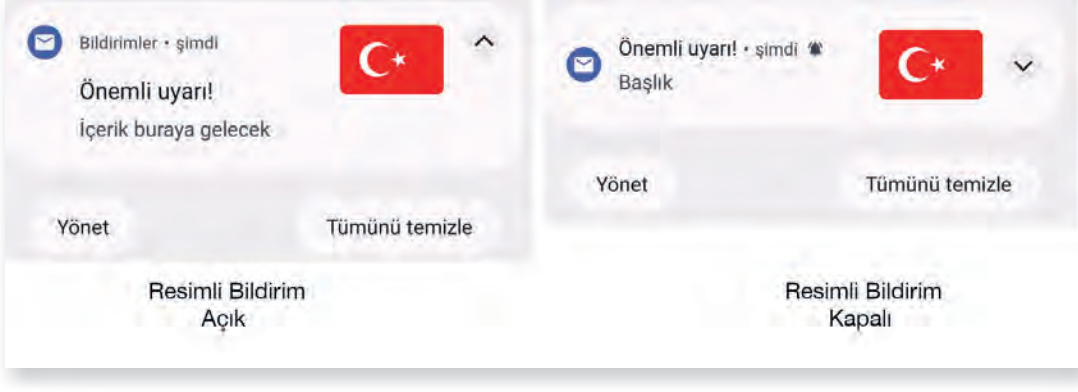


Görsel 8.14: Resimli Bildirim ekranı

14. Adım: Button nesnesinin onClick olayını şu şekilde değiştiriniz:

```
String KANAL_ID="Bildirimler";
Notification notification;
Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.tr);
NotificationCompat.Builder builder = new NotificationCompat.Builder(this, KANAL_ID)
        .setSmallIcon(R.drawable.ic_baseline_email)
        .setContentTitle("Uzun bildirim mesajı»)
        .setContentText("Uzun bildirim mesajı yazısı böyle olur.....")
        .setStyle(new NotificationCompat.BigTextStyle()
                .bigText("Daha fazla yazı ve fontlar biraz büyük yazıdırılır»))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
NotificationManagerCompat notificationManager = NotificationManagerCompat.
from(this);
notificationManager.notify(1, notification);
```

15. Adım: Uygulamayı çalıştırıp Button nesnesine basınız. Uygulamanın Görsel 8.15'teki gibi görünüp görünmediğini kontrol ediniz.

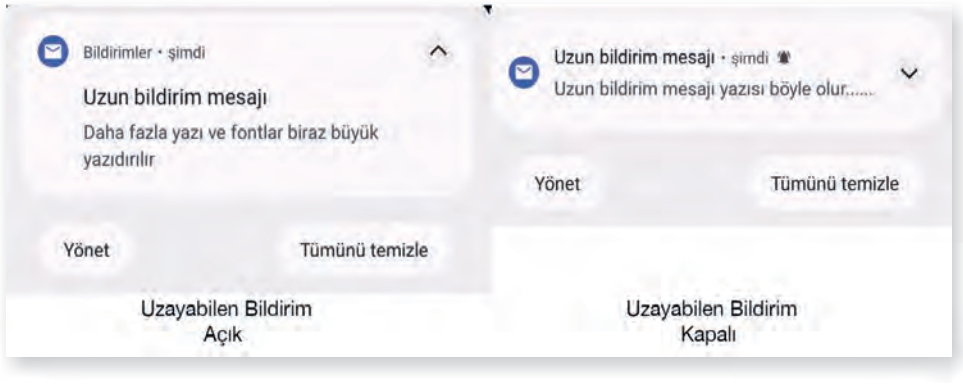


Görsel 8.15: Blok Bildirim ekranı

16. Adım: Button nesnesinin onClick olayını şu şekilde değiştiriniz:

```
String KANAL_ID="Bildirimler";
NotificationCompat.Builder builder = new NotificationCompat.Builder(this, KANAL_ID)
        .setSmallIcon(R.drawable.ic_baseline_email)
        .setContentTitle("Uzun bildirim mesajı»)
        .setContentText("Uzun bildirim mesajı yazısı böyle olur.....")
        .setStyle(new NotificationCompat.BigTextStyle()
                .bigText("Daha fazla yazı ve fontlar biraz büyük yazıdırılır»))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
NotificationManagerCompat notificationManager = NotificationManagerCompat.
from(this);
notificationManager.notify(1, notification.build());
```

17. Adım: Uygulamayı çalıştırıp Button nesnesine basınız. Uygulamanın Görsel 8.16'daki gibi görünüp görünmediğini kontrol ediniz.



Görsel 8.16: Uzayabilen Bildirim ekranı

SIRA SİZDE: Yeni bir uygulama geliştirerek uygulamaya üç Button yerleştiriniz. Her Buttonda farklı bir bildirim gösteriniz.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Tasarım ekranına üç tane Button yerleştirdi.		
3. Buttonlar için ayrı ayrı onClickListener yazdı.		

8.4. BAŞKA UYGULAMALARLA ETKİLEŞİM KURMAK

Implicit Intent kullanımı, mobil uygulama geliştirme ortamında başka uygulamalarla iletişim kurabilmek için kullanılan en kolay yöntemdir. Örneğin internet tarayıcısı ile iletişim şu şekilde kurulur:

```
Uri adres=Uri.parse("https://meb.gov.tr/");
Intent intent=new Intent(Intent.ACTION_VIEW,adres);
startActivity(intent);
```

Implicit Intent kullanımında seçilen IntentFilter nesnesi önemlidir. Veriler, intenti çalıştıracak IntentFilter nesnesine uygun bir şekilde ayarlanmalıdır.

8.4.1. Paylaş Butonuyla Diğer Uygulamalardan Veri Almak

Günümüzde hemen her uygulamada Paylaş butonu bulunur. Paylaş butonu sayesinde bir uygulama kullanılırken beğenilen bir içerik başka bir uygulamaya gönderilebilir. Mobil uygulama geliştirme ortamında IntentFilter nesnesi sayesinde bu özelliğin kullanımı sağlanır. Paylaşılan verinin türüne göre Paylaş menüsünde uygulamalar görüntülenir. Örneğin bir resim paylaşılırsa resim işleyebilen uygulamalar Paylaş menüsünde görünür. PDF paylaşılacağı zaman da bu listede PDF açabilen uygulamalar görünür. Bu özellik, IntentFilter nesneleri ile sağlanır.

Başka uygulamalardan bir resim alabilmek için manifest dosyasında IntentFilter ayarlanır. Resim almak için IntentFilter ayarı şu şekilde yapılır:

```
<intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="image/*" />
</intent-filter>
```

Manifest dosyasında yazılan IntentFilter, herhangi bir uygulamada Paylaş butonuna basıldığında uygulamanın Paylaş menüsünde görünmesini sağlar. IntentFilter nesnesinde data olarak istenen dosya tipi ayarlanır. Dosya türü olarak image seçilmişse ve sadece resim paylaşılırsa Paylaş menüsünde uygulamanın adı görünür.

Diğer MIME tipleri şunlardır:

- **Text Dosyaları:** text/plain, text/rtf, text/html, text/json ve tüm türler için text/* kullanılır.
- **Resim Dosyaları:** image/jpg, image/png, image/gif ve tüm resim dosyası türleri için image/* kullanılır.
- **Video Dosyaları:** video/mp4, video/3gp ve tüm video dosya türleri için video/* kullanılır.
- **Özel Dosyalar:** Uygulamaların dosya türleri kayıt altına alınmışsa application/pdf şekilde kullanılır.

Uygulamaya dosya gönderilirse şu şekilde alınır:

```
Intent intent = getIntent();
String action = intent.getAction();
String type = intent.getType();
if (action.equals(Intent.ACTION_SEND) && type != null) {
    doyayiAl(intent);
}
```

9. UYGULAMA: İşlem adımlarına göre mobil uygulama ekranına bir ImageView yerleştirerek uygulama arabirimi tasarımı yapınız. Uygulamanın başka bir dosyadan resim dosyası almasını ve gelen resmi göstermesini sağlayınız.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "DigerApp" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: activity_main.xml dosyasını açarak şu şekilde düzenleyiniz:

```
<?xml version=»1.0» encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="32dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:src="@drawable/ic_baseline_image_24" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

4. Adım: Manifest dosyasını açıp activity bölümüne şu şekilde ekleme yapınız:

```
<intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="image/*" />
</intent-filter>
```

5. Adım: MainActivity.java dosyasını açınız ve viewBinding nesnesi oluşturarak gerekli ayarlamaları yapınız.

6. Adım: dosyaAl isimli bir metot oluşturarak metodun kodlarını şu şekilde yazınız:

```
public void dosyaAl(Intent intent) {
    Uri imageUri = (Uri) intent.getParcelableExtra(Intent.EXTRA_STREAM);
    if (imageUri != null) {
        binding.imageView.setImageURI(imageUri);
    }
}
```

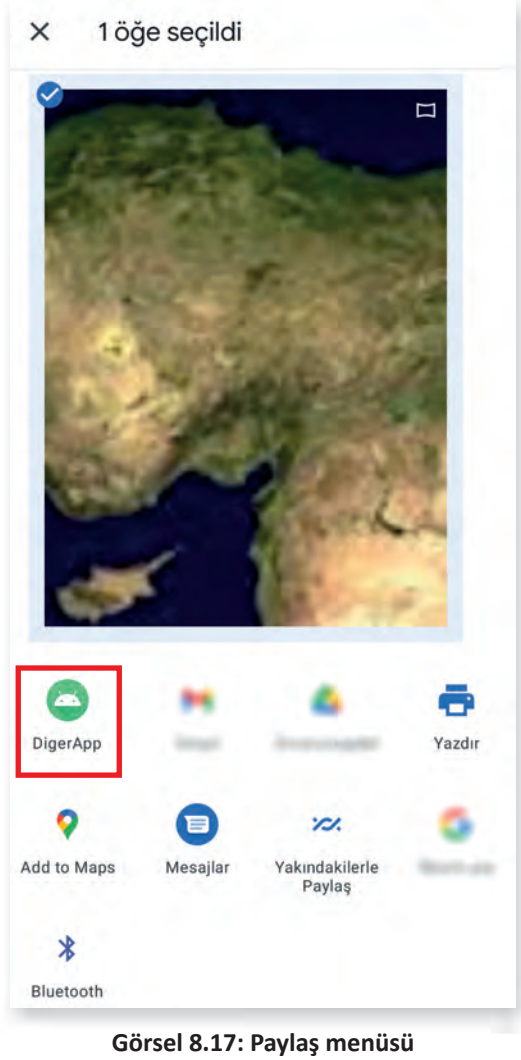
7. Adım: MainActivity.java dosyasını açınız ve onCreate metodunun kodlarını şu şekilde yazınız:

```
super.onCreate(savedInstanceState);
binding = ActivityMainBinding.inflate(getLayoutInflater());
View view = binding.getRoot();
```

```
setContentView(view);
Intent intent = getIntent();
String action = intent.getAction();
String type = intent.getType();
if (action.equals(Intent.ACTION_SEND) && type != null) {
    dosyaAl(intent);
}
```

8. Adım: Uygulamayı çalıştırınız. Emülatörde uygulama açıldıktan sonra uygulamayı kapatıp fotoğraf uygulamasını açınız. Emülatörde herhangi bir görsel dosyası yoksa internetten telifsiz bir görsel indiriniz.

9. Adım: Fotoğraf uygulamasında bir görsel seçip Paylaş butonuna basınız. Paylaş menüsü açıldıktan sonra listede geliştirilen uygulamayı seçiniz (Görsel 8.17).



Görsel 8.17: Paylaş menüsü

SIRA SİZDE: video/* MIME tipini kullanarak uygulamanıza bir video gönderilmesini sağlayınız.

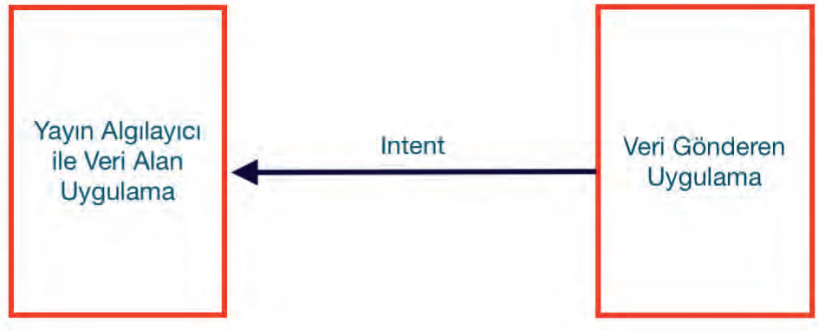
DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Video almak için gerekli kodları yazdı.		
3. Manifest dosyasına gerekli IntentFilter ayarlarını girdi.		

8.4.2. BroadcastReceiver Kullanarak Uygulamaların Arasında Haberleşmek

BroadcastReceiver sınıfları her ne kadar dinleme işlemi yapsa da veri gönderip almak için de kullanılır. Bir Broadcast sınıfı oluşturulurken parametre olarak Context ve Intent kullanılır. Intent nesneleri veri taşımak için elverişli nesnelerdir. putExtra metotları kullanılarak Intent nesneleri ile veri taşıma Görsel 8.18'deki gibi yapılır.



Görsel 8.18: BroadcastReceiver ile haberleşme

Veri gönderecek uygulama bu işlemleri şu şekilde yapar:

```
Intent intent=new Intent("com.example.broadcast.app.signal");
Bundle bundle=new Bundle();
bundle.putString("data1","Merhaba");
bundle.putString("data2",»Dünya");
```

```
intent.putExtras(bundle);
sendBroadcast(intent,"com.example.broadcast.app.permission");
```

Intent nesnesi oluşturulurken kullanılan anahtar, dinlemede kullanılacak özel anahtardır. sendBroadcast metodu ile yayın başlatılır ve bu metot, ikinci bir parametre olarak başka bir anahtar alır. İkinci parametre olarak dinleme yapabilmek için izin anahtarı oluşturulur.

Dinleme yapan uygulama bu işlemleri şu şekilde yapar:

```
public void onReceive(Context context, Intent intent) {
    Bundle bundle=intent.getExtras();
    Toast.makeText(context, bundle.getString("data1")+"-"
                    +bundle.getString("data2"),
            Toast.LENGTH_LONG).show();
    MainActivity.binding.textView.setText(bundle.getString("data1")+
            « « + bundle.getString("data2"))
}
```

Broadcast sınıfında onReceive metodu ile mesaj alınıp Intent nesnesi ile gelen veriler hem Toast mesajı hem de MainActivity dosyasında bir TextView nesnesine yazılır. Alıcı uygulamanın manifest dosyası da şu şekildedir:

```
<uses-permission android:name="com.example.broadcast.app.permission"/>
<receiver android:name=".BroadCasting" android:exported=»true»>
    <intent-filter>
        <action android:name="com.example.broadcast.app.signal"/>
    </intent-filter>
</receiver>
```

10. UYGULAMA: İşlem adımlarına göre alıcı ve verici şeklinde iki tane uygulama oluşturarak bir uygulamadan diğer uygulamaya veri gönderen uygulamaları yazınız.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "Uygulama01" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: activity_main.xml dosyasını şu şekilde düzenleyiniz:

```
<?xml version=»1.0» encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.072" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

4. Adım: MainActivity.java dosyasını açınız ve viewBinding nesnesi oluşturup gerekli ayarlamaları yapınız.

5. Adım: binding nesnesini static olarak şu şekilde tanımlayınız:

```
static ActivityMainBinding binding;
```

6. Adım: Broadcasting.java dosyası oluşturunuz ve Broadcasting sınıfını BroadcastReceiver sınıfından türetiniz.

7. Adım: Manifest dosyasına şu bilgileri giriniz:

```
<uses-permission android:name="com.example.broadcast.app.permission"/>
<receiver android:name=".BroadCasting" android:exported=»true»>
    <intent-filter>
        <action android:name="com.example.broadcast.app.signal"/>
    </intent-filter>
</receiver>
```

8. Adım: Uygulamayı çalıştırınız.

9. Adım: Yeni bir proje daha oluşturunuz. Empty Activity seçip uygulamanın adını "Uygulama02" yapınız.

10. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

11. Adım: activity_main.xml dosyasını şu şekilde düzenleyiniz:

```
<?xml version=»1.0» encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
```

```
        android:layout_marginBottom="16dp"
        android:text="Mesaj Gönder"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

12. Adım: MainActivity.java dosyasını açınız ve viewBinding nesnesi oluşturup gerekli ayarlamaları yapınız.

14. Adım: Button nesnesi için bir onClickListener tanımlayınız.

15.Adım: onClick olayına şu kodları yazınız:

```
Intent intent=new Intent("com.example.broadcast.app.signal");
Bundle bundle=new Bundle();
bundle.putString("data1","Merhaba");
bundle.putString("data2","Dünya");
intent.putExtras(bundle);
sendBroadcast(intent,"com.example.broadcast.app.permission");
```

16. Adım: Manifest dosyasını açıp şu bilgileri giriniz:

```
<permission android:name="com.example.broadcast.app.permission" android:protec-
tionLevel="signature"/>
```

17. Adım: Uygulamayı çalıştırınız. Uygulama01'in açık olduğundan emin olunuz.

18. Adım: Uygulama02 açıkken butona basınız. Hem bir Toast mesajı ekranda görülür hem de Görsel 8.19'daki gibi Intent ile gönderilen veri, Uygulama01'de yer alan TextView nesnesinde gösterilir.

Görsel 8.19: Broadcast yayının alınması ve TextView nesnesinde gösterilmesi

8.5. SERVİSLERLE ÇALIŞMAK

Servisler, uygulamanın arabiriminden bağımsız çalışan ve arabirimi olmayan uygulamalardır. Servisler genellikle arka planda çalışmak için kullanılır. Örneğin veri tabanının yedeklenmesi veya internetten veri indirmek gibi uzun sürecek bir işlem varsa bu işlemler servisler yardımıyla yapılır.

Servislerin çalışması için manifest dosyasına kaydedilmesi gereklidir. Manifest dosyasına eklenmemiş bir servis çalışmaz. Bir servis şu şekilde manifest dosyasına eklenir:

```
<service android:name=".YeniServis"/>
```

Servisler de aktiviteler gibi bir yaşam döngüsüne sahiptir. Bazı önemli servis metotları şunlardır:

- **onStartCommand():** Servis, aktiviteden startService metodu ile çalışmaya başlar. Servis kapatılıncaya kadar çalışır. Kullanıcı, servisi stopService veya stopSelf ile kapatıncaya kadar servisin onStartCommand metodu çalışmaya devam eder. onStartCommand ile yapılan işlemlerde dikkatli olmak gereklidir. Sisteme çok fazla iş yükü bindirilirse Google Play uygulamayı kabul etmeyebilir.
- **onBind():** Servislere görsel bileşenlerin bindService metodu ile bağlanmasını sağlar.
- **onUnBind():** Servis ile bağlanan bileşenin bağlantısı koptuğu zaman çalışır.
- **onRebind():** Servise yeni bir görsel bileşen bağlandığında çalışır.

- **onCreate():** Servis başlatıldığında çalışır.
- **onDestroy():** Servis kapatıldığı zaman çalışır. onDestroy çalıştıktan sonra servis yok edilir.

Android servisleri üç gruba ayrılır.

1. **Arka Plan Servisleri:** Bu servisler tamamen arka planda çalışır. Kullanıcılar bu servisleri göremezler. Arka plan servisleri tüm işlerini herhangi bir görsel bileşen kullanmadan yapar. İnternetten veri alma gibi işlemler çok sık yapılıyorsa genellikle arka plan hizmetleri tercih edilir.
2. **Ön Plan Servisleri:** Bir görsel bileşen ile çalışan servislerdir. Örneğin bir müzik çalar servisi kurularak uygulama kapansa bile müziğin çalışması sağlanır. Bazı ön plan servisleri Toast mesajı veya bir bildirim vererek çalışmayı durdurabilir.
3. **Bağlı Servisler:** Görsel bileşenin onBind metodu ile servise bağlandığı servislerdir. Görsel bileşen ile bağlantı koptuğunda servis otomatik olarak yok edilir.

Servisler sadece startService metodunun kullanılması ile çalışmaya başlar. stopService ile durduruluncaya kadar çalışmaya devam eder. Arka planda bir servis çalıştığında uzun süreli bir işlem yapılacaksa mutlaka yeni bir iş parçacığı üzerinde gerçekleştirilmelidir. Servisler her ne kadar arka planda çalışsa da ana aktivitenin iş parçacığı üzerine işlem yapar. Uzun süren bir işlem, ayrı bir iş parçacığı üzerinde çalıştırılmazsa uygulama kilitlenir.

Bir servis şu şekilde oluşturulur:

```
public class DataService extends Service {
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Servis oluşturulduktan sonra mobil uygulama geliştirme ortamı sadece onBind metodunu kodlara ekler. Servise ait diğer metotlar yazılarak kullanılır. Servisin son hâli şu şekildedir:

```
public class DataService extends Service {
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        return super.onStartCommand(intent, flags, startId);
    }
    @Override
    public void onCreate() {
        super.onCreate();
    }
    @Override
```

```
    public void onDestroy() {
        super.onDestroy();
    }
}
```

Servisle ilgili tüm işlemler onStartCommand metodunda yapılır. Servis hayatta olduğu sürece onStartCommand metodu çalışır.

Servisler de aktiviteler gibi bir Intent nesnesi ile başlatılır. Bir servis şu şekilde başlatılır:

```
Intent intent=new Intent(MainActivity.this,SayiciServis.class);
startService(intent);
```

Servis başlatıldıktan sonra kapatılmazsa çalışmaya devam eder. Bir servis şu şekilde kapatılır:

```
Intent intent=new Intent(MainActivity.this,SayiciServis.class);
stopService(intent);
```

Servisler sistem tarafından da kapatılabilir. Bir servis, sistem kaynaklarını çok fazla tüketirse işletim sistemi servisi kapatır.

11. UYGULAMA: İşlem adımlarına göre bir servis oluşturarak servisi çalıştıran uygulamayı yazınız.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "ServisApp01" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: main_activity.xml dosyasına Görsel 8.20'deki gibi iki buton yerleştiriniz.

Görsel 8.20: ServiceApp uygulaması tasarım ekranı

4. Adım: Butonlara btnBasla ve btnDur isimlerini veriniz.

5. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

6. Adım: MainActivity.java dosyasını açarak binding nesnesi tanımlayınız ve gerekli ayarlamaları yapınız.

7. Adım: btnBasla butonu için bir onClickListener tanımlayınız. onClick metodunu şu şekilde yazınız:

```
binding.btnBasla.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent(MainActivity.this,YeniServis.class);
        startService(intent);
    }
});
```

8. Adım: btnDur butonu için bir onClickListener tanımlayınız. onClick metodunu şu şekilde yazınız:

```
binding.btnDur.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent(MainActivity.this,YeniServis.class);
        stopService(intent);
    }
});
```

9. Adım: Service sınıfından türemiş yeni bir sınıf oluşturunuz ve adını "YeniServis" veriniz.

10. Adım: YeniServis sınıfının tüm metotlarını oluşturunuz. YeniServis sınıfına onStartCommand, onCreate ve onDestroy metotlarını ekleyiniz.

