DENEY NO: 4

MOBİL UYGULAMALAR

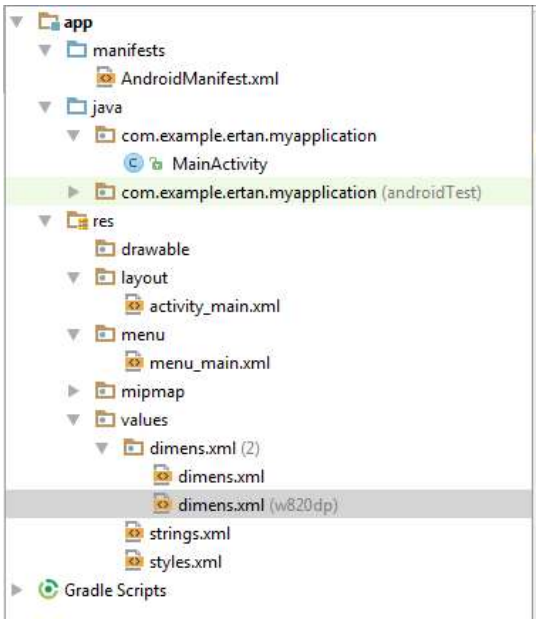
1. DENEYİN AMACI

Bu deneyde Android tabanlı akıllı telefonlarda uygulama geliştirmek için gerekli temel bilgilerin edinilmesi ve bazı örnek uygulamaların yapılması amaçlanmaktadır.

2. UYGULAMA GELİŞTİRMEYE GİRİŞ

2.1 Android Programlama

Bu bölümde android projenin temel bileşenlerinden bahsedilecektir. Android uygulama geliştirme ortamı için Android Studio kullanılacaktır. Android Studio ortamında bir android projesinin dosya ve klasörleri aşağıdaki şekildeki gibidir. Devam eden bölümde proje klasörü içerisindeki bazı temel bileşenlerden bahsedilecektir.



Şekil 1 Android Studio proje klasör ve dosyaları

- **src klasörü**: Kaynak dosyaları burada yer alır. Bu klasörün içinde Java dosyalarını tutulmaktadır.
- **res klasörü:** Kaynak kod dışında resimler, videolar, stiller, layout'lar gibi görsellikle ilgili her şey res klasöründe tutulur. Uygulamanın farklı cihaz konfigürasyonlarına göre kaynak kodu değiştirmeden kolay bir şekilde güncellenmesi sağlanır. Örneğin ekranı uzun olan ve geniş olan iki ayrı telefon için iki ayrı layout tanımlanarak uygulamanın iki cihaz için de uyumlu olması sağlanabilir. Res klasörü altında aşağıdaki klasörler bulunur.
  - **drawable klasörleri:** Bu klasörler uygulamada kullanılan resim dosyalarını içerir. Bu dosyalar PNG ya da JPEG formatında olabilir. Klasörün yanındaki hdpi (high dpi), ldpi (low dpi), mdpi (medium dpi) ve xhpi (extra hight dpi) cihaza özel ekran çözünürlüklerine göre

dosya çağırmamızı sağlar. Örneğin uygulamanın çalıştığı cihaz eski modelse ve ekranı düşük çözünürlük destekliyorsa ldpi klasörü içindeki resim dosyaları kullanılacaktır. Ama uygulama yeni nesil geniş ekran bir cihazda çalışıyorsa ona uygun olarak yüksek çözünürlüklü bir klasörde bulunan dosyalar kullanılır. Eğer uygulamanızda düzgün bir tasarım varsa ve geniş yelpazede cihazlar destekliyorsanız dosyalarınızın tüm farklı çözünürlüklere göre uygun formatlarının bulunması gerekir.

- **layout klasörü:** Burada ekranlara dair tasarım dosyaları bulunur. *xml* formatındaki bu dosyalar her ekrana ait tasarımları barındırır. Bir ekran **Activity** ile oluşturulduğunda **onCreate** metodu içinde ilgili layout çağırılır ve ekranda yer alacak öğeler oluşturulur.
- **menü klasörü:** Eğer bir ekranda cihazın **Menü** tuşuna basıldığında bir menü çıkmasını istiyorsak, menü elemanlarını bir **xml** dosyasında tanımlayarak bu klasör içine saklarız.
- **values klasörü:** Uygulamada kullanılan sabit değişkenler burada saklanabilir. **strings.xml** dosyası uygulamada kullanılan ve ekranlarda kullanıcıya gösterilen her türlü metni saklar. Anahtar – veri mantığıyla saklanan bu değerler kod içinde ya da layout dosyalarında çağırılır. Aynı zamanda bu değerler **R.java** dosyasında işaretlenir. **styles.xml** dosyası ise ekranlarda kullanılan ve yine layout dosyalarından çağrılan stilleri içerir. **values** klasörünün bir başka özelliği de cihazın ayarlanmış ana diline göre yerelleşebilmesidir.

- **AndroidManifest.xml dosyası:** Manifest dosyasının temel görevi uygulama bileşenlerinin sisteme bildirilmesidir. Uygulamayla ilgili her türlü özellik ve uygulamanın işletim sisteminden talep edeceği bütün izinler burada tanımlanır. Aynı zamanda uygulama içinde kullanılan her ekran burada kaydedilip tanımlanmak zorundadır. Aşağıdaki örnek bir AndroidManifest. xml dosyası verilmiştir.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.simpleproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="19" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Bu örnekte bir tane activity tanıtılmış ve intent-filter ile bu activity'nin başlangıçta çalıştırılacak activity olduğu belirtilmiştir. Yeni bir proje oluşturulduğunda varsayılan olarak bir tane MainActivity eklenmektedir ve AndroidManifest dosyası otomatik bir şekilde Android Studio tarafından oluşturulmaktadır.

Manifest dosyasında uygulamanın ihtiyaç duyduğu kullanıcı izinleri (user permissions) belirlenmekte, uygulamanın ihtiyaç duyduğu minimum API Level, donanımsal ve yazılımsal özellikler deklare edilmekte (kamera, bluetooth vs.), ihtiyaç duyulan api kütüphaneleri belirtilmektedir.

Örneğin bir uygulama kameraya ihtiyaç duyuyor ve Android 2.1 (API Level 7)'den itibaren ortaya çıkan api'leri kullanıyorsa manifest dosyasında aşağıdaki gibi belirtilmelidir.

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
    ...
</manifest>
```

Uygulamanın ihtiyaç duyduğu şeylerin manifest dosyasında belirtilmesi sayesinde Google Play arama sonuçlarında bu ihtiyaçları barındırmayan telefonlara söz konusu uygulamayı göstermeyecektir.
Bir uygulamada, gelen SMS mesajlarının görüntülenmesine ihtiyaç duyuluyorsa aşağıdaki gibi sms'lere erişim izni manifest dosyasında belirtilmelidir.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    ...
</manifest>
```

- **strings.xml dosyası:** Farklı stil ve formatta global olarak erişilebilen string değişkenlerin tanımlandığı yerdir. Kullanımı ile ilgi aşağıda bir örnek verilmiştir.
`res/values/strings.xml` dosyasına aşağıdaki xml kodun yazıldığını düşünelim.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
</resources>
```

`hello` string değişkeni layout dosyasında aşağıdaki gibi çağrılır.

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

Kod içerisindeki kullanımı ise aşağıdaki gibidir.

```
String string = getString(R.string.hello);
```

- **Activity sınıfı: Activity** sınıfı android programlamanın temel bileşenlerinden biridir. Android uygulamada kullanıcının herhangi bir şey yapmak için etkileşimde bulunduğu her ekran bir activity'dir. **Activity**'lerin uygulamada kullanılabilmesi için **manifest** dosyasında tanıtılması gerekmektedir. Aşağıdaki örnekte bir activity tanımlaması verilmiştir.