11. Adım: YeniServis sınıfını şu şekilde kodlayınız:

```
private static final String TAG = "YeniServis";
@Nullable
@Override
public IBinder onBind(Intent intent) {
    return null;
}
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.d(TAG, "onStartCommand: ");
    return super.onStartCommand(intent, flags, startId);
}
@Override
public void onCreate() {
    Log.d(TAG, "onCreate: ");
}
@Override
public void onDestroy() {
    Log.d(TAG, "onDestroy: ");
    super.onDestroy();
}
```

12. Adım: Manifest dosyasını açarak dosyaya şu eklemeyi yapınız:

```
<service android:name=".YeniServis"/>
```

13. Adım: Uygulamayı çalıştırıp önce BAŞLA butonuna sonra DURDUR butonuna basınız. Logcat penceresinde servisin çalışıp çalışmadığını kontrol ediniz.

```
YeniServis: onCreate:
YeniServis: onStartCommand:
YeniServis: onDestroy:
```

8.5.1. Arka Plan Servislerle Çalışmak

Tüm servis türleri arka planda çalışır ancak arka planda çalışan bir servis, kullanıcı ile hiçbir etkileşime girmez. Servisin yaptığı işler bir yerde saklanırsa sonradan kullanılabilir. Arka plan servisleri başlatıldıktan sonra kapatılıncaya kadar çalışmaya devam eder.

Arka plan servislerinde işlemler yeni bir iş parçacığı oluşturularak yapılmalıdır. Herhangi bir işlem, yeni bir iş parçacığı ile şu şekilde çalıştırılır:

```
new Thread(new Runnable() {
    @Override
    public void run() {
        getData();
    }
}).start();
```

Uzun işlemler, ana iş parçacığını sürekli meşgul eder ve uygulama, işlemler bitinceye kadar çalışamaz hâle gelir.

12. UYGULAMA: İşlem adımlarına göre bir servis yazarak servis ile "https://turkiye.gov.tr" internet sitesinin içeriğini arka planda alınız. Alınan verileri sharedPreferences nesnesini kullanarak kaydeden ve uygulamada sharedPreferences nesnesinden okuyup gösteren programı yazınız.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "ArkaPlanServis" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: main_activity.xml dosyasını şu şekilde düzenleyiniz:

```
<?xml version=»1.0» encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/editTextTextMultiLine"
```

```
        android:layout_width="406dp"
        android:layout_height="509dp"
        android:ems="10"
        android:gravity="start|top"
        android:inputType="textMultiLine"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
    <Button
        android:id="@+id/btnBaslat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="Servisi Başlat"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_editor_absoluteX="26dp" />
    <Button
        android:id="@+id/btnDur"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginTop="24dp"
        android:text="Servisi Durdur"
        app:layout_constraintStart_toEndOf="@+id/btnBaslat"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/btnGetir"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="48dp"
        android:text="Veriyi getir"
        app:layout_constraintTop_toBottomOf="@+id/btnBaslat"
        tools:layout_editor_absoluteX="139dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

4. Adım: ArkaPlanServis.java dosyası oluşturunuz. Oluşturduğunuz sınıfı Service sınıfından türetiniz.

5. Adım: Manifest dosyasını açarak servisi kaydediniz.

```
<service android:name=".ArkaPlanServis"/>
```

6. Adım: Manifest dosyasında Internet izni alınız.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

7. Adım: MainActivity.java dosyasını açarak viewBinding nesnesi tanımlayınız ve viewBinding için gerekli ayarlamaları yapınız.

8. Adım: btnBaslat butonuna bir onClickListener tanımlayınız. onClick metodunu şu şekilde yazınız:

```
Intent intent=new Intent(MainActivity.this,ArkaPlanServis.class);
startService(intent);
```

9. Adım: btnDur butonuna bir onClickListener tanımlayınız. onClick metodunu şu şekilde yazınız:

```
Intent intent=new Intent(MainActivity.this,ArkaPlanServis.class);
stopService(intent);
```

10. Adım: btnGetir butonuna bir onClickListener tanımlayınız. onClick metodunu şu şekilde yazınız:

```
SharedPreferences sharedPreferences=MainActivity.this.getSharedPreferences("do-
sya",MODE_PRIVATE);
String veri=sharedPreferences.getString("data","");
binding.editTextTextMultiLine.setText(veri);
```

11. Adım: ArkaPlanServis.java dosyasını açınız ve getData isimli bir metot oluşturunuz.

12. Adım: getData metodunu şu şekilde yazınız:

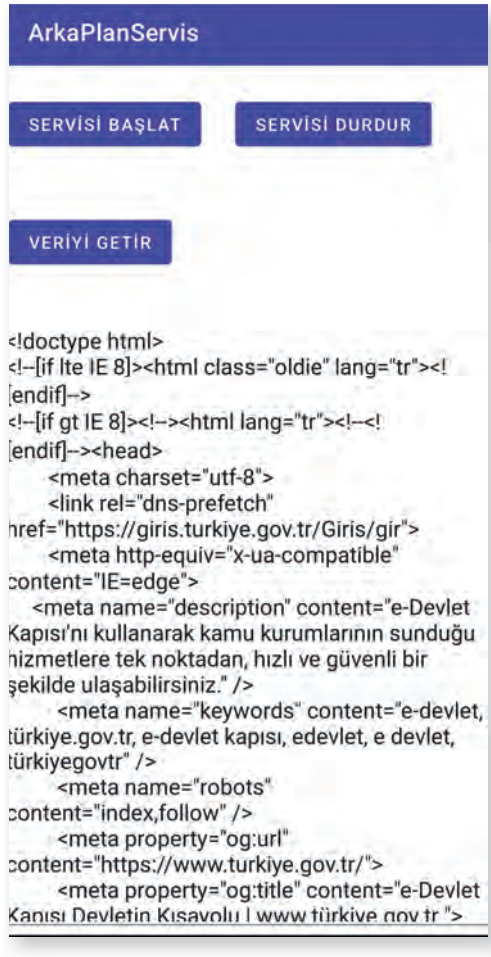
```
public void getData(){
    Context context=getApplicationContext();
    URL url;
    String sonuc="";
    HttpURLConnection httpURLConnection=null;
    try{
        url=new URL("https://www.turkiye.gov.tr/");
        URLConnection urlConnection=url.openConnection();
        httpURLConnection= (HttpURLConnection) url.openConnection();
        int lenghtData=urlConnection.getContentLength();
        InputStream inputStream = httpURLConnection.getInputStream();
        InputStreamReader inputStreamReader=new InputStreamReader(inputStream);
        int D=inputStreamReader.read();
        while(D!=-1){
            char c=(char) D;
            sonuc+=c;
            D=inputStreamReader.read();
        }
        SharedPreferences sharedPreferences=context.getSharedPreferences("do-
sya",MODE_PRIVATE);
        sharedPreferences.edit().putString("data",sonuc).apply();
    }catch (Exception e)
    {
        SharedPreferences sharedPreferences=context.getSharedPreferences("do-
sya",MODE_PRIVATE);
        sharedPreferences.edit().putString("data","Hata oluştu").apply();
    }
}
```

13. Adım: onStartCommand metodu yoksa oluşturunuz ve şu şekilde yazınız:

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            getData();
        }
    }).start();
    return super.onStartCommand(intent, flags, startId);
}
```

14. Adım: Uygulamayı çalıştırınız. Önce SERVİSİ BAŞLAT butonuna basınız.

15. Adım: Biraz bekleyip VERİYİ GETİR butonuna basınız. Sonucun Görsel 8.21'deki gibi görünüp görünmediğini kontrol ediniz.



Görsel 8.21: Arka Plan Servis uygulaması çalışma ekranı

UYARI: On ikinci uygulamada en temel veri okuma metotları kullanılmıştır. Başka bir web sitesi adresi yazıldığında hata alınabilir. Web sitesine herhangi bir tarayıcı yazılımı kullanılmadan bağlanıldığı için sunucular gelen bağlantıyı kabul etmeyebilir. Bu tür sitelerden veri okuma istendiğinde ekrana "Hata" yazacaktır. Web sitelerinden veri okumak için Retrofit gibi gelişmiş araçlar da kullanılabilir.

8.5.2. Ön Plan Servislerle Çalışmak

Ön plan servisler, kullanıcılara herhangi bir bildirim vermek için kullanılır. Arka plan servislerde kullanıcı ile etkileşim mümkün değildir. Kullanıcı ile etkileşim kurulması gerekli ise ön plan servisler kullanılır.

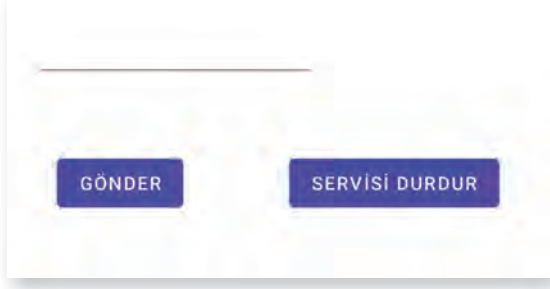
Ön plan servisi herhangi bir uyarı vermezse 5 saniye içinde yok edilir. Ön plan servisleri manifest dosyasından izin alınarak çalıştırılır. Normal servisler startService metodu ile çalıştırılırken ön plan servisler startForegroundService metodu ile çalıştırılır.

13. UYGULAMA: İşlem adımlarına göre bir ön plan servis yazarak servise bir mesaj gönderiniz. Alınan mesajı bir servis bildirimi ile gösteren uygulamayı yazınız.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "ArkaPlanServis" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: main_activity.xml dosyasını Görsel 8.22'deki gibi düzenleyiniz.



Görsel 8.22: Ön Plan Servis uygulaması tasarım ekranı

4. Adım: EditText için "EditText", butonlar için btnGonder ve btndUr id isimlerini veriniz.

5. Adım: MainActivity.java dosyasını açıp viewBinding nesnesi tanımlayınız ve viewBinding için gerekli ayarlamaları yapınız.

6. Adım: OnServis.java isimli bir dosya oluşturup Service sınıfından türetiniz.

7. Adım: Manifest dosyasını açarak ön plan servisi iznini şu şekilde alınız:

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
```

8. Adım: Manifets dosyasında servisi şu şekilde kaydediniz:

```
<service android:name=".OnServis"/>
```

9. Adım: MainActivity.java dosyasında bildirim için gerekli kanal ayarlarını onCreate metoduna şu şekilde ekleyiniz:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    int bildirimOnemi = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel BildirimKanali = new NotificationChannel("ForeGroundSer-
viceK1",
            "ForeGroundService", bildirimOnemi);
    BildirimKanali.setDescription("");
    NotificationManager notificationManager=getSystemService(NotificationManager.
class);
    notificationManager.createNotificationChannel(BildirimKanali);
}
```

10. Adım: MainActivity.java dosyasında btnGonder için bir onClickListener tanımlayınız.

11. Adım: onClick metodunu şu şekilde yazınız:

```
Intent intent=new Intent(MainActivity.this,OnServis.class);
intent.putExtra("mesaj",binding.editText.getText().toString());
startForegroundService(intent);
```

12. Adım: MainActivity.java dosyasında btnDur için bir onClickListener tanımlayınız.

13. Adım: onClick metodunu şu şekilde yazınız:

```
Intent intent=new Intent(MainActivity.this,OnServis.class);
stopService(intent);
```

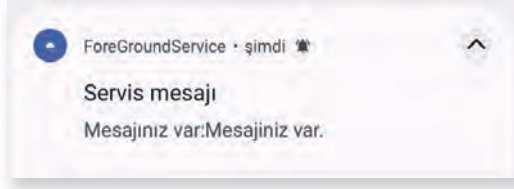
14. Adım: OnServis.java dosyasını açınız.

15. Adım: onStartComand dosyasını şu şekilde düzenleyiniz:

```
@RequiresApi(api = Build.VERSION_CODES.O)
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    String mesaj=intent.getStringExtra("mesaj");

    Intent bildirimIntent=new Intent(this,MainActivity.class);
    PendingIntent pendingIntent=PendingIntent.getActivity(this,0,
            bildirimIntent,PendingIntent.FLAG_MUTABLE);
    Notification notification=new NotificationCompat.Builder(this,"ForeGroundSer-
viceK1")
            .setContentTitle("Servis mesajı")
            .setContentText("Mesajınız var:"+mesaj)
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentIntent(pendingIntent)
            .build();
    startForeground(1,notification);
    return START_NOT_STICKY;
}
```

16. Adım: Uygulamayı çalıştırınız. Text kutusuna bir mesaj yazıp SERVİSİ BAŞLAT butonuna basınız.

17. Adım: Bildirim ekranında Görsel 8.23'teki bildirimi alıp almadığınızı kontrol ediniz.



Görsel 8.23: Ön Plan Servis uygulaması bildirim ekranı

8.5.3. Sticky Servislerle Çalışmak

Bir işlemin sürekli olarak çalışması gereken durumda Sticky servisler kullanılır. onStartCommand metodu servis kapatılıncaya kadar sürekli çalışır. Bu tür servisler genellikle müzik oynatıcılarda kullanılır. Uygulama ekranı kapatılsa bile arka planda müzik çalmaya devam eder. Uygulama, sistem belleğinden tamamen kaldırıldığında servis de işletim sistemi tarafından kapatılır.

14. UYGULAMA: İşlem adımlarına göre bir servis oluşturarak herhangi bir müzik dosyası çalan uygulamayı yazınız.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "StickyServisApp" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: main_activity.xml dosyasını Görsel 8.24'teki gibi düzenleyiniz.

Görsel 8.24: Sticky Servis App uygulaması tasarım ekranı

4. Adım: Butonlara btnBaslat ve btnDur id adlarını veriniz.

5. Adım: MainActivity.java dosyasını açıp viewBinding nesnesi tanımlayınız ve viewBinding için gerekli ayarlamaları yapınız.

6. Adım: StickyServis.java dosyası oluşturunuz ve sınıfı Service sınıfından türetiniz.

7. Adım: Manifest dosyasını açarak servisi kaydediniz.

8. Adım: MainActivity.java dosyasını açınız ve butonlar için onClickListener tanımlayınız.

9. Adım: btnBaslat butonunun onClick olayını şu şekilde yazınız:

```
Intent intent=new Intent(MainActivity.this,StickyServis.class);
startService(intent);
```

10. Adım: btnDur butonunun onClick olayını şu şekilde yazınız:

```
Intent intent=new Intent(MainActivity.this,StickyServis.class);
stopService(intent);
```

11. Adım: StickyServis.java dosyasını açınız. onStartCommand metodunu şu şekilde yazınız:

```
MediaPlayer mediaPlayer;
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    mediaPlayer = MediaPlayer.create(this, Settings.System.DEFAULT_RINGTONE_
URI);
    mediaPlayer.setLooping(true);
    mediaPlayer.start();
    return START_STICKY;
}
```

12. Adım: onDestroy metodunu şu şekilde yazınız:

```
@Override
public void onDestroy() {
    mediaPlayer.stop();
    super.onDestroy();
}
```

13. Adım: Uygulamayı çalıştırınız.

14. Adım: Çalışan uygulamada önce BAŞLAT butonuna basınız. Uygulama ekranını kapattığınızda hâlâ sesin gelip gelmediğini kontrol ediniz.

8.5.4. Bound Servislerle Çalışmak

Bound servisler bir aktivite ile bağlı olan servis türüdür. Servis ve aktivite birbirine bağlandıktan sonra aktivite yok oluncaya kadar servis kullanılmaya devam eder. Servis nesnesi aktivite nesnesinde tanımlandığı için veri almak çok kolaydır.

Bir Bound servis metoduna aktiviteden şu şekilde ulaşılabilir:

```
public String veriAl(){
    Random r=new Random();
    int sayi=r.nextInt(1000);
    return ""+sayi;
}
```

Aktiviteden veriAl metodu çağrılarak veri alınabilir. Bound servisler, ServiceConnection nesnesi

kullanılarak bağlanır. Ayrıca startService metodu ile de çalıştırılabilir. Servis, ServiceConnection nesnesi ile bağlandıktan sonra stopService ile durdurulmak istense bile çalışmaya devam eder. Activity ne zaman bağlantıyı keserse servis o zaman durur. ServiceConnection nesnesi tanımlandıktan sonra bindService metodu ile servis nesnesi çalıştırılır. unbindService metodu kullanılarak da servisin bağlantısı kesilir.

Bound servisler diğer servislerden farklı olarak bir Inner Class ile beraber şu şekilde tanımlanır:

```
public class BaglayiciSinif extends Binder {
    public BoundServis servisGetir(){
        return BoundServis.this;
    }
}
public IBinder baglayiciSinif=new BaglayiciSinif();
@Nullable
@Override
public IBinder onBind(Intent intent) {
    Log.d("SERVİS", "onBind: ");
    return baglayiciSinif;
}
```

Inner Class bir bağlayıcı sınıf olarak iş görür. Bağlayıcı sınıf, ServiceConnection nesnesi oluşturulurken kullanılır. Bu sınıf aslında bir ara sınıftır. Sadece servis sınıfının çalışan örneğinin referansını almayı sağlar.

ServiceConnection sınıfı şu şekilde kullanılarak servis bağlantısı yapılır:

```
if(serviceConnection==null){
    serviceConnection=new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder
iBinder) {
            BoundServis.BaglayiciSinif baglayiciSinif=
                                (BoundServis.BaglayiciSinif) iBinder;
            boundServis=baglayiciSinif.servisGetir();
            servisBaglimi=true;
        }
        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            servisBaglimi=false;
        }
    };
```

Booelan bir değişken kullanılarak servisin bağlı olup olmadığı kontrol edilir ve aktivite içinde gerekli işlemler yapılır.

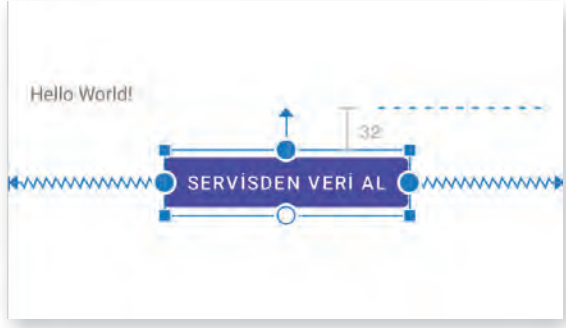
Bound servislerde servise işlem yaptırmak için startService kullanılmalıdır. ServiceConnection ile sadece servise bağlanılır, serviste onStartCommand metodu çalışmaz. onStartCommand metodunun çalışması için servisin startService ile başlatılması gereklidir.

15. UYGULAMA: İşlem adımlarına göre bir Bound servis yazarak servisten veri alan uygulamayı geliştiriniz.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "BoundServiceApp" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3.Adım: main_activity.xml dosyasını Görsel 8.25'teki gibi düzenleyiniz.



Görsel 8.25: Bound Servis uygulaması tasarım ekranı

4. Adım: TextView için textView, Button için btnVerial id isimlerini veriniz.

5. Adım: MainActivity.java dosyasını açıp viewBinding nesnesi tanımlayınız ve viewBinding için gerekli ayarlamaları yapınız.

6. Adım: BoundServis.java dosyası oluşturunuz ve sınıfı Service sınıfından türetiniz.

7. Adım: BoundServis.java dosyasını şu şekilde düzenleyiniz:

```
public class BoundServis extends Service {
    public class BaglayiciSinif extends Binder {
        public BoundServis servisGetir(){
            return BoundServis.this;
        }
    }
    public IBinder baglayiciSinif=new BaglayiciSinif();
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        Log.d("SERVİS", "onBind: ");
        return baglayiciSinif;
    }
    public String veriAl(){
        Random r=new Random();
        int sayi=r.nextInt(1000);
        return ""+sayi;
    }
    @Override
    public void onCreate() {
```

```
        super.onCreate();
        Log.d("SERVİS", "onCreate: ");
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d("SERVİS", "onDestroy: ");
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d("SERVİS", "onStartCommand: ");
        return super.onStartCommand(intent, flags, startId);
    }
}
```

8. Adım: Manifest dosyasını açarak servisi kaydediniz.

9. Adım: MainActivity.java dosyasını açınız. onCreate olayına şu kodları yazınız:

```
if(serviceConnection==null){
    serviceConnection=new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder
iBinder) {
            BoundServis.BaglayiciSinif baglayiciSinif=
                    (BoundServis.BaglayiciSinif) iBinder;
            boundServis=baglayiciSinif.servisGetir();
            servisBaglimi=true;
        }
        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            servisBaglimi=false;
        }
    };
    Intent intent=new Intent(this,BoundServis.class);
    bindService(intent,serviceConnection,BIND_AUTO_CREATE);
}
```

10. Adım: MainActivity.java dosyasında btnVerial butonu için bir onClickListener tanımlayınız.

11. Adım: onClick olayını şu şekilde yazınız:

```
public void onClick(View view) {
    if(servisBaglimi){
        binding.textView.setText(boundServis.veriAl());
    }
}
```

12. Adım: MainActivity.java dosyasında onDestroy olayına şu kodları yazınız:

```
protected void onDestroy() {
    super.onDestroy();
    unbindService(serviceConnection);
}
```

13. Adım: Uygulamayı çalıştırınız. Butona tıklayarak servisten bir veri alınız.

14. Adım: Uygulamayı kapatınız ve uygulamanın hareketlerini Logcat penceresinde izleyiniz.

15. Adım: LogCat penceresinde Görsel 8.26'daki değişikliklerin olup olmadığını kontrol ediniz.

```
D/SERVİS: onCreate:
D/SERVİS: onBind:
D/SERVİS: onDestroy:
```

Görsel 8.26: Bound Servis uygulaması LogCat çıktısı

8.5.5. IntentServislerle Çalışmak

Android servisleri bazı işlemleri kolaylaştırır ancak servislerden doğrudan veri alınması çok zordur. Bu sorunun daha kolay çözülmesini sağlayan IntentServis türüdür. IntentServisler oluşan sonucu olduğu gibi View nesnesine verebilir. IntentServislerde uzun süren işlemler için ayrıca bir iş parçacığı oluşturulmasına gerek yoktur. IntentServislerde tüm işlemler ayrı bir iş parçacığında yapılır.

IntentServislerin sadece onHandleIntent metodu vardır. Bu servisler de normal servis gibi startService ile başlatılır, stopService ile durdurulur.

IntentServisler bilgileri geriye ResultReceiver nesnesi ile gönderir. ResultReceiver, aktivite içinde Inner Class olarak şu şekilde tanımlanır:

```
public class MyResultReceiver extends ResultReceiver{
    public MyResultReceiver(Handler handler) {
        super(handler);
    }
    @Override
    protected void onReceiveResult(int resultCode, final Bundle resultData) {
        if(resultCode==1 && resultData!=null){
            handler.post(new Runnable() {
                @Override
                public void run() {
                    String resultString=resultData.getString("data");
                    binding.textView2.setText("Sonuç:"+resultString);
                }
            });
        }
```

```
        super.onReceiveResult(resultCode, resultData);
    }
}
```

ResultReceiver, servis içinde şu şekilde veri gönderir:

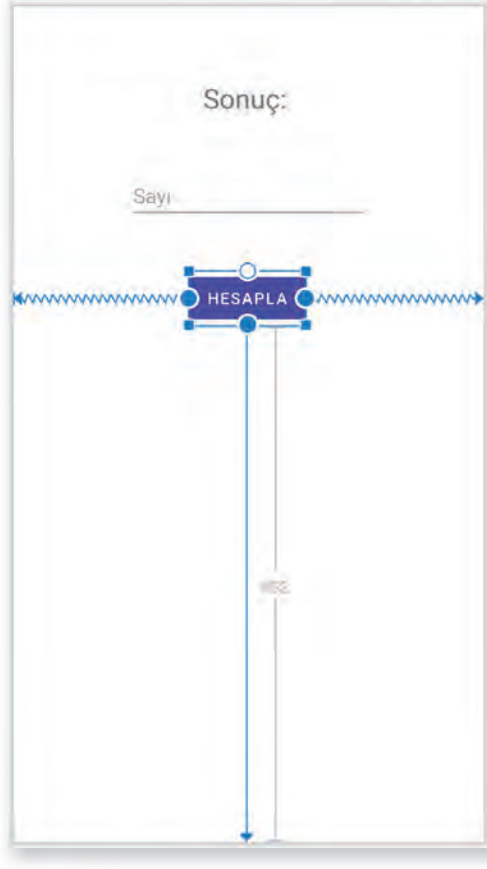
```
ResultReceiver resultReceiver=intent.getParcelableExtra("receiver");
Bundle bundle=new Bundle();
bundle.putString("data",r.nextInt(10)*sayi+"");
resultReceiver.send(1,bundle);
```

16. UYGULAMA: İşlem adımlarına göre bir IntentServis yazarak activity ile iletişim kurmasını sağlayınız. EditText kutusuna yazılan sayıyı servise gönderip, serviste rastgele bir sayı ile çarparak sonucu geri alan uygulamayı yazınız.

1.Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "IntentServiceApp" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: main_activity.xml dosyasını Görsel 8.27'deki gibi düzenleyiniz.



Görsel 8.27: IntentService uygulaması tasarım ekranı

4. Adım: Butonlara btnVerial, TextView nesnesine textSonuc, EditView nesnesine textSayi id adlarını veriniz.

5. Adım: MainActivity.java dosyasını açıp viewBinding nesnesi tanımlayınız ve viewBinding için gerekli ayarlamaları yapınız.

6. Adım: intentServis.java dosyasını oluşturunuz ve sınıfı IntentService sınıfından türetiniz.

7. Adım: Manifest dosyasını açarak servisi kaydediniz.

8. Adım: MainActivity.java dosyasını açıp şu şekilde Inner Class oluşturunuz:

```
public class MyResultReceiver extends ResultReceiver{
    public MyResultReceiver(Handler handler) {
        super(handler);
    }
    @Override
    protected void onReceiveResult(int resultCode, final Bundle resultData) {
        if(resultCode==1 && resultData!=null){
            handler.post(new Runnable() {
                @Override
                public void run() {
                    String resultString=resultData.getString("data");
                    binding.textSonuc.setText("Sonuç:"+resultString);
                }
            });
        }
        super.onReceiveResult(resultCode, resultData);
    }
}
```

9. Adım: MainActivity.java dosyasında btnVerial butonu için onClickListener tanımlayınız.

10. Adım: onClick olayını şu şekilde yazınız:

```
MyResultReceiver myResultReceiver=new MyResultReceiver(null);
Intent intent=new Intent(MainActivity.this, intentService.class);
intent.putExtra("sayi",binding.textSayi.getText().toString());
intent.putExtra("receiver",myResultReceiver);
startService(intent);
```

11. Adım: intentServis.java dosyasını açıp şu şekilde oluşturunuz:

```
public class intentService extends IntentService {
    public intentService() {
        super("");
    }
    @Override
    protected void onHandleIntent(@Nullable Intent intent) {
        Random r=new Random();
        ResultReceiver resultReceiver=intent.getParcelableExtra("receiver");
```

```
        String s=intent.getStringExtra("sayi");
        int sayi=Integer.parseInt(s);
        Bundle bundle=new Bundle();
        bundle.putString("data",r.nextInt(10)*sayi+"");
        resultReceiver.send(1,bundle);
    }
}
```

12. Adım: Uygulamayı çalıştırınız ve sonuçlarını gözlemleyiniz.

SIRA SİZDE: Uygulamanıza iki adet EditText yerleştiriniz. EditText nesnelerine yazılan sayıları IntentServise gönderip serviste matematiksel işlem yaptırınız. Sonucu ana ekrana alarak bir TextView nesnesine yazdırınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Ekran tasarımına iki adet EditText ekledi.		
3. Ekran tasarımına bir tane Button ekledi.		
4. IntentServisi yazdı.		
5. Verileri servise gönderip sonucu aldı.		

8.6. WORKMANAGERLE ÇALIŞMAK

Bazı uygulamalarda periyodik olarak bazı görevlerin yapılması gerekebilir. Bu tür işlemleri yapabilmek için mobil uygulama geliştirme ortamında birçok araç bulunur. WorkManager, zamanlanmış görev işlemlerinin çok kolay bir şekilde yapılması için oldukça uygundur. WorkManager; ertelenebilir görevler, anında yapılan görevler, periyodik olarak çalışan görevler olmak üzere üç türlü çalışmayı sağlar.

WorkManager basit ve tutarlı olmasının yanında çeşitli kurallar belirlenerek de çalıştırılabilir. Planlanmış tüm görevler cihazda bir veri tabanına kaydedilir. Zamanı geldiğinde tüm görevler kısıtlara uygun ise çalıştırılır.

WorkManager kullanabilmek için build.gradle dosyasının dependencies bölümüne şu kodlar eklenmelidir:

```
def work_version = "2.7.1"
implementation "androidx.work:work-runtime:$work_version"
```

WorkerManager, Worker sınıfından türemiş iş sınıflarını kullanır. Tüm işler bu sınıfta yapılır. Worker sınıfı şu şekilde tanımlanır:

```
public class YedekWorker extends Worker {
   public UploadWorker(
       @NonNull Context context,
       @NonNull WorkerParameters params) {
       super(context, params);
   }
   @Override
   public Result doWork() {
     logYedekle();
     return Result.success();
   }
}
```

Worker sınıfında asıl işi yapan doWork metodudur. Görevde gerekli tüm işler burada yapılır. Yapılması gereken tüm işlemler geliştirici tarafından belirtilmelidir. Görev sonunda bir sonuç gönderilmelidir. Görev sonuç çeşitleri şunlardır:

- **Result.success():** Görev başarılı olmuştur.
- **Result.failure():** Görev başarısız olmuştur.
- **Result.retry():** Görev başarısız olmuştur ancak tekrar başlatılabilir.

Görev tanımladıktan sonra WorkRequest nesnesi tanımlanır. WorkRequest, görev türünün nasıl olacağını belirler. Tek seferlik görev için OneTimeWorkRequest nesnesi, periyodik görevler için PeriodicWorkRequest nesnesi seçilir. Bir WorkRequest nesnesi şu şekilde tanımlanır:

```
WorkRequest yedekleWorkRequest =
   new OneTimeWorkRequest.Builder(yedekWorker.class)
       .build();
```

Günde bir defa çalışacak periyodik bir görev şu şekilde oluşturulur:

```
WorkRequest yedekPerioadic=new PeriodicWorkRequest.Builder(
  yedekWork.class,1,TimeUnit.DAYS)
       .build();
```

WorkManager nesnesine WorkRequest nesnesi verilebilir. WorkManager şu şekilde oluşturulur:

```
WorkManager
    .getInstance(myContext)
    .enqueue(uploadWorkRequest);
```

WorkManager, görev isteklerini enquee metodu ile listeye ekler. Aynı anda birden fazla görev tanımlanabilir. Her görev için bir id numarası otomatik olarak verilir. Görevleri karıştırmamak için görevlere etiket şu şekilde verilir:

```
WorkRequest yedekWorkRequest =
       new OneTimeWorkRequest.Builder(yedekWork.class)
       .addTag("yedekle")
       .build();
```

Görevler id numaraları ile çağrıldığı gibi etiketleri ile de çağrılabilir. Görevlerin id ve etiket ile çağrılması şu şekildedir:

```
WorkManager.getInstance(this).cancelWorkById(yedekleWorkRequest.getId());
WorkManager.getInstance(this).cancelAllWorkByTag("yedekle");
```

8.6.1. Kısıtlamaları Tanımlamak

Görevler tanımlanırken görevin çalışması esnasında bazı kısıtlar tanımlanabilir. Örneğin görevin düşük batarya seviyesinde çalışması istenmeyebilir veya sunucuya yedek gönderilirken ağa bağlı olması istenebilir. Bu tür kısıtlar koymak için Constraints nesnesi kullanılır. Bir Constraints nesnesi şu şekilde oluşturulur:

```
Constraints constraints = new Constraints.Builder()
       .setRequiredNetworkType(NetworkType.CONNECTED)
       .setRequiresCharging(true)
       .build();
WorkRequest myWorkRequest =
       new OneTimeWorkRequest.Builder(MyWork.class)
              .setConstraints(constraints)
              .build();
```

Birçok kısıtlama özelliği vardır. Tüm kısıtlamalara https://developer.android.com/reference/androidx/work/Constraints.Builder internet sitesinden bakılabilir. Tablo 8.1'de önemli bazı kısıtlamalar verilmiştir.

Tablo 8.1: Workrequest Nesnesi Kısıtları

Kısıt	Açıklama
setRequiresCharging	Cihaz şarja bağlıysa görevin yapılmasını sağlar.
setRequiresBatteryNotLow	Cihazın bataryası azsa görevi başlatmaz.
setRequiresDeviceIdle	Cihaz hiçbir işlem yapmadığı zaman görevin yapılmasını sağlar.
setRequiresStorageNotLow	Cihazda yeterince depolama alanı varsa görevin yapılmasını sağlar.
setRequiredNetworkType	Cihaz istenen ağa bağlıysa görevin yapılmasını sağlar.

8.6.2. Görevi Gecikmeli Başlatmak

Görevler istendiği kadar gecikmeli başlatılabilir. Görevler gecikmeli olarak şu şekilde başlatılır:

```
WorkRequest yedekWorkRequest =
      new OneTimeWorkRequest.Builder(yedekWork.class)
              .setInitialDelay(10, TimeUnit.MINUTES)
              .build();
```

Görevlerde zaman birimleri TimeUnit ile belirlenir. Gecikme tek seferlik görevler için uygulanır, periyodik görevlerde ise sadece ilk görev ertelenir. Tablo 8.2'de tüm TimeUnit birimleri verilmiştir.

Tablo 8.2: Periyodik Görevlerde Kullanılabilen Zaman Türleri

Birim	Açıklama
NANOSECONDS	Nanosaniye
MICROSECONDS	Mikrosaniye
MILLISECONDS	Milisaniye
SECONDS	Saniye
MINUTES	Dakika
HOURS	Saat
DAYS	Gün

8.6.3. Zincirleme Çalışmak

WorkManager nesnesi ile zincirleme işlemler yapılabilir. Bir görev başlatıldıktan sonra görevin bitmesi takip edilerek başka bir görevin çalıştırılması sağlanabilir. Zincir görevler sadece OneTimeWorkRequest görevleri ile yapılır. Periyodik görevler zincire eklenemez. Zincir bir görev şu şekilde oluşturulur:

```
WorkManager.getInstance(this).beginWith((OneTimeWorkRequest) p1WorkRequest)
        .then((OneTimeWorkRequest) p2WorkRequest)
        .then((OneTimeWorkRequest) p3WorkRequest)
        .enqueue();
```

Zincirleme görevlerde ilk görev bitinceye kadar diğer görevlerin talepleri engellenir. Görevler Result.success() ile biterse bir sonraki göreve geçilir. Herhangi bir görev başarısız olursa sonraki göreve geçilmez. Zincir görev esnasında sırası gelen görev, kısıtlardan birine takılırsa beklemeye alınır. Sonraki göreve geçilmez.

8.6.4. Görevleri İzlemek

Görevleri izlemek için WorkManager nesnesinin observe özelliği kullanılır. Bu özellik, mevcut tüm görevler hakkında bilgi alınmasını sağlar. Hangi görevin ne aşamada olduğu observe sayesinde görüntülenir. Observer özelliği şu şekilde kullanılır:

```
WorkManager.getInstance(this).getWorkInfoByIdLiveData(myWorkRequest.getId()).
observe(this,
        new Observer<WorkInfo>() {
    @Override
    public void onChanged(WorkInfo workInfo) {
        if(workInfo.getState()== WorkInfo.State.BLOCKED)
            Toast.makeText(MainActivity.this, "Görev bloklandı",
             Toast.LENGTH_SHORT).show();
    }
});
```

Görevler ile ilgili tüm bilgilere WorkInfo nesneleri ile ulaşılabilir.

8.6.5. Görevlere Dışarıdan Veri Göndermek

Görevlere dışarıdan veri göndermek için Data sınıfı kullanılır. Data sınıfı ile göreve veri şu şekilde gönderilir:

```
Data data=new Data.Builder().putInt("deger",100).build();
WorkRequest myPerioadic=new PeriodicWorkRequest.Builder(MyWork.class,
        1, TimeUnit.DAYS)
        .setInputData(data)
        .build();
```

Veriyi görev içinde almak için şu kodlar kullanılır:

```
public Result doWork() {
    Data data=getInputData();
    int deger=data.getInt("deger",0);
    return Result.success();
}
```

17. UYGULAMA: İşlem adımlarına göre bir WorkManager tanımlayarak Worker sınıfına rastgele bir sayı gönderiniz. Görev isteği için şu kısıtları tanımlayarak WorkManager ile her 15 dakikada periyodik görevler yapan uygulamayı yazınız:

- Batarya düşük olduğunda çalışmasın.
- Cihaz şarj edilirken görev çalışsın.
- Cihaz bir ağa bağlı olduğu zaman görev çalışsın.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "WorkManagerApp" yapınız.

2. Adım: YedekWork.java dosyası oluşturunuz ve sınıfı Worker sınıfından türetiniz.

3. Adım: YedekWork.java dosyasını şu şekilde kodlayınız:

```
public class YedekWork extends Worker {
    Context context;
    int deger;
    public YedekWork(@NonNull Context context, @NonNull WorkerParameters work-
erParams) {
        super(context, workerParams);
        this.context=context;
    }
    @RequiresApi(api = Build.VERSION_CODES.O)
    public void islemYap(){
        NotificationChannel channel=new NotificationChannel("Bildirim","Görev",
                NotificationManager.IMPORTANCE_DEFAULT);
        NotificationManager notificationManager=context
                .getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
```

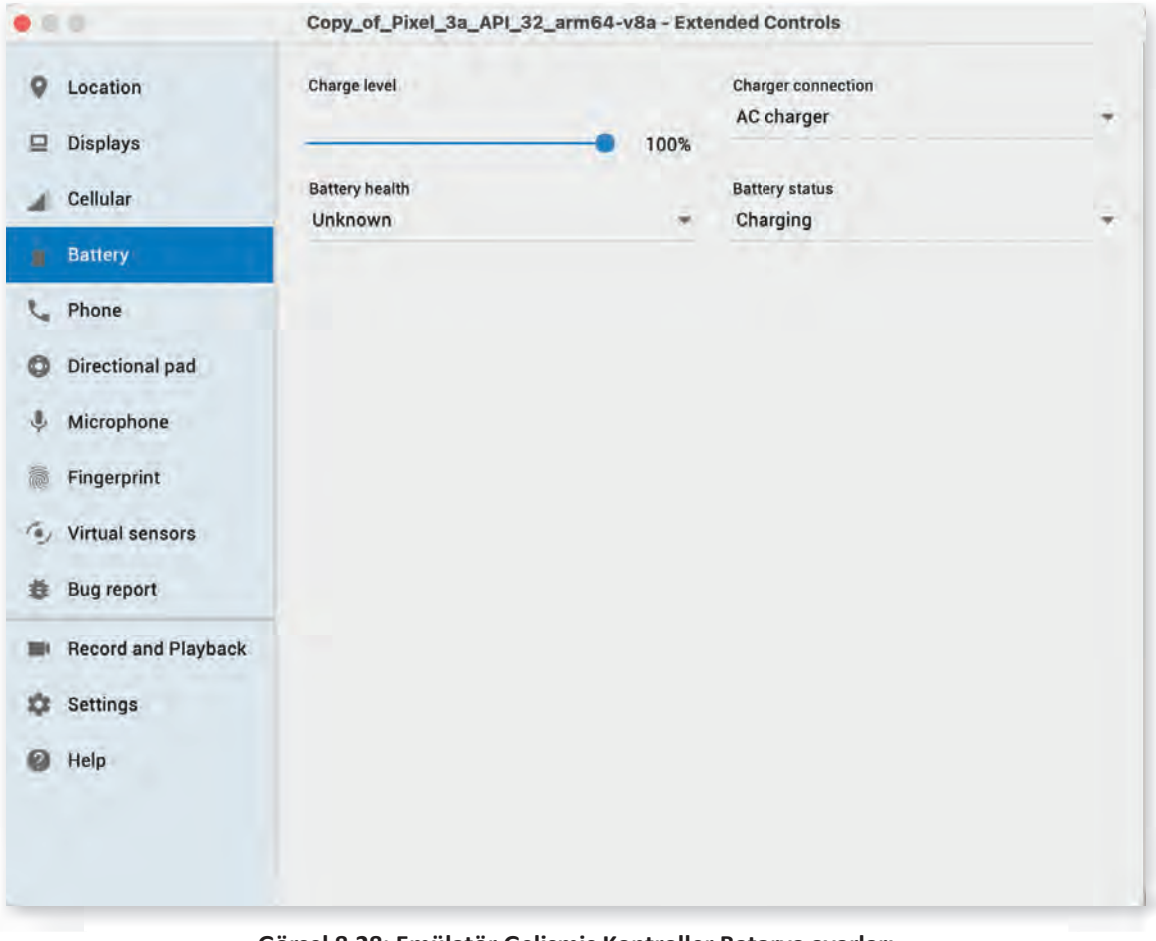
```
        Notification builder=new Notification.Builder(context,"Bildirim")
                .setContentTitle("Mesajınız var")
                .setContentText("WorkManager tarafından uyarıldınız! "+
                        deger + " değeri size gönderildi.")
                .setSmallIcon(R.drawable.ic_launcher_foreground)
                .build();
        notificationManager.notify(1,builder);
    }
    @NonNull
    @Override
    public Result doWork() {
        Data data=getInputData();
        this.deger=data.getInt("deger",0);
        return Result.success();
    }
}
```

4. Adım: MainActivity.java dosyasını açarak şu şekilde tanımlayınız:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Constraints constraints = new Constraints.Builder()
            .setRequiredNetworkType(NetworkType.CONNECTED)
            .setRequiresCharging(true)
            .setRequiresBatteryNotLow(true)
            .build();
    Random r=new Random();
    Data data=new Data.Builder().putInt("deger",r.nextInt(100)).build();
    WorkRequest yedekPeriodic=new PeriodicWorkRequest.Builder(YedekWork.class,
            15, TimeUnit.MINUTES)
            .setInputData(data)
            .setConstraints(constraints)
            .build();
    WorkManager.getInstance(this).enqueue(yedekPeriodic);
    Toast.makeText(this, "Workmanager kuruldu",
            Toast.LENGTH_LONG).show();
}
```

5. Adım: Uygulamayı çalıştırıp emülatörde Gelişmiş Kontroller penceresini açınız.

6. Adım: Battery sekmesini açarak cihazın ayarlarını Görsel 8.28'deki gibi yapınız.



Görsel 8.28: Emülatör Gelişmiş Kontroller Batarya ayarları

7. Adım: 15 dakika bekleyip bildirim çıkmasını gözlemleyiniz.

SIRA SİZDE: Bir tane OneTImeWorkRequest oluşturup zincir şeklinde üç defa çalıştıran uygulamayı yazınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. Worker sınıfından bir sınıf oluşturdu.		
3. Yeni bir OneTimeWorkRequest nesnesi oluşturdu.		
4. WorkManager nesnesi oluşturdu.		
5. Görevleri zincir olarak WorkManager nesnesine ekledi.		

8.7. SENSÖRLERLE ÇALIŞMAK

Mobil cihazlarda kullanıcının çevresindeki değişikleri ölçebilen algılayıcılar bulunur. Mobil uygulama geliştirme ortamı, sensörlerden gelen verileri alarak bu verilerin uygulama içinde kullanılmasını sağlar. Kullanılabilen sensör türleri şunlardır:

- **Hareket Sensörleri:** Bu sensörler, cihazın yer değiştirmesini tespit eder. İvmeölçer, jiroskop ve yerçekimi sensörleri bu tür sensörlerdendir.
- **Çevre Sensörleri:** Bu sensörler; sıcaklık, basınç, ışık ve nem değişimlerini tespit eder.
- **Konum Sensörleri:** Bu sensörler, cihazın konumunu tespit eder. Gerekli bilgileri manyetik sensörlerden alır.

Mobil uygulama geliştirme ortamında sensör işlemleri SensorManager sınıfı ile yapılır. SensorManager şu şekilde tanımlanır:

```
SensorManager sensorManager;
sensorManager= (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

Bir ışık sensörü şu şekilde tanımlanır:

```
Sensor sensorIsik;
sensorIsik=sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Sensör tipleri Sensor sınıfında tanımlanmıştır. Sensor.TYPE_LIGHT ile sensörün bir ışık sensörü belirtilir. Mobil uygulama geliştirme ortamının desteklediği sensör tiplerine https://developer.android.com/reference/android/hardware/Sensor adresinden bakılabilir.

Mobil uygulama geliştirme ortamında bulunan sensörlerin bazıları yazılım tabanlı bazıları ise donanım tabanlıdır (Tablo 8.3). Yazılım tabanlı sensörler, donanım tabanlı sensörleri taklit eder. Yazılım tabanlı sensörler, hız ve kesinlik olarak donanım tabanlı sensörler kadar iyi değildir.

Tablo 8.3: Mobil Uygulama Geliştirme Ortamında En Çok Kullanılan Sensörler

Sensör	Tip	Tanım	Ortak Kullanım Alanları
TYPE_ACCELEROMETER	D	Cihazın x, y, z eksenini ve ivme kuvvetini tespit eder.	Hareket algılama
TYPE_AMBIENT_TEMPERATURE	D	Ortam oda sıcaklığını santigrat derece cinsinden ölçer.	Sıcaklık
TYPE_GRAVITY	Y/D	Bir cihaza x, y, z eksenlerinde uygulanan yerçekimini tespit eder.	Hareket algılama
TYPE_GYROSCOPE	D	Bir aygıtın x, y, z eksenlerinde herbirinin kendi etrafında dönme hızını algılar.	Dönme algılama
TYPE_LIGHT	D	Ortam ışık seviyesini Lux cinsinden ölçer.	Ekran parlaklığı
TYPE_LINEAR_ACCELERATION	Y/D	Yerçekimi kuvveti hariç x, y, z eksenlerinde oluşan ivmeyi ölçer.	İvmeölçer

TYPE_MAGNETIC_FIELD	D	Cihazın x, y, z eksenleri için ortamdaki jeomanyetik alanı µT cinsinden ölçer.	Pusula
TYPE_ORIENTATION	Y	Bir aygıtın x, y, z tümü etrafında yaptığı dönüş derecelerini ölçer.	Konum
TYPE_PRESSURE	D	Ortam hava basıncını hPa veya mbar cinsinden ölçer.	Basınç
TYPE_PROXIMITY	D	Bir cihazın görüntüleme ekranına göre bir nesnenin yakınlığını cm cinsinden ölçer.	Telefon konumu
TYPE_RELATIVE_HUMIDITY	D	Bağıl ortam nemini yüzde (%) olarak ölçer.	Nem
TYPE_ROTATION_VECTOR	Y/D	Cihazın dönüş vektörünün üç ögesini sağlayarak bir cihazın yönünü ölçer.	Hareket algılama
TYPE_TEMPERATURE	D	Cihazın sıcaklığını santigrat derece cinsinden ölçer.	Sıcaklık

8.7.1. Tüm Sensörlere Erişmek

Sistemde tanımlı tüm sensörler şu şekilde alınır:

```
List<Sensor> deviceSensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
```

Sensör listesi alındıktan sonra cihazın hangi sensörleri destekleyip desteklemediği tespit edilir. Cihazda hangi sensörlerin desteklendiği şu şekilde belirlenir:

```
SensorManager sensorManager;
Sensor sensor;
    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    sensor = null;

    if (sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY) != null) {
         List<Sensor> gravSensors = sensorManager.getSensorList(Sensor.TYPE_GRAVITY);
       for (int i = 0; i < gravSensors.size(); i++) {
           if ((gravSensors.get(i).getVendor().contains("Google LLC")) &&
                   (gravSensors.get(i).getVersion() == 3)) {
               sensor = gravSensors.get(i);
           }
       }
   }
   if (sensor == null) {
       if (sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null) {
           sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
       } else {
           Toast.makeText(this, "Sensör desteklenmiyor",
```

```
                    Toast.LENGTH_SHORT).show();
            }
        }
}
```

Bir sensörden veri almak için öncelikle cihazın, sensörü destekleyip desteklemediği tespit edilmelidir. Cihaz, sensörü desteklemezse uygulama çökebilir.

8.7.2. Sensör Olaylarıyla Çalışmak

Sensör olayları kullanılarak sensörlerden gelen veriler eş zamanlı olarak elde edilir. Gelen verileri sağlıklı bir şekilde kullanmak için SensorEventListener arabirimi uygulanır. Arabirim ile onAccuracyChanged ve onSensorChanged metotları eklenir.

onAccuracyChanged olayı, sensörün doğruluğu değişirse çalışır. onSensorChanged olayında ise sensör verisi değiştiğinde metot çalışır.

Sensörler olaylara şu şekilde kaydedilir:

```
@Override
protected void onResume() {
    super.onResume();
    sensorManager.registerListener(this, mLight,
    SensorManager.SENSOR_DELAY_NORMAL);
}
@Override
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
}
```

Kayıt işleminde üçüncü parametre olarak sensörün çalışma sıklığı belirlenir. Sensörler ne kadar yüksek çalışma sıklığında ölçüm yaparsa daha hatasız sonuçlar elde eder. Ancak yüksek çalışma sıklığı pilin çok çabuk bitmesine neden olur. Bundan dolayı bir sensörü yüksek çalışma sıklığında çalıştırmak için özel izin almak gereklidir.

UYARI: Olayların çalışması için uygulama başladığında kayıt yapılması gereklidir. Uygulama kapandığında ise kayıt işlemi sonlandırılmalıdır. Kayıt işlemi sonlandırılmazsa sensör yine veri göndermeye devam eder. Bu durum, sensörün çok fazla pil tüketmesine neden olur.

Sensörün yüksek çalışma sıklığında olmasını sağlamak için manifest dosyasında şu değişiklik yapılarak izin alınır:

```
<uses-permission android:name="android.permission.HIGH_SAMPLING_RATE_SENSORS"/>
```

18.UYGULAMA: İşlem adımlarına göre bir ışık sensörü tanımlayarak sensörden gelen verileri okuyunuz.

1. Adım: Empty Activity şablonunu kullanarak yeni bir proje oluşturunuz. Projenin adını "LightSensorApp" yapınız.

2. Adım: build.gradle dosyasını açarak viewBinding özelliğini aktif ediniz.

3. Adım: main_activity.xml dosyasını açarak sadece bir tane TextView bırakınız. TextView nesnesine txtView id ismini veriniz.

4. Adım: MainActivity.java dosyasını açıp viewBinding nesnesi tanımlayınız ve viewBinding için gerekli ayarlamaları yapınız.

5. Adım: MainActivity.java dosyasını açarak SensorManager ve Sensor nesnelerini tanımlayınız.

6. Adım: MainActivity sınıfına SensorEventListener arabirimini şu şekilde tanımlayınız:

```
public class MainActivity extends AppCompatActivity implements SensorEventLi-
stener
```

7. Adım: SensorEventListener metotlarının sınıf içinde oluşmasını sağlayınız.

8. Adım: MainActivity.java dosyasını şu şekilde düzenleyiniz:

```
private SensorManager sensorManager;
Sensor mLight;
ActivityMainBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding=ActivityMainBinding.inflate(getLayoutInflater());
    View view=binding.getRoot();
    setContentView(view);

    sensorManager= (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    mLight=sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
}
```

9. Adım: onSensorChanged olayını şu şekilde düzenleyiniz:

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    float lux=sensorEvent.values[0];
    binding.txtView.setText(lux+" Lux");
}
```

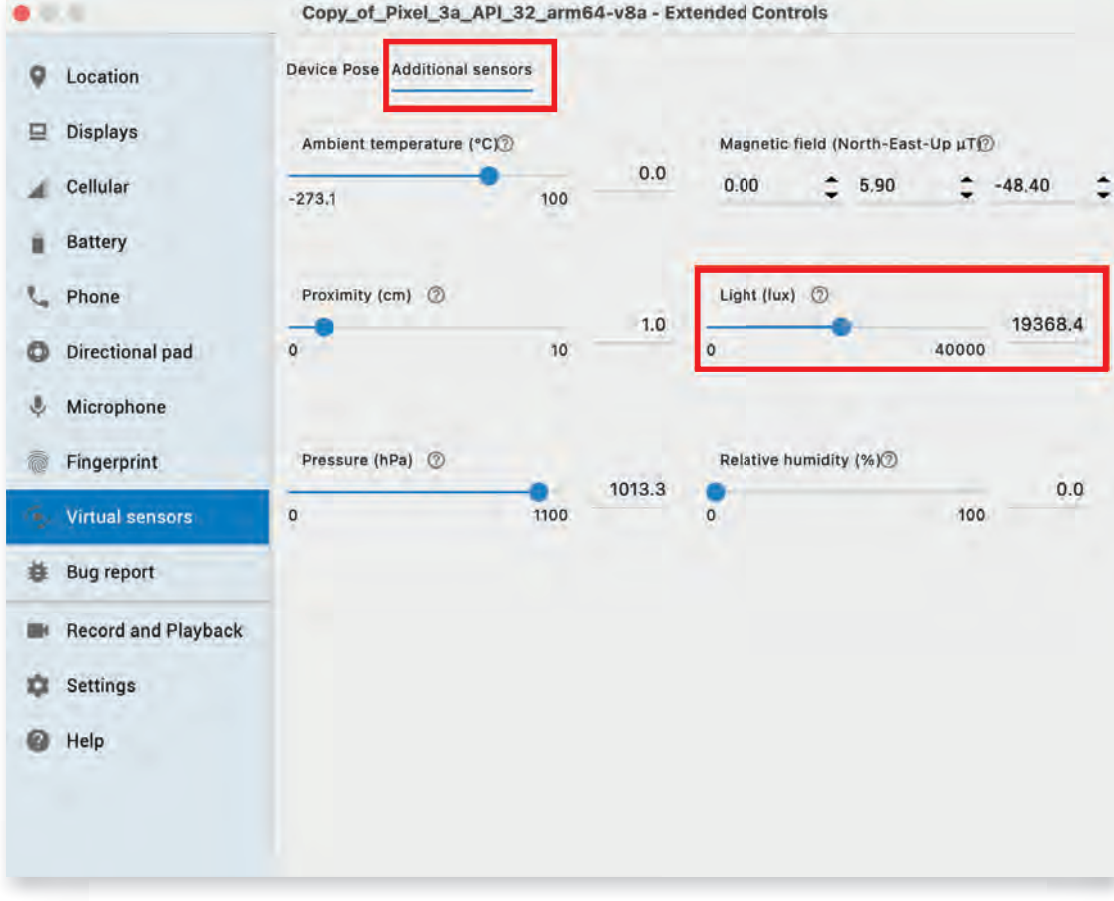
10. Adım: Sensörü şu şekilde kaydediniz:

```
@Override
protected void onResume() {
    super.onResume();
    sensorManager.registerListener(this,mLight,
    SensorManager.SENSOR_DELAY_NORMAL);
```

```
}
@Override
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
}
```

11. Adım: Uygulamayı çalıştırıp emülatörde Gelişmiş Kontroller penceresini açınız.

12. Adım: Gelişmiş Kontroller penceresinde Görsel 8.29'daki Virtual sensors sekmesini açınız. Additional sensors sekmesinde Light ayarları ile ayarlamalar yapınız.



Görsel 8.29: Emülatör Gelişmiş Kontroller penceresi Sanal Sensörler ayarları

13. Adım: Uygulamada Görsel 8.30'daki değişikliklerin olup olmadığını tespit ediniz.

Görsel 8.30: Sensör uygulaması

SIRA SİZDE: TYPE_AMBIENT_TEMPERATURE sensörünü kullanan bir uygulama yazınız.

DEĞERLENDİRME: Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

DEĞERLENDİRME ÖLÇÜTLERİ	EVET	HAYIR
1. Yeni Empty Activity ile proje oluşturdu.		
2. SensorManager tanımlandı.		
3. SensorManager ile TYPE_AMBIENT_TEMPERATURE değerlerinin dinlenmesini sağladı.		
4. SensorManager sınıfını aktiviteye kaydetti.		

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Tehlikeli izne bağlı olan servisler dinlenebilir.

2. ( ) Yayın alıcılar manifest dosyasında tanımlanmalıdır.

3. ( ) SMS alma işlemleri bir yayın alıcı ile yapılır.

4. ( ) E-posta işlemleri IntentFilter nesnesi kullanılarak yapılır.

5. ( ) Kapatılmayan servisler sonsuza kadar çalışabilir.

6. ( ) IntentServisler aktivite ile doğrudan iletişim kurabilir.

7. ( ) API 23 öncesi sistemlerde kanal nesnesi olmadan bildirim verilir.

8. ( ) Zincir görevlerde her türlü WorkerRequest nesnesi kullanılır.

9. ( ) Tek seferlik çalışan görevlerde kısıtlama kullanmak zorunlu değildir.

10. ( ) Yazılım türü sensörler daha yavaş çalışır.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcüğü yazınız.

Arka plan servisi	PendingIntent	SmsManager
Manifest	Bound	setRequiresBatteryNotLow
Donanım	BroadcastReceiver	Notification
	setSmallIcon	

11. İşletim sisteminden gelen mesajları alabilmek için ..... kullanılır.

12. Yayın alıcılar mutlaka ..... dosyasına kaydedilmelidir.

13. SMS göndermek ve almak için ..... nesnesi kullanılır.

14. En temel Android servisi ..... servisidir.

15. Activity ile bağlantı kurularak çalışan servis türü ..... servistir.

16. Bildirim yapabilmek için NotificationManager nesnesine ..... verilmelidir.

17. Bildirimlerde ..... özelliği mutlaka olmalıdır.

18. Bildirimlerde kullanıcının bildirime dokunduğu ..... ile algılanır.

19. Görevler tanımlanırken düşük batarya kısıtı ..... ile ayarlanır.

20. En hızlı sensörler ..... sensörlerdir.

C) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

21. Aşağıdakilerden hangisi bir yayın alıcı sınıfın metotlarından biridir?

A) onResume
B) onReceive
C) onKill
D) onPush
E) onDestroy

22. Aşağıdakilerden hangisi temel Android uygulamalarından biri değildir?

A) Activity
B) BroadcstReceiver
C) Service
D) Worker
E) Content Provider

23. Sürekli olarak çalışan ve stopService ile duruncaya kadar çalışan servis türü aşağıdakilerden hangisidir?

A) Broadcast
B) Arka plan servisi
C) Bound
D) Ön plan servisi
E) Sticky

24. Activityler arasında veri alışverişlerinde aşağıdaki nesnelerden hangisi kullanılır?

A) Data
B) Worker
C) Intent
D) Service
E) Bundle

25. Bildirim kanallarını ayarlayan nesne aşağıdakilerden hangisidir?

A) NotificationChannel
B) PendinIntent
C) Notification
D) WorkManager
E) Service

26. Aşağıdakilerden hangisi bir WorkManager kısıtı değildir?

A) setRequiresCharging
B) setRequiresBatteryNotLow
C) setRequiresDeviceIdle
D) setDeviceNotConnected
E) setManageCommand

27. Işık şiddetini ölçen sensörün anahtar adı aşağıdakilerden hangisidir?

A) TYPE_ORIENTATION
B) TYPE_GRAVITY
C) TYPE_GYROSCOPE
D) TYPE_ACCELEROMETER
E) TYPE_LIGHT

28. Uygulamaya vektör grafikleri eklemeyi sağlayan aracın adı aşağıdakilerden hangisidir?

A) Asset Studio
B) App Inspection
C) LogCat
D) Manifest
E) Image Studio

29. Sensör verilerini almak için SensorEventListener aşağıdaki metotlardan hangisi ile kaydedilir?

A) register
B) receiver
C) regListener
D) registerListener
E) registerSensor

30. WorkManager nesnesi, oluşturulan görevleri aşağıdaki metotlardan hangisi ile işleme alır?

A) add
B) enqueue
C) insert
D) append
E) kill

Ç) Aşağıdaki cümlelerde verilen işlemleri yapınız.

31. BroadcastReceiver oluşturmak için bir sınıf türetiniz.

32. E-posta göndermek için bir Intent oluşturup gerekli veri yazıp gönderiniz.

33. Servis.java servisini çalıştırınız.

34. Düşük pil seviyesinde çalışmayı engelleyen ve cihaz şarjdayken çalışmasını sağlayan Constraints nesnesini yazınız.

9. ÖĞRENME BİRİMİ

UYGULAMA YAYIMLAMA

KONULAR

9.1. UYGULAMA MARKET GELİŞTİRİCİ AYARLARI

9.2. UYGULAMA YAYIMLAMA

NELER ÖĞRENECEKSİNİZ?

- Oluşturulan uygulamaların imzalanmış olarak paketlenmesi
- Paketi alınmış bir uygulamanın market üzerinde paylaşım süreçleri
- Market üzerinde uygulamayı paylaşma veya teste alma işlemleri
- Uygulamayı ücretli veya ücretsiz paylaşım seçenekleri

ANAHTAR KELİMELER

- App Market
- Google Play
- Play Store
- Play Console
- Uygulama marketi
- Uygulama yayımlama

HAZIRLIK ÇALIŞMALARI

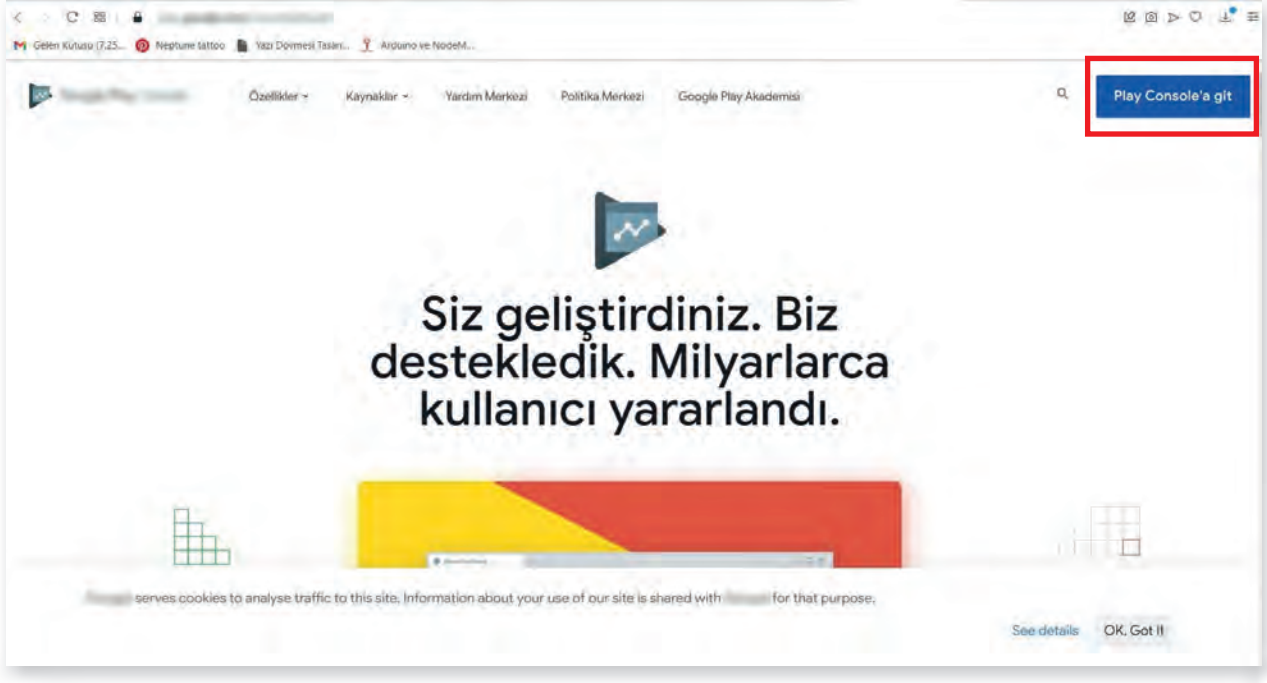
1. Cep telefonlarında ve tabletlerde bulunan ücretli ve ücretsiz uygulamalardan hatırladıklarınızın benzerliklerini ve farklılıklarını arkadaşlarınızla birlikte listeleyiniz.
2. Market üzerinde yer alan uygulama kategorileri hakkında bildiklerinizi arkadaşlarınızla paylaşınız.

9.1. UYGULAMA MARKET GELİŞTİRİCİ AYARLARI

Mobil uygulama geliştirme platformu üzerinden tasarlanan uygulamalar her zaman taslak hâlindedir. ***.apk** olarak çıktı alınmadıkça üzerinde değişiklikler yapılabilir. Geliştirilen mobil uygulamanın .apk uzantılı çıktısının alınması, başka bir Android işletim sistemine sahip cihazda kullanılmasını sağlar. Bu .apk dosyası herhangi bir yol ile başka bir Android cihaza gönderilebilir. Uygulama paketi çalışır hâldedir fakat bunun en işlevsel ve yayılmaya uygun yolu, market üzerinde uygulamanın yayımlanmasıdır.

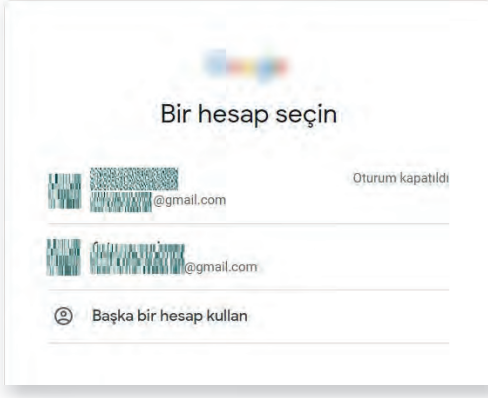
Uygulamaların yayımlanmasını sağlamak için Android market web sitesi kullanılır. Uygulama marketine ulaşılıp "console" linkine giriş yapılır. Ayrıca bir uygulamayı yayımlayabilmek için de geliştirici hesabına sahip olmak gerekir. Her oluşturulmuş hesap, potansiyel şekilde bir geliştirici olabilecek düzeydedir. Bunun için yapılması gerekenler şunlardır:

- Verilen linkten Play Console sayfasına gidilerek **"Play Console'a git"** butonuna tıklanır (Görsel 9.1).

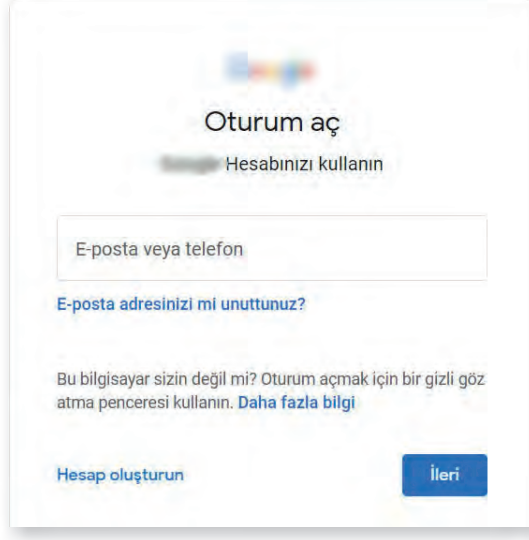


Görsel 9.1: Play Console ana sayfa ekranı

- Açılan sayfa üzerinden var olan bir hesap seçilir veya “Başka bir hesap kullan” butonuna basılır (Görsel 9.2).
- Yeni bir hesap oluşturulmak istenirse “Başka bir hesap kullan” tıklanır ve bir hesap girilir veya “Hesap oluşturun” butonuna tıklanır (Görsel 9.3).

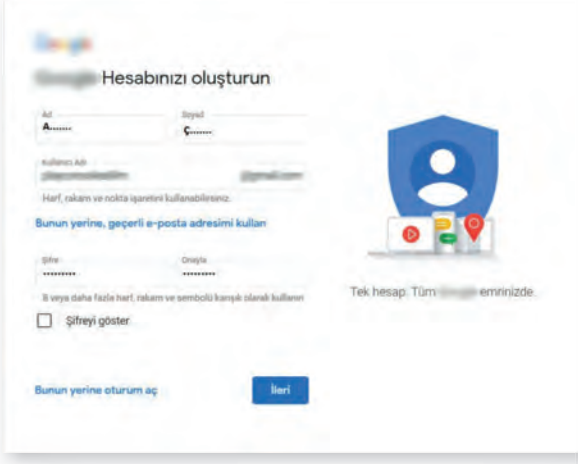


Görsel 9.2: Bir hesap seç ekranı

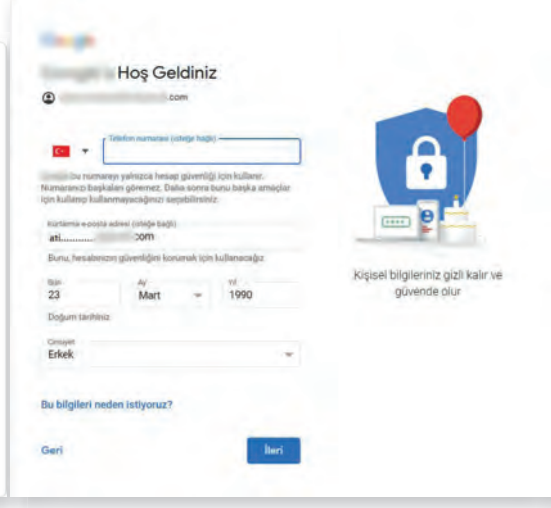


Görsel 9.3: Hesap oluşturun ekranı

- “Kendim için” seçeneği seçilir ve Görsel 9.4’teki gibi Ad, Soyad, Kullanıcı Adı ve Şifre bilgileri girilir. İleri butonuna basılır.
- İstenen bilgiler girilir ve İleri butonuna tıklanarak çıkan sözleşme kabul edilir (Görsel 9.5). Ekranda iki adımlı doğrulama çıkarsa kabul edilerek telefon numarası girilir.