```
<manifest ... >
 <application ... >
   <activity android:name=".ExampleActivity"
```

```
            android:label="ExampleActivity"/>
    </application ... >
    ...
</manifest >
```

android:name java class'ının adı, **android:label** activity'nin uygulamada görünen ismidir.

- **intent sınıfı:** Bir Android uygulaması birçok farklı Activity içerir. Her Activity yeni arayüzler gösterirken belirli bir görevi (haritanın gösterilmesi, fotoğraf çekilmesi gibi) yerine getirir. Kullanıcıyı bir Activity'den diğerine geçirmek için uygulamanızda Intent sınıfını kullanmalısınız. Uygulamanızın yapacağı herhangi bir işte "amacı" belirtmek için Intent sınıfını kullanmalısınız. Uygulamanızda startActivity() metoduyla sisteme Intent geçirdiğinizde, sistem doğru uygulamayı ve eylemi belirlemek için bu Intent'i kullanır. Intent sınıfı, farklı uygulamalar tarafından kullanılan bir activity'yi çalıştırmaya da şans verir.

  Bir Intent belli bir bileşeni başlatmak (örneğin: belirli bir Activity'yi) için açık (**explicit**) olabilir. Explicit intent oluşturulurken başlatılacak bileşenin class adı mutlaka verilmelidir. Bu bileşen uygulumanız içindeki bir activity ya da servis olabilir.

  Uygulama geliştirirken sıklıkla **activity** başlatmaya ihtiyaç duyulacaktır. Aşağıda bir **activity** içerisinde başka bir **activity**'nin **explicit** şeklinde çağrılması gösterilmiştir.

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

  Hedeflenen amacı cihazda gerçekleştirebilecek (örn: fotoğraf çekilmesi) herhangi bir uygulamayı başlatmak için örtülü (**implicit**) intent kullanılır. Kendinizin gerçekleştirmediği ancak ihtiyaç duyduğunuz bir işi cihazda yapabilen başka bir uygulamaya yaptırmak için implicit intent kullanılır. Örneğin yaptığınız uygulamada bir içeriği kullanıcının diğer insanlarla paylaşabilmesini istiyorsunuz. Bunun için ACTION_SEND action'ı ile bir intenti aşağıdaki gibi oluşturabilirsiniz.

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

  Bu şekilde intenti oluşturduğunuz adnroid sistem diğer insanlara text paylaşılabilen uygulamaları kullanıcıya gösterir. Örneğin sms, whatsapp, gmail vb. Kullanıcı da hangi uygulama üzerinden text mesajı paylaşacağını seçer.

  Uygulamanızda email, sms gönderme, durum güncelleme, veri kullanımı gibi bazı olaylar gerçekleştirmek isteyebilirsiniz. Bunları gerçekleştirmek için bu işleri yapan **activity**'lerin sizin activiy'niz olması gerekmez. **Intent** ile gerçekleştirmek istediğiniz işleri yapacak **activity**'leri başlatabilirsiniz. Örneğin kullanıcıya uygulamanızda email göndermek istiyorsanız aşağıdaki

gibi **implicit intent** oluşturabilirsiniz. Dikkat edilecek olursa intent oluşturulurken başlatılacak bileşenin class adı verilmemiş sadece yapılacak eylemin türü `ACTION_SENDTO verilmiş.`

```
public void composeEmail(String[] addresses, String subject) {
    Intent intent = new Intent(Intent.ACTION_SENDTO);
    intent.setData(Uri.parse("mailto:")); // only email apps should handle this
    intent.putExtra(Intent.EXTRA_EMAIL, addresses);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

Bazen activity'lerden bir geri dönüş değeri almak isteyebilirsiniz. Bunun için **startActivity()** yerine **startActivityForResult()** metodu kullanılır. Sonuç almak için **onActivityResult()** callback metodunun override edilmesi gerekmektedir. Başlatılan activity, işini tamamladığında intent içindeki sonuç ile **onActivityResult()** metoduna geri döner. Aşağıda rehberden bir kişinin getirilmesi örneği verilmiştir.

```
private void pickContact() {
    // Create an intent to "pick" a contact, as defined by the content
provider URI
    Intent intent = new Intent(Intent.ACTION_PICK, Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
    // If the request went well (OK) and the request was
PICK_CONTACT_REQUEST
    if (resultCode == Activity.RESULT_OK && requestCode ==
PICK_CONTACT_REQUEST) {
        // Perform a query to the contact's content provider for the
contact's name
        Cursor cursor = getContentResolver().query(data.getData(),
        new String[] {Contacts.DISPLAY_NAME}, null, null, null);
        if (cursor.moveToFirst()) { // True if the cursor is not empty
            int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
            String name = cursor.getString(columnIndex);
            // Do something with the selected contact's name...
        }
    }
}
```

Android'in uygulamalara sağladığı önemli özelliklerinden biri de kullanıcıyı gerçekleştirmek istedikleri "eyleme göre" başka bir uygulamaya yönlendirme olanağı sunmasıdır. Örneklemek gerekirse, çeşitli resim işleme eylemleri yaptığınız bir uygulamada fotoğraf çekmek için bir activity yazmanıza gerek yoktur. Bunun yerine fotoğraf çekilmesi için bir Intent oluşturmanız ve çalıştırmanız yeterlidir. Sistem sizi kamera uygulamasına kendiliğinden yönlendirecektir.

Uygulamanızın Activity'leri arasında geçmek için Intent kullanmalısınız. Bunu genellikle adresi belli intent'ler (explicit intent) kullanarak, çalıştırmak istediğiniz bileşenin class ismini

kullanarak yaparsınız, fakat başka uygulamada bir eylem gerçekleştirmek istediğinizde, örneğin "haritayı göster" gibi, **üstü kapalı intent (implicit intent)** kullanmak zorundasınız.

Üstü kapalı (Implicit) intent oluşturma
Üstü kapalı intent, çalıştırmak istediğiniz bileşenin class'ını çağırmak yerine gerçekleştirmek istediğiniz eylemi belirttiğiniz intent türüdür. Belirttiğiniz eylem ne yapılacağını belirtir, örneğin bir şeyi göster (view), gönder (send), getir (get) gibi. Intentler eylemle ilgili çeşitli bilgiler de taşır. Örneğin göstermek istediğiniz adres, e-posta atılacak kişi, aranacak numara gibi. Oluşturmak istediğiniz Intent'e göre bu veri Uri veya diğer veri tiplerinin biçiminde olabilir veya Intent'in hiçbir veri taşımaya ihtiyacı da olmayabilir.

Eğer veriniz Uri şeklindeyse, basit bir Intent() yapılandırıcı metoduyla (constructor) eylemi ve verileri tanımlayabilirisiniz.

Örneğimizde bir telefon araması yaparken Uri verisini nasıl kullandığımızı görebilirsiniz.

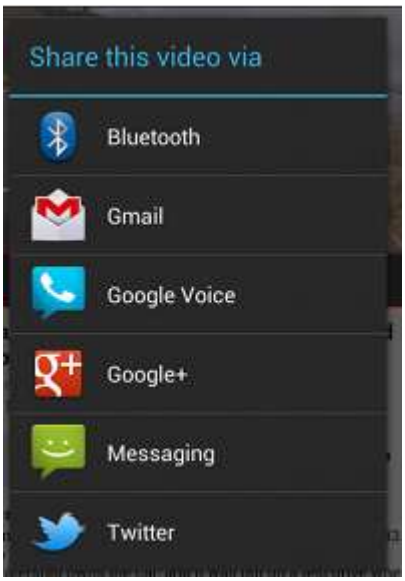
Uri number = Uri.parse("tel:5323334455");

Intent callIntent = new Intent(Intent.ACTION_DIAL, number);

Uygulamanız startActivity() ile callIntent isimli Intent'i çağırdığında Telefon uygulaması ilgili numarayı arar.