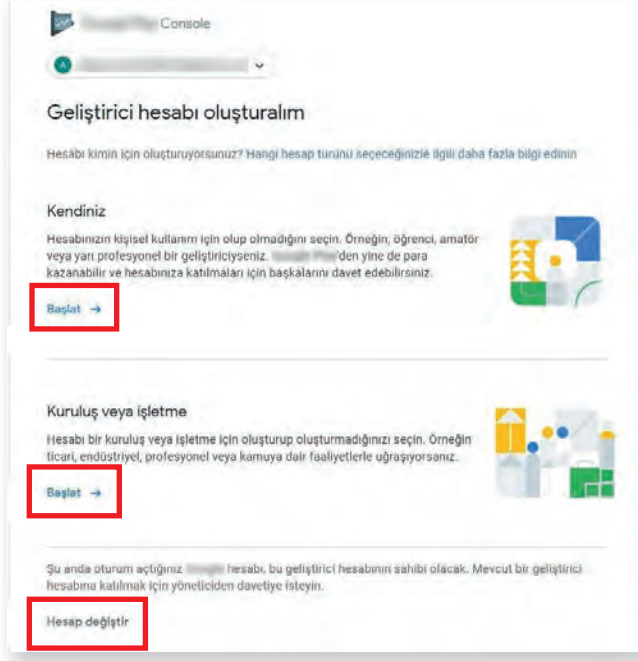


Görsel 9.4: Yeni hesap oluşturma ekranı



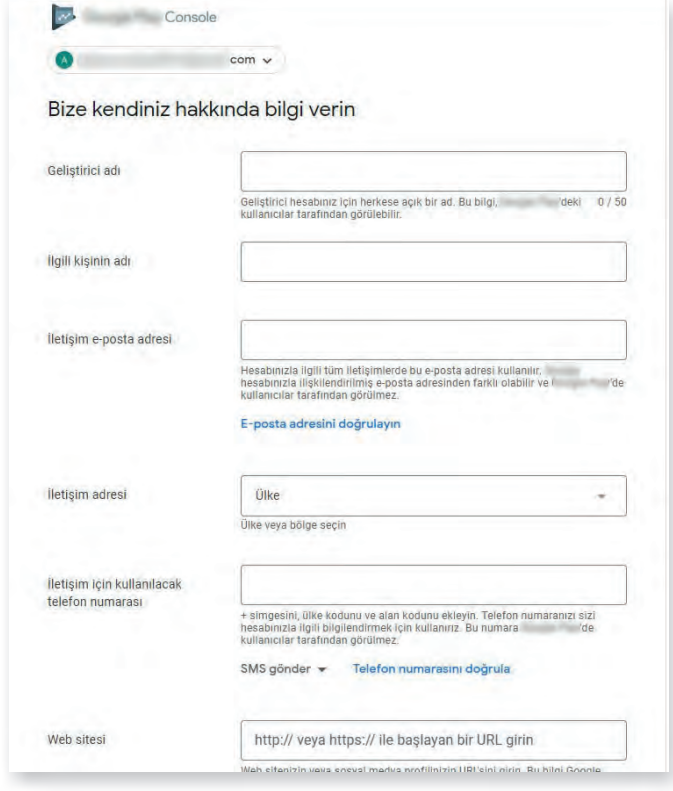
Görsel 9.5: Ek bilgiler ekranı

- Play Console için verilen linke tekrar bağlanılarak “Play Console’a git” tıklanır ve yeni oluşturulan hesap seçilir. Ardından hesabı geliştirici hesabına dönüştürme adımlarına geçilir. Görsel 9.6’da uygun olan seçeneklerden biri seçilir.



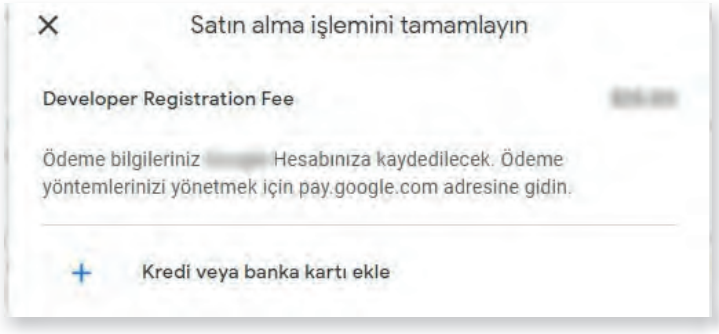
Görsel 9.6: Geliştirici hesabı oluşturma ekranı

- Kendiniz olarak "Başlat" butonuna tıklanır. Sonrasında gelen "Geliştirici Bilgi Ekranı"ndan geliştirici bilgileri girilir. Telefon numarası doğrulaması yapılır ve "Hesap oluştur ve öde" butonuna tıklanır (Görsel 9.7).



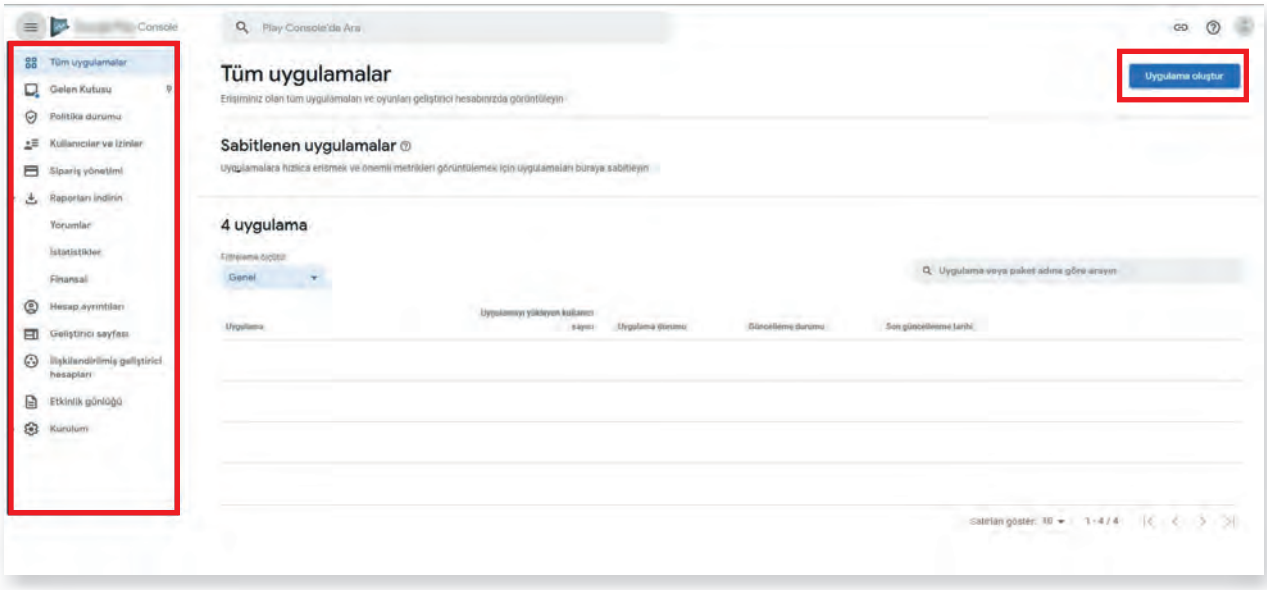
Görsel 9.7: Geliştirici bilgi ekranı

- Geliştirici hesabını tamamlamak için öğretmeniniz veya bir yetişkinle birlikte market ücreti olarak bir defaya mahsus belirtilen ücretin ödenmesi gerekir (Görsel 9.8).



Görsel 9.8: Satın alma işlem ekranı

- Satın alma işlemi sonrasında "Play Console" geliştirme ekranı açılır. Görsel 9.9'da sol tarafta menüler, sağ üst kısımda da "Uygulama oluştur" butonu bulunur.



Görsel 9.9: Play Console ekranı

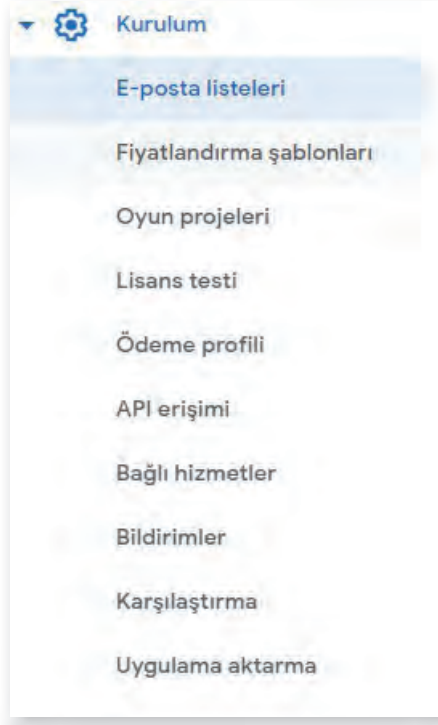
► **Gelen Kutusu:** Firma tarafından gelen mesajların yer aldığı bölümdür. Erişim hesabı olarak geliştirici hesabının kendi tanımlanmış ise uygulamaya gönderilen mesajların da bulunduğu mesaj kutusudur.

► **Politika durumu:** Firmaya ait olan politikalara uygunsuz hareketlerden kaynaklı durumlar varsa bunların yer aldığı bölümdür. Ayrıca uygulama politika ihlalleri veya uygunluğu da bu bölümden görülüp incelenebilir.

► **Kullanıcılar ve izinler:** Play Console ekranını kullanabilecek maillere ait izinler bu bölümden ayarlanır. Birden fazla hesap ile oluşturulan üyeliğe ait Play Console kullanılabilir.

► **Sipariş yönetimi:** "Payments satıcı hesabı" oluşturulduktan sonra bu bölüm kullanılabilir. Adsence gibi yöntemlerle para kazanmak için ödeme planı bu bölümden yapılmalıdır.

- **Raporları indirin:** "Yorumlar", "İstatistikler", "Finansal" adında üç raporun yer aldığı bölümdür. Raporlar Google tarafından güncel şekilde bildirilir.
- **Hesap ayrıntıları:** Kişisel veya kurumsal hesap şeklinde düzenlenip telefon, adres, mail gibi birçok bilginin yeniden oluşturulabileceği bölümdür.
- **Geliştirici sayfası:** Geliştirici Adı, Fiziksel Adresi, web sitesi, promosyonlar, geliştirici simgesi, öne çıkan uygulamalar gibi ayarların düzenlendiği seçenektir.
- **İlişkilendirilmiş geliştirici hesapları:** Bir başka geliştirici hesabı ile ilişkilendirme yapılırsa ilişkilendirilmiş o hesap hakkında bilgilerin alındığı bölümdür.
- **Kurulum:** Görsel 9.10'da görülen menülerin yer aldığı, Play Console'a ayrıntılı bilgilerin detaylı biçimde düzenlendiği bölümdür.

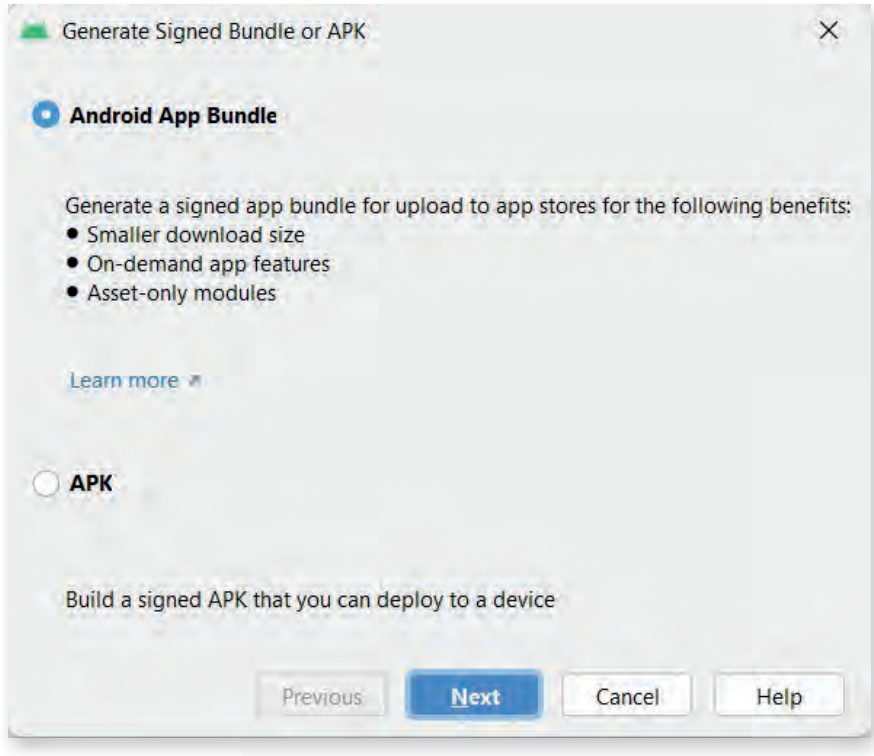


Görsel 9.10: Kurulum seçenekleri ekranı

9.2. UYGULAMA YAYIMLAMA

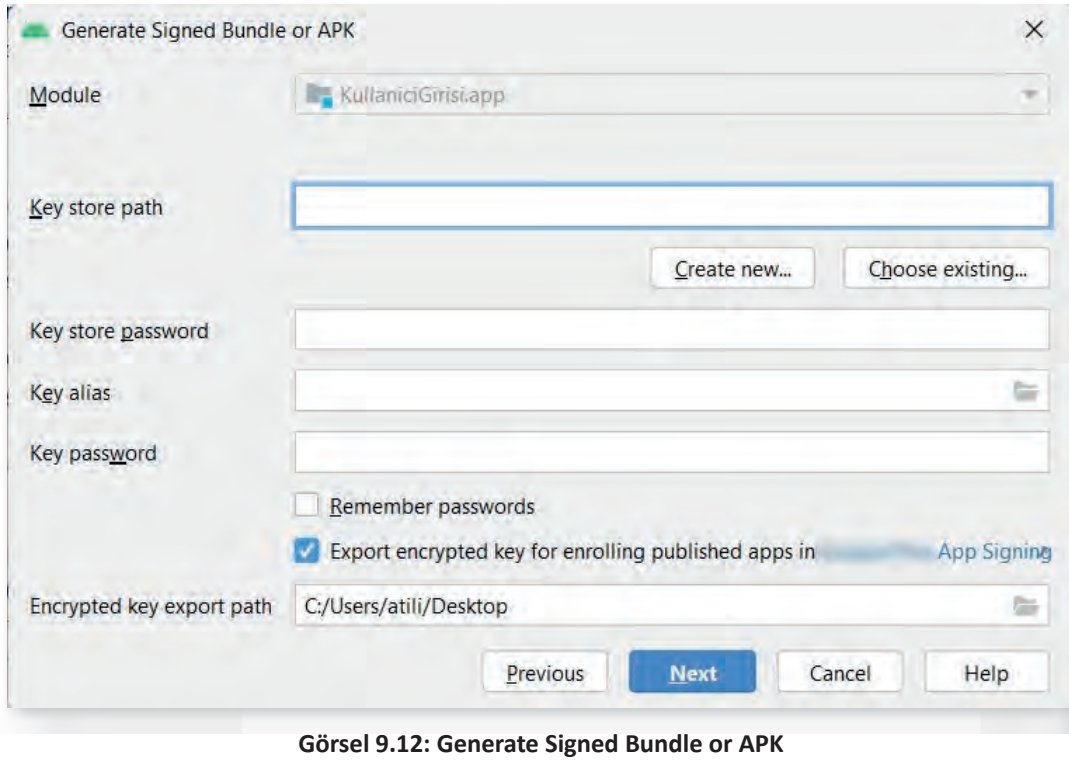
Uygulama Yayımlama Firması, market olarak politikalarını oldukça sıkı tutar. İhlallere, hilelere, çalıntı uygulamalara karşı katı kuralları vardır. Örneğin 2019 yılında başlayan pandemide COVID-19 ile ilgili bir oyun vb. uygulamalar yayımlanmak istenirse öncelikle yüksek sağlık kuruluşlarından (Sağlık Bakanlığı vb.) alınmış izin belgeleri gerekir. Market, her türlü yanlış bilgilendirmeden kendini arındırmak ister ve böyle bir izin yoksa Covid imgesi, ismi, resmi, iconu bulunan uygulamayı reddeder. Uygulamaların market içinde yayımlanmadan önce de uygulamanın geliştiricisi tarafından imzalanması zorunluluğu bulunur. Mobil uygulama geliştirme ortamından sadece "APK" (Android Package Kit) olarak veya "Generate Signed APK" olarak iki türlü çıktı alınır.

Normal "APK" çıktısı, paketin başka bir Android cihaza yüklenmesini ve o cihazda kullanılmasını sağlar. Bu durumda önce cihazın "Ayarlar" menüsündeki "Gelişmiş" seçeneğinden "Bilinmeyen uygulamaları yükle" açılmalıdır. Ancak bu şekilde uygulama paketi o cihaza yüklenebilir. Bu paket, markete yüklenmek istendiğinde imza bulunmadığı için kabul edilmez. Bu nedenle uygulamanın markette paylaşılmadan önce "Generate Signed APK" olarak paket şekline getirilmesi gerekir. Öncelikle **Build** sekmesinden **Generate Signed Bundle/APK** seçeneği tıklanır. Görsel 9.11'deki pencere görünür. Android App Bundle seçeneği seçilerek Next butonuna tıklanır.



Görsel 9.11: Generate Signed Bundle/APK

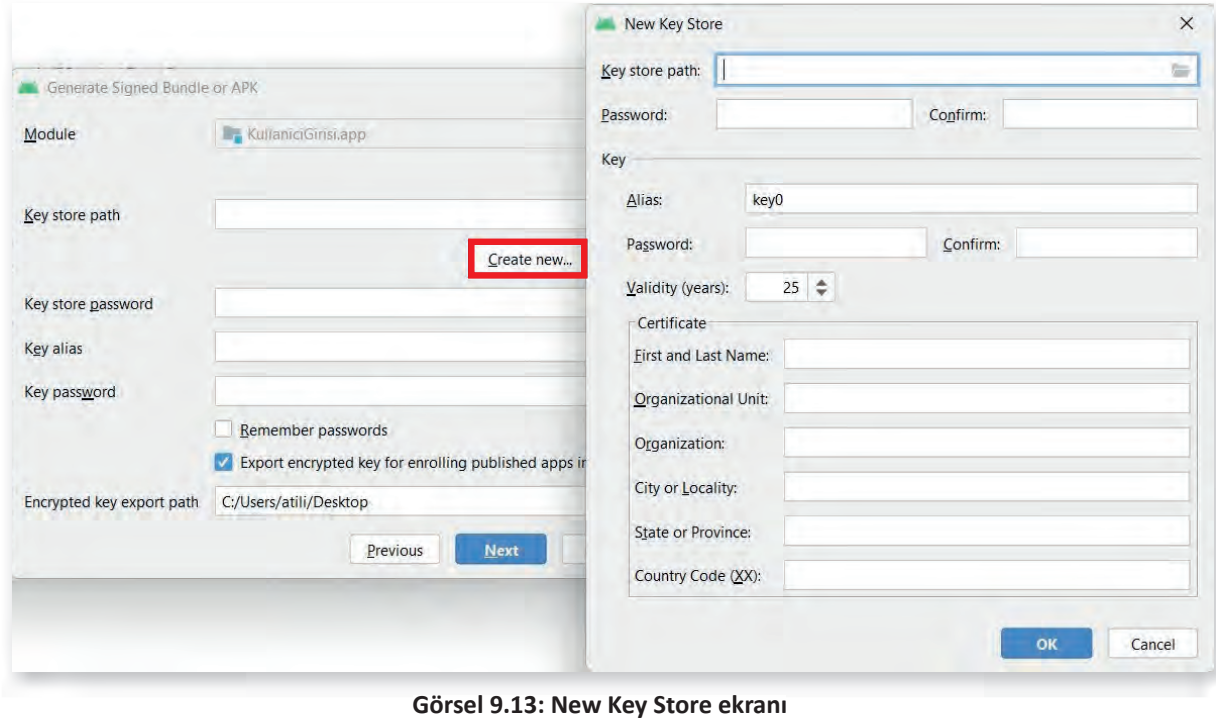
Görsel 9.12'deki gibi bir ekranla karşılaşılır.



Görsel 9.12: Generate Signed Bundle or APK

- **Module:** Mobil uygulama ekranında açık olan ve APK imzası çıkarılacak uygulamayı temsil eder.
- **Key store path:** APK imzalama işleminden sonra çıkan imza dosyasının yolunun, şifresinin ve sertifikasının belirtildiği bölümdür. **Create New** butonuna tıklandığında "New Key Store" penceresi ile karşılaşılır (Görsel 9.13).

UYARI: Bu imza dosyası saklanmak zorundadır. Bu dosya olmadan market üzerinde uygulama ile ilgili güncelleştirmeler yapılamaz.



Görsel 9.13: New Key Store ekranı

► **Key store path:** Anahtar dosyasının çıkartılacağı yol buradan seçilir.

► **Password:** Anahtar dosyasının kullanılabilmesi için gerekli olan şifre oluşturulur.

► **Confirm:** Anahtar dosyasının kullanılabilmesi için gerekli olan şifre tekrar edilir.

► **Alias:** Anahtara verilecek takma isim ifade edilir.

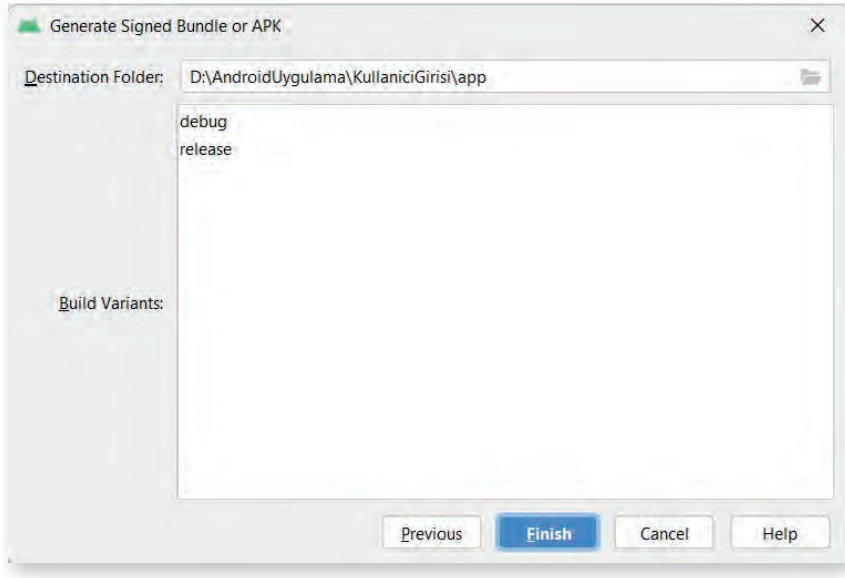
► **Password:** Anahtar dosyasının kullanılabilmesi için gerekli olan şifre oluşturulur.