Uygulama seçim diyalogu gösterme

Bir Activity'yi oluşturduğunuz Intent'i startActivity() metoduna geçirerek başlattığınızda ve Intent'inize yanıt verebilecek birden fazla uygulama olduğunda kullanıcı varsayılan olarak kullanılacak uygulamayı seçebilir. (Diyalog pencereciğinin altındaki seçenek kutusunu seçerek; Bkz: Resim 1) Bu durumlarda oluşan pencereden kullanıcı her zaman kullanacağı uygulamayı seçebilir. Web sitesi açılışlarında kullandığı web tarayıcıyı veya fotoğraf çekmek için kullandığı kamera uygulamasını insanlar genelde değiştirmek istemezler. Bu gibi durumlar için hoş bir özelliktir.



Resim 2: Seçim diyalogu

Bunun yanında bazı uygulamalarda bu tercih sık sık değişebilir, örneğin paylaşma aksiyonunu bazen Bluetooth, bazen e-posta bazen mesaj tercih edilebilir. Bunun gibi değişkenlik gösteren isteklere karşı bir seçim ekranı kullanmanız ve her zaman kullanmanız gereklidir.

Seçim pencereciğini göstermek için createChooser() metodunu kullanarak bir Intent oluşturup, startActivity()'ye geçirebilirsiniz. Örneğin,

```
Intent intent = new Intent(Intent.ACTION_SEND);

...

// arayüz metinleri için her zaman string kaynakları kullanın
// örneğin burada title değişkenine "Fotoğrafı şununla paylaş"
// metnini atamış oluyoruz
String title = getResources().getString(R.string.secici_basligi);
// seçici ekranı göstermek için Intent oluşturuyoruz
Intent chooser = Intent.createChooser(intent, title);

// Intent'in en az bir Activity çözümleneceğini doğruluyoruz
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```

Bu kodla createChooser() metoduna geçirilmiş Intent'e karşılık gelebilecek uygulamaların listesini gösteren bir diyalog pencereciğinde gösterilir ve R.string.secici_basligi değişkeninden sağlanan metin diyalog başlığı olarak kullanılır.

Diğer Uygulamaların Sizin Activity'nizi Başlatmasına İzin Vermek

Bundan önceki eğitim içeriklerinde genelde şunu yapıyorduk: kendi uygulamamızda başka uygulamaların Activity'sini başlatıyorduk. Uygulamanız başka bir uygulama için kullanışlı olabilecek bir eylemi gerçekleştiriyorsa, onu başka uygulamalardan gelen isteklere yanıt verecek şekilde hazırlamanız gerekir. Örneğin kullanıcının arkadaşlarıyla mesaj veya fotoğraf paylaştığı bir sosyal ağ uygulaması yapıyorsanız, ACTION_SEND Intent'ini desteklemelisiniz ki, kullanıcılar başka bir uygulamadaki "paylaş" eylemini kullandıklarında bu tür eylemleri uygulamanız üzerinden gerçekleştirebilsinler.

Diğer uygulamaların Activity'nizi başlatmasını sağlamak için manifest dosyasında uygun <activity> elementinin içine <intent-filter> elementini koymalısınız.

Uygulamanız cihaza yüklendiğinde, sistem, uygulamanızın Intent filter'ına bakarak buradan ürettiği bilgileri kurulu uygulamaların Intent'lerinden oluşan dâhili bir kataloğa ekler. Başka bir uygulama startActivity() ya da startActivityForResult() metotlarını örtülü (implicit) Intent ile çağırdığında, sistem hangi Activity'lerin çağrıyı cevaplayabileceğini bu kataloğa bakarak bulur.

Intent filter ekleme

Activity'nizin karşılayabileceği Intent'leri belirtmek için her birine (Activity'nin kabul ettiği veri ve eyleme özel şekilde) özel intent filter'ı uygun şekilde mümkün mertebe spesifik tanımlamalısınız.
Aşağıdaki örnekte, ACTION_SEND Intent'ine, veri tipi metin veya resim oldukça karşılık verebilen Activity için tanımlanmış intent filter'ını görüyorsunuz:

```
<activity android:name="ShareActivity">
    <intent-filter>
```

```
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
        <data android:mimeType="image/*"/>
    </intent-filter>
</activity>
```

Uygulamaya gelecek her Intent bir eylem ve bir veri tipiyle gelir. Fakat tanımladığınız her <intent-filter>'ın içinde <action>, <category> ve <data> elementlerinin örneklerini birden fazla ilan edebilirsiniz ki, bu sorun oluşturmayacaktır.

Herhangi iki eylem ve veri çifti birbiriyle çakışıyorsa, onları kabul ettikleri veri tiplerine göre iki ayrı intent filter olarak ayırmalısınız.

Örneğin, Activity'nizin ACTION_SEND ve ACTION_SENDTO Intent'leriyle metin ve resme karşılık verebildiğini varsayın. Böyle bir senaryoda iki ayrı intent filter tanımlamalısınız çünkü ACTION_SENDTO Intent'i send veya sendto Uri şemasını kullanarak, alıcının ismini tanımlarken Uri verisini kullanmaya ihtiyaç duyar. Örneğimizi inceleyelim:

```
<activity android:name="ShareActivity">
    <!-- metin göndermeye uygun intent-filter; SENDTO eylemini sms URI
şemalarıyla kabul eder -->
    <intent-filter>
        <action android:name="android.intent.action.SENDTO"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="sms" />
        <data android:scheme="smsto" />
    </intent-filter>

    <!-- metin veya resim göndermek için intent-filter; SEND eylemini metin
veya resim verisiyle kabul eder -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="image/*"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

A.ş linklerde activity'ler arası veri gönderimi örnekleri vardır.
http://developer.android.com/training/basics/firstapp/starting-activity.html
http://www.youtube.com/watch?v=SaXYFHYGLj4

2.1.1 Layout'lara giriş

Layout'lar her ekrana ait tasarımları barındırır. Aşağıda örnek bir layout verilmiştir.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
```

```
            android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Bu layout `res/layout/main_activity.xml` olarak kaydedildikten sonra bir Activity'de yüklenmesi için **onCreate()** metodunda aşağıdaki kodun yazılması gereklidir.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_activity);
}
```

Resources kod içinde **R** ile çağrılır. Yukarıdaki örnekte görüldüğü gibi `res/layout` klasörü altındaki main_**activity.xml** layout dosyası `R.layout.main_activity` şeklinde çağrılmıştır.

android:layout_height ve **android:layout_width** attribute'leri ya birim değer, (ör: 10dp) ya da `match_parent`, `fill_parent`, `wrap_content` anahtar değerlerinden birini alırlar. `match_parent` ile `fill_parent` aynı amaç için kullanılır elemanın parent'inin boyutlarını alırlar. `wrap_content` içerik için ihtiyaç duyulan minimum boyutu alır.

`layout_weight` ile yatay ya da dikey olarak ekranda kalan alanın elemanlara hangi oranda verileceğini belirlenir. Aşağıda **`layout_weight`** ile ilgil bir örnek verilmiştir.

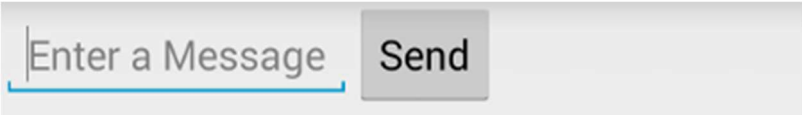
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

Yukarıdaki kodun ekran görüntüsü aşağıda verilmiştir.

Yukarıdaki örnekte EditText aşağıdaki gibi değiştirilirse

```
<EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
```

Aşağıdaki görünüm elde edilir.



Yaygın layout'lar: **Linear**, **Relative** ve **Web View**'dir. **Linear layout** ile elemanlar yatay ya da dikey doğrultuda yerleştirilir. **Relative layout**'da elemanların konumları birbirlerine göre tanımlanır. Örneğin B elemanının konumu A elemanın solundadır şeklinde tanımlanabilir. **Web view** ile web sayfaları görüntülenir.

Relative layout'un bazı özellikleri aşağıda verilmiştir.

android:layout_alignParentLeft="true",elemanın sol kenar hizasını parent elemanın sol kenar hizasına getirir.

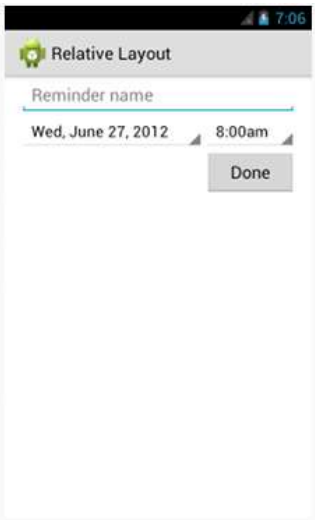
android:layout_alignParentTop="true", elemanın üst kenar sol hizasını parent elemanın üst kenar hizasına getirir.

android:layout_below="@id/name", elemanı id'si "name" olan elemanın altına yerleştirir.

android:layout_toLeftOf="@+id/times", elemanı id'si "times" olan elemanının soluna yerleştirir.

Aşağıda relative layout ile ilgili bir örnek verilmiştir.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

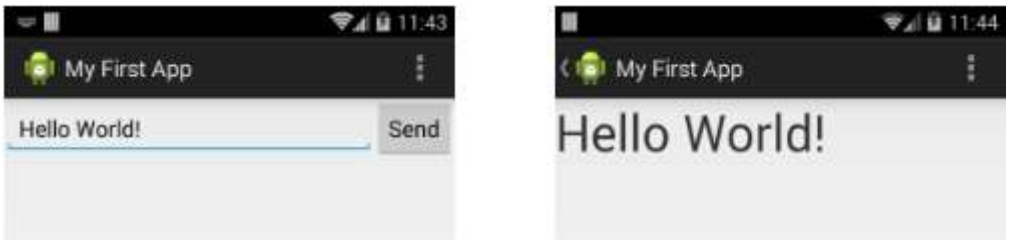


2.1.2 Basit Bir Proje Yapma ve Telefona Yükleme

Basit bir örnek yapmak için 3. Bölümde anlatılan Android Studio Kurulumu ve Akıllı Telefon yapılandırılması adımlarını uygulayınız.

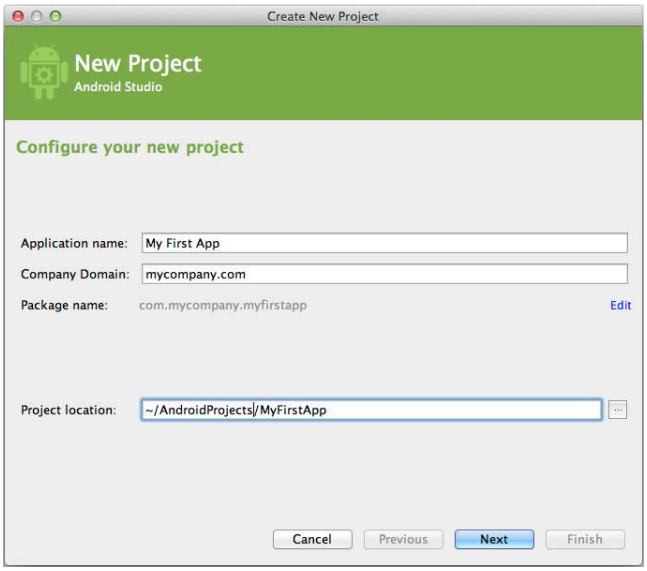
Bu bölümde başka bir **activity** başlatma ve **activity**'ler arası veri gönderme örneği yapılacaktır. Örneği http://developer.android.com/training/basics/firstapp/index.html adreslerindeki adımları takip ederek de deneyebilirsiniz.

Activity ekranları aşağıdaki şekildeki gibidir.



Şekil 2 Örnek uygulamanın ekran görüntüleri

Android Studio ortamında yeni bir proje açıp proje adını giriniz.

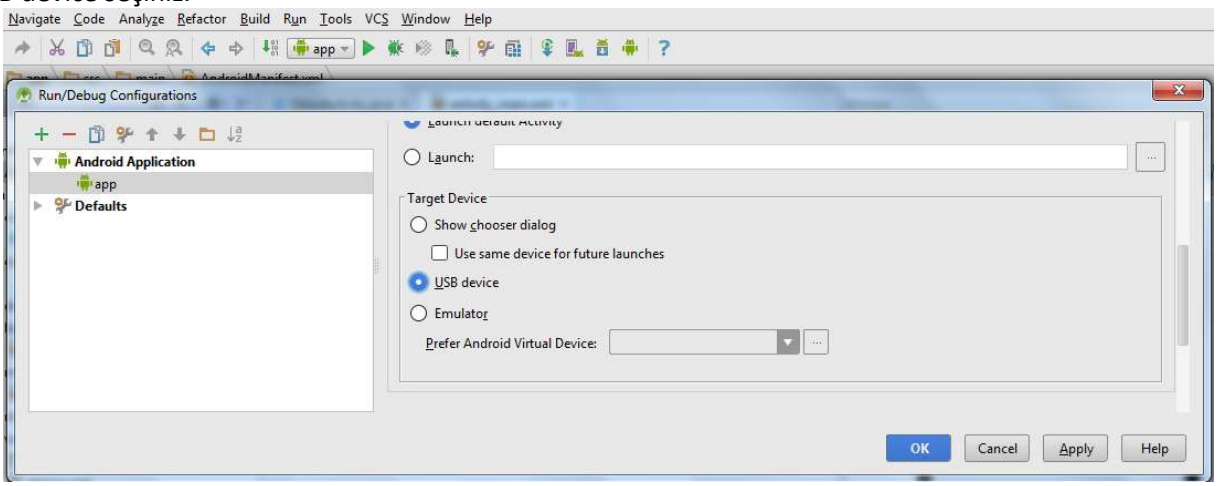


Şekil 3 Yeni proje ekranı

Uygulamanızı Google Play Store'ye yüklemeyi düşünürseniz, uygulamanızın paket adı (**package name**) Google Play Store'de bulunan uygulamalarla aynı isimde olmamalıdır. Başka uygulamalarla karıştırılmaması için mümkün olduğunca kendine has bir isim verilmelidir. Minimum Required SDK ile yaptığımız uygulamanın hangi minimum SDK'da çalışabileceği belirlenmektedir. Burada seçilen min SDK'nın altındaki SDK'larda uygulama çalışmayacaktır. Next'lerle ilerleyip projeyi oluşturunuz. Aşağıdaki gibi bir ekran gelecektir.

Şekil 4 Android Studio proje ortamı

Uygulamayı Telefonda çalıştırmak için **Run > Edit Configurations** kısmında **Target Device** olarak USB device seçiniz.



Telefonunuzu bilgisayara bağlayıp run butonu ile uygulamayı telefonunuzda çalıştırınız.

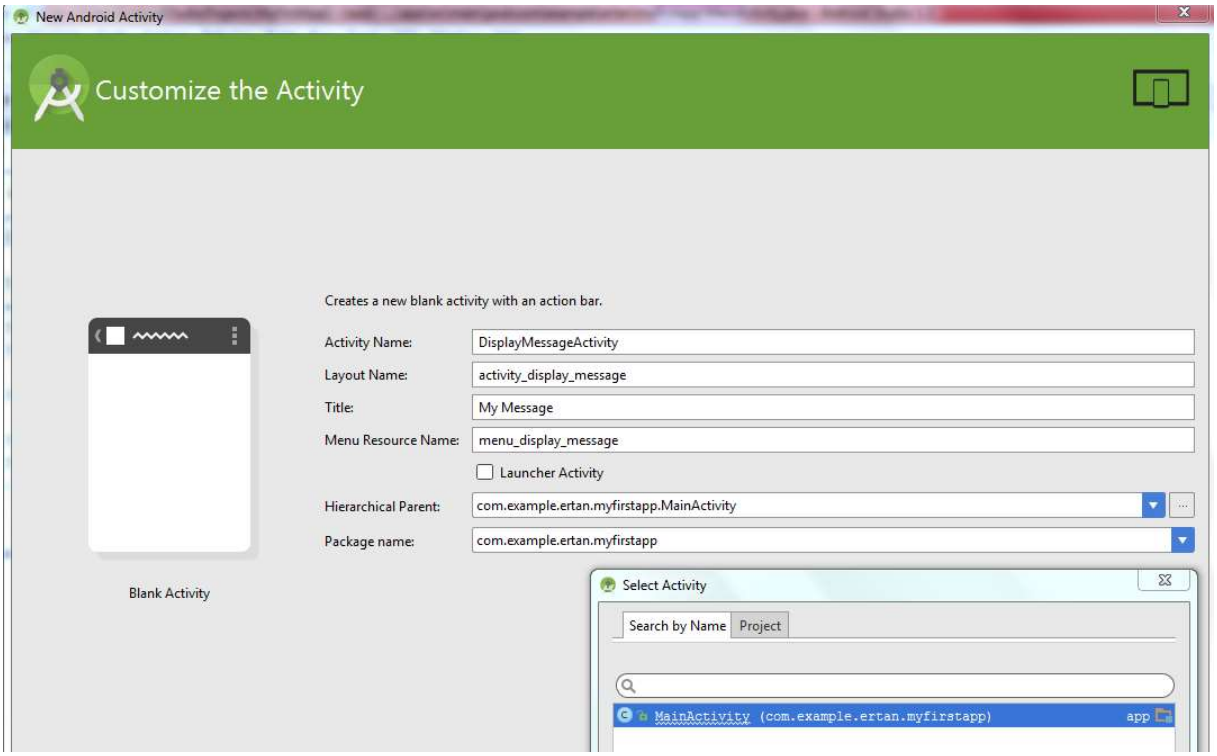
res/values/string.xml dosyasına **edit_message**, **button_send** değişkenlerini aşağıdaki gibi ekleyiniz.