► **Confirm:** Anahtar dosyasının kullanılabilmesi için gerekli olan şifre tekrar edilir.

► **Validity (Years):** Anahtar dosyasının geçerli olacağı yıl sayısı girilir.

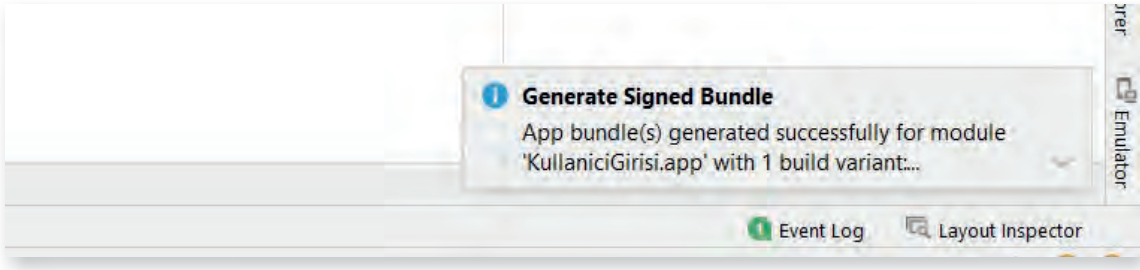
► **Certificate:** Sertifika bilgilerini içerir. First and Last Name ile kişi adı soyadı, Organizational Unit ile şirketin birimi, Organization ile şirketin adı, City and Locality ile şehir, State or Province ile ilçe ve Country Code ile de ülke kodu girişi yapılır.

Bilgiler uygun şekilde doldurulur. Ardından "OK" butonuna tıklanır. Girilen bilgiler New Key Store penceresine gelir. Next butonuna tıklanır. Görsel 9.14'te görülen menü ile karşılaşılır.



Görsel 9.14: Generate Signed Bundle or APK

"release" seçeneğine tıklanır ve Finish butonuna basılır. Bir süre beklendikten sonra imzalama işlemi tamamlanır ve mobil geliştirme ortamının sağ alt bölümünde imzalı çıktının oluşturulduğu bilgisi verilir (Görsel 9.15). locate seçeneği seçilirse dosyanın dizinine erişim sağlanır.

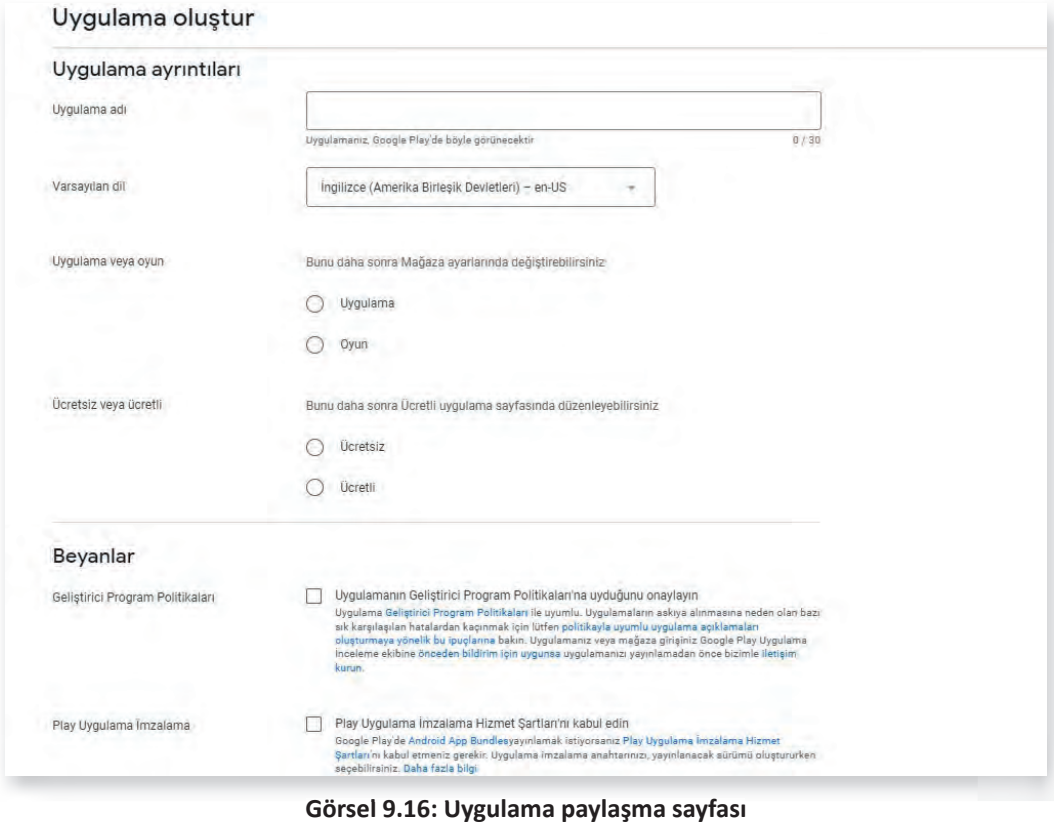


Görsel 9.15: Generate Signed Bundle onay ekranı

NOT:

Oluşturulan dosya **.aab** uzantısındadır. Paket, .apk uzantılı istenirse Görsel 9.11'deki menüden Android App Bundle yerine APK seçeneği seçilmelidir.

İmzalanmış çıktısı alınan mobil uygulamanın markette yayımlanması için tarayıcı üzerinden geliştirici hesabı ile Play Console açılır. Görsel 9.9'da görülen **Uygulama oluştur** butonuna tıklanır. Görsel 9.16'da görülen sayfa açılır.



Görsel 9.16: Uygulama paylaşma sayfası

- **Uygulama adı:** Markette görüntülenecek isim bu bölüme girilir.
- **Varsayılan dil:** Dil seçeneği bu bölümden seçilir.
- **Uygulama veya oyun:** Geliştirilen uygulamanın bir program mı yoksa oyun mu olduğu seçilir.
- **Ücretsiz veya ücretli:** Geliştirilen uygulamanın markette ücretli olarak mı yoksa free olarak mı yayımlanacağı seçilir.

NOT:

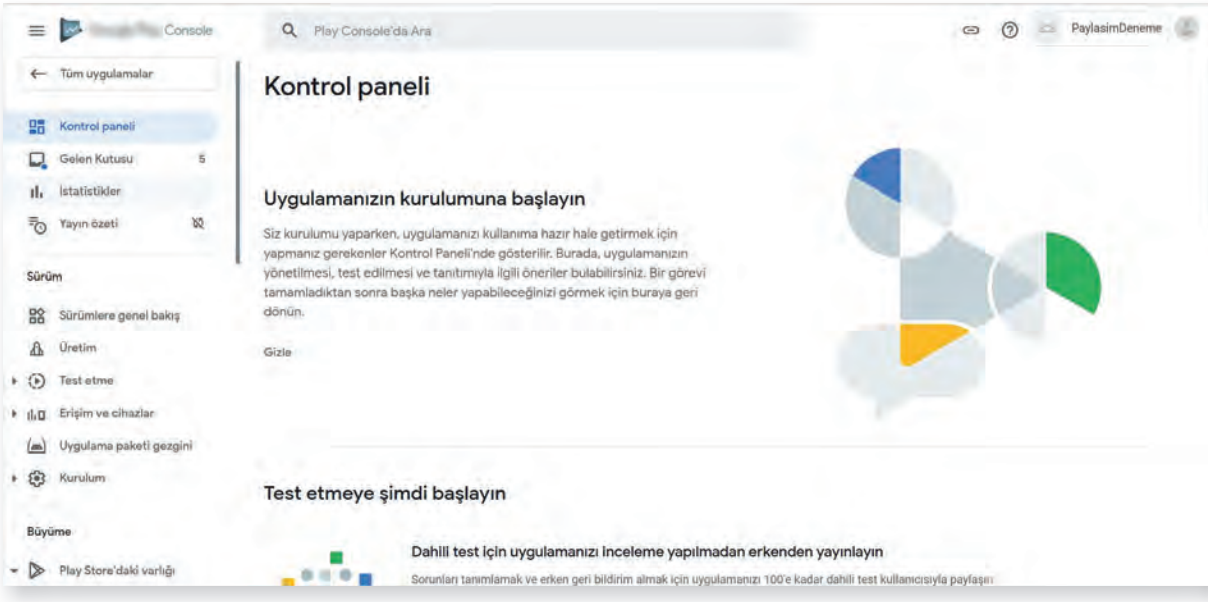
Uygulamayı ücretsiz veya ücretli indirme özelliğinde yayımlama seçimi, uygulama yayımlanıncaya kadar değiştirilebilir. Uygulama yayımlandıktan sonra ücretsiz bir uygulama, ücretli olarak değiştirilemez.

- **Beyanlar:** Geliştirici program politikalarına, Play uygulama imzalama ve ABD ihracat yasalarına uygun olduğuna dair beyan verilir.

Tüm ayarlar yapılıp, girişler tamamlandığında "Oluştur" butonuna tıklanır ve uygulama, paylaşıma açılır. Görsel 9.17'deki ekranla karşılaşılır. Bu panele Kontrol paneli ismi verilir.

NOT:

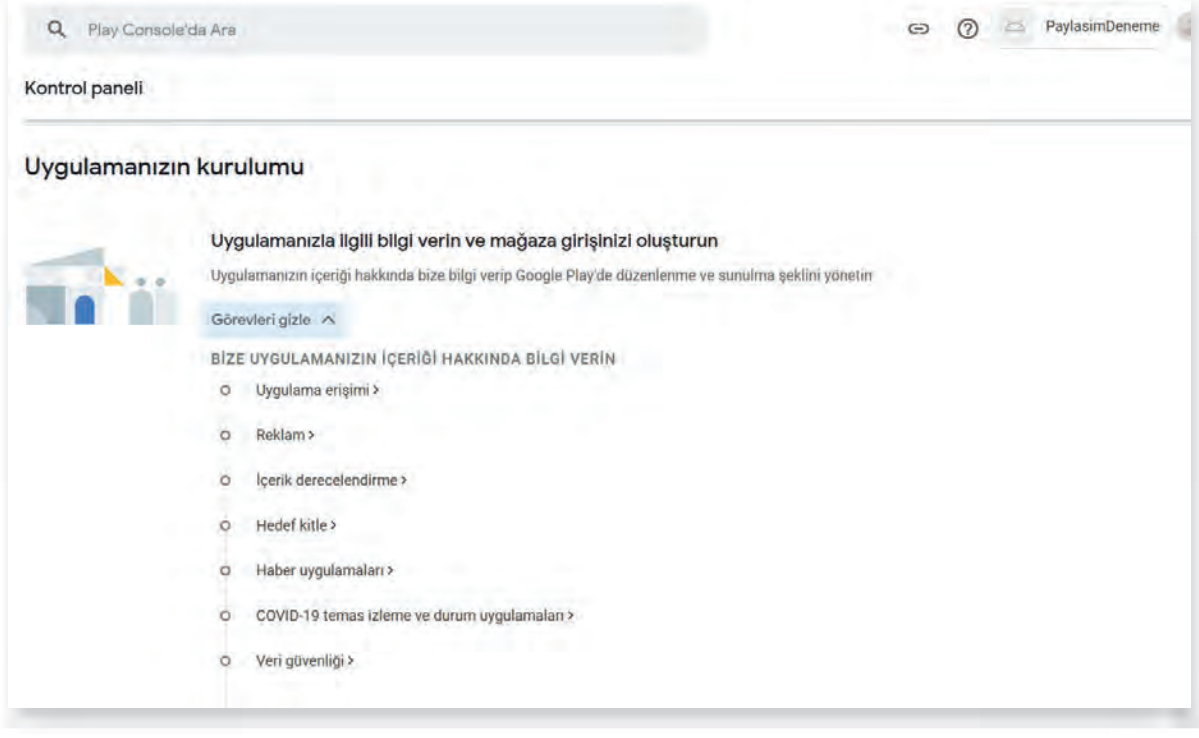
Henüz uygulama paylaşılmamış sadece gerekli ayarlar ve isimlendirmeler yapılmıştır. Uygulamanın kurulumu, Kontrol paneli aracılığı ile bu aşamadan sonra yapılır.



Görsel 9.17: Kontrol paneli ekranı

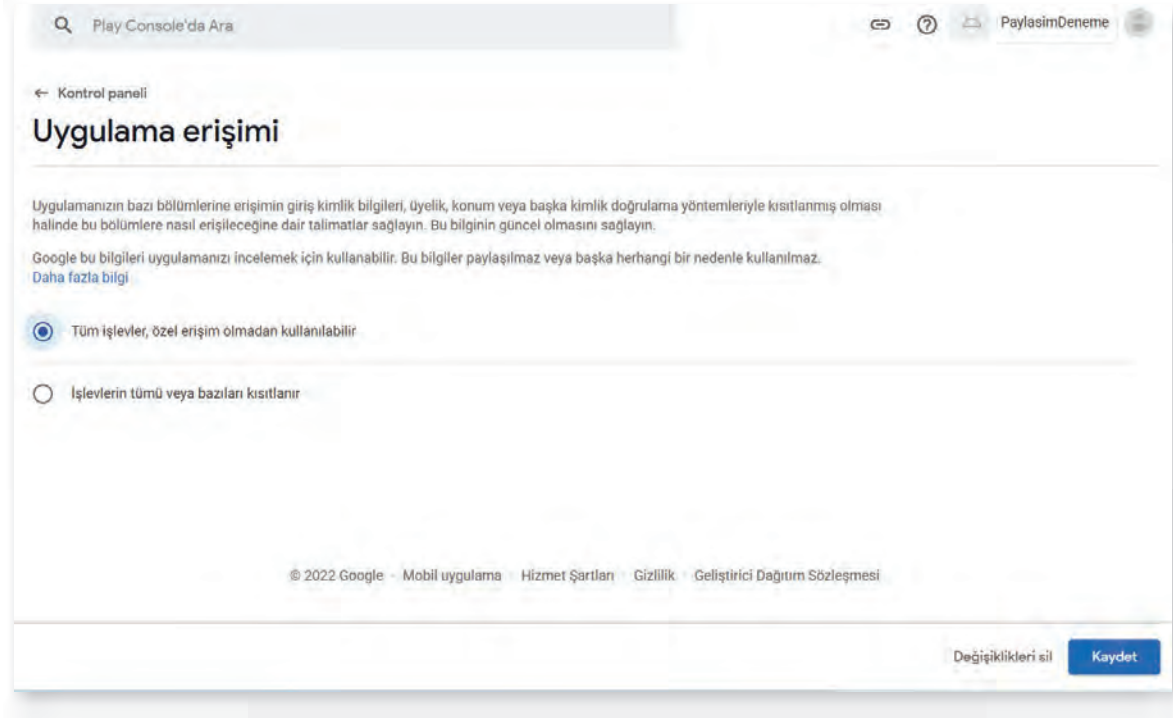
Uygulamanın Play Console ekranına yüklenmesi ve markette yayımlanması için görevler verilir. Bu görevlerin sırasıyla yapılması gerekir.

- Görsel 9.18'de görüldüğü gibi Kontrol panelindeki "Uygulamanızın kurulumu" başlığı altında yer alan "Görevleri göster" sekmesine tıklanır. Görevlerin her biri tek tek yapılmalıdır. Görevler tamamlandıktan sonra tamamlanan her bir görevin üzeri sistem tarafından otomatik çizilerek onay işareti alınmalıdır. "Uygulama erişimine" tıklanır.



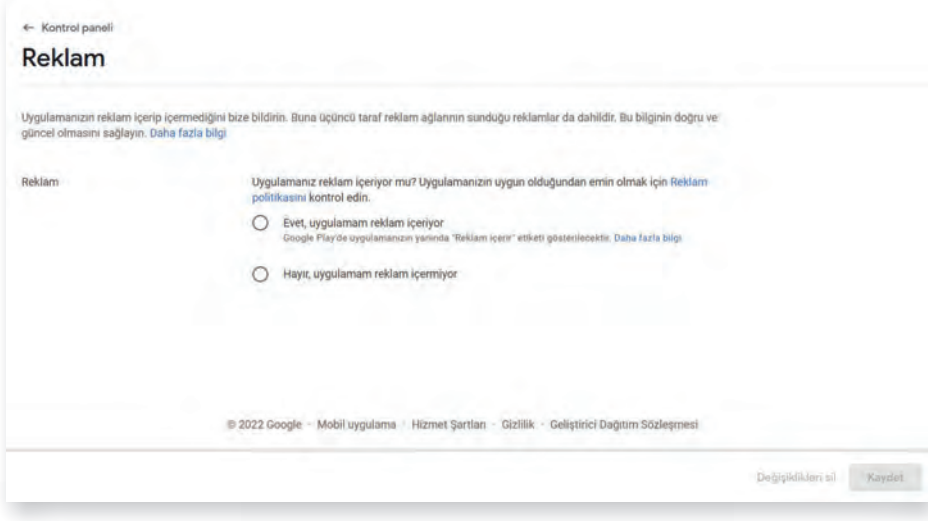
Görsel 9.18: Uygulamanızın kurulumu başlığı

- Görsel 9.19'da yer alan sayfa açılır. "Tüm işlevler, özel erişim olmadan kullanılabilir" seçeneğine tıklanır. Diğer seçenek ile bazı işlevler kısıtlanabilir. Kaydet butonu tıklanır. "Değişiklikleriniz kaydedildi" mesajı sonrasında tekrar "Kontrol paneli" seçeneğine tıklanır ve görevlere dönülür.



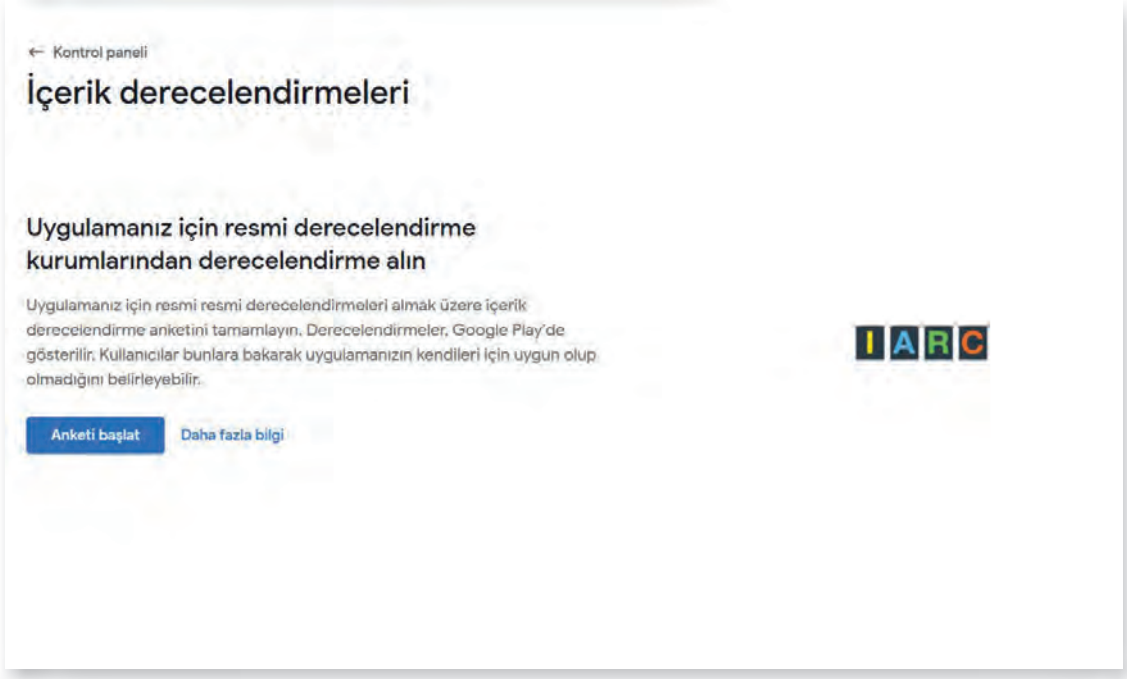
Görsel 9.19: Uygulama erişimi ekranı

- **Reklam** görevine tıklanır. Uygulama içinde reklam varsa "Evet, uygulamam reklam içeriyor" seçeneği seçilir. Bu durumda market içinde uygulamanın yanında "Reklam içerir" ibaresi yer alır. Uygulama içinde reklam yoksa "Hayır, uygulamam reklam içermiyor" seçeneği seçilir. Kaydet butonuna tıklanır (Görsel 9.20). "Değişiklik Kaydedildi" seçeneğinden sonra Kontrol paneline tıklanıp görevlere dönülür.

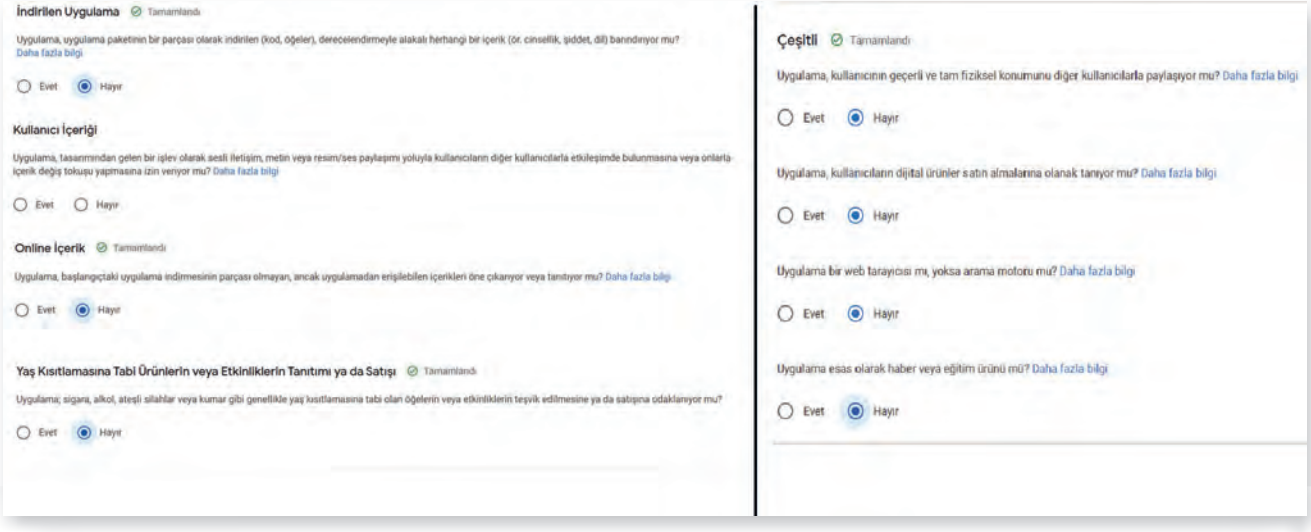


Görsel 9.20: Reklam seçeneği ekranı

- **İçerik derecelendirmeleri** görevine tıklanır. Görsel 9.21'deki sayfa açılır. "Anketi başlat" seçeneğine tıklanır. Paylaşılan uygulamanın bir oyun mu yoksa uygulama mı olduğu sorulur. İlgili cevap verilip Kaydet seçeneği tıklanır ve İleri butonuna basılır. İçerik değerlendirme bölümü oldukça önemlidir. Uygulama hakkında birçok bilgi burada sorulur ve bu bölüme doğru bilgi girişi yapılmalıdır. Görsel 9.22'de bu bölümde sorulan sorular verilmiştir. Bilgiler girildikten sonra Kaydet seçeneği tıklanır ve İleri butonuna basılır.