```
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
    <string name="title_activity_display_message">My Message</string>
</resources>
```

res/layout/activity_main.xml dosyasına EditText ve Button elemanlarını aşağıdaki gibi ekleyiniz. Button elemanının **onClick** event'ine **sendMessage** fonksiyon adını veriniz. Bu fonksiyonun java tarafındaki tanımını ileride verilecektir.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage"/>
</LinearLayout>
```

Şimdi projeye yeni bir **Activity** daha eklenecektir. Sol taraftaki java dosyalarının bulunduğu pakete sağ tıklayıp **New>Activity>Blank Activity**'i seçip aşağıdaki gibi isim verip Hierarchical Parrent'ten projedeki MainActivity'yi seçiniz.



Şekil 5 Yeni bir activity ekleme penceresi

res/layout/activity_display__text_message.xml dosyasında TextView ya da başka bir eleman varsa siliniz. Dosyanın içinde aşağıdaki gibi olacaktır.

```
<RelativeLayout 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" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context="com.example.ertan.myfirstapp.DisplayMessageActivity">

</RelativeLayout>
```

MainActivity.java dosyasına **EXTRA_MESSAGE** adında bir string ve **sendMessage** metodunu ekleyiniz. Bu metot ile **Display_TextMessage activity'**si başlatılmakta ayrıca bu activity'ye bir string mesaj gönderilmektedir. Eklenilen kodlar için gerekli kütüphaneleri de import ile ekleyiniz.

```
public final static String EXTRA_MESSAGE = "com.mycompany.myfirstapp.MESSAGE";

public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

Display_TextMessage.java dosyasının **onCreate** metodunda gelen **intent** nesnesini ve gönderilen string mesajını alınız. Gelen mesajı bir TextView'de gösteriniz. Eklenilen kodlar için gerekli kütüphaneleri de import ile ekleyiniz.

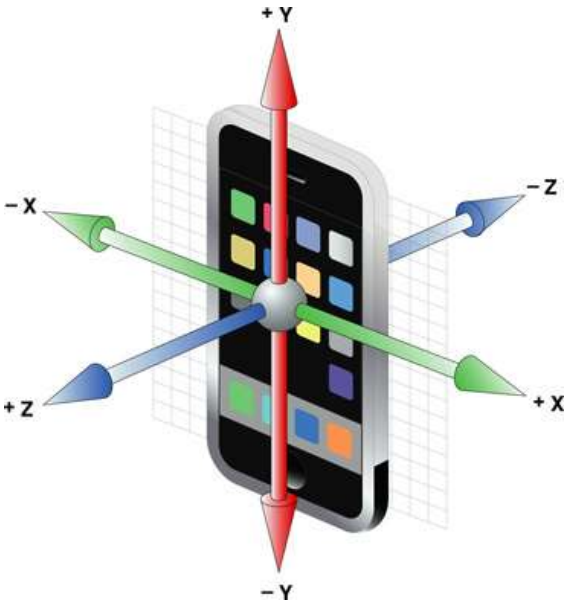
```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Get the message from the intent
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

    // Create the text view
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);

    // Set the text view as the activity layout
    setContentView(textView);
}
```

Bu aşamada activity'ler arası string mesajı için gerekli kodlamalar tamamlanmıştır. Şimdi sıra uygulamayı çalıştırmaktadır. Uygulamanın emülatör ortamında mı yoksa bilgisayara bağlı bir telefonda mı çalıştırılacağı **Run** menüsü altındaki **Edit Configurations** ile belirlenmektedir.

2.1.3 Accelerometer Sensor Örneği

Bu bölümde Accelerometer Sensor (ivme ölçer sensör) kullanımı ile ilgi bir örnek yapılacaktır. Accelerometer sensor koordinat eksenlerine (x,y,z) göre ivmeyi ölçmek için kullanılmaktadır. Telefon dik bir şekilde tutulduğunda koordinat eksenlerinden y ekseni yukarı ve aşağı yönlerini, x ekseni sağ

ve sol yönlerini z ekseni de ileri ve geri yönlerini göstermektedir. Telefon yatay bir şekilde tutulduğunda ise telefonun dikey konumdaki durumuna göre y ve z eksenleri yer değiştirmektedir.



Şekil 6 Telefon konumuna göre koordinat eksenleri

Telefon hangi yönde ivmeli hareket ederse bu değer sensörlere hemen yansıtılmaktadır. Accelerometer sensör yerçekimini de dikkate almaktadır. Telefon resimdeki gibi sabit bir şekilde tutulduğunda sensörün gönderdiği değerler y ekseni için 9.8m/s2, x ve z eksenleri için 0m/s2 olacaktır.

Aşağıda kodları verilen uygulama x, y ve z eksenleri için accelerometer sensörünü dinlemekte ve değerleri anlık olarak göstermektedir. Uygulama telefona yüklenip telefon eksenlerden herhangi bir yöne doğru ivmeli hareket ettirildiğinde değerlerin önceki değerlere göre bariz bir şekilde değiştiği görülecektir. Uygulamanın ekran görüntüsü aşağıdaki görüntüye benzer olacaktır.

Şekil 7 Accelerometer sensör örneği ekran görüntüsü

Uygulamada sensör kullanımı için Activity sınıfınıza SensorEventListener arayüzünü ekleyiniz.

```
public class MainActivity extends Activity implements SensorEventListener
```

Activity içerisindeki **onCreate** metodunda sistem servislerinden sensör servisini çağırarak **Accelerometer** sensörünü dinleyici olarak atayınız.

```
sm = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
accelerometer = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sm.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

Burada yapılan işlem sonrasında Accelerometer adlı sensörden gelen hareketler **onSensorChanged** metodu içerisinde dinlenmeye başlanacaktır. **registerListener** metodu içerisinde

tanımlanan **SENSOR_DELAY_NORMAL** değişkeni sensörün duyarlılığını belirler. Burada kullanabileceğimiz değerler aşağıdaki gibidir:
SENSOR_DELAY_NORMAL: Kullanıcının basit hareketlerini düşük duyarlılıkla takip eder. (215 - 230 ms)
SENSOR_DELAY_FASTEST: Sensörden gelen hareketlerin mümkün olduğu kadar fazla kısmının değerlendirilmesini sağlar. (15-20 ms)
SENSOR_DELAY_GAME: Oyun uygulamaları için yüksek duyarlılıkla dinleme gerçekleştirir. (35-40 ms)
SENSOR_DELAY_UI: Normal kullanım için idealdir. (85-90 ms)

Örnek uygulamanın kodları aşağıda verilmiştir.

#MainActivity.java

```
package com.example.accelerometer;
import android.app.Activity;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity implements SensorEventListener {
	Sensor accelerometer;
	SensorManager sm;
	TextView acceleration;

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

		sm = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
		accelerometer = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
		sm.registerListener(this, accelerometer,
				SensorManager.SENSOR_DELAY_NORMAL);

		acceleration = (TextView) findViewById(R.id.acceleration);
	}

	@Override
	public void onSensorChanged(SensorEvent arg0) {
		if (arg0.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
			float[] values = arg0.values;
			float x = values[0];
			float y = values[1];
			float z = values[2];

			acceleration.setText("X: " + x + "\nY: " + y + "\nZ: " + z);
		}
	}

	@Override
	public void onAccuracyChanged(Sensor arg0, int arg1) {
		// TODO Auto-generated method stub
	}
}
```

activity_main.xml layout'una x,y,z eksenlerindeki ivmeleri göstermek için bir TextView ekleyiniz.

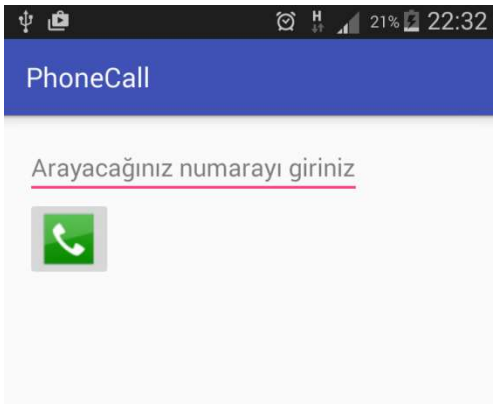
#activity_main.xml

```
<RelativeLayout 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"
    tools:context="${relativePackage}.${activityClass}" >
```

```
    <TextView
        android:id="@+id/acceleration"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="X: \nY: \nZ:" />

</RelativeLayout>
```

2.1.3 Telefon Arama Örneği

Bu örnekte girilen telefon numarasını arayan uygulama yapılmıştır. Uygulamanın ekran görüntüsü aşağıdaki gibidir.



MainActivity.java

```
package com.example.ertan.phonecall;

import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private static final int RESULT_PICK_CONTACT = 855;

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

        numara = (EditText) findViewById(R.id.id_numara);

        ImageButton ara_button = (ImageButton)
findViewById(R.id.id_call_image);
        ara_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent call_intent = new Intent(Intent.ACTION_CALL);

                call_intent.setData(Uri.parse("tel:" + numara.getText()));
                try{
```

```
                    startActivity(call_intent);

                }catch(android.content.ActivityNotFoundException e){
                    Toast.makeText(getApplicationContext(), "yourActivity
is not founded", Toast.LENGTH_SHORT).show();
                }

            }
        });
            }


}
```

activity_main.xml

Arama butonu imageButton'dur. İmage drawable klasörü içindedir, drawable/call.png.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity"
    android:orientation="vertical">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/id_numara"
        android:hint="Arayacağınız numarayı giriniz"/>

    <ImageButton
        android:id="@+id/id_call_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/call"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        />
</LinearLayout>
```

AndroidManifest.xml

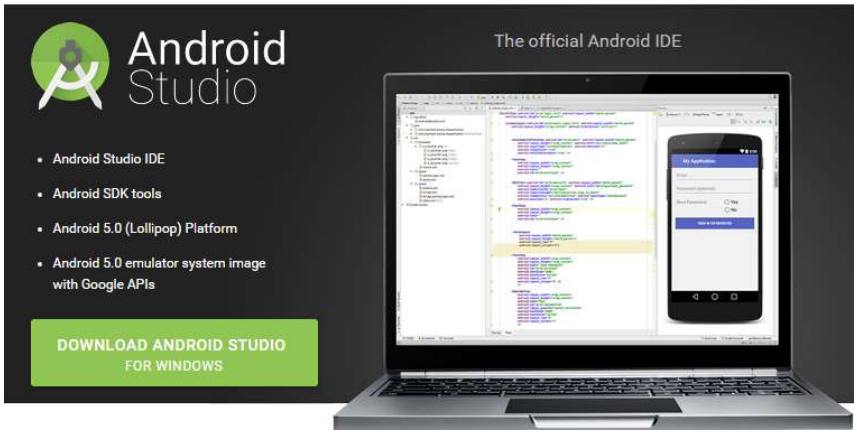
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ertan.phonecall" >
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
```

```
            </intent-filter>
        </activity>
    </application>

</manifest>
```

3. ANDROİD STUDİO KURULUMU VE TELEFON AYARLARININ YAPILMASI

3.1 Android Studio Kurulumu

1) Aşağıdaki linkten Android Studio'yu kurunuz.
http://developer.android.com/sdk/index.html



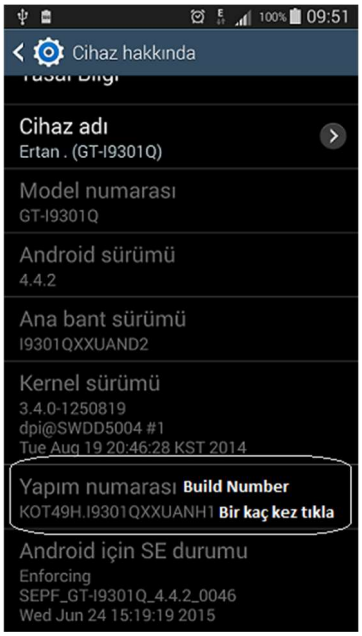
2) Tools menüsü altında **Android > SDK Manager** 'ı çalıştırıp http://developer.android.com/sdk/installing/adding-packages.html adresinde belirtilen güncellemeleri yapınız.

3.2 Akıllı Telefon Ayarlarının Yapılması

1) Eğer işletim sistemi akıllı telefonunuzun USB sürücüsünü yükleyememişse http://developer.android.com/tools/extras/oem-usb.html#Drivers adresinden telefonunuzun USB sürücüsünü yükleyiniz.

2) Telefonun **Ayarlar> Güvenlik > Bilinmeyen kaynaklar** kısmını aktif ediniz.

3) Androdi 4.2 ve sonrasında Geliştirici seçenekleri default olarak gizlidir. Bunu açmak için **Ayarlar> Genel >Cihaz Hakkında** bölümüne girip Yapım numarası kısmına arka arkaya birkaç kez tıklayınız.



Daha sonra **Ayarlar > Genel > Geliştirici seçenekleri** kısmında USB hata ayıklama (USB debugging) kısmını aktif ediniz.