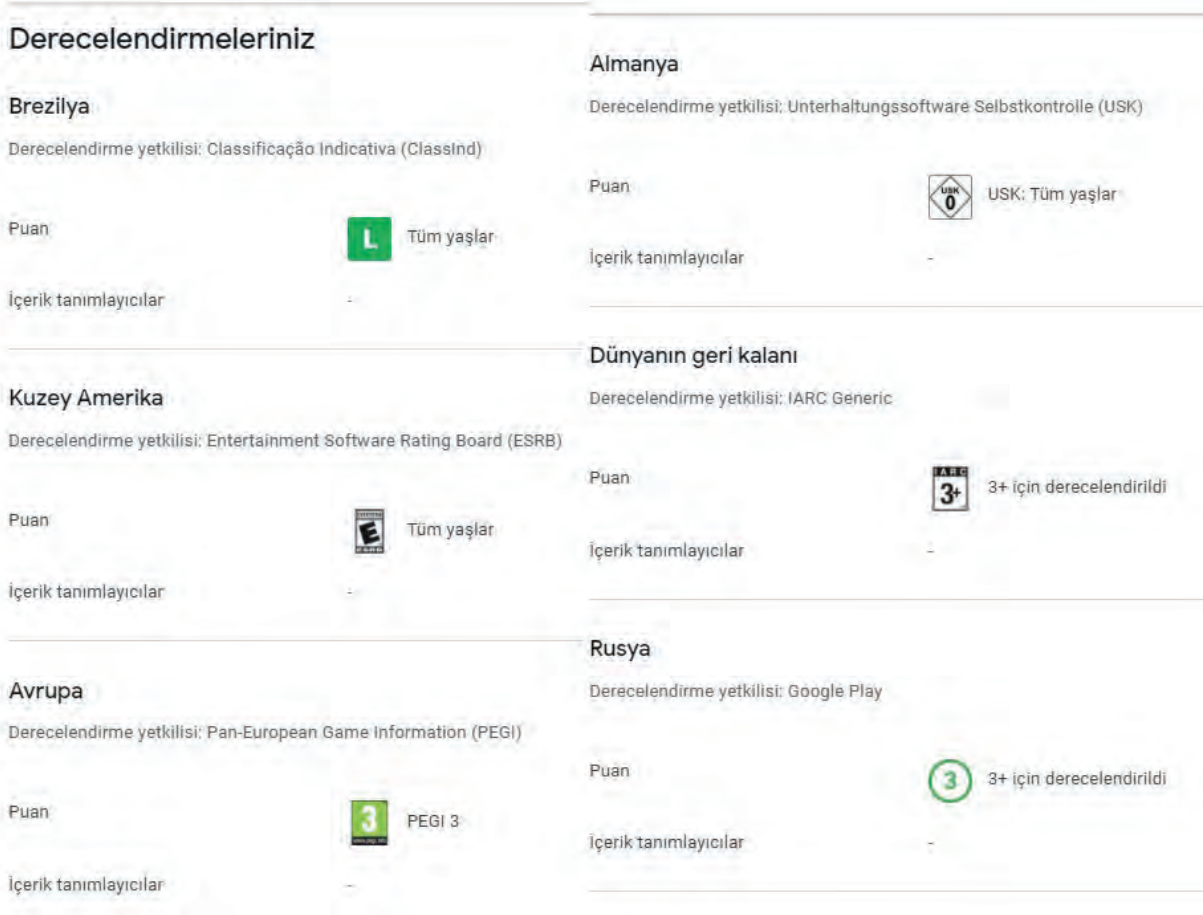


Görsel 9.21: İçerik derecelendirmeleri ekranı



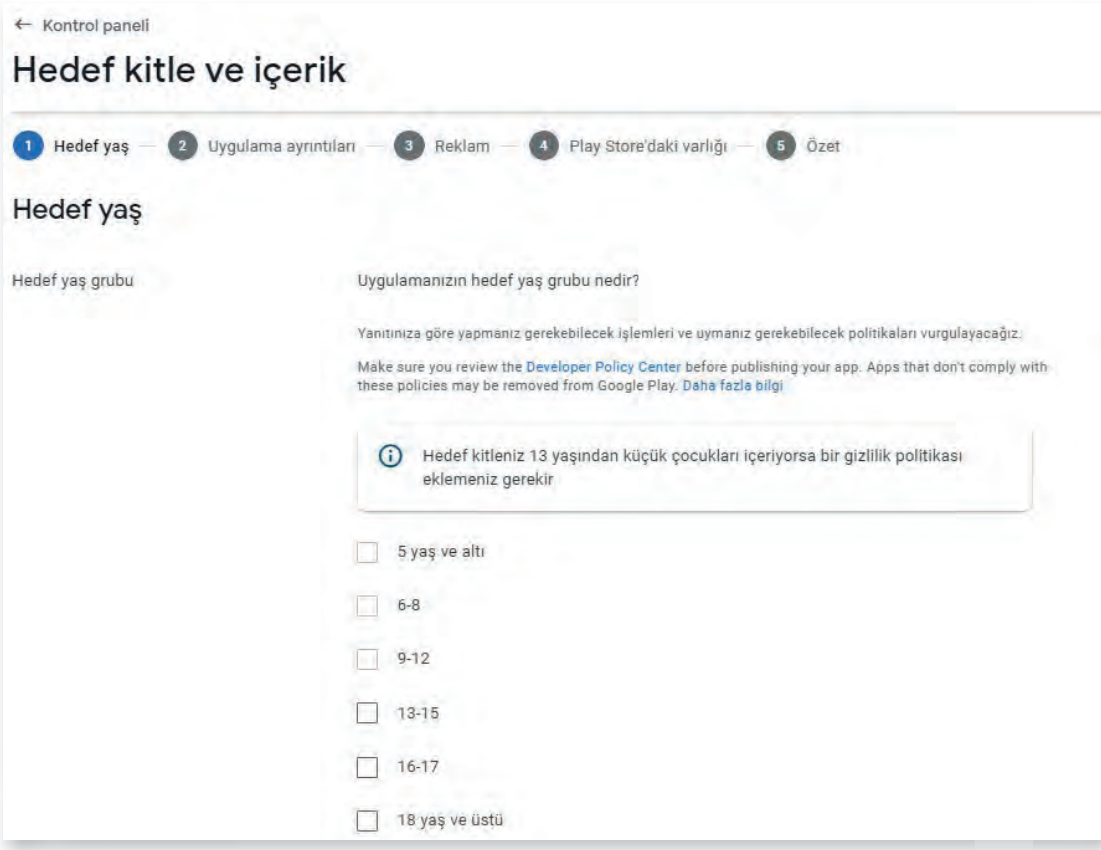
Görsel 9.22: İçerik soruları ekranı

Görsel 9.23'te görüldüğü gibi her ülkede geçerli olacak bir derecelendirme puanı ortaya çıkar. Bu derecelendirme puanı o ülkedeki yasal statüyü belirler. Gönder seçeneğine basılır ve görev tamamlanır.



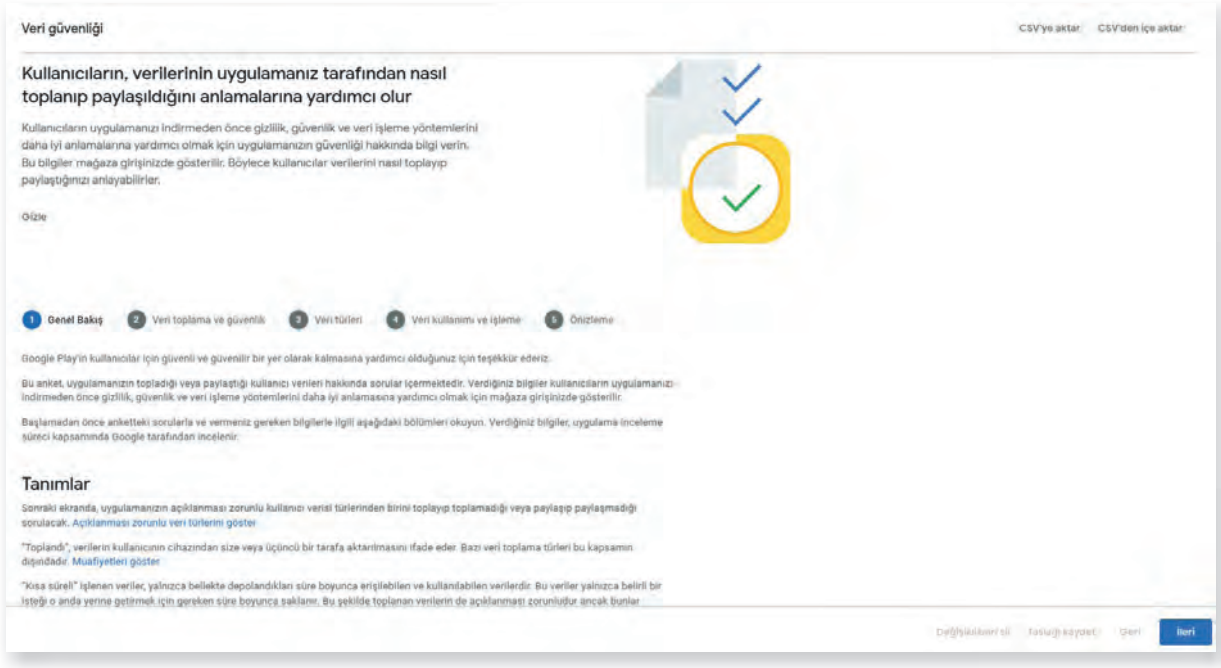
Görsel 9.23: Ülkelere göre içerik derecelendirme sonuç ekranı

- **Hedef Kitle** görevine tıklanır. Görsel 9.24'te görülen menü ile karşılaşılır. Hedef kitle olarak 13 yaş üzerinde seçim serbesttir fakat 13 yaş altındaki seçimler pasif durumdadır. Bunun nedeni, firmanın politikalarıdır. 13 yaş altındaki kitlelere uygulama paylaşılabilmesi için bir **gizlilik politikası** eklenmesi zorunludur. 13 yaş üzeri seçimlerle işleme devam edilir. 13 yaş altına hitap eden uygulamalar için gizlilik politikasına ait bir metin oluşturulup, bu metin Play Console üzerine yüklendiğinde daha sonradan hedef kitlede güncelleme yapılabilir. İleri butonuna tıklandığında "Mağaza girişiniz kasıtlı olmadan çocuklara hitap ediyor olabilir mi?" diye sorulur ve cevap beklenir. Seçim yapıldıktan sonra İleri butonuna tıklanır. Bir özet metni ortaya çıkar ve hedef kitle yaşının markette neden yer aldığının, aileler programına kayıtlı olunup olunmadığının bir özetini sunar. Kaydet butonuna basılır. "Değişiklikler Kaydedildi" mesajı görülür. Kontrol paneli seçeneğine tıklanır ve görevlere dönülür.



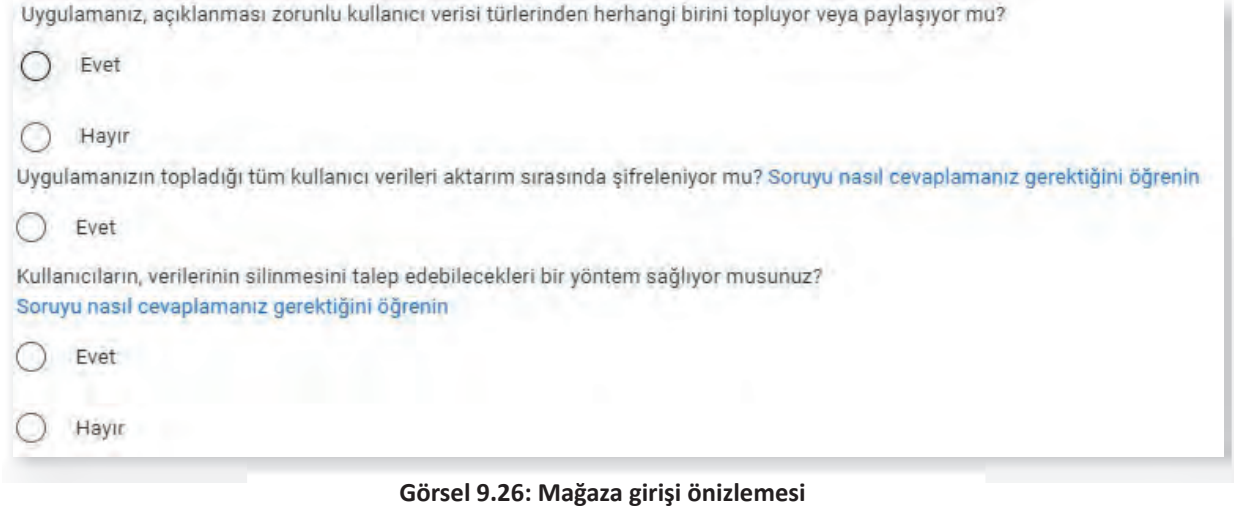
Görsel 9.24: Hedef kitle ve içerik sayfası

- **Haber uygulamaları** görevine tıklanır ve "Uygulamanız haber uygulaması mı?" sorusu Play Console tarafından sorulur. Hayır cevabı verilirse herhangi bir uyarı çıkmaz, Evet cevabı verilirse uygulamadaki haberlerin içeriği sorulup uygulama geliştiriciden "Haber Politikalarına uygunluğunu onaylıyorum" şeklinde bir taahhüt alınır. Uygun cevap verilir ve Kaydet butonuna basılır. Kontrol paneli seçeneğine tıklanır ve görevlere dönülür.
- **COVID-19 temas izleme ve durum uygulamaları** görevine tıklanır. Uygulamanın COVID-19 ile ilgili olup olmadığı Play Console tarafından sorulur. Uygulamanın COVID-19 ile ilgili olmadığı taahhüt edilir ve Kaydet butonuna tıklanır. Kontrol paneli seçeneğine tıklanır ve görevlere dönülür.
- **Veri güvenliği** görevine tıklanır. Bu görev, kullanıcıların verilerinin uygulama tarafından nasıl toplanıp paylaşıldığını anlamalarına yardımcı olur. Görsel 9.25'te görülen sayfa açılır.



Görsel 9.25: Veri güvenliği ekranı

Verilerin toplanıp toplanmadığı, veriler toplanıyorsa nasıl toplandığı ve paylaşıldığı soruları Görsel 9.26'da olduğu gibi sorulur.

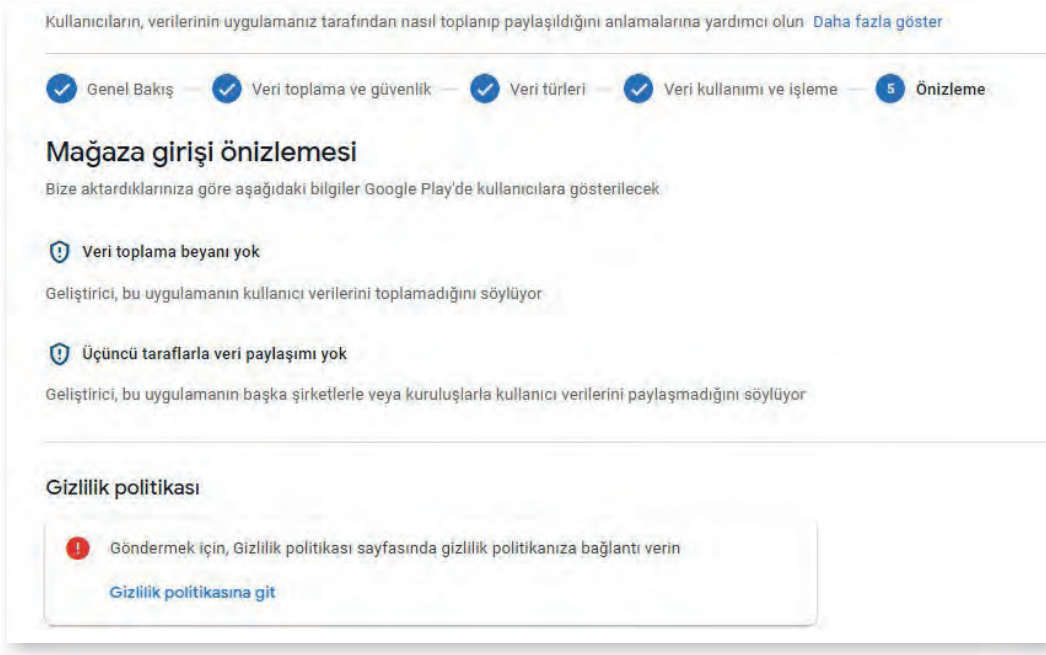


Görsel 9.26: Mağaza girişi önizlemesi

Uygun şekilde cevap verilip İleri butonuna tıklanır. Ekrana **Gizlilik beyanının** olmadığı, bu nedenle görevin tamamlanamayacağını belirten bir mesaj gelir (Görsel 9.27).

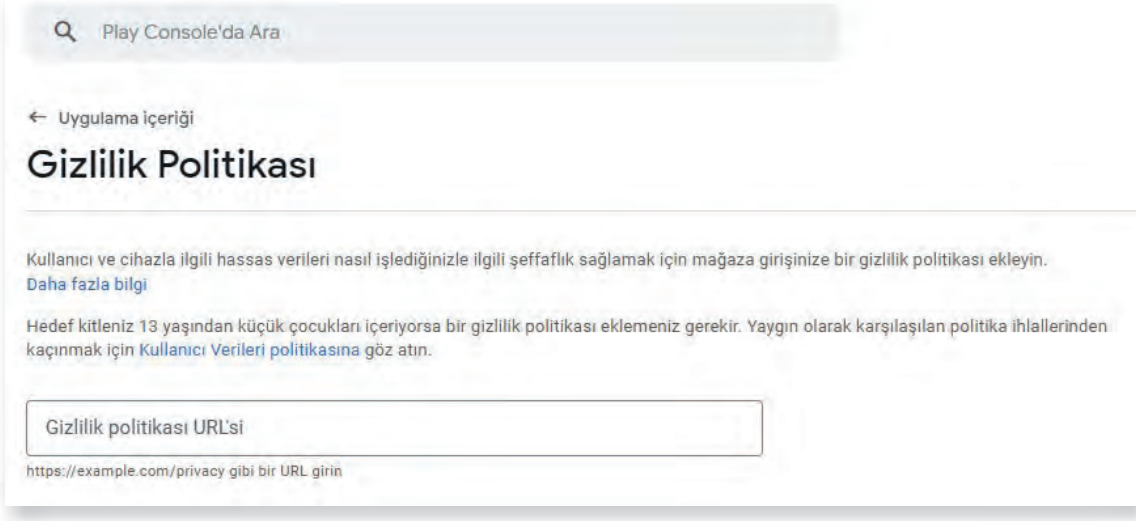
NOT:

Play Console İçin Gizlilik Politikası Metni Oluşturma şeklinde internet üzerinden dokümanlar aracılığı ile word dosyasında gizlilik politikası metni oluşturulmalı ve bilgisayara kaydedilmelidir. Bu belge, uygulama içinde gizlilik politikası olarak sisteme yüklenir.



Görsel 9.27: Mağaza girişi gizlilik politikası yok ekranı

- "Gizlilik politikasına git" seçeneğine tıklanır ve sayfadan çıkılır. Gizlilik Politikası sayfasında URL adresi sorulur. Oluşturulan Gizlilik Politikası metni herhangi bir upload sitesi yardımı ile sisteme internet ortamına yüklenir ve URL adresi bu bilgi kutusuna yazılır (Görsel 9.28).

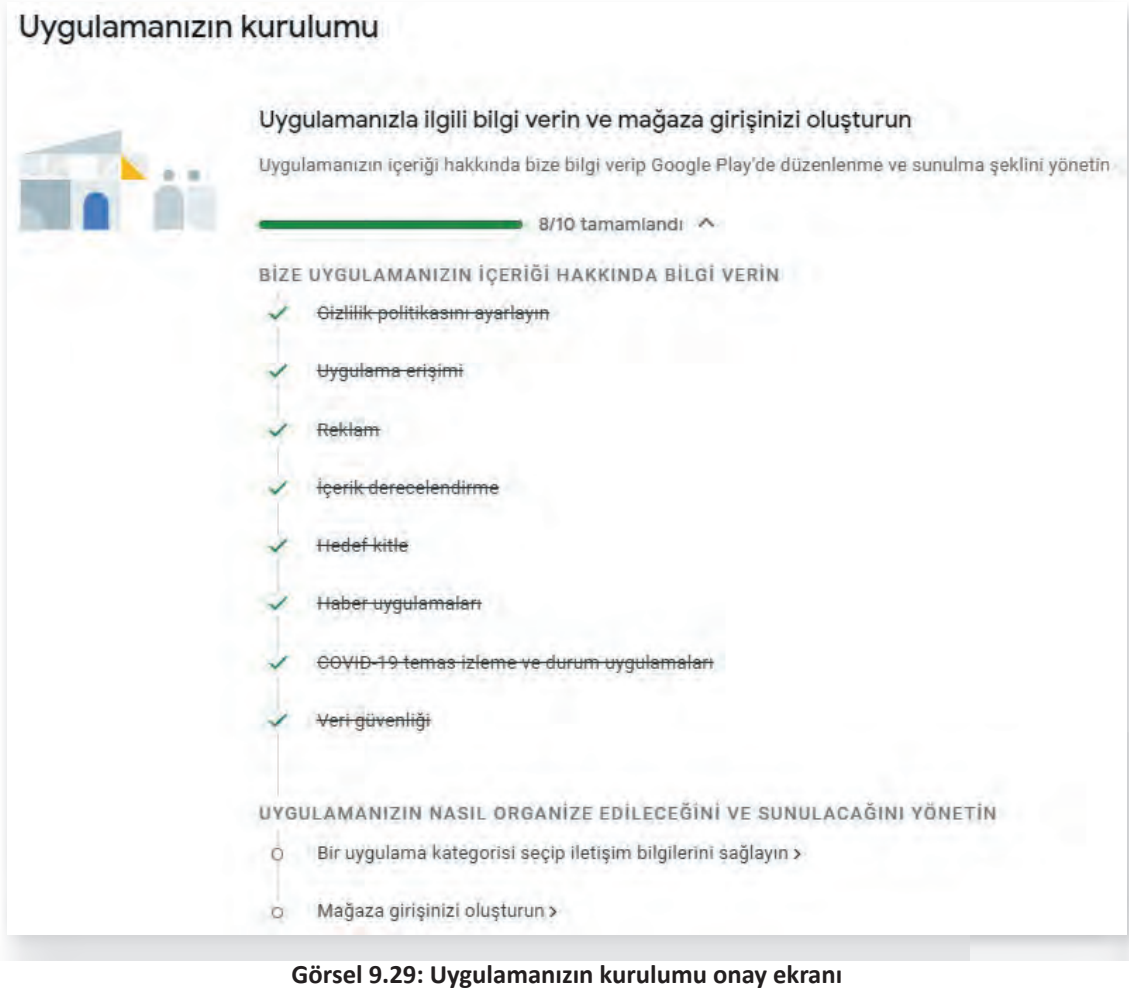


Görsel 9.28: Gizlilik Politikası kayıt ekranı

NOT:

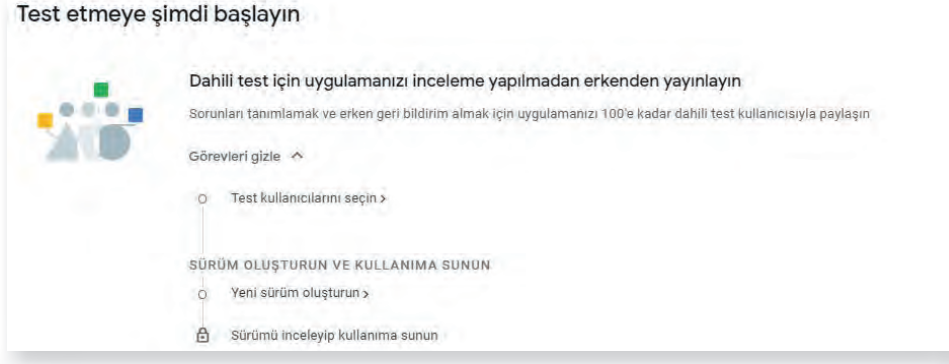
Görsel 9.28'de görüldüğü gibi 13 yaş ve üzeri hedef kitlesi için nasıl bir gizlilik politikası eklenmesi gerektiğini bildiren bir doküman sistem üzerinde paylaşılır. Bu linke tıklanıp, ilgili doküman incelenerek uygulama için uygun bir Gizlilik Politikası metni oluşturulursa hedef kitle 13 yaş ve altı seçilebilir.