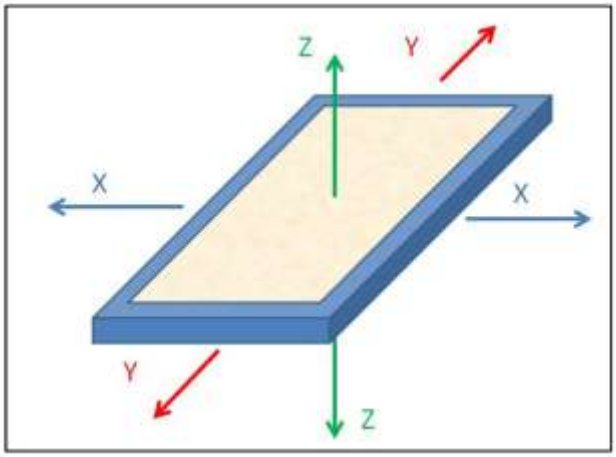
3) Son adımda Android Studio ortamından telefonunuza uygulama yüklemek için güvenlik iznini onaylayınız.

4. UYGULAMA ÖDEVLERİ

Android Ödev Uygulamalarında sensörler kullanılacaktır. Bu nedenle uygulama sınavı sırasında accelerometer sensöre sahip android telefon bulundurulmalıdır. Uygulama sınavı öncesi hazırlık çalışmalarının da gerçek telefon üzerinden yapılması tavsiye edilmektedir.

4.1 Y-Ekseni Boyunca Adım Sayıcı Uygulaması

Telefon Şekil 8'deki gibi yatay tutulup y ekseni boyunca hareket ettirildiğinde accelerometer sensörünü dinleyiniz belirlediğiniz uygun bir eşik değeri ile adım atma hareketini yakalayıp adım sayısını ekranda gösteriniz. Uygulamaya "**Başla**", "**Bitir**", "**Hedef Belirle**", "**Geçmişi Göster**" butonlarını ekleyiniz. Adım sayıcıyı "**Başla**" butonuna tıklandığında başlatınız ve ekranda atılan adım sayısını anlık olarak gösteriniz. "**Bitir"** butonuna tıklandığında adım sayıcıyı sonlandırıp adım sayısını gösteriniz; tarih, saat ve adım sayısı bilgilerini bir dosyaya yazınız. "**Hedef Belirle"** butonuna tıklandığında hedeflenen adım sayısını kullanıcıdan alınız, hedeflenen adım sayısına yaklaştıkça kullanıcıya "son 50 adım", "son 10 adım" gibi uyarı veriniz. "**Geçmişi Göster**" butonuna basıldığında dosyaya kaydedilen tarih, saat ve adım sayısı kayıtlarını yeni bir activity'de gösteriniz.



Şekil 8 Y-ekseninde adım sayıcı için telefonun yatay pozisyonu

4.2 Yukarı, Aşağı, Sağa ve Sola Doğru Yapılan Hareketleri Tanıyan Servis Uygulaması

Yukarı, aşağı, sağa ve solo yöne doğru yapılan hareketleri tanıyan bir **android servis uygulaması** yapınız. Telefon dik şekilde tutulup y-ekseninde + yönde (yukarı) ivmeli bir hareket ettirildiğinde telefonun sesini kısınız, aşağı yönde ivmeli hareket ettirildiğinde ise telefonun sesini azaltınız, sağa doğru hareket ettirildiğinde telefonun varsayılan müzik çaları çalıyorsa sonraki müziği başlatınız, sola doğru hareket ettirildiğinde telefonun varsayılan müzik çaları çalıyorsa önceki müziği başlatınız. Yaptığınız uygulamayı telefonun varsayılan müzik çaları ile test ediniz.

4.3 Telefonu Titretme Oyunu

-Uygulamada oyuncunun ismi alınacak ya da var olan daha önce eklenen oyuncu isimlerinin ve skorlarının tutulduğu listeden bir oyuncu seçilecektir.

-Başla butonuna tıklanıldığında 10 sn'lik bir bir sayaç başlatınız. Süre içerisinde telefon titretildikçe belirlenen bir eşik değerinin üstündeki titretme sayısını tutunuz.

-Süre bittiğinde titretme hareketini saymayı sonlandırınız ve titretme sayısını gösteriniz.

-Oyuncu listesinde oyuncuların isimleri ve yapmış oldukları maksimum titretme sayısı tutulacaktır.

-Yeni bir rekor kırıldığında oyuncunun adı ve rekoru notification(bildirim) olarak gösteriniz.

4.4 Telefonu Sallama Hareketini Tanıyan ve Müzik Çalan Uygulama

Telefonu sallama hareketini tanıyan (shake detection) ve bu hareket olduğunda müzik çalan bir android uygulama yapınız. Uygulamada çalınacak müzik kullanıcı tarafından seçilecektir. Bunun için uygulamaya bir dosya seçici (file chooser) ekleyiniz. Telefon sallandığında kullanıcının seçtiği mp3 dosyasını MediaPlayer sınıfını kullanarak yürütünüz, ekranda "çalıyor" yazısını gösteriniz. Telefon tekrar sallandığında media player'i durdurup ekranda "durduruldu" yazısını gösteriniz. Telefon sallandıkça çalma ve durdurma işlemleri tekrar edilecektir. Müzik bittiğinde ise "bitti" uyarısını gösteriniz.

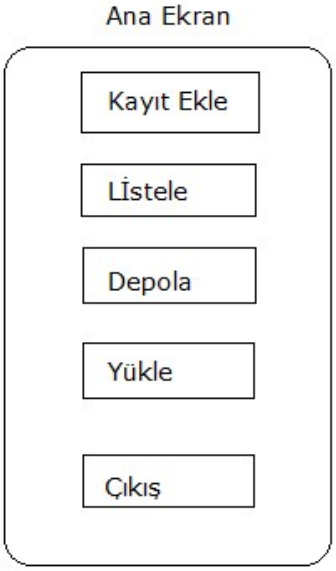
Not: Düzgün bir şekilde test yapmak için müzik içerisinde duraklamaların olmadığı bir mp3 dosyası kullanınız.

4.5 Android Kayıt Defteri Uygulaması

Uygulama Kayıt Ekle, Listele, Depola, Yükle işlemlerinden oluşacaktır.

- **Ana Ekran**

Ana aactivty'de uygulamada yapılacak işlemler buttonlarla gösterilecektir. Çıkış buttonuna tıklanıldığında veriler dosyaya kaydedilmişse dosyaya kaydedilip program kapatılacak. Dosya adı olarak daha önce bir şey girilmemişse dosya adına sistem tarihi verilecek.

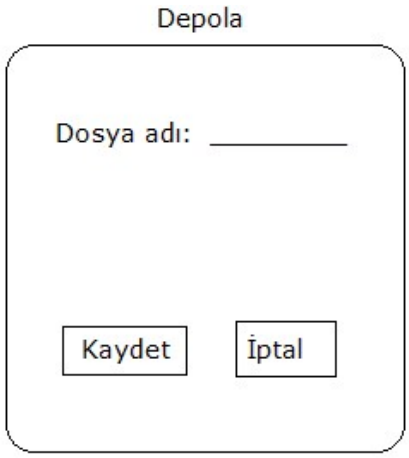


- **Kayıt Ekle**

Kayıt Ekle activity'de **Ekle** buttonuna tıklanıldıkça girilen kayıtlar bellekte istenildiği şekilde tutulacak, **Tamam** buttonuna basılınca Ana Ekran'a geri dönülecek.

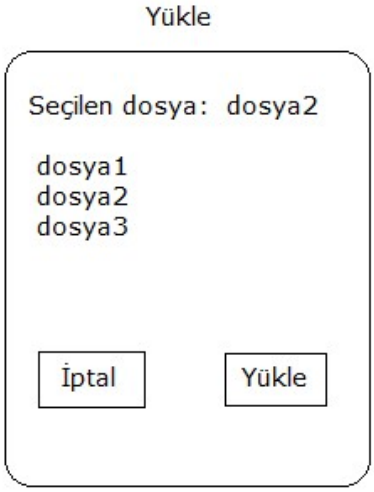
- Depola

Depola activity'de bellekte tutulan veriler girilen dosya adında bir dosyaya "isim,yaş,D.Yeri " formatında satır satır kaydedilecektir



- **Yükle**

Yükle activity'de daha önceden kayıtlı dosyalar listelenecek. Seçilen dosyadaki veriler belleğe yüklenecek.



- **Listele**

Listele activity'de bellekte tutulan kayıtlar listelenecek.

4.6 Mp3 Çalar

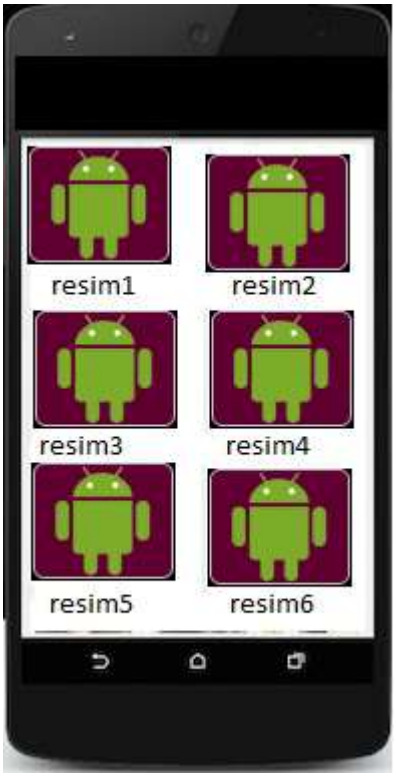
-Uygulamada belirtilen klasördeki mp3 dosyaları oynatılacak.
- Oynatma ekranına "SeekBar" eklenecek, oynatılan müziğin herhangi bir anına geçiş yapılabilecek
-Oynat, durdur, sonrakini öncekini oynat işlemleri yapılacak.

-Çalma listesi oluşturulabilecek. Çalma listesi ekranında daha önce oluşturulan listeler listelenecek, seçilen liste oynatılabilecek, yeni liste oluşturulabilecek.

-Çalma listesine geçiş oynatma ekranındaki menüden yapılacaktır.

4.7 Resim Galerisi

-Uygulamada belirtilen dosyadaki resimler GridView, GridViewAdapter kontrolleri kullanılarak küçük boyutlarda listelenecektir. Örnek ekran görüntüsü aşağıda verilmiştir.



-GridView içindeki herhangi bir resme tıklanıldığında başka bir ekranda resmin büyük hali gösterilecektir.

- GridView içindeki herhangi bir resme basılı tutulduğunda resmin oluşturulma tarihi, boyutu, yolu gibi ayrıntı bilgileri pop up pencerede gösterilecektir.

4.8 Telefon Sallandığında Arama Yapan Servis Uygulaması

-Accelerometer sensörü kullanılarak telefonda sallama hareketini (shake detection) tanıyan
-Sallama hareketi olduğunda son aranan numarayı arayan
-Arama bittiğinde arama süresini notification (bildirim) olarak gösteren **Servis uygulaması** yapınız.

5. UYGULAMA SINAVI

5.1 Uygulama Sınavında Bulunması Gerekenler

Uygulama sınavına girecek her öğrenci Android Studio programlarının kurulu olduğu bir bilgisayar ve accelerometer sensöre sahip android tabanlı bir akıllı telefon getirmelidir.

5.2 Uygulama Sınavında Bilinmesi Gerekenler

-Android Ödev Uygulamalarında sensörler kullanılacaktır. Bu nedenle uygulama sınavı sırasında accelerometer sensöre sahip android telefon bulundurulmalıdır. Uygulama sınavı öncesi hazırlık çalışmalarının da gerçek telefon üzerinden yapılması tavsiye edilmektedir.

-Android programlamanın temel bileşenleri hakkında aşağıdaki dokümanlar okunacaktır.

http://developer.android.com/guide/components/fundamentals.html
http://developer.android.com/guide/practices/compatibility.html
http://developer.android.com/guide/topics/security/permissions.html
http://developer.android.com/guide/components/intents-filters.html
http://developer.android.com/guide/components/activities.html
(Not: Uygulama sınavında bu kaynaklarla ilgili sorular sorulacaktır.)

-Her öğrenci kendisine verilen ödevde istenilen şeylerle ilgili temel bileşenlerin nasıl kullanıldığını bilmelidir. Bu bileşenlerin kullanımı ile ilgili küçük bir uygulama veya ödevde değişiklik istenecektir.

-Android uygulamaları için temel arayüz elemanlarının (text field, buton, list) kullanımı, ekranlar arası geçiş ve veri aktarımı gibi temel işlemlerin ve ivme sensörü kullanımının öğrenilmesi beklenmektedir.

Her öğrenci kendisi için belirlenen ödevi yapacaktır. Ödevlerde yapılacaklar "4. UYGULAMA ÖDEVLERİ" bölümünde açıklanmıştır.

5.3. Birinci Öğretim Ödev Dağılımları

"4.1 Y-Ekseni Boyunca Adım Sayıcı Uygulaması" ödevini ödev detayında açıklanan şekliyle eksiksiz gerçekleştiriniz.

Her öğrenci kendisine verilen ödevde istenilen şeylerle ilgili temel bileşenlerin nasıl kullanıldığını bilmelidir. Bu bileşenlerin kullanımı ile ilgili küçük bir uygulama veya ödevde değişiklik istenecektir.

5.4. İkinci Öğretim Ödev Dağılımları

"4.3 Telefonu Titretme Oyunu" ödevini ödev detayında açıklanan şekliyle eksiksiz gerçekleştiriniz.

Her öğrenci kendisine verilen ödevde istenilen şeylerle ilgili temel bileşenlerin nasıl kullanıldığını bilmelidir. Bu bileşenlerin kullanımı ile ilgili küçük bir uygulama veya ödevde değişiklik istenecektir.

6. KAYNAKLAR

6.1. Genel kaynaklar

- http://developer.android.com/training/index.html
- http://developer.android.com/samples/index.html
- http://developer.android.com/guide/index.html
- http://www.vogella.com/tutorials/android.html
- http://www.tutorialspoint.com/android/
- http://www.coreservlets.com/android-tutorial/
- http://androidexample.com/Android_Basic_Tutorial/index.php?view=category&categoryid=23
- https://www.technopat.net/tag/android-programlama/
- http://www.devfright.com/
- https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim/android-201/android-cihazlar-ve-android-isletim-sistemi-uzerine-genel-bilgiler

6.2 Yabancı üniversitelerden kaynaklar

- http://cs76.tv/2012/spring/
- http://www.youtube.com/watch?v=HgSMtGXq1aA&index=1&list=PLxrgvRt_aRIEum8eeVp0apeML4AZMapJ5

6.3 Android Sensor için kaynaklar

- http://www.vogella.com/tutorials/AndroidSensor/article.html
- http://nebomusic.net/androidlessons/Pedometer_Project.pdf
- http://jasonmcreynolds.com/?p=388 (ShakeDetector)

6.4 Android Servis uygulaması için kaynaklar

- http://developer.android.com/guide/components/services.html
- http://developer.android.com/training/best-background.html
- http://developer.android.com/samples/RepeatingAlarm/AndroidManifest.html
- http://www.vogella.com/tutorials/AndroidServices/article.html

6.5 Android dosya seçici için kaynak

- https://github.com/iPaulPro/aFileChooser/blob/master/aFileChooserExample/src/com/ipaulpro/afilechooserexample/FileChooserExampleActivity.java