- Kontrol paneline dönüş yapılır ve veri güvenliği görevine geçilir. Veri güvenliği ile ilgili işlemler tekrar edilir. Gizlilik politikası metni paylaşıldığı için sistem ayrıca bir hata veya uyarı vermez. Kaydet butonu tıklanır. Kontrol paneli seçeneğine tıklanır ve görevlere dönülür.
- **Uygulamanızın kurulumu** başlığında yer alan görevler Görsel 9.29'da olduğu gibi çizgi ve onay işaretini alınca Test aşamasına geçilir.



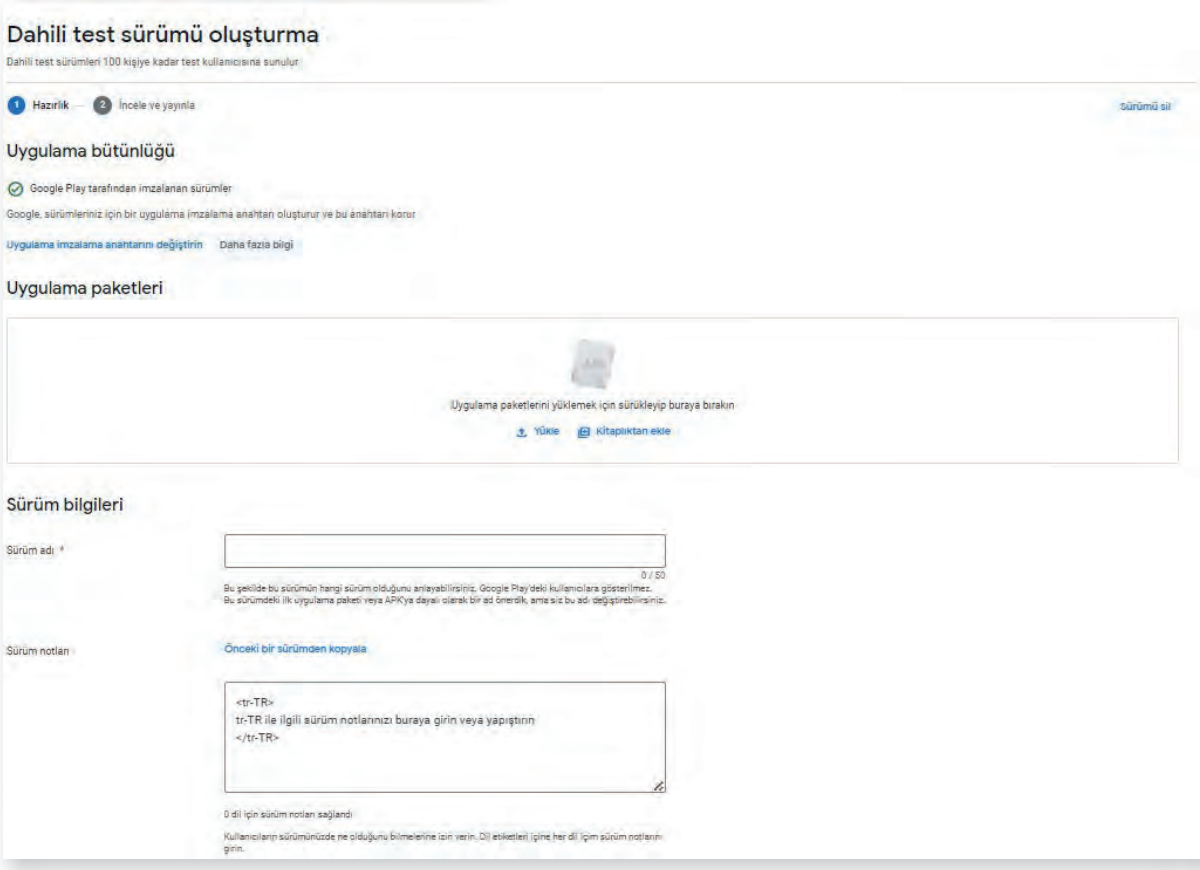
Görsel 9.29: Uygulamanızın kurulumu onay ekranı

- Görsel 9.30'da görülen **Yeni sürüm oluşturun** bağlantısına tıklanır.



Görsel 9.30: Test etmeye şimdi başlayın ekranı

- Görsel 9.31'de görüldüğü şekilde Yeni Sürüm kaydı yapılır. Bu işlem, uygulama ilk yüklenirken oluşturulur. Yükle seçeneğine tıklanıp imzalı çıktısı alınan mobil uygulama paketi seçilir.



Görsel 9.31: Uygulama paketi yükleme ekranı

NOT:

Paket yükleme işlemi tamamlandığında uygulamanın imzalı paketi Play Console tarafından içeriye yüklenir. Bu paket yayımlanmaya başladığında imza paketi kaybedilirse (".jks" uzantılı dosya) uygulama güncelleme veya değişiklik işlemleri yapılamaz.

Paket içeriye yüklendikten, sürüm ve not bilgileri girildikten sonra Kaydet butonuna tıklanır. Ardından "Sürüm İncelemesi Başlat" butonuna tıklanır. Görsel 9.32'de görülen hata durumları ayıklanır ve uygulama gösterilir. "Dahili test kanalına sunumu başlat" seçeneğine tıklanır. Gelen mesaj üzerine "Kullanıma Sun" seçeneğine tıklanır.

Görsel 9.32: Sürüm inceleme ekranı

- Kontrol paneline dönüş yapılır. Uygulamanızın kurulumu başlığı altında yer alıp henüz tamamlanmayan iki adet görevi tamamlamak üzere "Bir uygulama kategorisi seçip iletişim bilgilerini sağlayın" seçeneğine tıklanır. Yüklenen mobil uygulama paketinin uygulama mı, oyun mu olduğu tekrar girilir. Kategori bilgisi sorulur ve etiket girilmesi istenir. Ayrıca kullanıcılara gösterilecek mail adresi ve telefon numarası gibi bilgiler istenir. Mail adresi zorunlu iken telefon numarası opsiyonel bir seçenektir. Girilmesi zorunlu değildir. Bilgiler girildikten sonra Kaydet butonuna tıklanır. Kontrol paneline dönülür.

- "Uygulamanızın kurulumu" başlığı altında yer alan son görev yerine getirilir. "Mağaza girişinizi oluşturun" seçeneğine tıklanır. Bu bölümde şu bilgilerin girilmesi istenir:

- **Uygulamanın adı**

- Maksimum 80 harflik **kısa açıklama bilgisi**

- Maksimum 4.000 harflik **tam açıklama bilgisi**

- 512x512 en fazla 1 MB'lık şeffaf bir PNG veya JPEG **uygulama simgesi**

- 1024x500 piksel ölçülerinde, en fazla 1 MB'lık bir PNG veya JPEG **özellik grafiği**

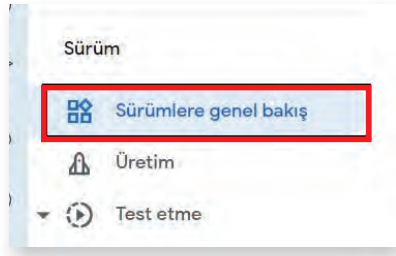
- Opsiyonel olarak **video linki**

- Telefon, 7 inçlik Tablet ve 10 inçlik Tablet **ekranından görüntü**

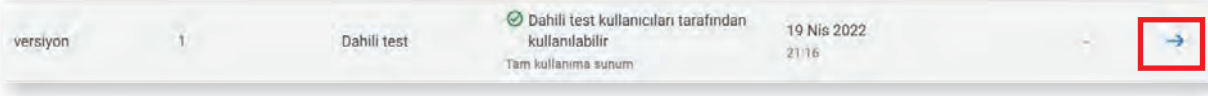
İstenen tüm bilgiler yüklendikten sonra Kaydet seçeneğine tıklanır. "Değişiklikler Kaydedildi" mesajı görüldüğünde Kontrol paneline geri dönülür.

Test ve Uygulama kurulumunda tüm görevler tamamlandığına göre uygulama yayımlanmaya hazırdır. Öncelikle Görsel 9.33'te görülen "Üretim" seçeneğine tıklanır. Bu seçenekten "Ülkeler ve Bölgeler" sekmesi altından yayımlanması istenen ülkeler seçilir ve kaydedilir. Görsel 9.33'teki Sürüm başlığı altında yer alan "Sürümlere genel bakış" menüsüne tıklanır. Ekrandaki sürüm bilgisinin Görsel 9.34'te görüldüğü şekilde sağ bölgesinde yer alan ok işaretine tıklanır. Açılan menü-

den “Sürümü Yükselt” seçeneği seçilir. “Üretim” butonuna tıklanır.

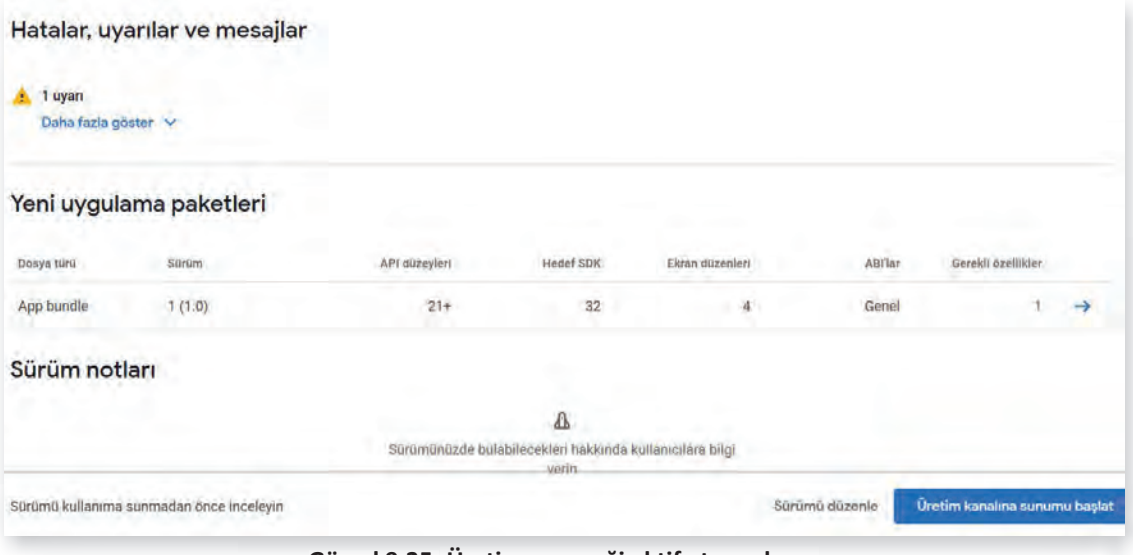


Görsel 9.33: Sürüm menüsü



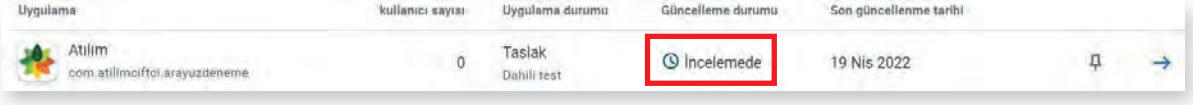
Görsel 9.34: Sürüm inceleme ekranı

- Üretim bölümüne geçen, menüdeki yüklenmiş sürüm güncellenmek istenirse “Yükle” seçeneğine tıklanır ve yeni sürüm yüklenir. Aynı sürüm ile devam edilmek istenirse “Sürümü inceleme” butonuna basılır ve inceleme aşamasına geçilir. Görsel 9.35’te görüldüğü gibi **Üretim kanalına sunumu başlat** seçeneğine tıklanır ve çıkan menüden **Kullanıma sun** seçeneği seçilir.



Görsel 9.35: Üretim seçeneği aktif etme ekranı

Bu noktadan sonra uygulama, market üzerinde yayımlanması için incelemeye gönderilir. Görsel 9.36’daki gibi “Tüm uygulamalarım” bölümünde uygulamanın “İncelemede” şeklinde görünmesi gerekir. İnceleme süreci, istisnai durumlar haricinde 1 saat ile 7 gün arasında sürebilir. İnceleme süreci sonrası uygulama reddedilebilir. Uygulama askıya alınıp bilgilendirme veya uygulamada düzenleme talep edilebilir. Uygulama doğrudan yayına da alınabilir. Uygulama durumunun bilgisi hem Play Console mesajlarına hem de mail adresine gönderilir.



Görsel 9.36: Uygulama “İncelemede” ekranı

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi uygulama paylaşımı için kullanılan marketin yönetim işlemlerinin yapıldığı ortamdır?

A) Play Store B) Play Console C) Play Market

D) Play Game E) Play Mobil

2. Aşağıdakilerden hangisi mobil uygulamaların dışarıya çıkartılmış paket uzantısıdır?

A) jdk B) jks C) apk

D) xml E) doc

3. Aşağıdakilerden hangisi geliştirilen mobil uygulamanın imzalı paketi çıkartılırken Built Variants bölümünden seçilir?

A) begin B) release C) new

D) pick E) browse

4. Aşağıdakilerden hangisi market üzerinde oluşturulan uygulamanın, paylaşım öncesinde görevlerinin görülüp tamamlanabileceği panelin adıdır?

A) İstatistikler paneli B) Üretim paneli C) Kontrol paneli

D) Geliştirme paneli E) Test paneli

5. Aşağıdakilerden hangisi mobil uygulamanın paylaşım öncesi ülkeler standardına göre yaş gruplarının belirlendiği görev aşamasıdır?

A) İçerik derecelendirme B) Reklam C) Hedef kitle

D) Veri güvenliği E) Haber uygulamaları

6. Aşağıdakilerden hangisi uygulama marketinde uygulamanın yayımlanmadan önce paylaşılması zorunlu olan belgedir?

A) Veri metni B) İçerik anketi C) Gizlilik politikası

D) Haber analizi E) Yaş kitle belgesi

7. Aşağıdakilerden hangisi uygulama paylaşma sırasında istenen market ikonu boyutudur?

A) 1024x768 B) 1024x1024 C) 512x512

D) 1024x500 E) 512x768

8. Aşağıdakilerden hangisi Google Play'e ait politikalara uygunsuz hareketlerden kaynaklı durumların yer aldığı bölümdür?

A) Kullanıcılar ve izinler B) Sipariş yönetimi C) Hesap ayrıntıları

D) Geliştirici seçenekleri E) Politika durumu

CEVAP ANAHTARLARI

1. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D
2. D
3. Y
4. D
5. Y
6. D
7. D
8. D
9. B
10. A
11. C
12. B
13. D
14. A

2. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D
2. D
3. D
4. D
5. Y
6. Y
7. D
8. E
9. B
10. A
11. B
12. A
13. A

3. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D
2. Y
3. D
4. D
5. D
6. Y
7. Y
8. B
9. D
10. A
11. B
12. B
13. C
14. D
15. E

4. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D
2. Y
3. Y
4. Y
5. D
6. Y
7.

textView Değerleri

```
0 3 6 9 12 15 21 24
30 33 36 39 42 45 48
```

8.

textView Değerleri

```
1 2 6 24 120 720
5040
```

9.
```
if(sayi1>=sayi2 && sayi1<sayi3) {

}
```

10.
```
if(editText_Adi.getText().equals(adiSoyadi)){

}
```

11.
```
int not1,not2,not3;
not1 = Integer.parseInt(editText1.getText().toString());
not2 = Integer.parseInt(editText2.getText().toString());
not3 = Integer.parseInt(editText3.getText().toString());
if(not1 >= not2){
    if(not1>=not3){
        textView.setText("En Büyük Not : "+not1);
    }else{
        textView.setText("En Büyük Not : "+not3);
    }
}else{
    if(not2>=not3){
        textView.setText("En Büyük Not : "+not2);
    }else{
        textView.setText("En Büyük Not : "+not3);
    }
}
```

12.
```
String gun;
gun=editText4.getText().toString();
switch (gun){
    case "Pazartesi":
        Toast.makeText(this, "Pazartesinden Günaydın", Toast.LENGTH_SHORT).show();
        break;
    case "Salı":
        Toast.makeText(this, "Salıdan Günaydın", Toast.LENGTH_SHORT).show();
        break;
    case "Çarşamba":
        Toast.makeText(this, "Çarşambadan Günaydın", Toast.LENGTH_SHORT).show();
        break;
    case "Perşembe":
        Toast.makeText(this, "Perşembeden Günaydın", Toast.LENGTH_SHORT).show();
        break;
    case "Cuma":
        Toast.makeText(this, "Cumadan Günaydın", Toast.LENGTH_SHORT).show();
        break;
    case "Cumartesi":
        Toast.makeText(this, "Cumartesinden Günaydın", Toast.LENGTH_SHORT).show();
        break;
    case "Pazar":
        Toast.makeText(this, "Pazardan Günaydın", Toast.LENGTH_SHORT).show();
        break;
    default:
        Toast.makeText(this, "Yanlış Gün Bilgisi", Toast.LENGTH_SHORT).show();
}
```

5. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D
2. D
3. Y
4. D
5. Y
6. D
7. Y
8. B
9. A
10. D
11. C
12. D

6. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D
2. D
3. D
4. Y
5. Y
6. Y
7. Y
8. D
9. Y
10. Y
11. return
12. uses-permission
13. Dangerous
14. Serializable
15. ViewBinding
16. Intent
17. match_parent
18. D
19. E
20. A
21. A

22.
```
<TextView
    android:id="@+id/textView_Adi"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView"/>
```

23.
```
<uses-permission android:name="android.permission.INTERNET"></
uses-permission>
```

24.
```
OgrenciBilgileri.java
Intent intent = new Intent(this, OgrenciDetay.class);
intent.putExtra("ogrAdi",adi);
intent.putExtra("ogrSoyadi",soyadi);
intent.putExtra("ogrNumara",numara);
startActivity(intent);
OgrenciDetay.java
Intent intent = getIntent();
String gelenOgrAdi = intent.getStringExtra("ogrAdi");
String gelenOgrSoyadi = intent.getStringExtra("ogrSoyadi");
int gelenOgrNum = intent.getIntExtra("ogrNumara",0);
```

7. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D	11. sharedPreferences	21. E
2. Y	12. put	22. B
3. Y	13. openOrCreateDatabase	23. C
4. D	14. Cursor	24. A
5. Y	15. execSQL	25. D
6. Y	16. BaseAdapter	26. E
7. D	17. getInstance	27. C
8. D	18. getCurrentUser	28. B
9. Y	19. update	29. A
10. D	20. response	30. D

31.
```
SharedPreferences sharedPreferences=getSharedPreferences("isim",MODE_PRIVATE);
```

32.
```
SharedPreferences.Editor editor=sharedPreferences.edit();
editor.putString("ad","deneme");
editor.apply();
```

33.
```
while (cursor.moveToNext())
{
        String ad= cursor.getString(0)
        liste.add(ad);
}
```

34.
```
if(FirebaseAuth.getInstance().getCurrentUser()!={
        // Yetkili
}else {
        // Yetkisiz giriş
}
```

35.
```
whereEqualTo("user",FirebaseAuth.getInstance().getCurrentUser().getUid())
```

8. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. D
2. D
3. D
4. Y
5. Y
6. D
7. D
8. Y
9. D
10. D
11. BroadcastReceiver
12. Manifests
13. SmsManager
14. Arka plan servisi
15. Bound
16. Notification
17. setSmallIcon
18. PendingIntent
19. setRequiresBatteryNotLow
20. Donanım
21. B
22. D
23. E
24. C
25. A
26. D
27. E
28. A
29. D
30. B

31.

```
public class MesajAlgilayici extends BroadcastReceiver
```

32.

```
Intent intent=new Intent(Intent.ACTION_SEND);
String email="eposta@posta.com;
String konu="başlık";
String mesaj="mesaj";
intent.putExtra(Intent.EXTRA_EMAIL,new String[]{email});
intent.putExtra(Intent.EXTRA_SUBJECT,konu);
intent.putExtra(Intent.EXTRA_TEXT,mesaj);
intent.setType("message/rfc822");
startActivity(intent);
```

33.

```
Intent intent=new Intent(MainActivity.this,Servis.class);
startService(intent);
```

34.

```
Constraints constraints = new Constraints.Builder()
            .setRequiredNetworkType(NetworkType.CONNECTED)
            .setRequiresCharging(true)
            .setRequiresBatteryNotLow(true)
            .build();
```

9. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

1. B
2. C
3. B
4. C
5. A
6. C
7. C
8. E

KAYNAKÇA

Android Studio Developers web sitesi. https://developer.android.com/studio/intro, Mayıs 15 2022, kaynağından alınmıştır.

Aliferi, C. (2016). Android Programming Cookbook. Yunanistan: Exelixis Media. P.C.

Bilişim Teknolojileri Alanı Çerçeve Öğretim Programı

Cardle, J. P. (2017). Android App Development in Android Studio. İngiltere: CreateSpace Independent Publishing Platform.

DiMarzio, J. F. (2017). Android Programming with Android Studio. Amerika Birleşik Devletleri: John Wiley & Sons, Inc.

Horstmann, C. (2010). Big Java. Amerika Birleşik Devletleri: John Wiley & Sons, Inc.

Türk Dil Kurumu, sözlükleri. (t.y.). https://sozluk.gov.tr/

Türk Dil Kurumu (t.y.). https://www.tdk.gov.tr/

* Kaynakça, APA6 referanslama sistemi kullanılarak oluşturulmuştur.

GÖRSEL KAYNAKÇASI

Görsel Kaynakçasına ulaşmak için kodu tarayınız.

http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=2399

NOTLAR: