P.S.R.ENGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Sevalpatti (P.O), Sivakasi - 626140.
Virudhunagar Dt.

LECTURE NOTES

COURSE CODE	**: 191CS53**
COURSE NAME	**: MOBILE APPLICATION DEVELOPMENT**
SEMSTER	**: V**
ACADEMIC YEAR	**: 2024-2025 (ODD SEMESTER)**

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

PREPARED BY

Dr.C. NATARAJAN
ASP/CSE

191CS53 **MOBILE APPLICATION DEVELOPMENT**

L	T	P	C
3	0	0	3

Programme: B.E. Computer Science and Engineering **Sem:** 5 **Category:** PC

Prerequisites: 191CS42 – Object Oriented Programming

Aim: To provide students with the tools and knowledge necessary to create mobile applications that can run on mobile devices.

Course Outcomes: The Students will be able to

CO1: Classify the various Android Applications using layout and menus.

CO2: Demonstrate the audio, video and animation application.

CO3: Explain the fundamentals concepts of web database application.

CO4: Develop various types of sensors in Android.

CO5: Interpret applications to hand-held devices.

CO6: Design and Develop the mobile application (Mini Project).

ANDROID OVERVIEW **9**

Overview of Android platform- Android SDK features - setting up the mobile app development environment along with an emulator- Android Application Development Architecture - creating an applications and activities -Application manifest

INTERFACE TOOLS **9**

Creating user interface - Views - creating views - Layouts - Drawable resources - resolution and density independence - Menus - Intents - Adapters - Using Internet resources - Dialogs.

FILES AND DATABASES **9**

Saving Simple Application Data - creating and saving preferences - preferences activity -saving activity state - loading files - file management tools-sending emails through application - Introducing Android databases -SQLite - Web Database- Firebase, MySQL-PHP

SMALL COMPUTING TECNOLOGY AND SENSORS **9**

Audio, Video Using the Camera - Telephony And SMS - Bluetooth Networks - Managing network connectivity - WI-FI - **Sensors**-Sensors and the Sensor Manager - Interpreting sensor values-Using the compass, Accelerometer and Orientation sensor.

ADVANCED TECHNOLOGY **9**

Paranoid Android - Using Wake Locks - AIDL to Support IPC for Services -General API's- Payment-gateway, Android jetpack-Technology II-IOS-Introduction to Objective C-IOS features.

Total Periods: 45

Text Book:

1. Jeff Mc Wherter and Scott Gowell, " Professional Mobile Application Development" Wrox,2012
2. David Mark, Jack Nutting, Jeff LaMarche and Frederic Olsson "Beginning ios 6 Development: Exploring the iOS SDK", Apress, 2013
3. "Teach Yourself Android Application Development in 24 Hours", SAMS publication, 3/e, 2013

References:

1. Anubhav Pradhan, Anil V. Deshpande, "Composing Mobile Apps: Learn. Explore. Apply. Using Android", Wiley publication, 2014.
2. Barry Burd, "Android Application Development All in one for Dummies", John Wiley & Sons publication,2011.
3. http://developer.android.com/develop/index.htm
4. https://www.tutorialspoint.com/android
5. http://www.androidhive.info/
6. https://www.codeschool.com/learn/ios

Course Outcomes	**Program Outcomes (POs)**												**Program Specific Outcomes (PSOs)**			
	PO1	**PO 2**	**PO 3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PS O1**	**PSO 2**	**PSO3**	**PSO 4**
CO1	3	2			2							3	3	3		3
CO2	3			2								3	2		1	
CO3	3	2										3		2		
CO4	3		2									3	3		3	3
CO5	3				2							3		2		1
CO6	3		2									3	3		3	3

PREPARED BY **APPROVED BY**

(Mr.Natarajan.C, ASP/CSE) **(HOD/CSE)**

UNIT-I

ANDROID OVERVIEW

Overview of Android platform- Android SDK features - setting up the mobile app development environment along with an emulator- Android Application Development Architecture - creating an applications and activities -Application manifest

Android

Android is a software package and linux based operating system for mobile devices such as tablet computers and smartphones.

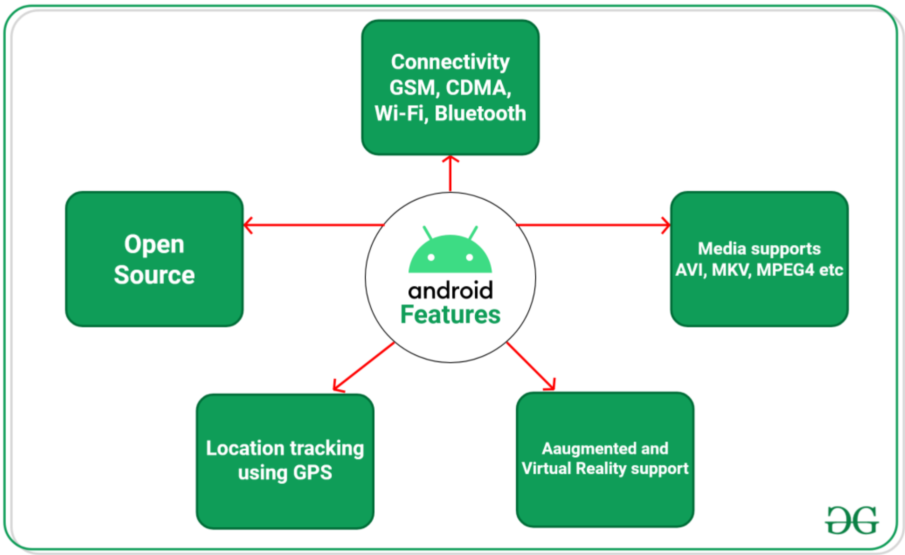
It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

There are many code names of android such as Lollipop, Kitkat, Jelly Bean, Ice cream Sandwich, Froyo, Ecliar, Donut etc which is covered in next page.

Features of Android SDK

Android is a powerful open-source operating system that open-source provides immense features and some of these are listed below.


Connectivity GSM, CDMA, Wi-Fi, Bluetooth
Open Source
android Features
Media supports AVI, MKV, MPEG4 etc
Location tracking using GPS
Aaugmented and Virtual Reality support

- Android Open Source Project so we can customize the OS based on our requirements.
- Android supports different types of connectivity for GSM, CDMA, Wi-Fi, Bluetooth, etc. for telephonic conversation or data transfer.
- Using wifi technology we can pair with other devices while playing games or using other applications.
- It contains multiple APIs to support location-tracking services such as GPS.
- We can manage all data storage-related activities by using the file manager.

- It contains a wide range of media supports like AVI, MKV, FLV, MPEG4, etc. to play or record a variety of audio/video.
- It also supports different image formats like JPEG, PNG, GIF, BMP, MP3, etc.
- It supports multimedia hardware control to perform playback or recording using a camera and microphone.
- Android has an integrated open-source WebKit layout-based web browser to support User Interfaces like HTML5, and CSS3.
- Android supports multi-tasking means we can run multiple applications at a time and can switch between them.
- It provides support for virtual reality or 2D/3D Graphics.

Categories of Android applications

There are many android applications in the market. The top categories are:

- Entertainment
- Tools
- Communication
- Productivity
- Personalization
- Music and Audio
- Social
- Media and Video
- Travel and Local etc.

Programming Languages used in Developing Android Applications

1. **Java**
2. **Kotlin**

Developing the Android Application using Kotlin is preferred by Google, as Kotlin is made an official language for Android Development, which is developed and maintained by JetBrains. Previously before Java is considered the official language for Android Development. Kotlin is made official for Android Development in Google I/O 2017.

Advantages of Android Development

- The Android is an open-source Operating system and hence possesses a vast community for support.
- The design of the Android Application has guidelines from Google, which becomes easier for developers to produce more intuitive user applications.
- Fragmentation gives more power to Android Applications. This means the application can run two activities on a single screen.
- Releasing the Android application in the Google play store is easier when it is compared to other platforms.

Disadvantages of Android Development

- Fragmentation provides a very intuitive approach to user experience but it has some drawbacks, where the development team needs time to adjust to the various screen sizes of mobile smartphones that are now available in the market and invoke the particular features in the application.
- The Android devices might vary broadly. So the testing of the application becomes more difficult.

- As the development and testing consume more time, the cost of the application may increase, depending on the application's complexity and features.

Android Architecture

Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services.

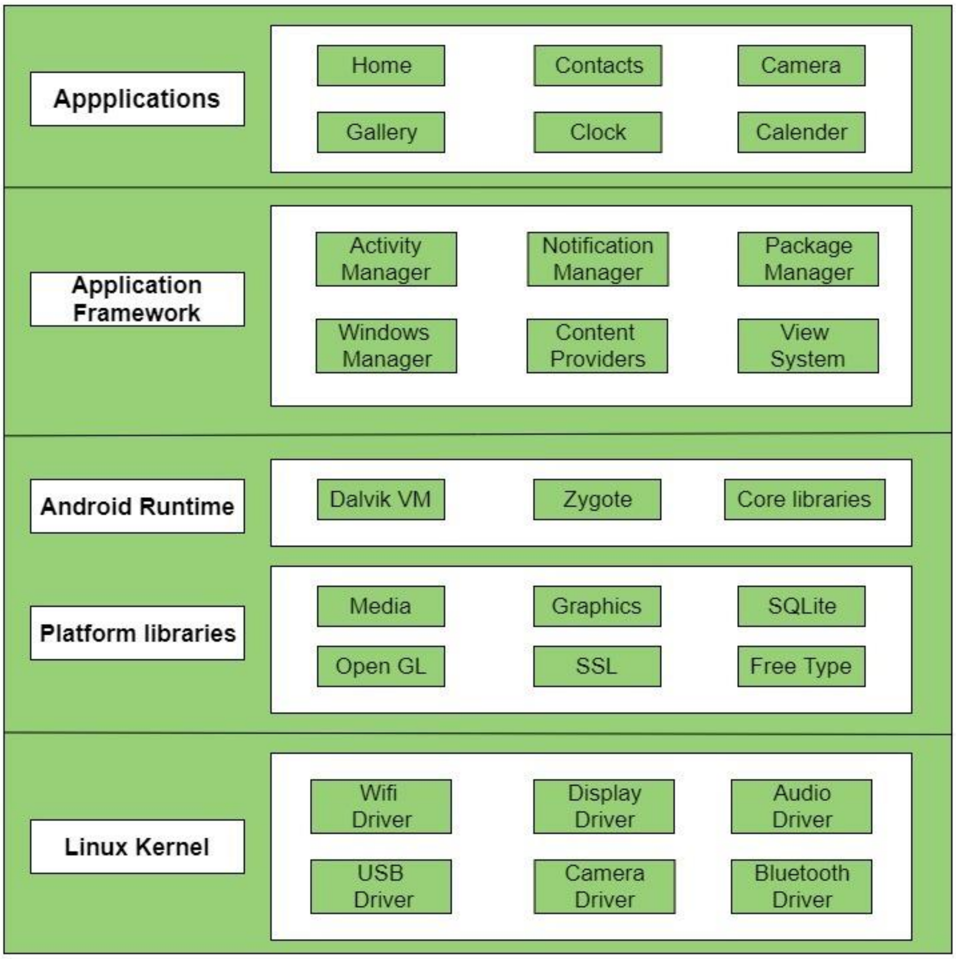
Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

The main components of android architecture are following:-

- Applications
- Application Framework
- Android Runtime
- Platform Libraries
- Linux Kernel

android architecture

Pictorial representation of android architecture with several main components and their sub components –



Applications –

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only. It runs within the Android run time with the help of the classes and services provided by the application framework.

Application framework –

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

Activity Manger:

gives information about, and interacts with, activities, services, and the containing process.

Windows Manager:

The interface that apps use to talk to the window manager. Each window manager instance is bound to a Display.

Notification Manager

Android allows to put notification into the titlebar of your application. The user can expand the notification bar and by selecting the notification the user can trigger another activity.

Package Manager

Package Manager is a highly powerful application to manage apps, both system and user, installed on an android device.

Content Provider

Content Provider is a part of an android application and it will act like more like a relational database to store the app data. We can perform multiple operations like insert, update, delete and edit on the data stored in content provider using **insert()**, **update()**, **delete()** and **query()** methods.

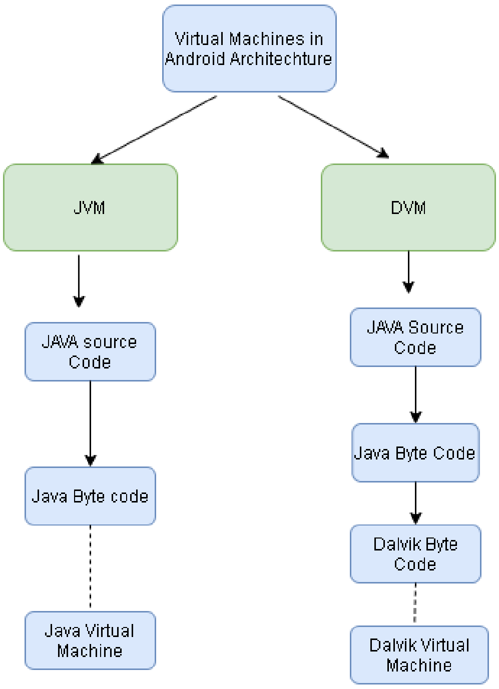
View sytems:

View system's is a basic building block of UI (User Interface) in android. A view is a small rectangular box that responds to user inputs. Eg: EditText, Button, CheckBox, etc.

Application runtime –

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), Dalvik Virtual Machine (DVM) is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.



Platform libraries –

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

- Media library provides support to play and record an audio and video formats.
- Surface manager responsible for managing access to the display subsystem.
- SGL and OpenGL both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- SQLite provides database support and FreeType provides font support.
- Web-Kit This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- SSL (Secure Sockets Layer) is security technology to establish an encrypted link between a web server and a web browser.

Linux Kernel –

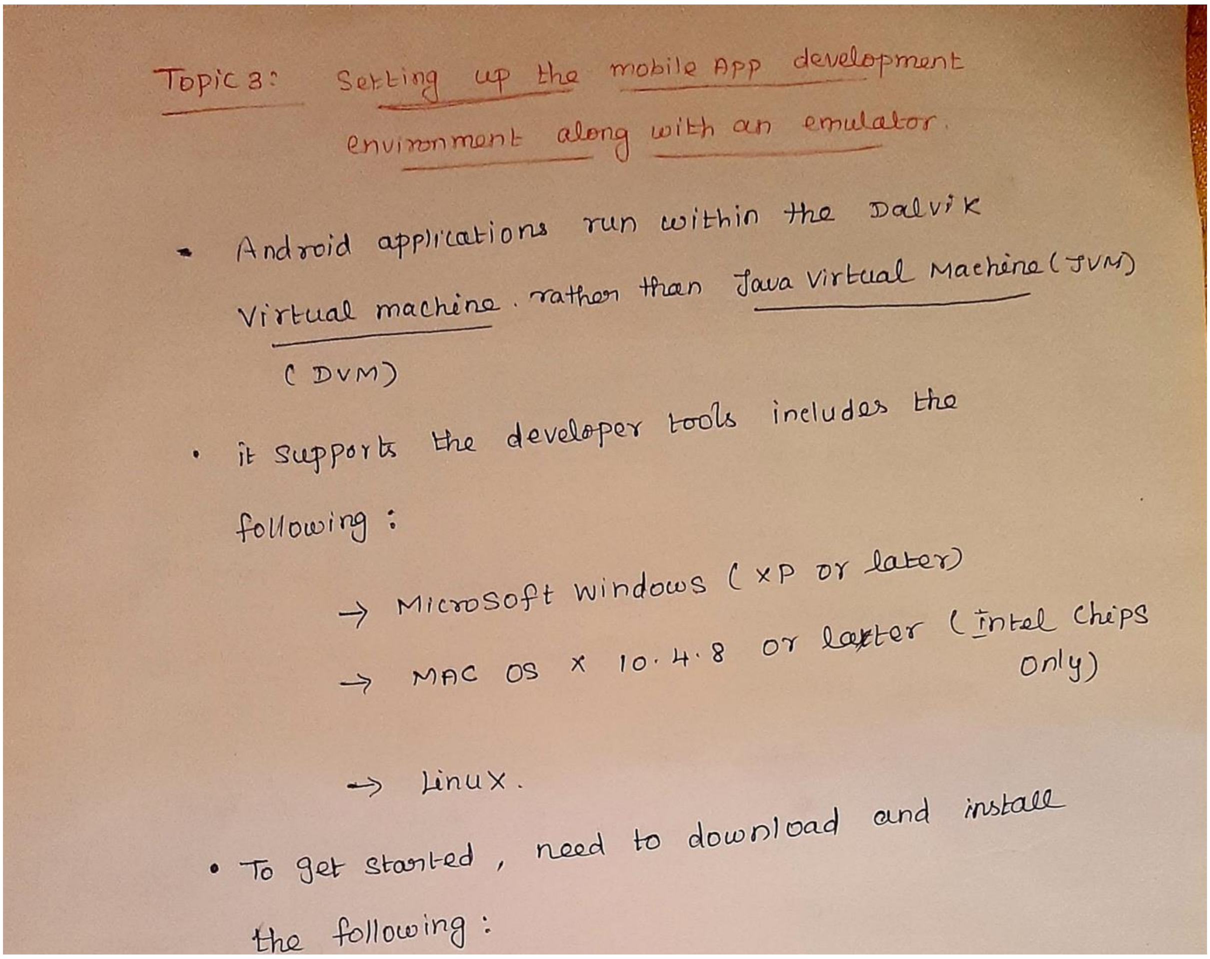
Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

The features of Linux kernel are:

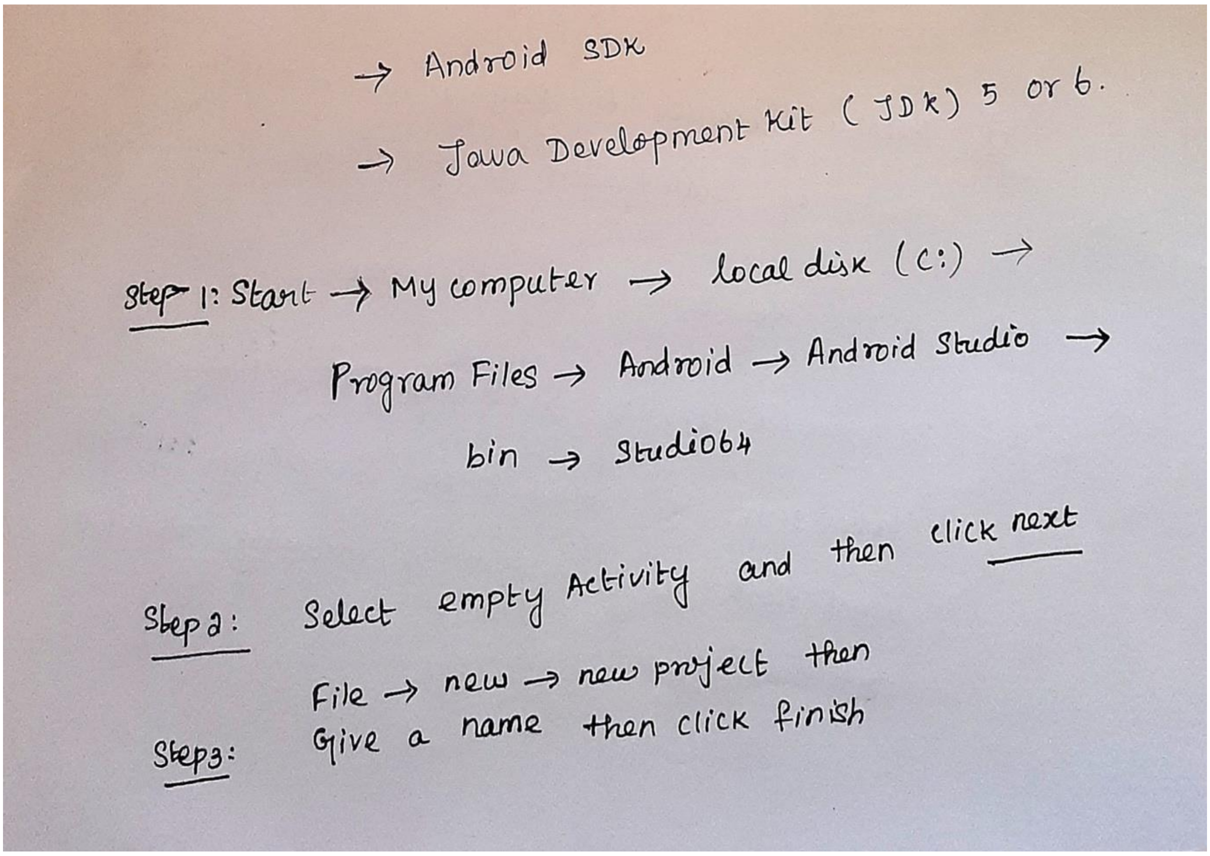
- Security: The Linux kernel handles the security between the application and the system.
- Memory Management: It efficiently handles the memory management thereby providing the freedom to develop our apps.
- Process Management: It manages the process well, allocates resources to processes whenever they need them.
- Network Stack: It effectively handles the network communication.
- Driver Model: It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

Topic 3:



Topic 3: Setting up the mobile App development environment along with an emulator.

- Android applications run within the Dalvik Virtual machine . rather than Java Virtual Machine (JVM) (DVM)
- it supports the developer tools includes the following :
  → Microsoft windows (XP or later)
  → MAC OS X 10.4.8 or latter (intel Chips only)
  → Linux.
- To get started , need to download and install the following :



→ Android SDK

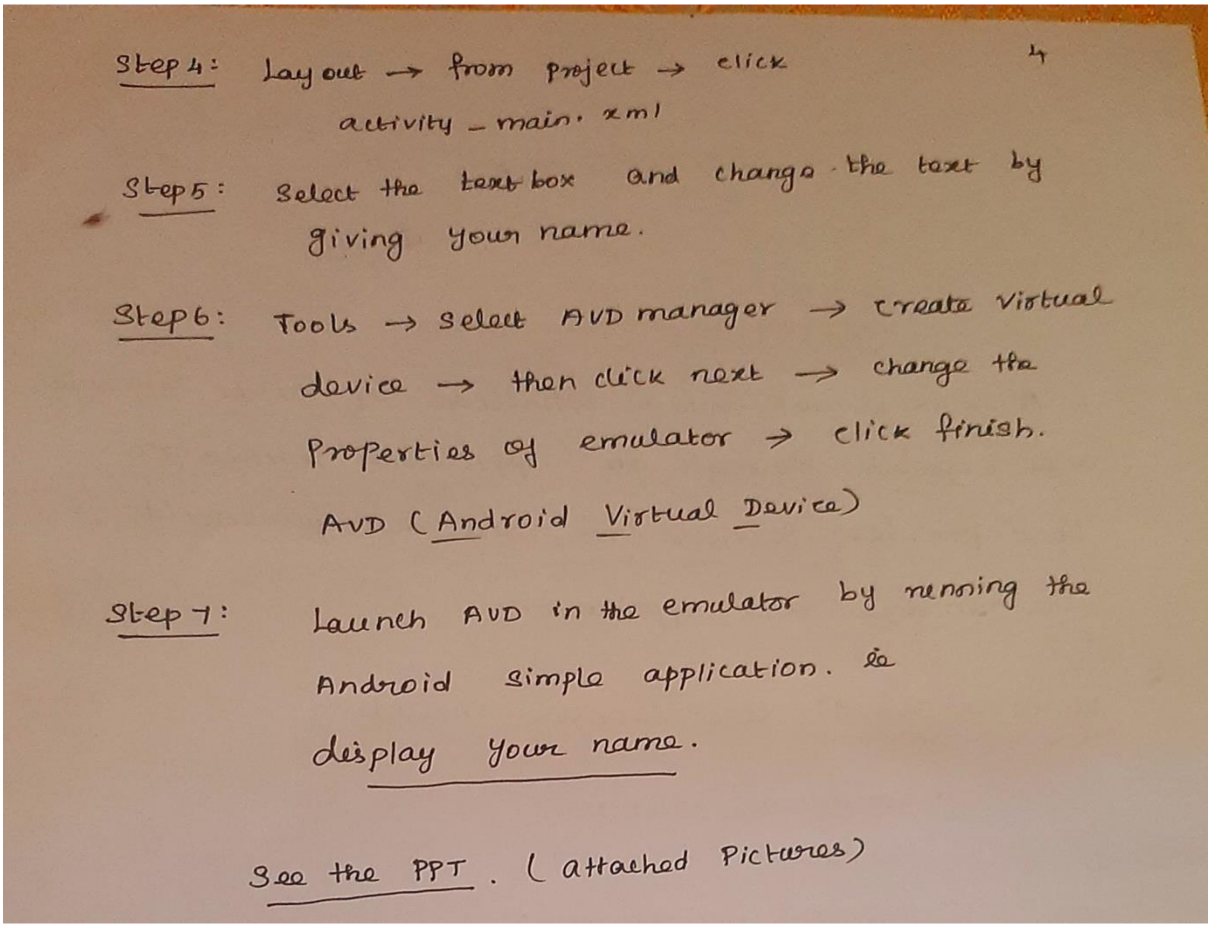
→ Jawa Development Kit (JDK) 5 or 6.

Step 1: Start → My computer → local disk (C:) → Program Files → Android → Android Studio → bin → Studio64

Step 2: Select empty Activity and then click next

File → new → new project then

Step3: Give a name then click finish



4

Step 4: Lay out → from project → click activity _ main. xml

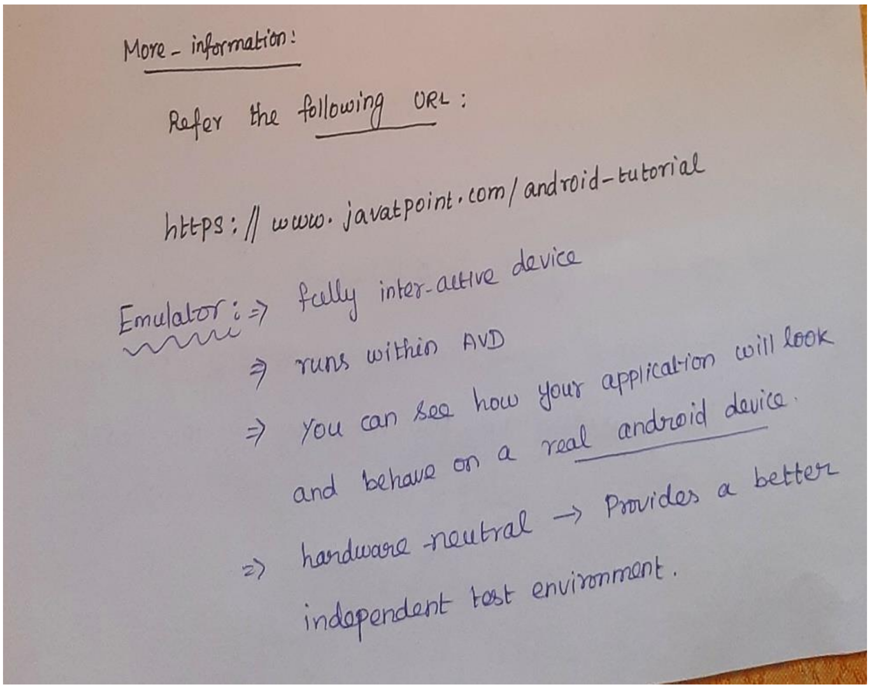
Step5: Select the text box and change the text by giving your name.

Step6: Tools → Select AVD manager → Create Virtual device → then click next → change the Properties of emulator → click finish.

AVD (Android Virtual Device)

Step 7: Launch AVD in the emulator by renning the Android simple application. ie display your name.

See the PPT. (attached Pictures)



More - information:

Refer the following URL :

https://www.javatpoint.com/android-tutorial

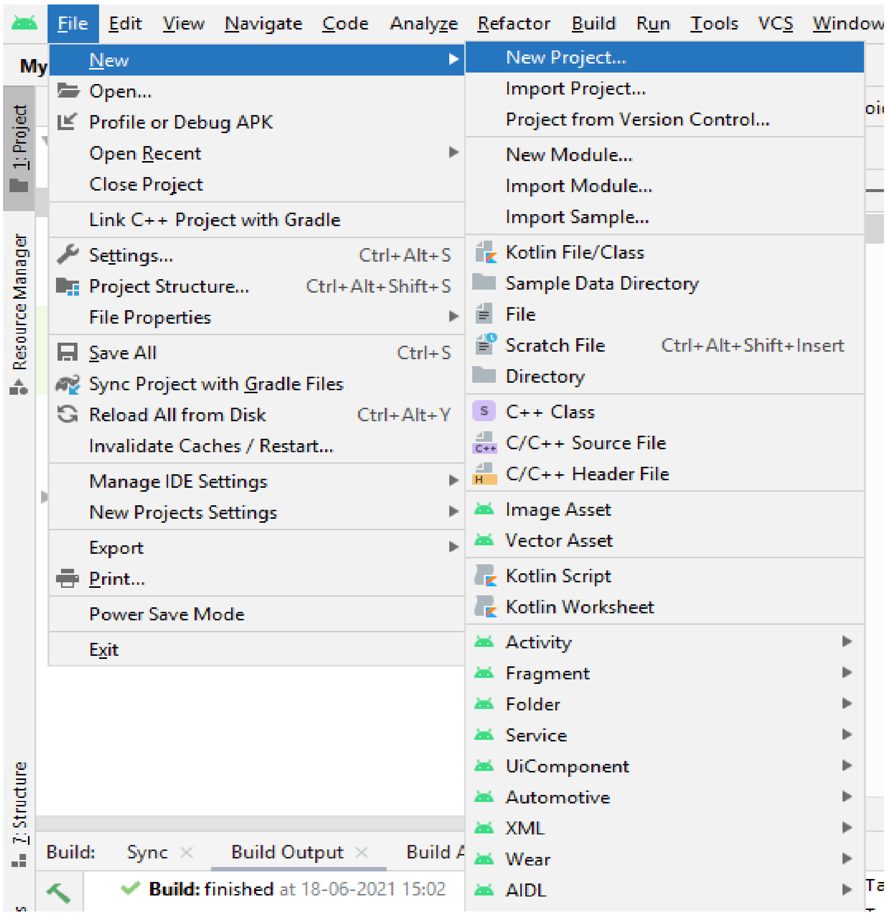
Emulator :=> fully inter-active device

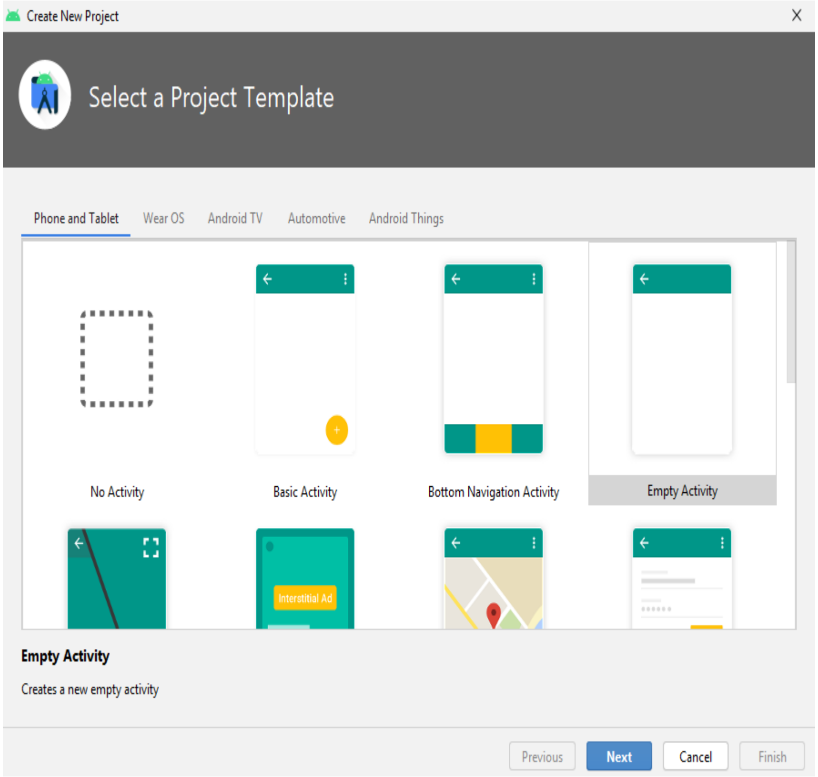
=> runs within AVD

=> You can see how your application will look and behave on a real android device.

=> hardware-neutral -> Provides a better independent test environment.

- ➢ Start-> My computer -> local disk(c:) -> Program Files ->Android -> AndroidStudio -> bin -> studio64

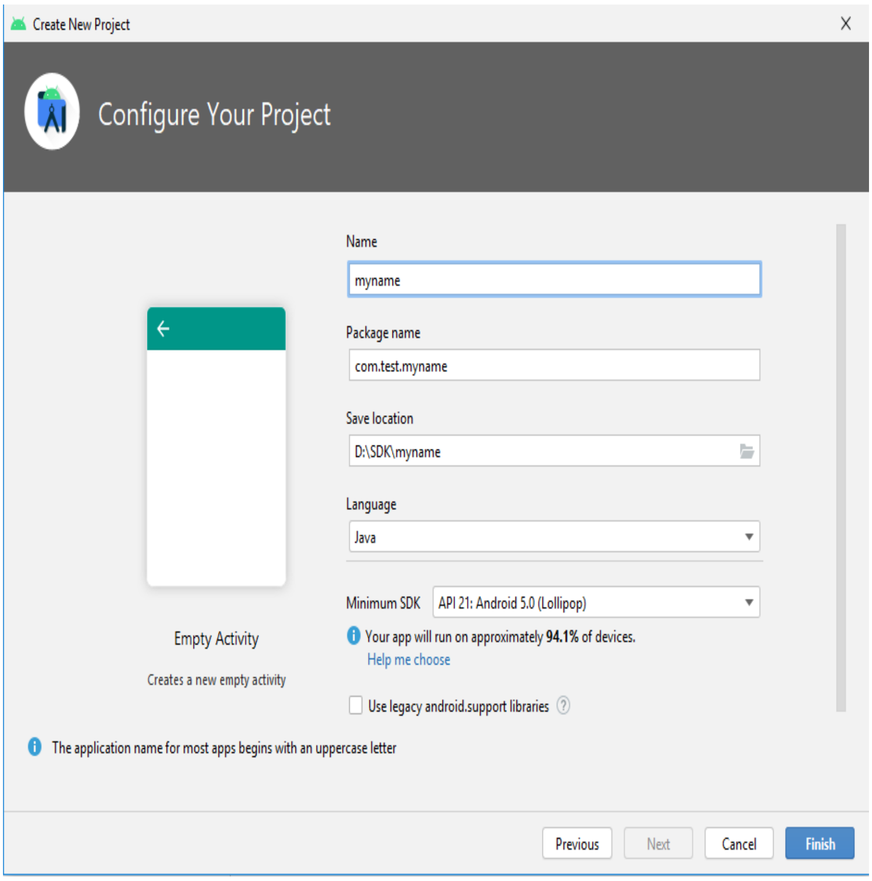




Select a Project Template

Empty Activity

Creates a new empty activity



Configure Your Project

Name: myname

Package name: com.test.myname

Save location: D:\SDK\myname

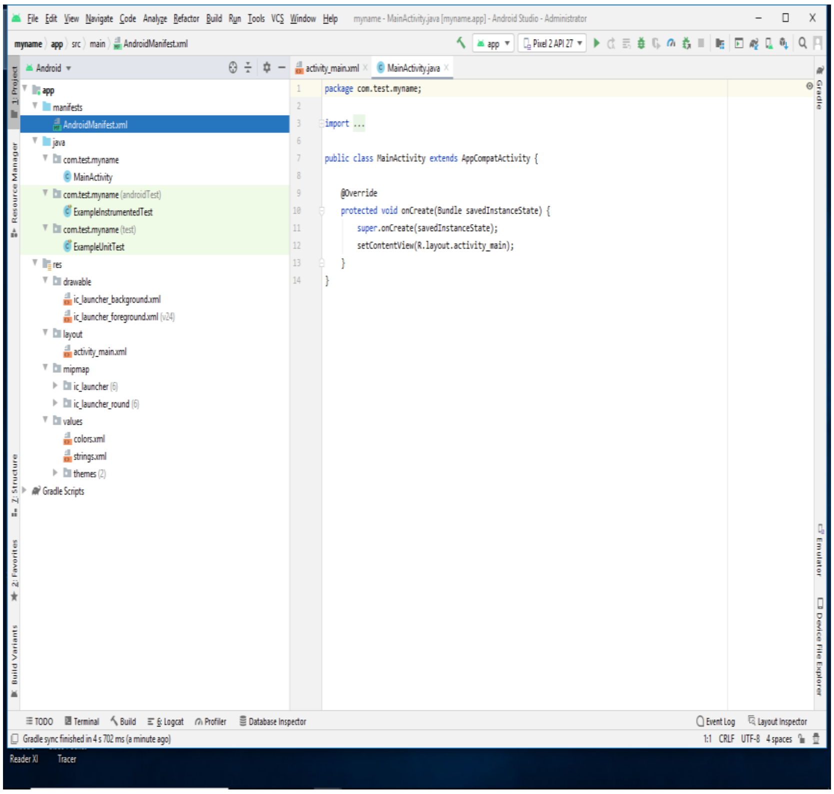
Language: Java

Minimum SDK: API 21: Android 5.0 (Lollipop)

Your app will run on approximately **94.1%** of devices.

□ Use legacy android.support libraries

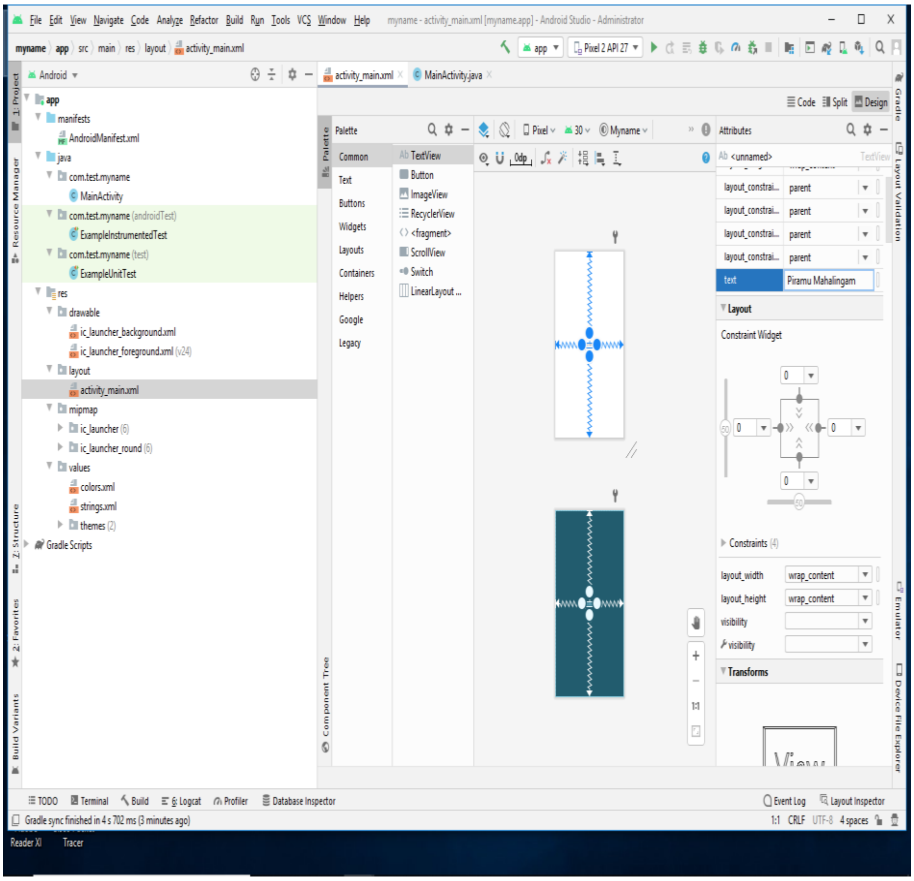
The application name for most apps begins with an uppercase letter



myname - MainActivity.java [myname.app] - Android Studio - Administrator

```
package com.test.myname;

import ...

public class MainActivity extends AppCompatActivity {

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

Gradle sync finished in 4 s 702 ms (a minute ago)

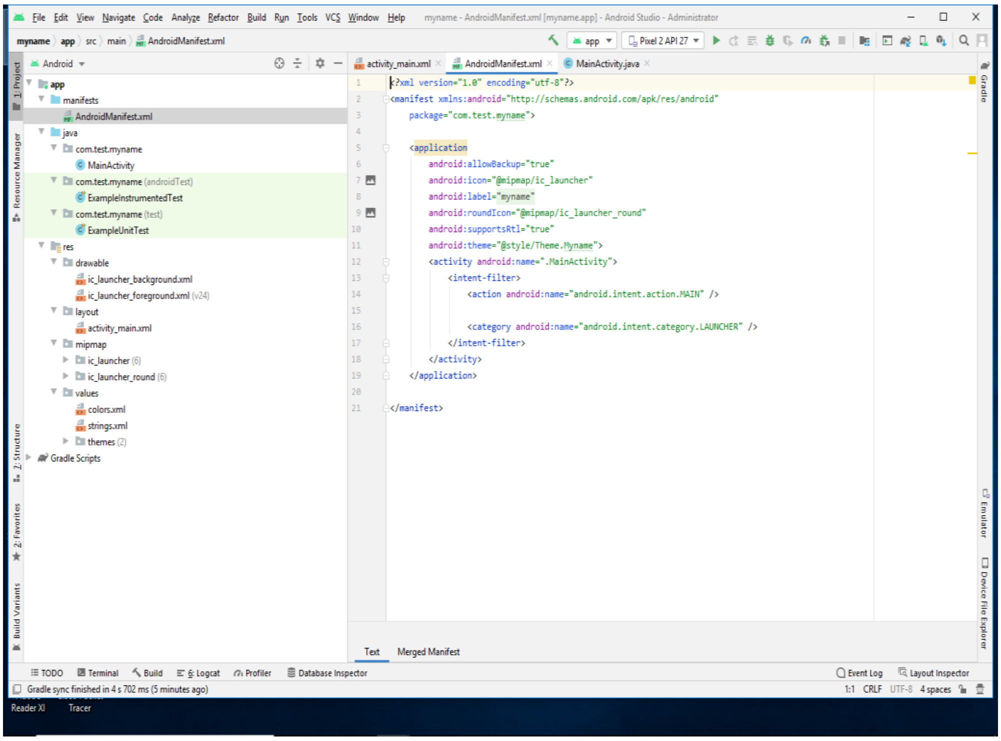


myname - activity_main.xml [myname.app] - Android Studio - Administrator

Attributes <unnamed> TextView

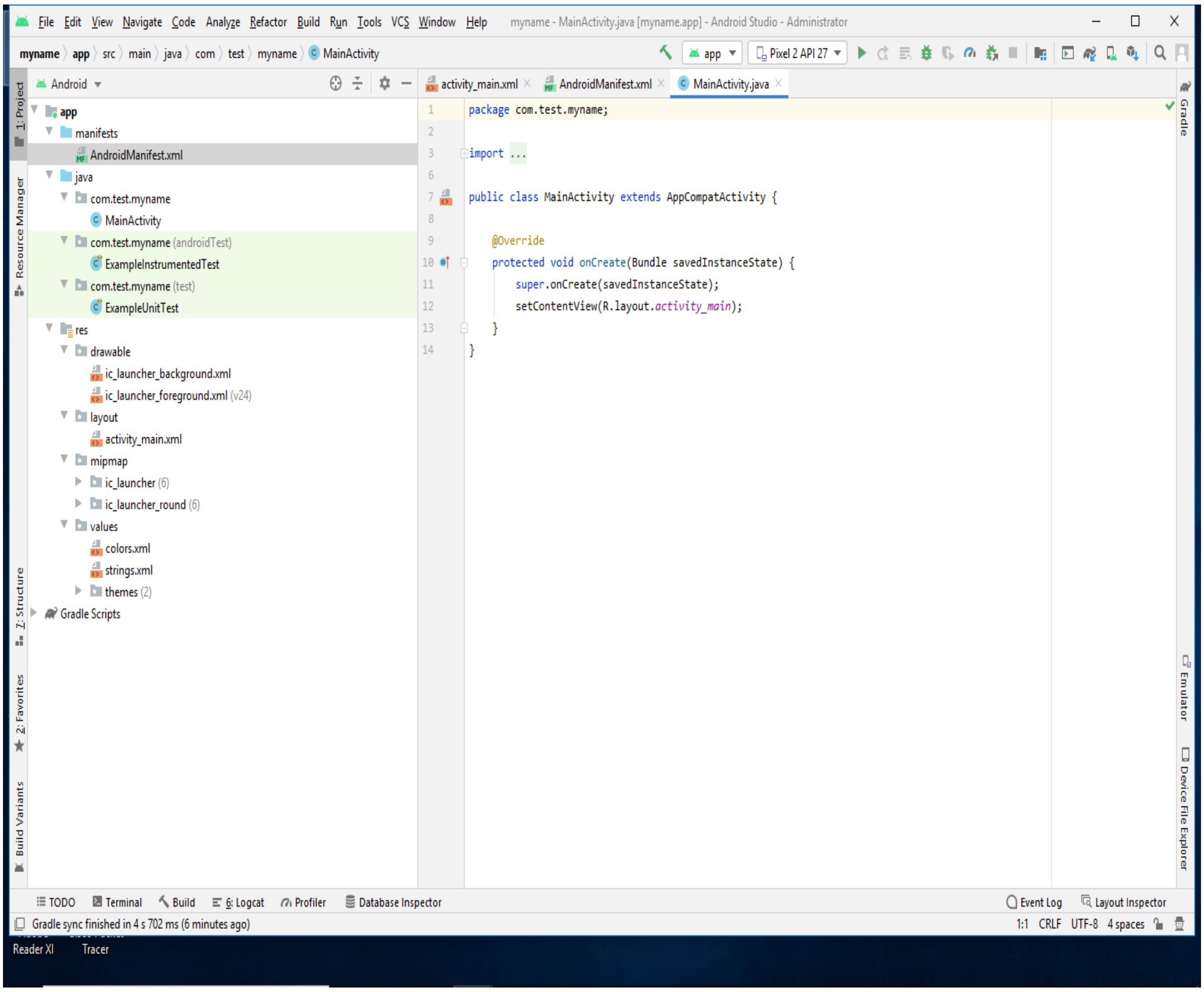
layout_constrai...	parent
layout_constrai...	parent
layout_constrai...	parent
layout_constrai...	parent
text	Piramu Mahalingam
layout_width	wrap_content
layout_height	wrap_content
visibility	

Gradle sync finished in 4 s 702 ms (3 minutes ago)



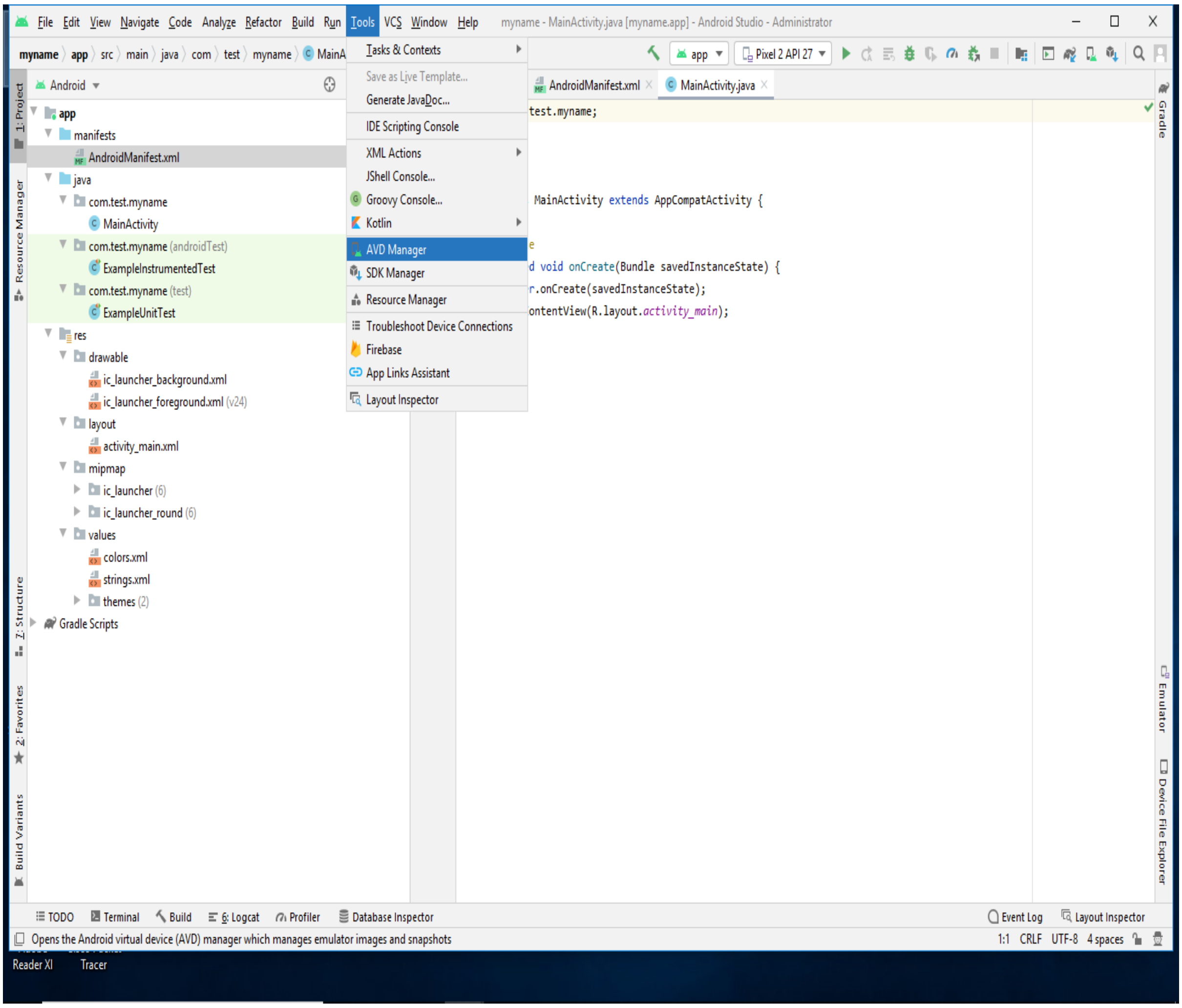
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.myname">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="myname"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Myname">
        <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>
```

Gradle sync finished in 4 s 702 ms (5 minutes ago)



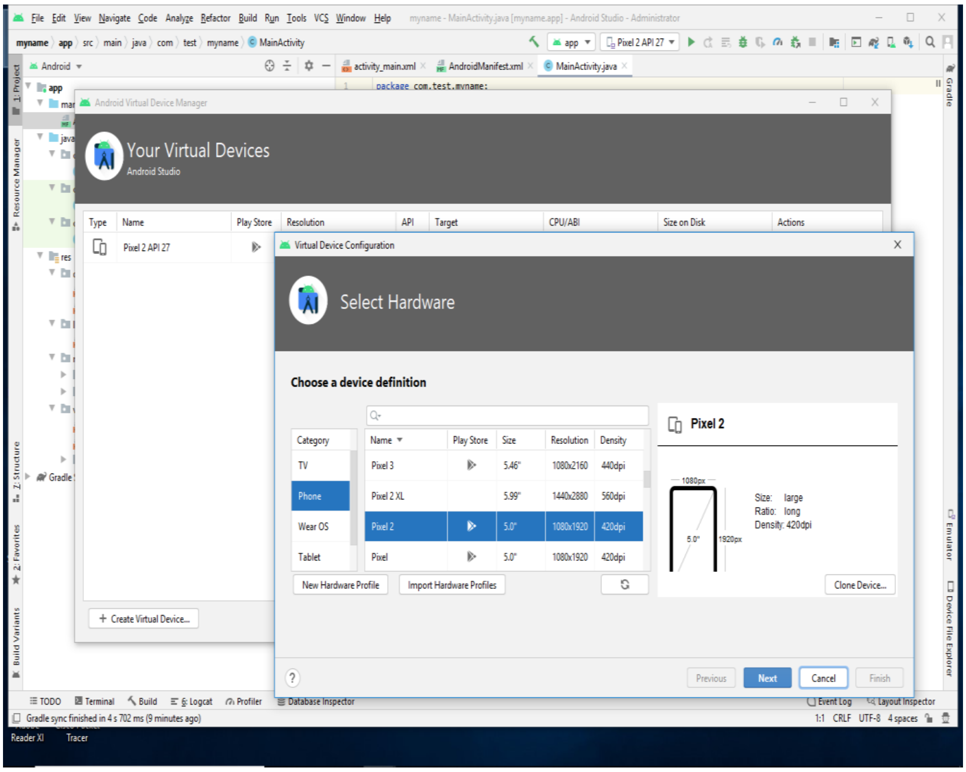
```
package com.test.myname;

import ...

public class MainActivity extends AppCompatActivity {

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

Gradle sync finished in 4 s 702 ms (6 minutes ago)



```
test.myname;

 MainActivity extends AppCompatActivity {

e
d void onCreate(Bundle savedInstanceState) {
r.onCreate(savedInstanceState);
ontentView(R.layout.activity_main);
```

Opens the Android virtual device (AVD) manager which manages emulator images and snapshots



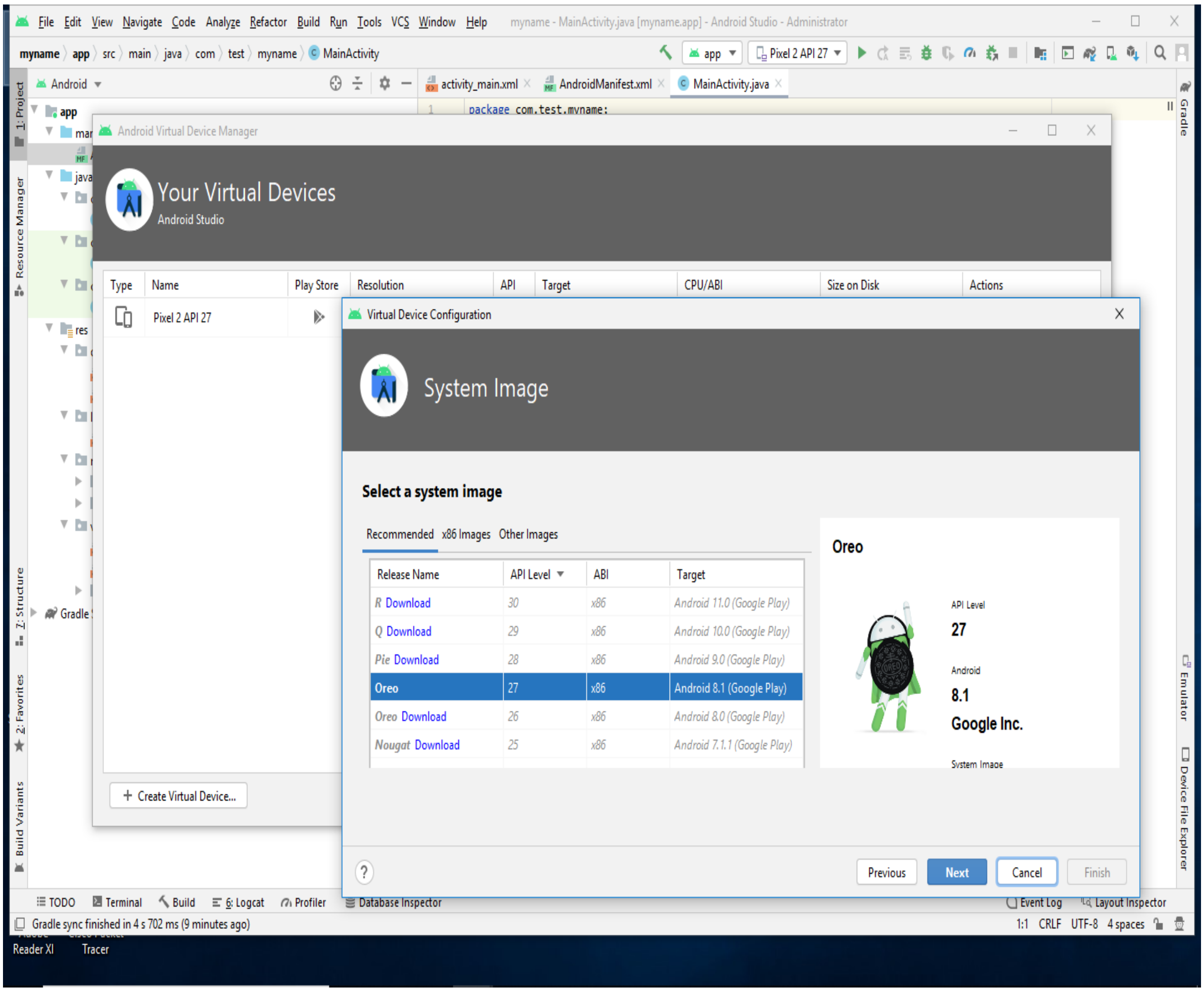
```
package com.test.myname;
```

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 2 API 27	▶						

Category
TV
Phone
Wear OS
Tablet

Name	Play Store	Size	Resolution	Density
Pixel 3	▶	5.46"	1080x2160	440dpi
Pixel 2 XL		5.99"	1440x2880	560dpi
Pixel 2	▶	5.0"	1080x1920	420dpi
Pixel	▶	5.0"	1080x1920	420dpi

Gradle sync finished in 4 s 702 ms (9 minutes ago)



Select a system image

Release Name	API Level	ABI	Target
R Download	30	x86	Android 11.0 (Google Play)
Q Download	29	x86	Android 10.0 (Google Play)
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)

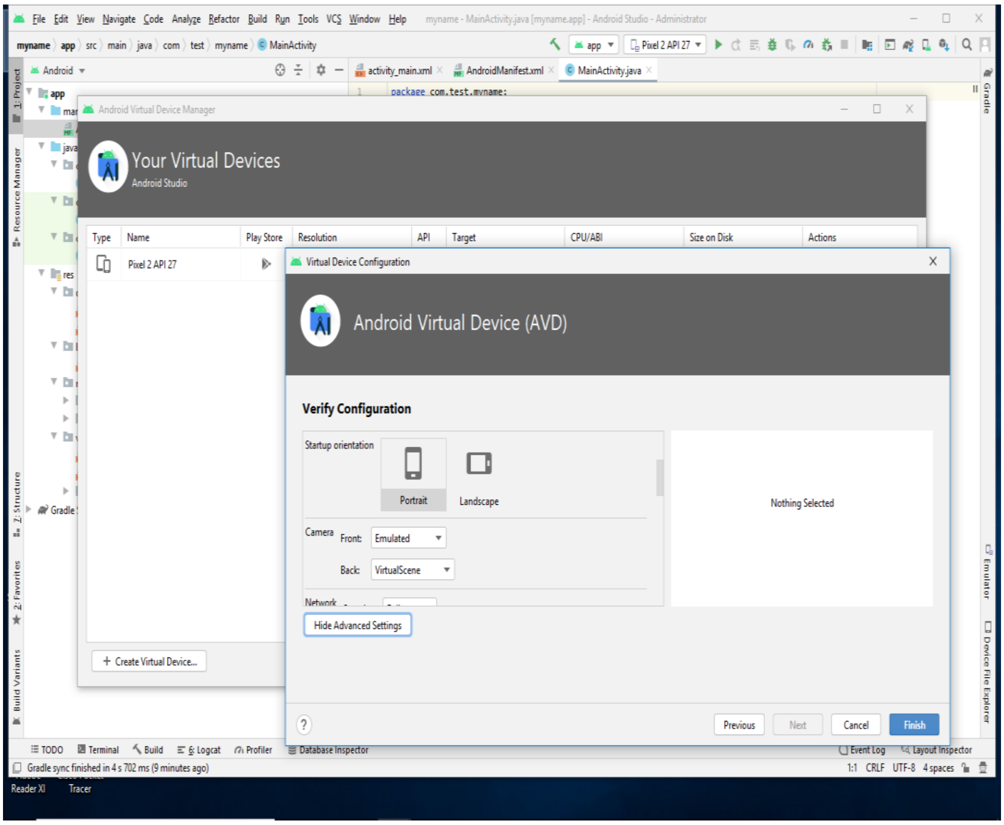
Oreo

API Level
27

Android
8.1

Google Inc.

System Image


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
myname - MainActivity.java [myname.app] - Android Studio - Administrator
Android Virtual Device Manager
Your Virtual Devices
Android Studio
Type
Name
Play Store
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Pixel 2 API 27
+ Create Virtual Device...
Virtual Device Configuration
Android Virtual Device (AVD)
Hide Advanced Settings
Previous
Next
Cancel
Finish
Gradle sync finished in 4 s 702 ms (9 minutes ago)

Verify Configuration

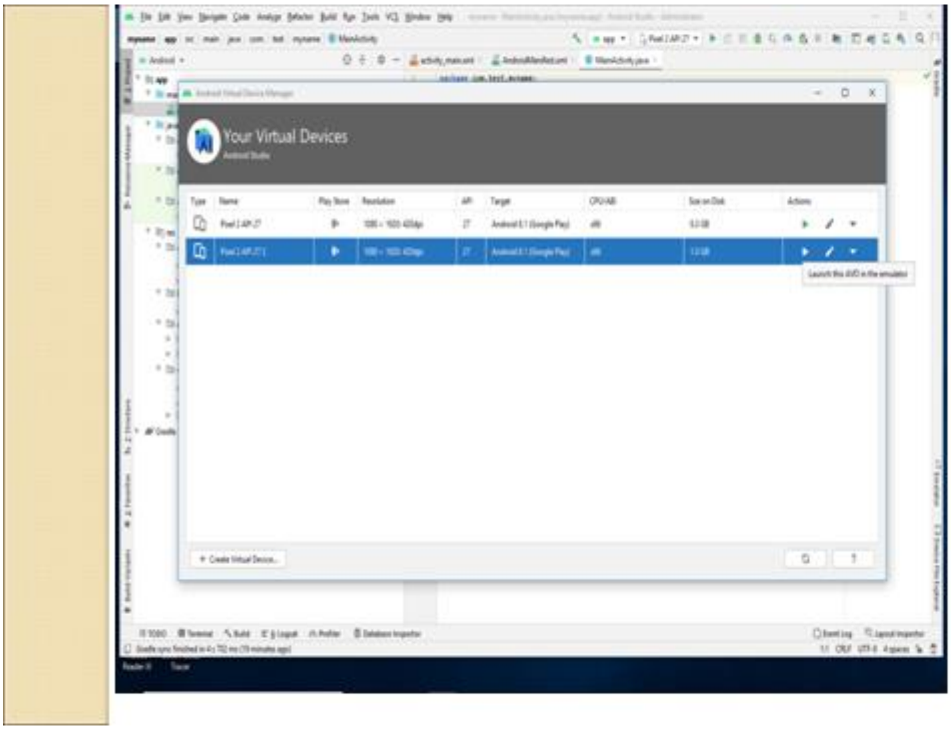
Startup orientation: Portrait | Landscape

Camera Front: Emulated

Back: VirtualScene

Network

Nothing Selected



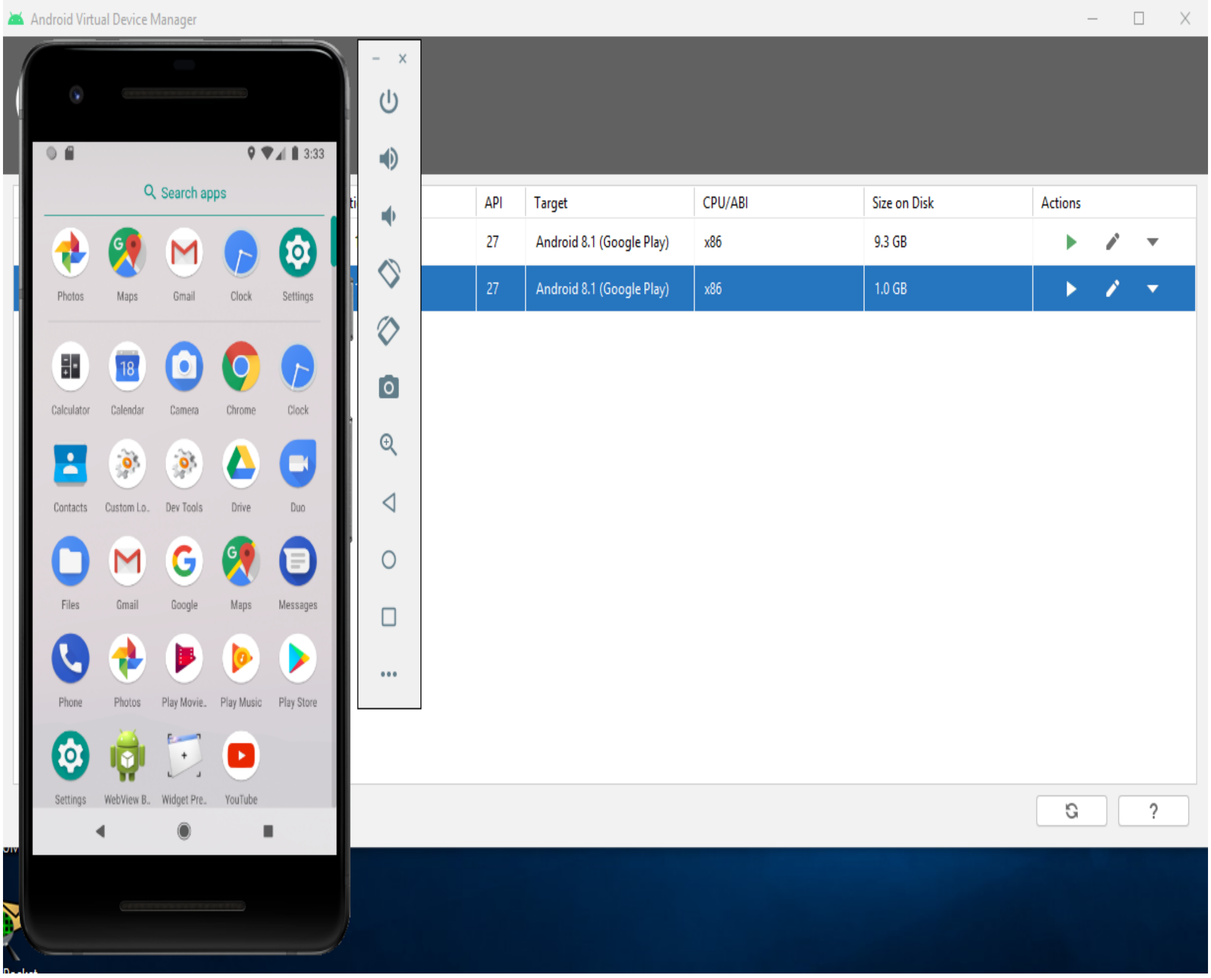
Your Virtual Devices

Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 2 API 27		1080 × 1920: 420dpi	27	Android 8.1 (Google Play)	x86	9.3 GB	
	Pixel 2 API 27 1		1080 × 1920: 420dpi	27	Android 8.1 (Google Play)	x86	1.0 GB	

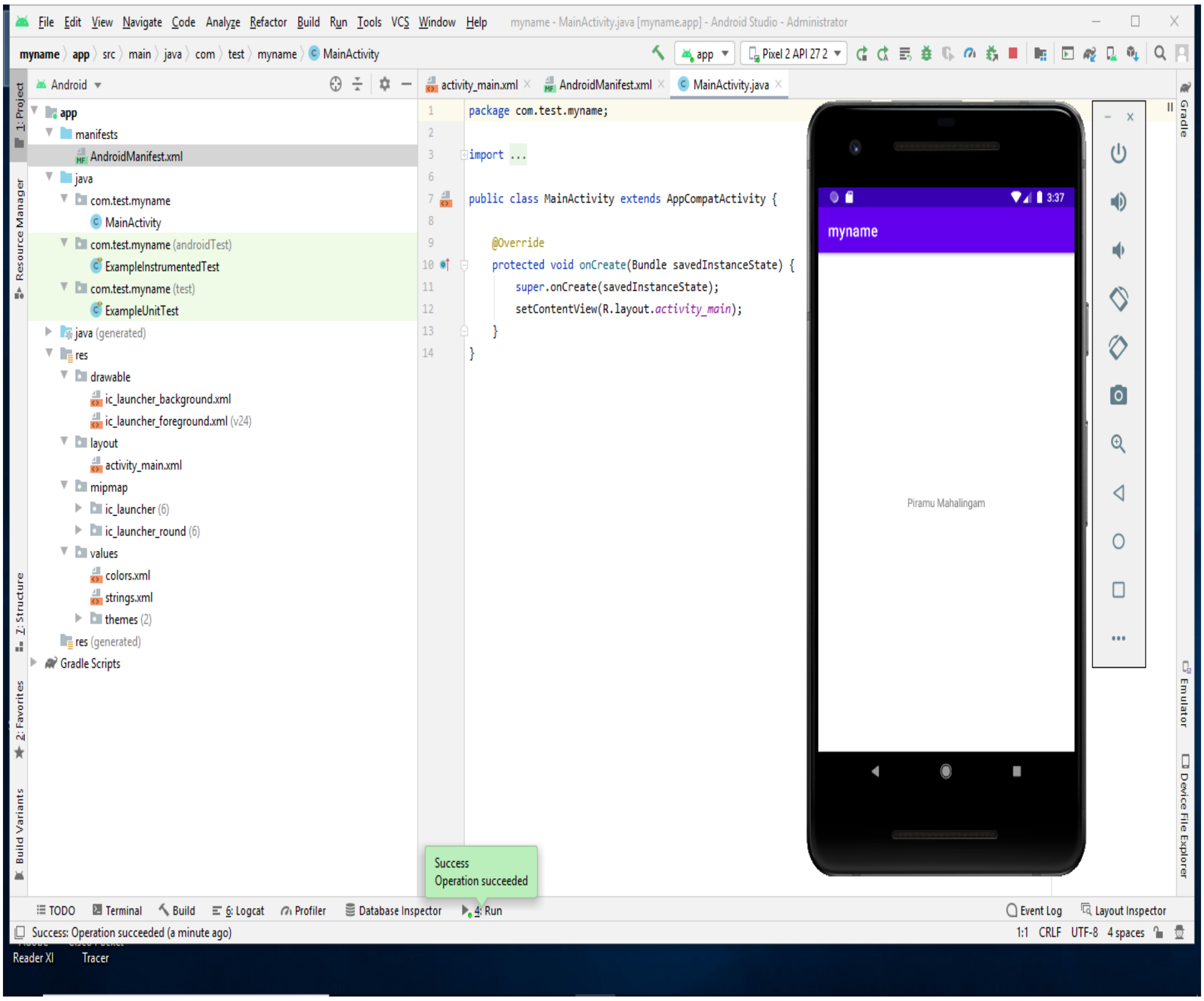
Launch this AVD in the emulator

+ Create Virtual Device...



Android Virtual Device Manager

API	Target	CPU/ABI	Size on Disk	Actions
27	Android 8.1 (Google Play)	x86	9.3 GB	
27	Android 8.1 (Google Play)	x86	1.0 GB	



```
package com.test.myname;

import ...

public class MainActivity extends AppCompatActivity {

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

Success
Operation succeeded

Success: Operation succeeded (a minute ago)

Dalvik Virtual Machine | DVM

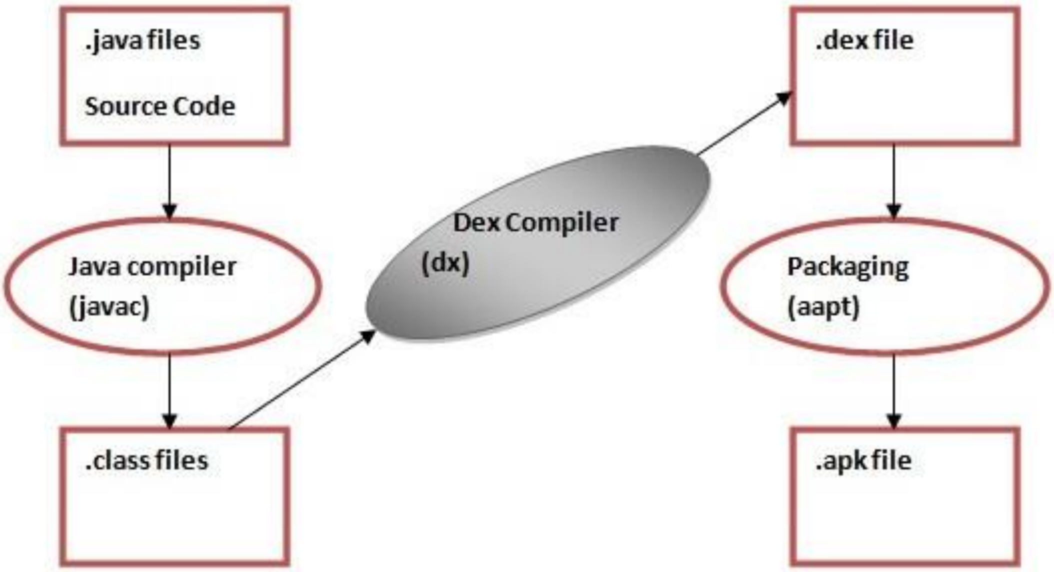
As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.

The Dalvik Virtual Machine (DVM) is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for *memory*, *battery life* and *performance*.

Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.

The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.

Let's see the compiling and packaging process from the source file:



The javac tool compiles the java source file into the class file.

The dx tool takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.

The Android Assets Packaging Tool (aapt) handles the packaging process.

S.NO	DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
1	It is Register based which is designed to run on low memory.	It is Stack based.
2	DVM uses its own byte code and runs the “.Dex” file. From Android 2.2 SDK Dalvik has got a Just in Time compiler	JVM uses java byte code and runs “.class” file having JIT (Just In Time).
3	DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	A single instance of JVM is shared with multiple applications.
4	DVM supports the Android operating system only.	JVM supports multiple operating systems.
5	For DVM very few Re-tools are available	For JVM many Re-tools are available.
6	There is a constant pool for every	It has a constant pool for

S.NO	DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
	application.	every class.
7	Here the executable is APK.	Here the executable is JAR.

CREATING ON APPLICATION AND ACTIVITIES

Android application:

It consists of loosely coupled components

Bound by an application manifest-that describes each component and how they all interact, as well as application metadata including its hardware and platform requirements

The following six components provides the building block for your applications

- Activities
- Services
- Content Providers
- Intents
- Broadcast receivers
- Widgets

Activity

Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

An activity is the single screen in android. It is like window or frame of Java.By the help of activity, you can place all your UI components or widgets in a single screen.The 7 lifecycle method of Activity describes how activity will behave at different states.Android Activity Lifecycle methods

Let's see the 7 lifecycle methods of android activity.

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.

onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="example.javatpoint.com.activitylifecycle.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Android Activity Lifecycle Example

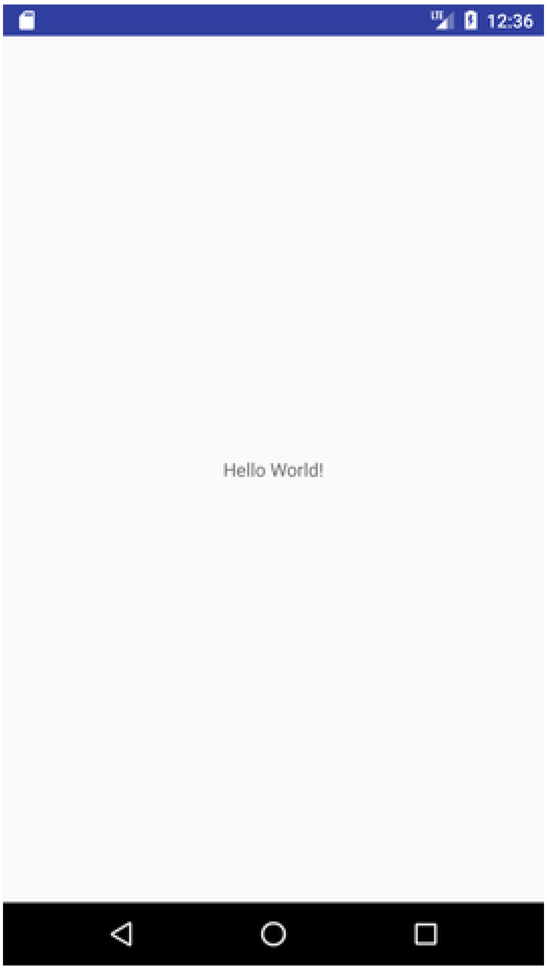
It provides the details about the invocation of life cycle methods of activity. In this example, we are displaying the content on the logcat.

File: MainActivity.java

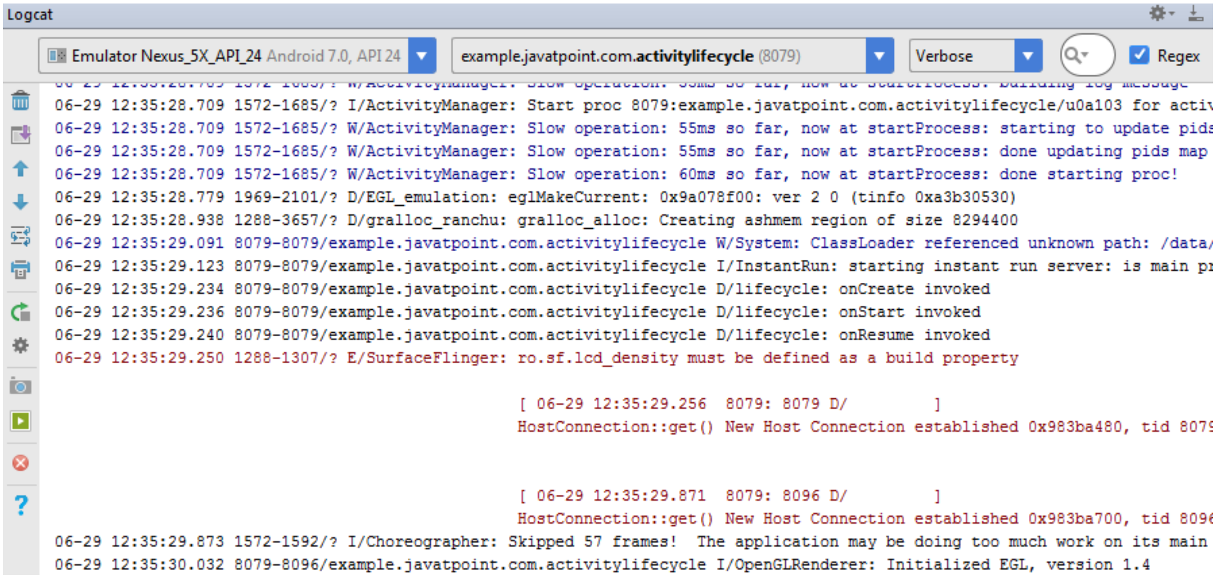
```
package example.javatpoint.com.activitylifecycle;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("lifecycle","onCreate invoked");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d("lifecycle","onStart invoked");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d("lifecycle","onResume invoked");
    }
    @Override
    protected void onPause() {
        super.onPause();
        Log.d("lifecycle","onPause invoked");
    }
    @Override
    protected void onStop() {
        super.onStop();
        Log.d("lifecycle","onStop invoked");
    }
    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d("lifecycle","onRestart invoked");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("lifecycle","onDestroy invoked");
    }
}
```

Output:

You will not see any output on the emulator or device. You need to open logcat.

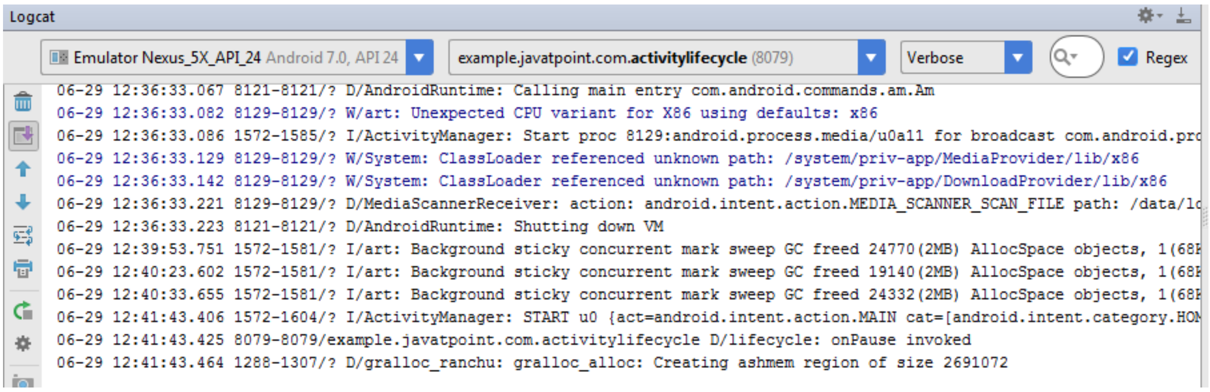


Now see on the logcat: onCreate, onStart and onResume methods are invoked.


Logcat
Emulator Nexus_5X_API_24 Android 7.0, API 24
example.javatpoint.com.activitylifecycle (8079)
Verbose
☑ Regex

```
06-29 12:35:28.709 1572-1685/? I/ActivityManager: Start proc 8079:example.javatpoint.com.activitylifecycle/u0a103 for acti
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 55ms so far, now at startProcess: starting to update pids
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 55ms so far, now at startProcess: done updating pids map
06-29 12:35:28.709 1572-1685/? W/ActivityManager: Slow operation: 60ms so far, now at startProcess: done starting proc!
06-29 12:35:28.779 1969-2101/? D/EGL_emulation: eglMakeCurrent: 0x9a078f00: ver 2 0 (tinfo 0xa3b30530)
06-29 12:35:28.938 1288-3657/? D/gralloc_ranchu: gralloc_alloc: Creating ashmem region of size 8294400
06-29 12:35:29.091 8079-8079/example.javatpoint.com.activitylifecycle W/System: ClassLoader referenced unknown path: /data/
06-29 12:35:29.123 8079-8079/example.javatpoint.com.activitylifecycle I/InstantRun: starting instant run server: is main pr
06-29 12:35:29.234 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onCreate invoked
06-29 12:35:29.236 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onStart invoked
06-29 12:35:29.240 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onResume invoked
06-29 12:35:29.250 1288-1307/? E/SurfaceFlinger: ro.sf.lcd_density must be defined as a build property

                                                 [ 06-29 12:35:29.256  8079: 8079 D/         ]
                                                 HostConnection::get() New Host Connection established 0x983ba480, tid 807


                                                 [ 06-29 12:35:29.871  8079: 8096 D/         ]
                                                 HostConnection::get() New Host Connection established 0x983ba700, tid 809
06-29 12:35:29.873 1572-1592/? I/Choreographer: Skipped 57 frames!  The application may be doing too much work on its main
06-29 12:35:30.032 8079-8096/example.javatpoint.com.activitylifecycle I/OpenGLRenderer: Initialized EGL, version 1.4
```

Now click on the HOME Button. You will see onPause method is invoked.



Logcat
Emulator Nexus_5X_API_24 Android 7.0, API 24
example.javatpoint.com.activitylifecycle (8079)
Verbose
Regex

```
06-29 12:36:33.067 8121-8121/? D/AndroidRuntime: Calling main entry com.android.commands.am.Am
06-29 12:36:33.082 8129-8129/? W/art: Unexpected CPU variant for X86 using defaults: x86
06-29 12:36:33.086 1572-1585/? I/ActivityManager: Start proc 8129:android.process.media/u0a11 for broadcast com.android.pr
06-29 12:36:33.129 8129-8129/? W/System: ClassLoader referenced unknown path: /system/priv-app/MediaProvider/lib/x86
06-29 12:36:33.142 8129-8129/? W/System: ClassLoader referenced unknown path: /system/priv-app/DownloadProvider/lib/x86
06-29 12:36:33.221 8129-8129/? D/MediaScannerReceiver: action: android.intent.action.MEDIA_SCANNER_SCAN_FILE path: /data/l
06-29 12:36:33.223 8121-8121/? D/AndroidRuntime: Shutting down VM
06-29 12:39:53.751 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 24770(2MB) AllocSpace objects, 1(68
06-29 12:40:23.602 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 19140(2MB) AllocSpace objects, 1(68
06-29 12:40:33.655 1572-1581/? I/art: Background sticky concurrent mark sweep GC freed 24332(2MB) AllocSpace objects, 1(68
06-29 12:41:43.406 1572-1604/? I/ActivityManager: START u0 {act=android.intent.action.MAIN cat=[android.intent.category.HO
06-29 12:41:43.425 8079-8079/example.javatpoint.com.activitylifecycle D/lifecycle: onPause invoked
06-29 12:41:43.464 1288-1307/? D/gralloc_ranchu: gralloc_alloc: Creating ashmem region of size 2691072
```

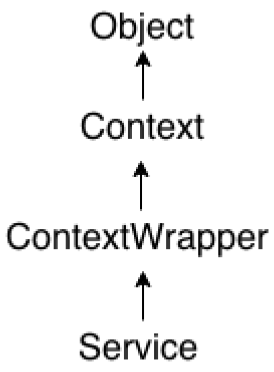
After a while, you will see onStop method is invoked.

Android service is a component that is *used to perform operations on the background* such as playing music, handle network transactions, interacting content providers etc. It doesn't has any UI (user interface).

The service runs in the background indefinitely even if application is destroyed.

Moreover, service can be bounded by a component to perform interactivity and inter process communication (IPC).

The android.app.Service is subclass of ContextWrapper class.



Life Cycle of Android Service

There can be two forms of a service.The lifecycle of service can follow two different paths: started or bound.

1. Started
2. Bound

1) Started Service

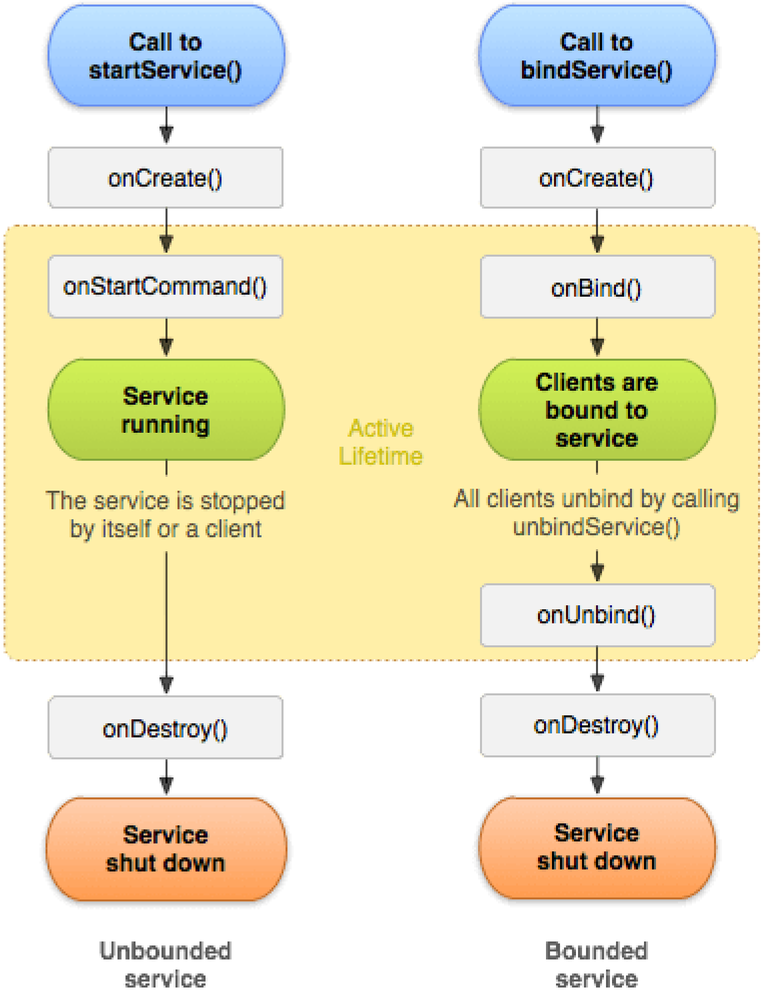
A service is started when component (like activity) calls startService() method, now it runs in the background indefinitely. It is stopped by stopService() method. The service can stop itself by calling the stopSelf() method.

2) Bound Service

A service is bound when another component (e.g. client) calls bindService() method. The client can unbind the service by calling the unbindService() method.

The service cannot be stopped until all clients unbind the service.

A bound service is the server in a client-server interface. It allows components (such as activities) to bind to the service, send requests, receive responses, and perform interprocess communication (IPC)

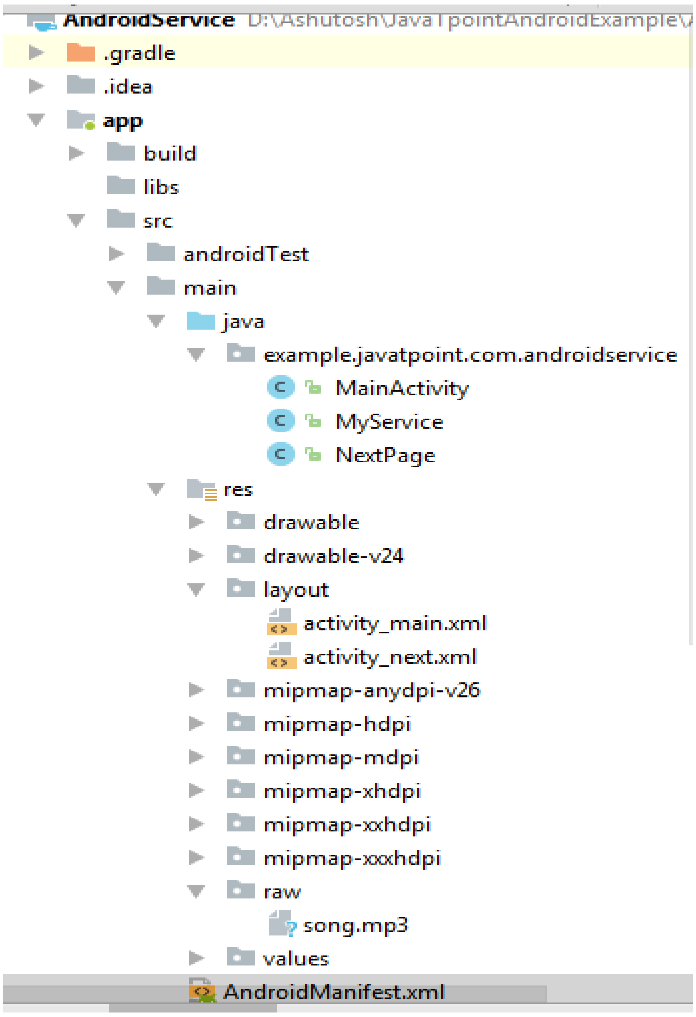


Understanding Started and Bound Service by background music example

Suppose, I want to play music in the background, so call startService() method. But I want to get information of the current song being played, I will bind the service that provides information about the current song.

Android Service Example

Let's see the example of service in android that plays an audio in the background. Audio will not be stopped even if you switch to another activity. To stop the audio, you need to stop the service.



activity_main.xml

Drag the 3 buttons from the pallete, now the activity_main.xml will look like this:

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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="example.javatpoint.com.androidservice.MainActivity">


    <Button
        android:id="@+id/buttonStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="74dp"
        android:text="Start Service" />

    <Button
        android:id="@+id/buttonStop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="Stop Service" />

    <Button
        android:id="@+id/buttonNext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="63dp"
        android:text="Next Page" />
</RelativeLayout>
```

activity_next.xml

It is the layout file of next activity.

File: activity_next.xml

It contains only one textview displaying the message Next Page

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="example.javatpoint.com.androidservice.NextPage">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="200dp"
        android:text="Next Page"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Service class

Now create the service implemenation class by inheriting the Service class and overridding its callback methods.

MyService.java

```
package example.javatpoint.com.androidservice;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.support.annotation.Nullable;
import android.widget.Toast;

public class MyService extends Service {
    MediaPlayer myPlayer;
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public void onCreate() {
        Toast.makeText(this, "Service Created", Toast.LENGTH_LONG).show();

        myPlayer = MediaPlayer.create(this, R.raw.sun);
        myPlayer.setLooping(false); // Set looping
    }
    @Override
    public void onStart(Intent intent, int startid) {
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        myPlayer.start();
    }
    @Override
    public void onDestroy() {
        Toast.makeText(this, "Service Stopped", Toast.LENGTH_LONG).show();
        myPlayer.stop();
    }
}
```

Activity class

Now create the MainActivity class to perform event handling. Here, we are writing the code to start and stop service. Additionally, calling the second activity on buttonNext.

File: MainActivity.java

```
package example.javatpoint.com.androidservice;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    Button buttonStart, buttonStop,buttonNext;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        buttonStart = findViewById(R.id.buttonStart);
        buttonStop = findViewById(R.id.buttonStop);
        buttonNext =  findViewById(R.id.buttonNext);

        buttonStart.setOnClickListener(this);
        buttonStop.setOnClickListener(this);
        buttonNext.setOnClickListener(this);


    }
    public void onClick(View src) {
        switch (src.getId()) {
            case R.id.buttonStart:

                startService(new Intent(this, MyService.class));
                break;
            case R.id.buttonStop:
                stopService(new Intent(this, MyService.class));
                break;
            case R.id.buttonNext:
                Intent intent=new Intent(this,NextPage.class);
                startActivity(intent);
                break;
        }
    }
}
```

NextPage class

Now, create another activity.

File: NextPage.java

```
package example.javatpoint.com.androidservice;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

```

public class NextPage extends AppCompatActivity {

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

Declare the Service in the AndroidManifest.xml file

Finally, declare the service in the manifest file.

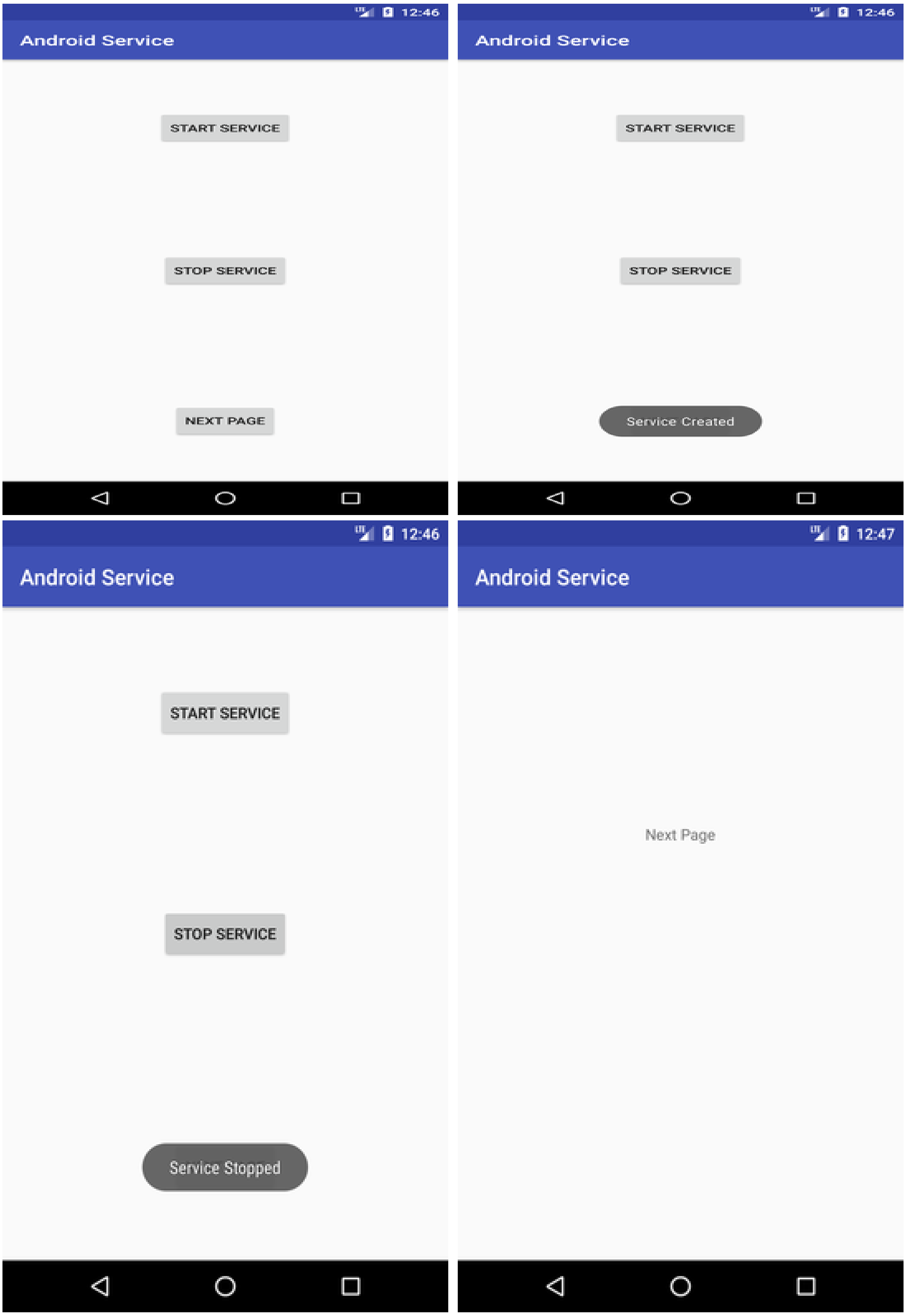
File: AndroidManifest.xml

Let's see the complete AndroidManifest.xml file

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

    <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/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".NextPage"></activity>
        <service
```

```
            android:name=".MyService"
            android:enabled="true" />
    </application>

</manifest>
```

Output:



Service Created

Service Stopped

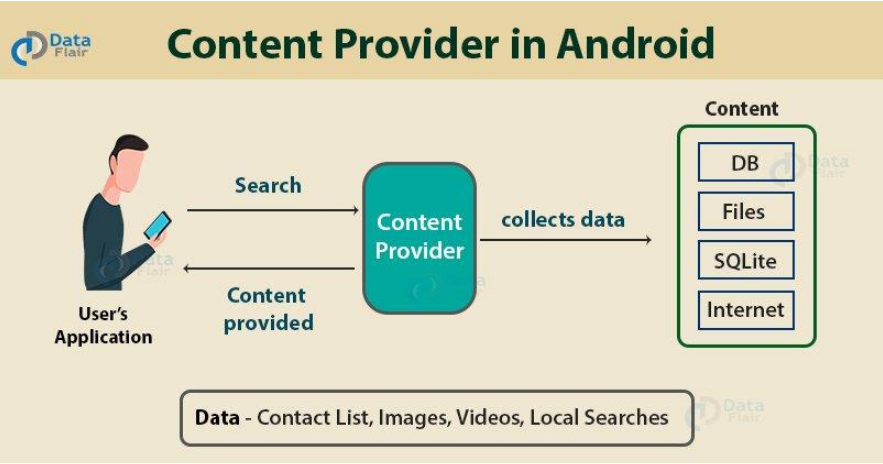
Next Page

https://data-flair.training/blogs/content-provider-in-android/

Content Providers in Android

Content Providers are an important component of Android. They handle the access to the central repository and supply data from one application to another on request. This task of handling is done by methods of ContentResolver class. So, content providers can store data in various ways such as files, database or over the internet.

Many a time we need to share our data with applications and that's where it becomes useful. The following diagram depicts how the content provider helps in sharing data with applications from data stores.



As can be seen, the content provider lets us collect the data centrally and provide them to applications as shown in the above diagram. Content providers act the same as database and also we can query it to add, delete, insert or update the data.

It can be understood that a content provider hides the database details and also, it lets an application share data among other applications. Content providers are not limited to texts, but also contains images and videos as well.

CRUD Operations

Content providers provide the following four basic operations. These are also known as CRUD operations, where

C – Create
R – Read
U – Update
D – Delete
The above are the four operations of content providers :

- Create: It is used for the creation of data in content providers.

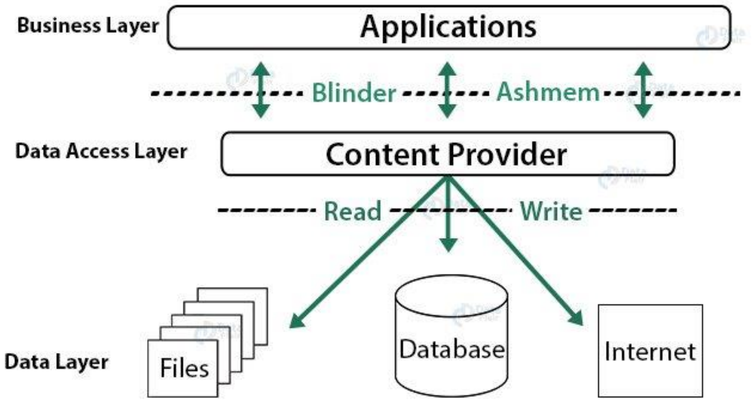
- Read: It reads the data stored in the content provider.
- Update: It lets the editing in existing data in content providers.
- Delete: It deletes the existing data stored in its Storage.

Examples of Android Content Providers

To understand content providers with examples, consider the following examples :

- The Gallery that contains images.
- Contact lists that contain Contact details.
- A dictionary that has collections of all the words that are used.
- The music playlist has a list of songs.
- Call logs contain the call details.

This is how exactly content providers work.

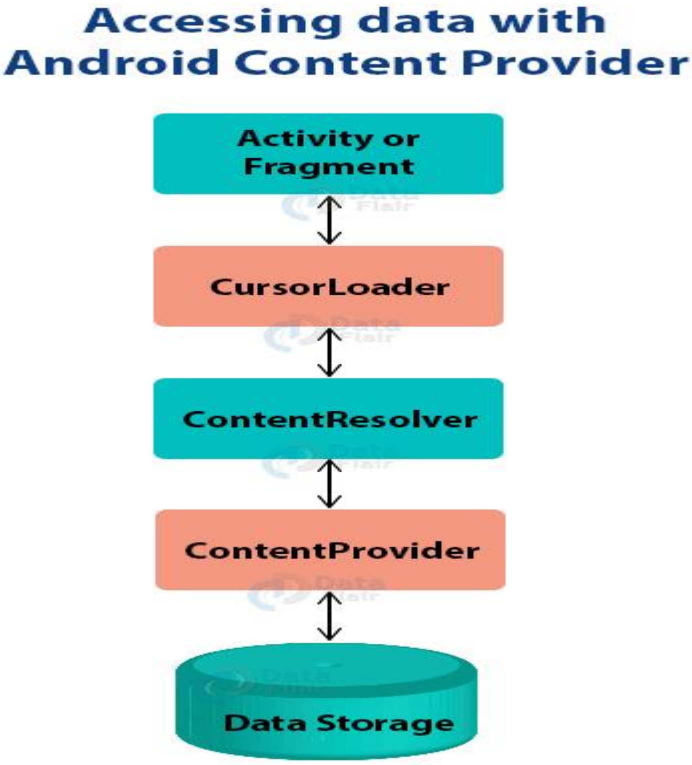


Working of Android Content Provider

Accessing Data with Content Provider

We need to use the ContentResolver object in our application, in order to communicate with the content providers for data access. Now to enable communication between the user interface and ContentResolver, we use another object, CursorLoader to run query asynchronously. This CursorLoader will be called using Activity/Fragment of the application. Then, the content provider receives the query from the client and executes and returns the result.

To understand how it works, consider the following diagram:



Methods of Content Provider in Android

Let’s see the following methods of content provider:

- onCreate() – This method in Android initializes the provider as soon as the receiver is created.
- query() – It receives a request in the form of a query from the user and responds with a cursor in Android.
- insert() – This method is used to insert the data into our content provider.
- update() – This method is used to update existing data in a row and return the updated row data.
- delete() – This method deletes existing data from the content provider.
- getType() – It returns the Multipurpose Internet Mail Extension type of data to the given Content URI.

Intent

Android uses Intent for communicating between the components of an Application and also from one application to another application.

Intent are the objects which is used in android for passing the information among Activities in an Application and from one app to another also. Intent are used for communicating between the Application components and it also provides the connectivity between two apps.

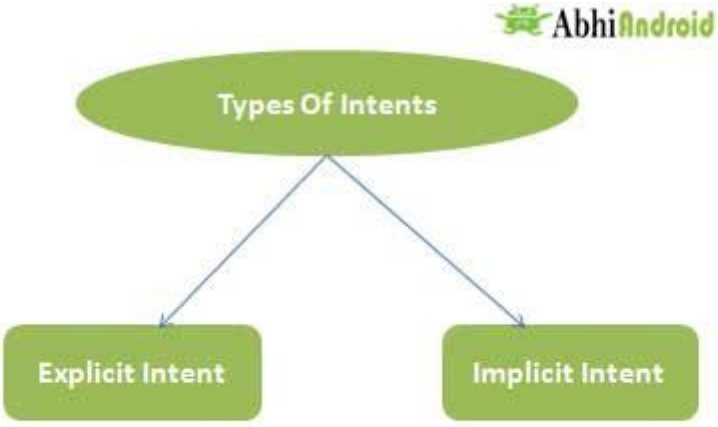
For example: Intent facilitate you to redirect your activity to another activity on occurrence of any event. By calling, startActivity() you can perform this task.

```
Intentintent=newIntent(getApplicationContext(),SecondActivity.class);
startActivity(intent);
```

In the above example, foreground activity is getting redirected to another activity i.e. SecondActivity.java. getApplicationContext() returns the context for your foreground activity.

Types of Intents:

Intent are of two types: Explicit Intent and Implicit Intent



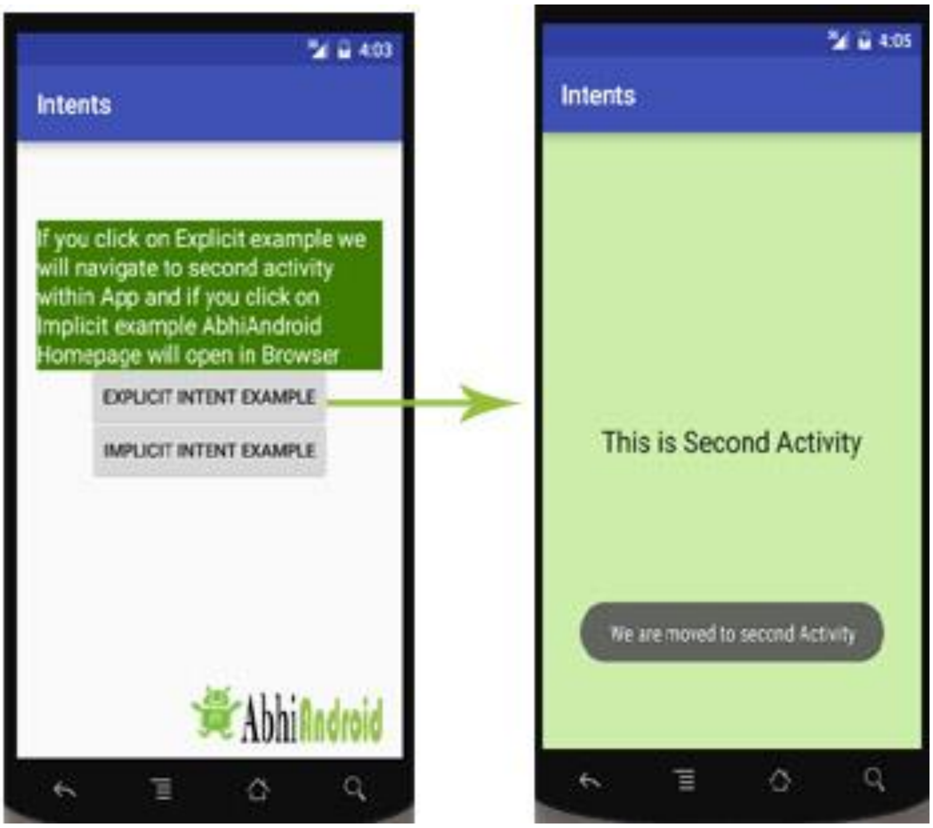
Explicit Intent:

- Explicit Intents are used to connect the application internally.
- In Explicit we use the name of component which will be affected by Intent. For Example: If we know class name then we can navigate the app from One Activity to another activity using Intent. In the similar way we can start a service to download a file in background process.

Explicit Intent work internally within an application to perform navigation and data transfer. The below given code snippet will help you understand the concept of Explicit Intents

```
Intentintent=newIntent(getApplicationContext(),SecondActivity.class);
startActivity(intent);
```

Here SecondActivity is the JAVA class name where the activity will now be navigated. Example with code in the end of this post will make it more clear.



If you click on Explicit example we will navigate to second activity within App and if you click on Implicit example AbhiAndroid Homepage will open in Browser

This is Second Activity

We are moved to second Activity

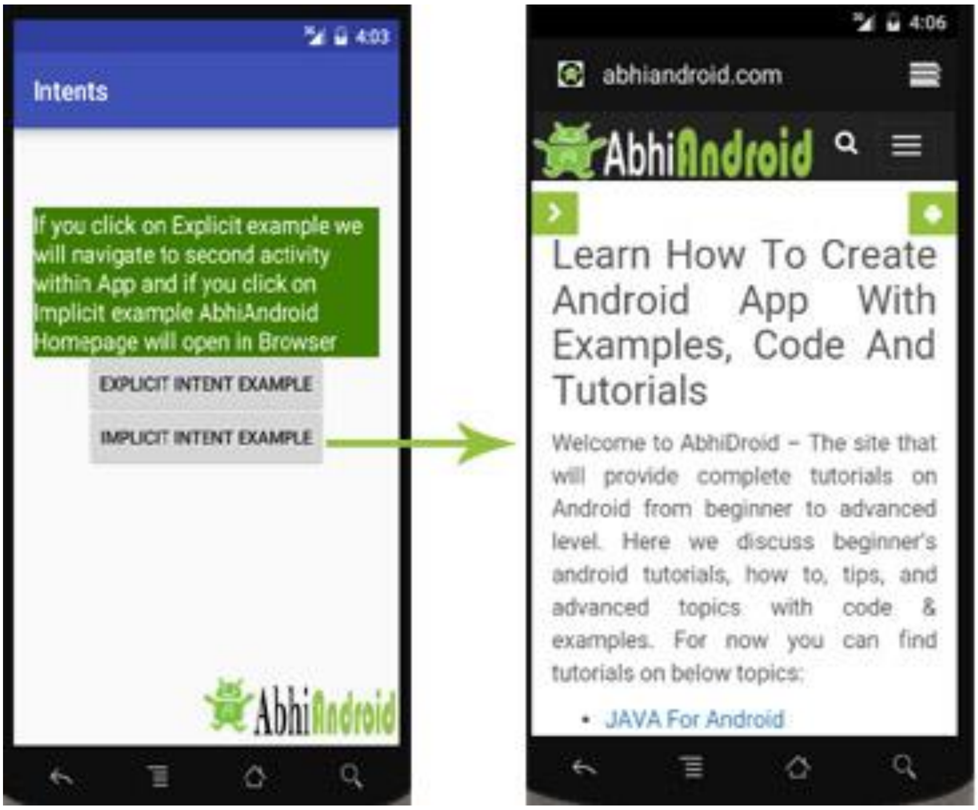
Implicit Intent:

- In Implicit Intents we do need to specify the name of the component. We just specify the Action which has to be performed and further this action is handled by the component of another application.
- The basic example of implicit Intent is to open any web page

Let's take an example to understand Implicit Intents more clearly. We have to open a website using intent in your application. See the code snippet given below

```
IntentintentObj=newIntent(Intent.ACTION_VIEW);
intentObj.setData(Uri.parse("https://www.abhiandroid.com"));
startActivity(intentObj);
```

Unlike Explicit Intent you do not use any class name to pass through Intent(). In this example we have just specified an action. Now when we will run this code then Android will automatically start your web browser and it will open AbhiAndroid home page.



If you click on Explicit example we will navigate to second activity within App and if you click on Implicit example AbhiAndroid Homepage will open in Browser

Learn How To Create Android App With Examples, Code And Tutorials

Welcome to AbhiDroid – The site that will provide complete tutorials on Android from beginner to advanced level. Here we discuss beginner's android tutorials, how to, tips, and advanced topics with code & examples. For now you can find tutorials on below topics:

- JAVA For Android

Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

There are following two important steps to make BroadcastReceiver works for the system broadcasted intents –

- Creating the Broadcast Receiver.
- Registering Broadcast Receiver

There is one additional steps in case you are going to implement your custom intents then you will have to create and broadcast those intents.

Creating the Broadcast Receiver

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and overriding the onReceive() method where each message is received as a Intent object parameter.

```
publicclassMyReceiverextendsBroadcastReceiver{
@Override
publicvoidonReceive(Context context,Intent intent){
Toast.makeText(context,"Intent Detected.",Toast.LENGTH_LONG).show();
}
}
```

Registering Broadcast Receiver

An application listens for specific broadcast intents by registering a broadcast receiver in *AndroidManifest.xml* file. Consider we are going to register *MyReceiver* for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.

Broadcast-Receiver

```
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<receiverandroid:name="MyReceiver">

<intent-filter>
<actionandroid:name="android.intent.action.BOOT_COMPLETED">
</action>
</intent-filter>

</receiver>
</application>
```

Now whenever your Android device gets booted, it will be intercepted by BroadcastReceiver *MyReceiver* and implemented logic inside *onReceive()* will be executed.

There are several system generated events defined as final static fields in the Intent class. The following table lists a few important system events.

Sr.No	Event Constant & Description
1	android.intent.action.BATTERY_CHANGED Sticky broadcast containing the charging state, level, and other information about the battery.
2	android.intent.action.BATTERY_LOW Indicates low battery condition on the device.
3	android.intent.action.BATTERY_OKAY Indicates the battery is now okay after being low.
4	android.intent.action.BOOT_COMPLETED This is broadcast once, after the system has finished booting.

5	android.intent.action.BUG_REPORT Show activity for reporting a bug.
6	android.intent.action.CALL Perform a call to someone specified by the data.
7	android.intent.action.CALL_BUTTON The user pressed the "call" button to go to the dialer or other appropriate UI for placing a call.
8	android.intent.action.DATE_CHANGED The date has changed.
9	android.intent.action.REBOOT Have the device reboot.

Android Widgets

There are given a lot of android widgets with simplified examples such as Button, EditText, AutoCompleteTextView, ToggleButton, DatePicker, TimePicker, ProgressBar etc.

Android widgets are easy to learn. The widely used android widgets with examples are given below:

Android Button

Let's learn how to perform event handling on button click.

Android Toast

Displays information for the short duration of time.

Custom Toast

We are able to customize the toast, such as we can display image on the toast

ToggleButton

It has two states ON/OFF.

CheckBox

Let's see the application of simple food ordering.

AlertDialog

AlertDialog displays a alert dialog containing the message with OK and Cancel buttons.

Spinner

Spinner displays the multiple options, but only one can be selected at a time.

AutoCompleteTextView

Let's see the simple example of AutoCompleteTextView.

RatingBar

RatingBar displays the rating bar.

DatePicker

Datepicker displays the datepicker dialog that can be used to pick the date.

TimePicker

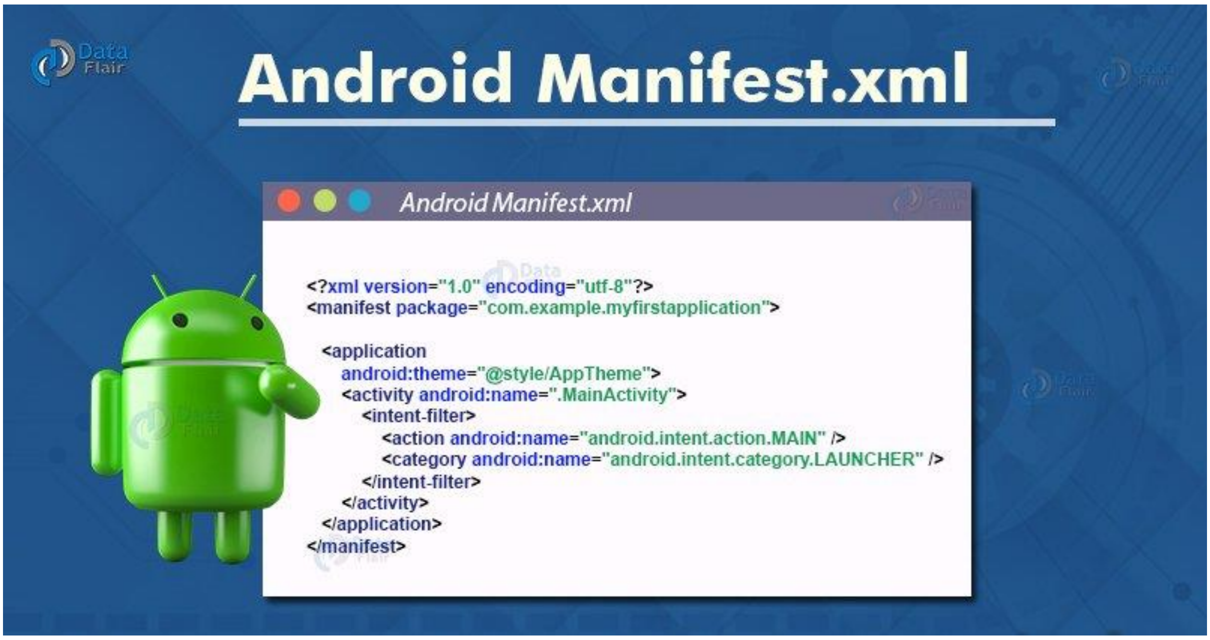
TimePicker displays the timepicker dialog that can be used to pick the time.

ProgressBar

ProgressBar displays progress task.

Application Android Manifest:

As we know that every android project that we make, need to have an Android Manifest file within. This file is present in the root of the project source set. The reason why it is a must to have it is that it describes all the important information of the application to the Android build tools.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="com.example.myfirstapplication">

  <application
    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>
```

The manifest file in Android is generally created automatically as soon as the app is built in Android Studio.

Structure of a Manifest file in Android is:

```
<manifest>
<application>
<activity android:name="com.example.applicationname.MainActivity">
</activity>
</application>
</manifest>
```

The information that is stored in the Manifest file is as follows:

- The name of the application's package, it is generally the code's namespace. This information is used to determine the location of the code while building the project.
- Another component is the one, that includes all the activities, services, receivers, and content providers.
- The permissions that are required by the application to access the protected parts of the system and other apps.
- The features required by the app, that affect which devices can install the app from Google Play. These features include both hardware and software features.
- It also specifies the application metadata, which includes the icon, version number, themes, etc.

The android manifest.xml file generally looks like the following:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.myfirstapplication">
<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/AppTheme">
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<meta-data
android:name="preloaded_fonts"
android:resource="@array/preloaded_fonts" />
</application>
</manifest>
```

Element Tags of Manifest File

Following are the essential element tags of Manifest.xml files:

1. <manifest>
It is the root element of this element. It consists of a package attribute package that tells the activity's package name.

2. <application>
It is the subelement of the manifest file that includes the declaration of the namespace. It contains certain attributes. These attributes declare the application components, and these attributes include:

- icon
- allowBackup
- label
- theme

3. <activity>
Activity is a subelement of application. It has the declaration of the activity that must be there in the manifest file. It also has certain attributes like name, label, theme, etc.

4. <intent-filter>
It is an element in the activity, it describes the type of intent in which the Android components can respond.

5. <action>
This element provides an action for the intent filter. Each intent filter must have at least one action element in it.

6. <category>
This element adds the category name in an intent-filter.

7. <service>
This element contains the operations that are provided by libraries or APIs.

Manifest File Application Property Elements

Following is a list of important Application Property Elements in manifest.xml (Sub- Node Elements)

1. <uses-permission>: This element specifies the Android Manifest permissions that are requested for the purpose of security.
2. <permission>: This element sets permission to provide access to control for some components of the app.
3. <permission-groups>: This element sets permission to provide access to control for a set of components of the app.
4. <permission-tree>: This element refers to a specific component that is the owner of the set of components.
5. <instrumentation>: This element tells the interaction between the app and the system.
6. <uses-sdk>: This one specifies the compatibility of the app.
7. <uses-configuration>: This element specifies the permissions that are requested for the purpose of security.
8. <uses-feature>: This element specifies one hardware or software feature that is required by the Application.
9. <supports-screen, compatible-screen>: These elements tell the screen size and configuration.

How to set Manifest File Permissions in Android?

Now in this part of the tutorial, we’ll see how to set permission in a Manifest file.

An android application must get some permissions to get access to other apps or the Internet. While we build our app, the manifest file gets automatically generated as manifest.xml. This manifest file contains permissions that are configured by us. A few permissions are applied by default if there is no permission provided by us.

These are written in the manifest file as:

```
<manifest >

<uses-permission .../>

...

<manifest >
```

The permission mentioned in the manifest file can be either granted or rejected by the users. Once the user grants permission to the app, the app can use the protected features. If the user denies permission the app doesn’t get permission to access the protected features.

The following permissions are added by default and set to TRUE:

- INTERNET
- ACCESS_NETWORK_STATE
- READ_PHONE_STATE

There are many permissions that are set to FALSE by default but can be set to TRUE as per requirements. A few of these permissions are as follows:

ACCESS_WIFI_STATE	AUTHENTICATE_ACCOUNT	BLUETOOTH	BATTERY_STATS
BIND_APPWIDGET	BROADCAST_WAP_PUSH	BROADCAST_STICKY	BIND_INPUT_METHOD
CALL_PHONE	CHANGE_CONFIGURATION	CAMERA	CLEAR_APP_DATA
CHANGE_WIFI_STATE	CLEAR_APP_USER_DATA	DEVICE_POWER	DELETE_CACHE_FILES
DISABLE_KEYGUARD	DELETE_PACKAGES	EXPAND_STATUS_BAR	EXPAND_STATUS_BAR
FACTORY_TEST	GET_PACKAGE_SIZE	FLASHLIGHT	GLOBAL_SEARCH
HARDWARE_TEST	INTERNAL_SYSTEM_PROCESSES	USE_CREDENTIALS	MANAGE_ACCOUNTS
MANAGE_APP_TOKENS	MODIFY_AUDIO_SETTINGS	MODIFY_PHONE_STATE	NFC
SEND_SMS	PROCESS_OUTGOING_CALLS	SET_ALARM	SET_ALWAYS_FINISH
READ_CALENDAR	KILL_BACKGROUND_PROCESSES	SET_WALLPAPER	VIBRATE
WAKE_LOCK	WRITE_APN_SETTINGS	WRITE_CALENDAR	WARITE_SETTINGS

UNIT-II

INTERFACE TOOLS

Creating user interface - Views - creating views - Layouts - Drawable resources - resolution and density independence - Menus - Intents - Adapters - Using Internet resources - Dialogs.

Creating user interface

In android, **Layout** is used to define the user interface for an app or activity and it will hold the UI elements that will appear to the user.

The user interface in an android app is made with a collection of View and ViewGroup objects. Generally, the android apps will contain one or more activities and each activity is a one screen of app. The activities will contain a multiple UI components and those UI components are the instances of View and ViewGroup subclasses.

The user interface in an android app is made with a collection of View and ViewGroup objects. Generally, the android apps will contain one or more activities and each activity is a one screen of the app. The activities will contain multiple UI components and those UI components are the instances of View and ViewGroup subclasses.

Android View

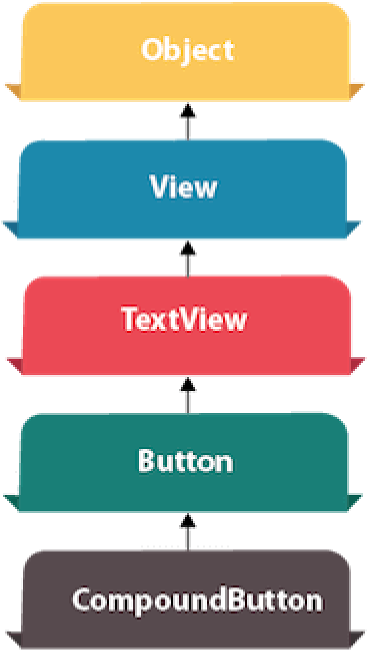
The View is a base class for all UI components in android. For example, the **EditText** class is used to accept the input from users in android apps, which is a subclass of View.

Following are the some of common View subclasses that will be used in android applications.

- TextView
- EditText
- Button
- CheckBox
- RadioButton
- ImageButton
- Progress Bar
- Spinner

Like these we have many View subclasses available in android.

Android Button Example



Android Button represents a push-button. The android.widget.Button is subclass of TextView class and CompoundButton is the subclass of Button class.

There are different types of buttons in android such as RadioButton, ToggleButton, CompoundButton etc.

Android Button Example with Listener

Here, we are going to create two textfields and one button for sum of two numbers. If user clicks button, sum of two input values is displayed on the Toast.

We can perform action on button using different types such as calling listener on button or adding onClick property of button in activity's xml file.

```
button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
               //code
            }
});
```

```
<Button
        android:onClick="methodName"
/>
```

File: activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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="example.javatpoint.com.sumoftwonumber.MainActivity">

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="61dp"
        android:ems="10"
        android:inputType="number"
        tools:layout_editor_absoluteX="84dp"
        tools:layout_editor_absoluteY="53dp" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:inputType="number"
        tools:layout_editor_absoluteX="84dp"
        tools:layout_editor_absoluteY="127dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="109dp"
        android:text="ADD"
        tools:layout_editor_absoluteX="148dp"
        tools:layout_editor_absoluteY="266dp" />
</RelativeLayout>
```

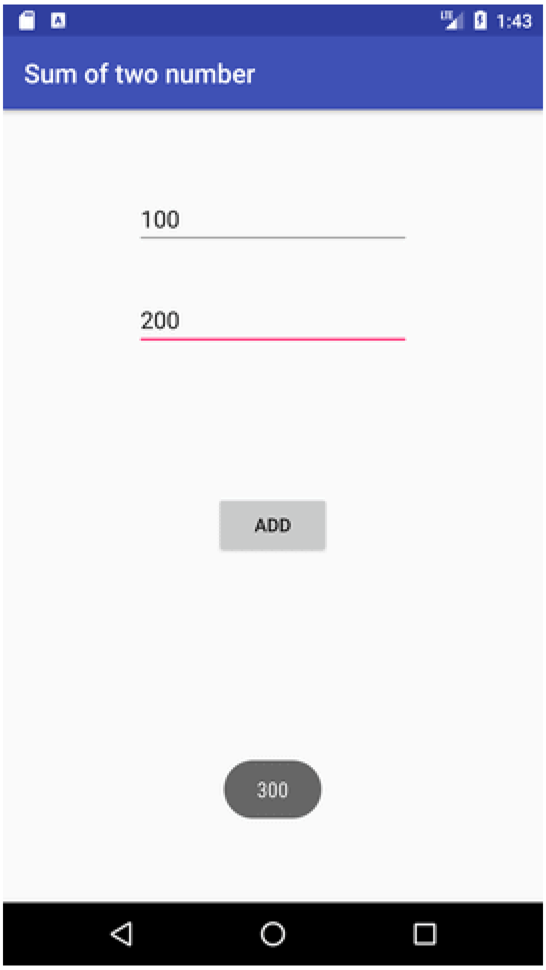
Activity class

Now write the code to display the sum of two numbers.

File: MainActivity.java

```
package example.javatpoint.com.sumoftwonumber;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private EditText edittext1, edittext2;
    private Button buttonSum;

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

        addListenerOnButton();
    }

    public void addListenerOnButton() {
        edittext1 = (EditText) findViewById(R.id.editText1);
        edittext2 = (EditText) findViewById(R.id.editText2);
        buttonSum = (Button) findViewById(R.id.button);

        buttonSum.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String value1=edittext1.getText().toString();
                String value2=edittext2.getText().toString();
                int a=Integer.parseInt(value1);
                int b=Integer.parseInt(value2);
                int sum=a+b;
                Toast.makeText(getApplicationContext(),String.valueOf(sum), Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

Ouput:



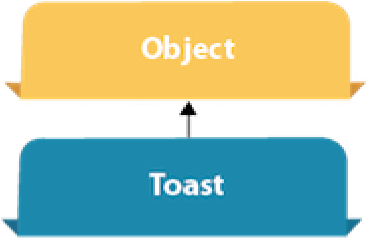
Sum of two number

100

200

300

Android Toast Example



Andorid Toast can be used to display information for the short period of time. A toast contains message to be displayed quickly and disappears after sometime.

The android.widget.Toast class is the subclass of java.lang.Object class.

You can also create custom toast as well for example toast displaying image. You can visit next page to see the code for custom toast.

Toast class

Toast class is used to show notification for a particular interval of time. After sometime it disappears. It doesn't block the user interaction.

Constants of Toast class

There are only 2 constants of Toast class which are given below.

Constant	Description
public static final int LENGTH_LONG	displays view for the long duration of time.
public static final int LENGTH_SHORT	displays view for the short duration of time.

Android Toast Example

```
1. Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT).show();
```

Another code:

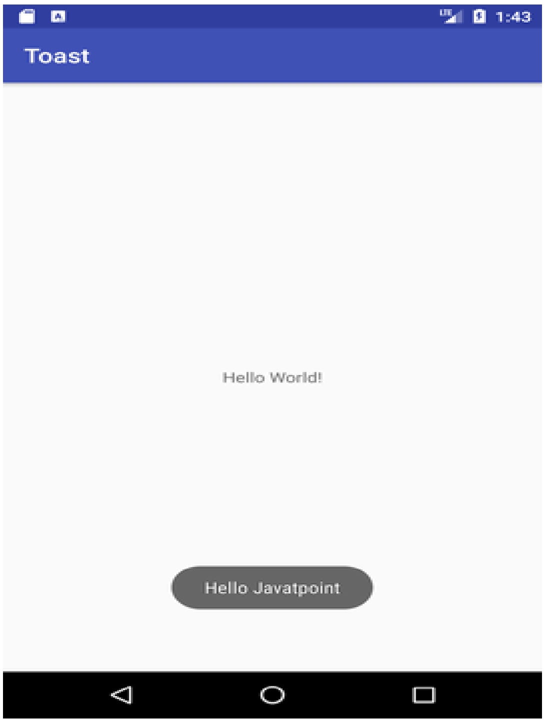
```
1. Toast toast=Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT);
2. toast.setMargin(50,50);
3. toast.show();
```

Here, getApplicationContext() method returns the instance of Context.

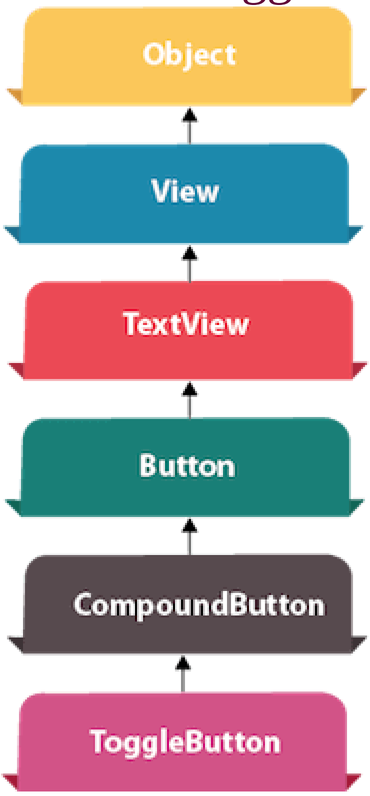
Example

```
package example.javatpoint.com.toast;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

        //Displaying Toast with Hello Javatpoint message
        Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT).show();
    }
}
```

Output:



Android ToggleButton Example



Android Toggle Button can be used to display checked/unchecked (On/Off) state on the button.

It is beneficial if user have to change the setting between two states. It can be used to On/Off Sound, Wifi, Bluetooth etc.

Since Android 4.0, there is another type of toggle button called *switch* that provides slider control.

Android ToggleButton and Switch both are the subclasses of CompoundButton class.

Android ToggleButton class

ToggleButton class provides the facility of creating the toggle button.

XML Attributes of ToggleButton class

The 3 XML attributes of ToggleButton class.

XML Attribute	Description
android:disabledAlpha	The alpha to apply to the indicator when disabled.
android:textOff	The text for the button when it is not checked.
android:textOn	The text for the button when it is checked.

Methods of ToggleButton class

The widely used methods of ToggleButton class are given below.

Method	Description
CharSequence getTextOff()	Returns the text when button is not in the checked state.
CharSequence getTextOn()	Returns the text for when button is in the checked state.
void setChecked(boolean checked)	Changes the checked state of this button.

activity_main.xml

Drag two toggle button and one button for the layout. Now the activity_main.xml file will look like this:

File: activity_main.xml

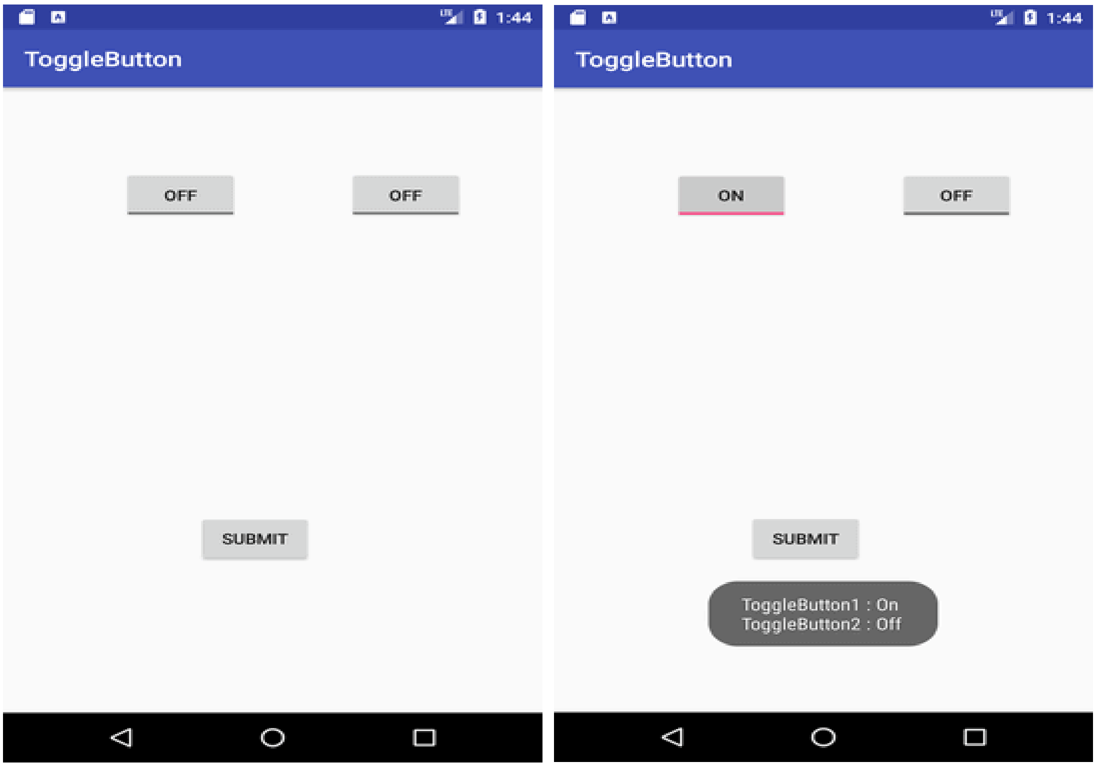
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

    <ToggleButton
        android:id="@+id/toggleButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="80dp"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintEnd_toStartOf="@+id/toggleButton2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ToggleButton
        android:id="@+id/toggleButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="60dp"
        android:layout_marginTop="80dp"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="144dp"
        android:layout_marginLeft="148dp"
        android:text="Submit"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Activity class

File: MainActivity.java

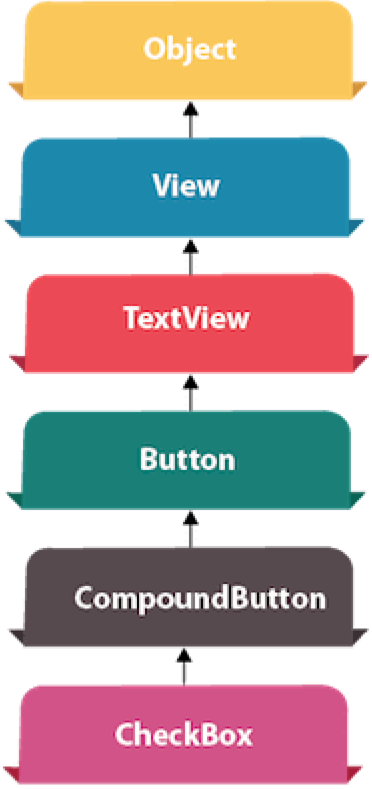
```
package example.javatpoint.com.togglebutton;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {
    private ToggleButton toggleButton1, toggleButton2;
    private Button buttonSubmit;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        addListenerOnButtonClick();
    }

    public void addListenerOnButtonClick(){
        //Getting the ToggleButton and Button instance from the layout xml file
        toggleButton1=(ToggleButton)findViewById(R.id.toggleButton);
        toggleButton2=(ToggleButton)findViewById(R.id.toggleButton2);
        buttonSubmit=(Button)findViewById(R.id.button);

        //Performing action on button click
        buttonSubmit.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View view) {
                StringBuilder result = new StringBuilder();
                result.append("ToggleButton1 : ").append(toggleButton1.getText());
                result.append("\nToggleButton2 : ").append(toggleButton2.getText());
                //Displaying the message in toast
                Toast.makeText(getApplicationContext(), result.toString(),Toast.LENGTH_LONG).show()
;
            }

        });

    }
}
```

Output:



ToggleButton1 : On
ToggleButton2 : Off

Android CheckBox Example



Android CheckBox is a type of two state button either checked or unchecked.

There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

Android CheckBox class is the subclass of CompoundButton class.

Android CheckBox class

The android.widget.CheckBox class provides the facility of creating the CheckBoxes.

Methods of CheckBox class

There are many inherited methods of View, TextView, and Button classes in the CheckBox class. Some of them are as follows:

Method	**Description**
public boolean isChecked()	Returns true if it is checked otherwise false.
public void setChecked(boolean status)	Changes the state of the CheckBox.

Android CheckBox Example

activity_main.xml

Drag the three checkboxes and one button for the layout. Now the activity_main.xml file will look like this:

File: activity_main.xml

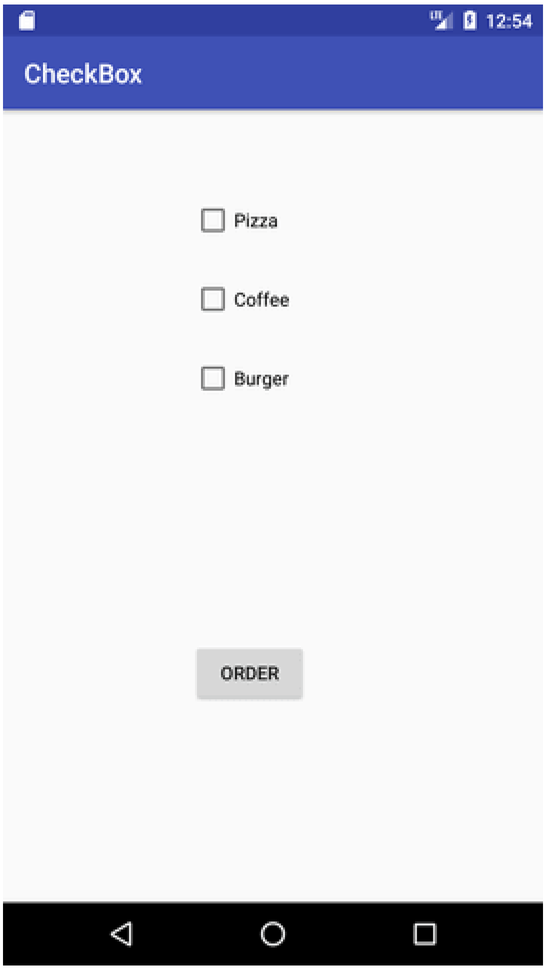
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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


    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="68dp"
        android:text="Pizza"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="28dp"
        android:text="Coffee"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox" />

    <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="28dp"
        android:text="Burger"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox2" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="184dp"
        android:text="Order"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox3" />

</android.support.constraint.ConstraintLayout>
```

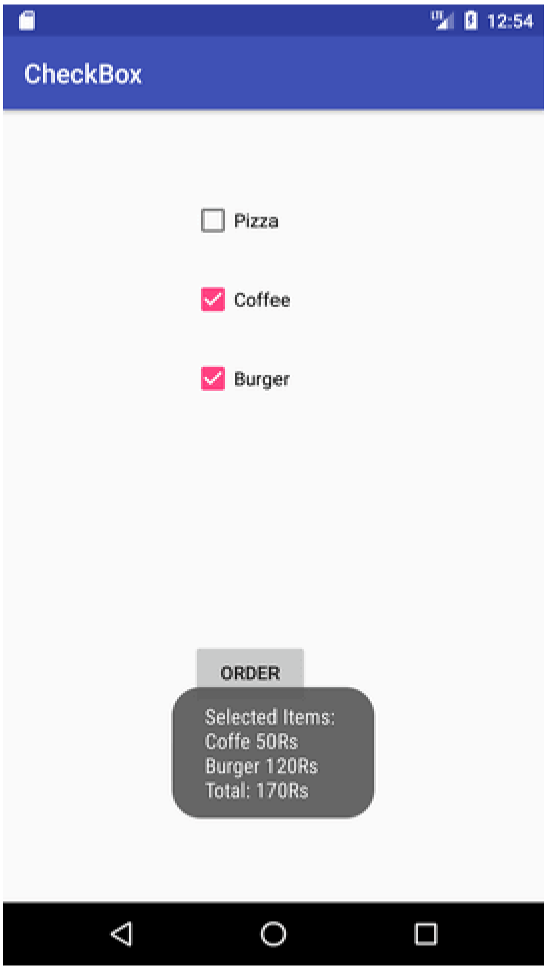
Activity class

File: MainActivity.java

```
package example.javatpoint.com.checkbox;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    CheckBox pizza,coffe,burger;
    Button buttonOrder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addListenerOnButtonClick();
    }
    public void addListenerOnButtonClick(){
        //Getting instance of CheckBoxes and Button from the activty_main.xml file
        pizza=(CheckBox)findViewById(R.id.checkBox);
        coffe=(CheckBox)findViewById(R.id.checkBox2);
        burger=(CheckBox)findViewById(R.id.checkBox3);
        buttonOrder=(Button)findViewById(R.id.button);

        //Applying the Listener on the Button click
        buttonOrder.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View view) {
                int totalamount=0;
                StringBuilder result=new StringBuilder();
                result.append("Selected Items:");
                if(pizza.isChecked()){
                    result.append("\nPizza 100Rs");
                    totalamount+=100;
                }
                if(coffe.isChecked()){
                    result.append("\nCoffe 50Rs");
                    totalamount+=50;
                }
                if(burger.isChecked()){
                    result.append("\nBurger 120Rs");
                    totalamount+=120;
                }
                result.append("\nTotal: "+totalamount+"Rs");
                //Displaying the message on the toast
                Toast.makeText(getApplicationContext(), result.toString(), Toast.LENGTH_LONG).show();
            }

        });
    }
}
```

Output:





Selected Items:
Coffe 50Rs
Burger 120Rs
Total: 170Rs

Android ViewGroup

The ViewGroup is a subclass of View and it will act as a base class for **layouts** and **layouts parameters**. The ViewGroup will provide an invisible containers to hold other **Views** or **ViewGroups** and to define the layout properties.

For example, Linear Layout is the ViewGroup that contains a UI controls like button, textview, etc. and other layouts also.

Following are the commonly used ViewGroup subclasses in android applications.

- Linear Layout
- Relative Layout
- Table Layout
- Frame Layout
- Web View
- List View
- Grid View

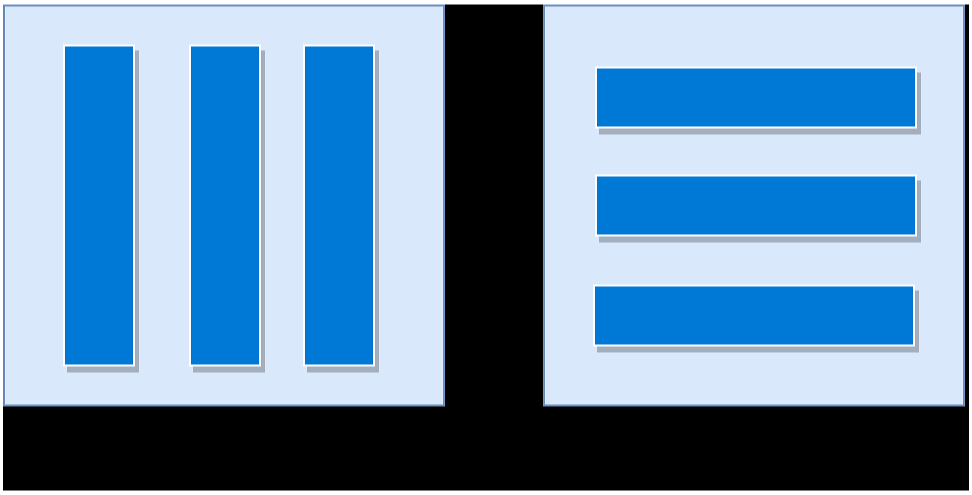
Both View and ViewGroup subclasses together will play a key role to create a layouts in android applications.

Android LinearLayout with Examples

In android, **LinearLayout** is a **ViewGroup** subclass which is used to render all child **View** instances one by one either in **Horizontal** direction or **Vertical** direction based on the **orientation** property.

In android, we can specify the linear layout orientation using **android:orientation** attribute.

Following is the pictorial representation of linear layout in android applications.



In **LinearLayout**, the child **View** instances arranged one by one, so the horizontal list will have only one row of multiple columns and vertical list will have one column of multiple rows.

Android LinearLayout Declaration

Following is the way we need to define the LinearLayout in android applications.

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

If you observe above code snippet, here we defined orientation as vertical, so this aligns all its child layout / views vertically.

Android LinearLayout Example

Following is the example of creating a **LinearLayout** with different controls in android application.

Create a new android application using android studio and give names as **LinearLayout**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/txtTo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="To"/>
    <EditText
        android:id="@+id/txtSub"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Subject"/>
    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="Message"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Send"/>
</LinearLayout>
```

Once we are done with creation of layout, we need to load the XML layout resource from our activity **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.linearlayout** path and write the code like as shown below.

MainActivity.java

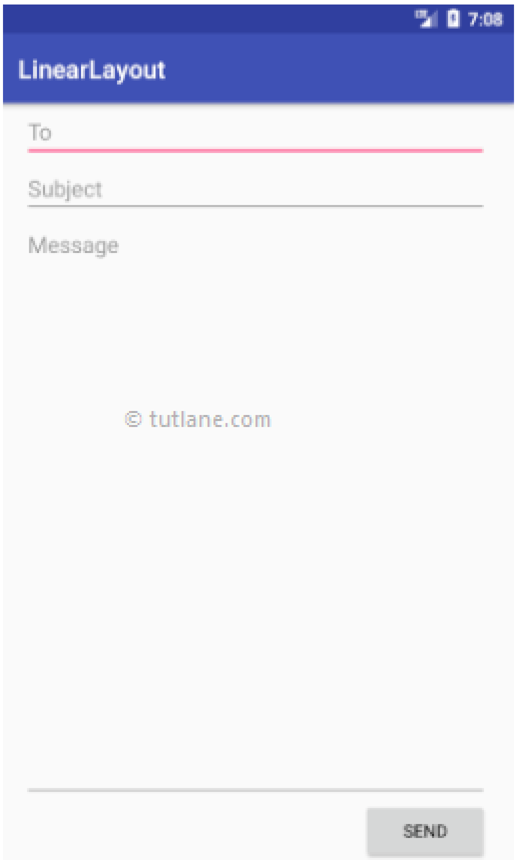
```
package com.tutlane.linearlayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Generally, during the launch of our activity, the **onCreate()** callback method will be called by the android framework to get the required layout for an activity.

Output of Android LinearLayout Example

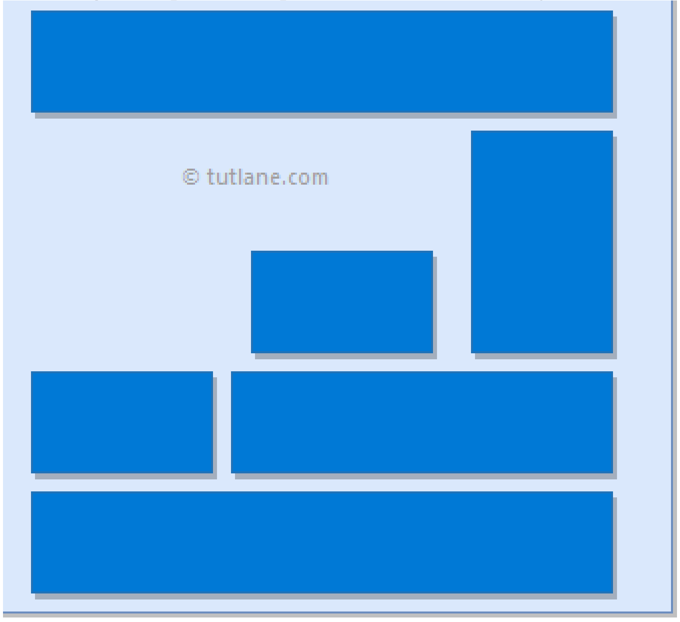
When we run above example using the android virtual device (AVD) we will get a result like as shown below.



Android RelativeLayout with Examples

In android, **RelativeLayout** is a **ViewGroup** which is used to specify the position of child **View** instances relative to each other (Child **A** to the left of Child **B**) or relative to the parent (Aligned to the top of parent).

Following is the pictorial representation of relative layout in android applications.



n android, **RelativeLayout** is very useful to design user interface because by using relative layout we can eliminate the nested view groups and keep our layout hierarchy flat, which improves the performance of application.

Android Positioning Views in Relative Layout

As we discussed, in **RelativeLayout** we need to specify the position of child views relative to each other or relative to the parent. In case if we didn't specify the position of child views, by default all child views are positioned to top-left of the layout.

Following are the some of most useful layout properties available to views in RelativeLayout.

Attribute	Description
layout_alignParentTop	If it specified "true", the top edge of view will match the top edge of the parent.
layout_alignParentBottom	If it specified "true", the bottom edge of view will match the bottom edge of parent.
layout_alignParentLeft	If it specified "true", the left edge of view will match the left edge of parent.
layout_alignParentRight	If it specified "true", the right edge of view will match the right edge of the parent.
layout_centerInParent	If it specified "true", the view will be aligned to the centre of parent.
layout_centerHorizontal	If it specified "true", the view will be horizontally centre aligned within its parent.
layout_centerVertical	If it specified "true", the view will be vertically centre aligned within its parent.
layout_above	It accepts another sibling view id and places the view above the specified view id.
layout_below	It accepts another sibling view id and places the view below the specified view id.
layout_toLeftOf	It accepts another sibling view id and places the view left of the specified view id.
layout_toRightOf	It accepts another sibling view id and places the view right of the specified view id.
layout_toStartOf	It accepts another sibling view id and places the view to start of the specified view id.

Attribute	Description
layout_toEndOf	It accepts another sibling view id and places the view to the end of the specified view id.

Android RelativeLayout Example

Following is the example of creating a **RelativeLayout** with different controls in android application.

Create a new android application using android studio and give names as **RelativeLayout**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?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"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Button1" />
    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:text="Button2" />
    <Button
        android:id="@+id/btn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:text="Button3" />

    <Button
        android:id="@+id/btn4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Button4" />
    <Button
        android:id="@+id/btn5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btn2"
        android:layout_centerHorizontal="true"
        android:text="Button5" />
    <Button
        android:id="@+id/btn6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btn4"
        android:layout_centerHorizontal="true"
        android:text="Button6" />
    <Button
        android:id="@+id/btn7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/btn1"
        android:layout_toRightOf="@+id/btn1"
        android:layout_alignParentRight="true"
        android:text="Button7" />
</RelativeLayout>
```

Once we are done with creation of layout, we need to load the XML layout resource from our activity **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.relativelayout** path and write the code like as shown below.

MainActivity.java

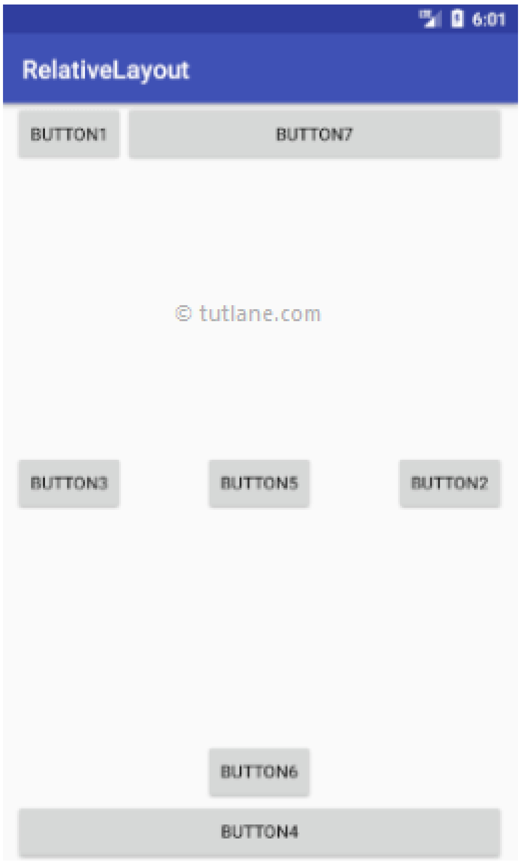
```
package com.tutlane.linearlayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Generally, during the launch of our activity, **onCreate()** callback method will be called by the android framework to get the required layout for an activity.

Output of Android RelativeLayout Example

When we run above example using the android virtual device (AVD) we will get a result like as shown below.

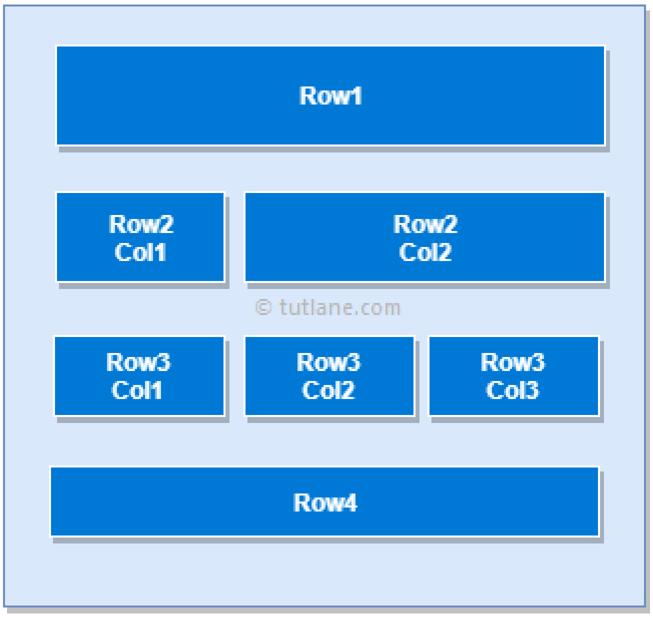


This is how we can use RelativeLayout in android applications based on our requirements.

Android TableLayout with Examples

In android, **TableLayout** is a **ViewGroup** subclass that is used to display the child View elements in rows and columns.

Following is the pictorial representation of table layout in android applications.



In android, TableLayout will position its children elements into rows and columns and it won't display any border lines for rows, columns or cells.

The TableLayout in android will work same as the HTML table and the table will have as many columns as the row with the most cells. The TableLayout can be explained as **<table>** and TableRow is like **<tr>** element.

Android TableLayout Example

Following is the example of creating a **TableLayout** with different controls in android application.

Create a new android application using android studio and give names as **TableLayout**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_marginTop="100dp"
   android:paddingLeft="10dp"
   android:paddingRight="10dp" >
   <TableRow android:background="#0079D6" android:padding="5dp">
      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="UserId" />
      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
```

```
        android:layout_weight="1"
        android:text="User Name" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Location" />
    </TableRow>
    <TableRow android:background="#DAE8FC" android:padding="5dp">
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="1" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Suresh Dasari" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Hyderabad" />
    </TableRow>
    <TableRow android:background="#DAE8FC" android:padding="5dp">
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="2" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Rohini Alavala" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Guntur" />
    </TableRow>
    <TableRow android:background="#DAE8FC" android:padding="5dp">
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="3" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
```

```
            android:text="Trishika Dasari" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Guntur" />
    </TableRow>
</TableLayout>
```

Once we are done with creation of layout, we need to load the XML layout resource from our activity **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.tablelayout** path and write the code like as shown below.

MainActivity.java

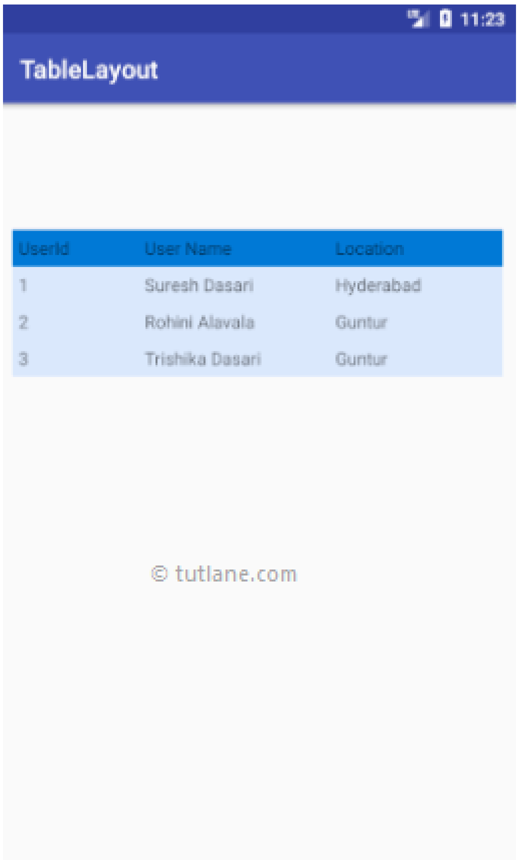
```
package com.tutlane.linearlayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

If you observe above code, we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Generally, during the launch of our activity, **onCreate()** callback method will be called by android framework to get the required layout for an activity.

Output of Android TableLayout Example

When we run above example using the android virtual device (AVD) we will get a result like as shown below.



UserId	User Name	Location
1	Suresh Dasari	Hyderabad
2	Rohini Alavala	Guntur
3	Trishika Dasari	Guntur

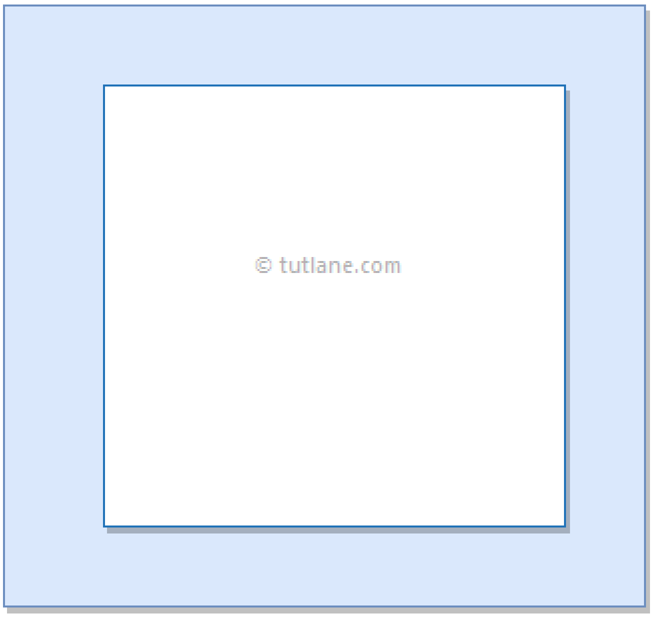
This is how we can use the table layout in android applications based on our requirements.

Android FrameLayout with Examples

In android, **Framelayout** is a **ViewGroup** subclass that is used to specify the position of **View** instances it contains on the top of each other to display only single **View** inside the FrameLayout.

In simple manner, we can say FrameLayout is designed to block out an area on the screen to display a single item.

Following is the pictorial representation of frame layout in android applications.



In android, **FrameLayout** will act as a placeholder on the screen and it is used to hold a single child view.

In FrameLayout, the child views are added in a stack and the most recently added child will show on the top. We can add multiple children views to FrameLayout and control their position by using gravity attributes in FrameLayout.

Android FrameLayout Example

Following is the example of creating a **FrameLayout** with different controls in android application.

Create a new android application using android studio and give names as **FrameLayout**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?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"
   android:orientation="vertical">
   <ImageView
      android:id="@+id/imgvw1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:scaleType="centerCrop"
      android:src="@drawable/flimg" />
   <TextView
      android:id="@+id/txtvw1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="40dp"
      android:background="#4C374A"
      android:padding="10dp"
      android:text="Grand Palace, Bangkok"
      android:textColor="#FFFFFF"
      android:textSize="20sp" />
   <TextView
```

```
        android:id="@+id/txtvw2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:background="#AA000000"
        android:padding="10dp"
        android:text="21/Aug/2017"
        android:textColor="#FFFFFF"
        android:textSize="18sp" />
</FrameLayout>
```

If you observe above code we used **ImageView** to show the image (**flimg**) from drawable folder in framelayout. So add your image to **drawable** folder and replace **@drawable/flimg** path with your image path.

Once we are done with the creation of layout, we need to load the XML layout resource from our activity **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.framelayout** path and write the code like as shown below.

MainActivity.java

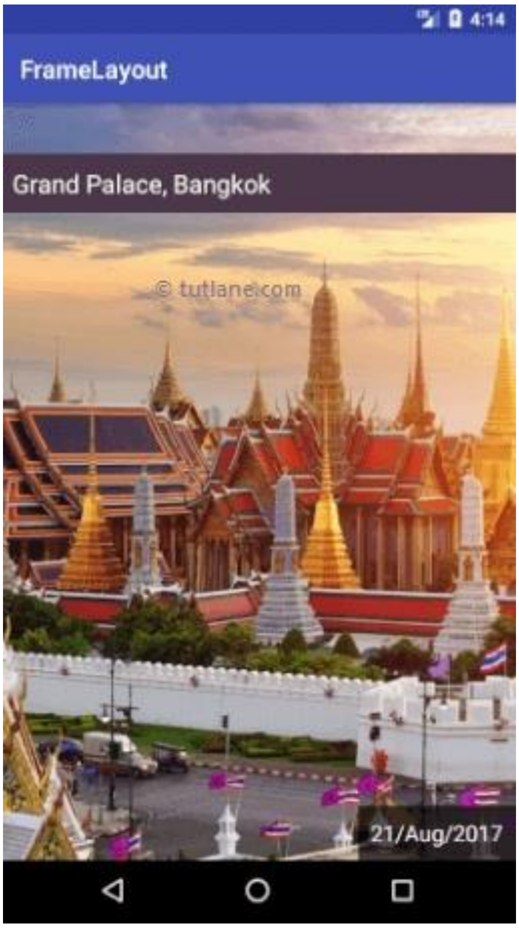
```
package com.tutlane.linearlayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

If you observe above code, we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Generally, during the launch of our activity, the **onCreate()** callback method will be called by the android framework to get the required layout for an activity.

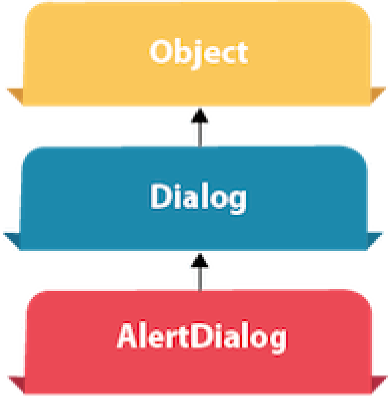
Output of Android FrameLayout Example

When we run the above example using the android virtual device (AVD) we will get a result like as shown below.



This is how we can use frame layout in android applications based on our requirements.

Android AlertDialog Example

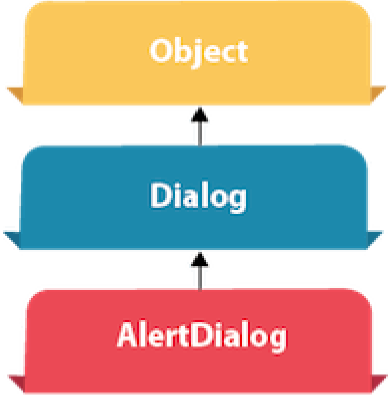


Android AlertDialog can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

Android AlertDialog is composed of three regions: title, content area and action buttons.

Android AlertDialog is the subclass of Dialog class.

Android AlertDialog Example



Android AlertDialog can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

Android AlertDialog is composed of three regions: title, content area and action buttons.

Android AlertDialog is the subclass of Dialog class.

Methods of AlertDialog class

Method	Description
public AlertDialog.Builder setTitle(CharSequence)	This method is used to set the title of AlertDialog.
public AlertDialog.Builder setMessage(CharSequence)	This method is used to set the message for AlertDialog.
public AlertDialog.Builder setIcon(int)	This method is used to set the icon over AlertDialog.

Android AlertDialog Example

Let's see a simple example of android alert dialog.

activity_main.xml

You can have multiple components, here we are having only a textview.

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<android.support.constraint.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="example.javatpoint.com.alertdialog.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Close app"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

strings.xml

Optionally, you can store the dialog message and title in the strings.xml file.

File: strings.xml

```
<resources>
    <string name="app_name">AlertDialog</string>
    <string name="dialog_message">Welcome to Alert Dialog</string>
    <string name="dialog_title">Javatpoint Alert Dialog</string>
</resources>
```

Activity class

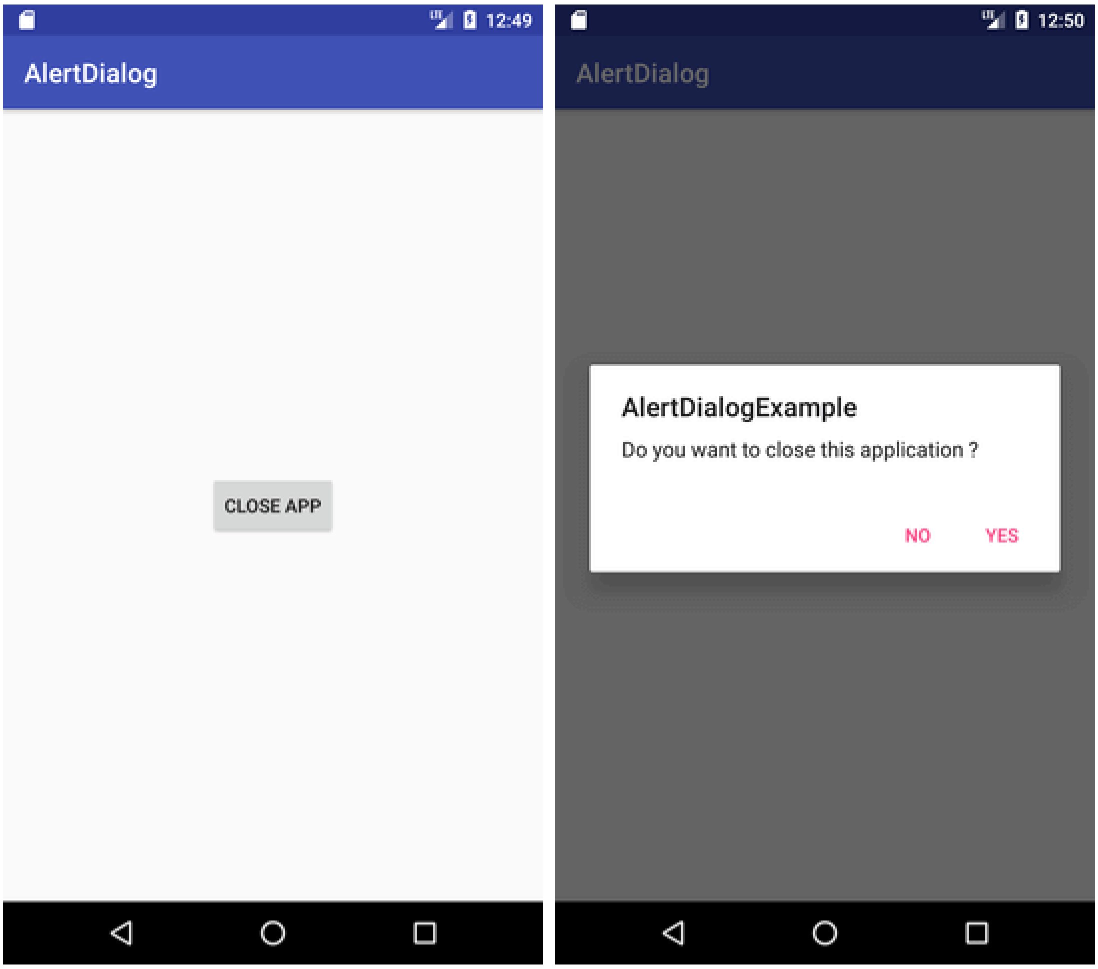
Let's write the code to create and show the AlertDialog.

File: MainActivity.java

```
package example.javatpoint.com.alertdialog;

import android.content.DialogInterface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

```java
import android.view.View;
import android.widget.Button;
import android.app.AlertDialog;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    Button closeButton;
    AlertDialog.Builder builder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        closeButton = (Button) findViewById(R.id.button);
        builder = new AlertDialog.Builder(this);
        closeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                //Uncomment the below code to Set the message and title from the strings.xml file

                builder.setMessage(R.string.dialog_message) .setTitle(R.string.dialog_title);

                //Setting message manually and performing action on button click
                builder.setMessage("Do you want to close this application ?")
                        .setCancelable(false)
                        .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                finish();
                         Toast.makeText(getApplicationContext(),"you choose yes action for alertbox",

                                Toast.LENGTH_SHORT).show();
                            }
                        })
                        .setNegativeButton("No", new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                //  Action for 'NO' Button
                                dialog.cancel();
                          Toast.makeText(getApplicationContext(),"you choose no action for alertbox",

                                Toast.LENGTH_SHORT).show();
                            }
                        });
                //Creating dialog box
                AlertDialog alert = builder.create();
                //Setting the title manually
                alert.setTitle("AlertDialogExample");
                alert.show();
            }
        });
    }
}
```

Output:



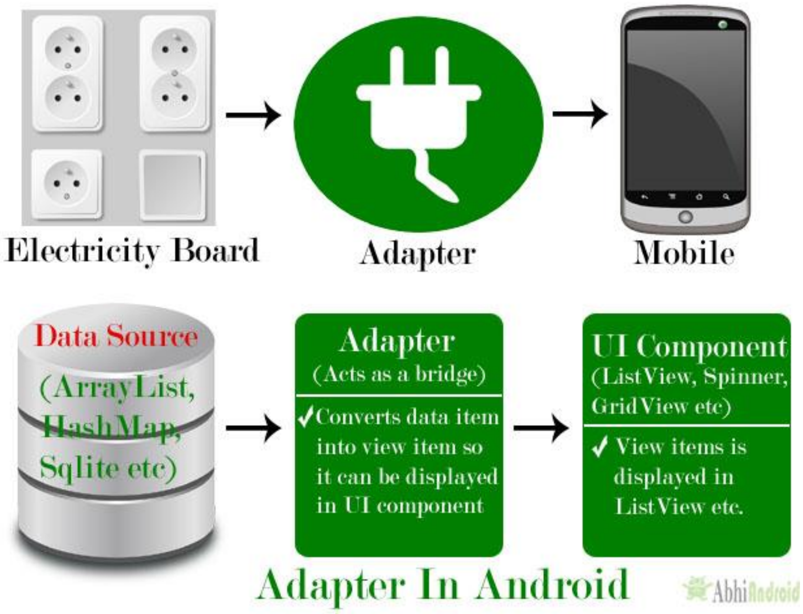
AlertDialogExample

Do you want to close this application ?

Adapter

Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner etc. For more customization in Views we uses the base adapter or custom adapters.

To fill data in a list or a grid we need to implement Adapter. Adapters acts like a bridge between UI component and data source. Here data source is the source from where we get the data and UI components are list or grid items in which we want to display that data.


Electricity Board
Adapter
Mobile
Data Source
(ArrayList, HashMap, Sqlite etc)
Adapter
(Acts as a bridge)
✓Converts data item into view item so it can be displayed in UI component
UI Component
(ListView, Spinner, GridView etc)
✓ View items is displayed in ListView etc.
Adapter In Android
AbhiAndroid

Adapters In Android:

There are the some commonly used Adapter in Android used to fill the data in the UI components.

1. BaseAdapter – It is parent adapter for all other adapters
2. ArrayAdapter – It is used whenever we have a list of single items which is backed by an array
3. Custom ArrayAdapter – It is used whenever we need to display a custom list
4. SimpleAdapter – It is an easy adapter to map static data to views defined in your XML file
5. Custom SimpleAdapter – It is used whenever we need to display a customized list and needed to access the child items of the list or grid

Topic 5: Drawable Resources:

- It includes
    - shapes
    - transformative
    - composite
- it defines how to use these resources to create UI that are independent of screen size and resolution.

1) Shapes:

- define simple primitive shapes by defining their dimensions, background, and storke/ outline using the <Shape> tag.

- Android currently supports the following shape types as values for the shape attribute.

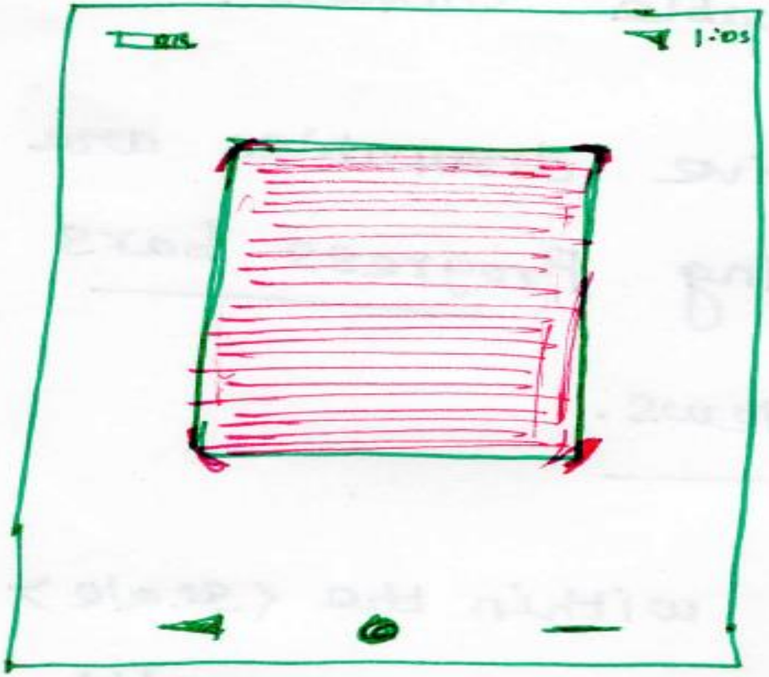
  - oval
  - rectangle
  - ring

Program:

Rectanglular shape drawable with a Solid fill, rounded edges , 10 dP out and 10 dP of Padding around each .

```
< ? xml Version = "1.0" encoding = "Utf - 8" ? >
< shape xmlns : androd = "http://schemas.and
            com/ apk/ res/ android"
       android : shape = "rectangle" >

  <solid
       android: color = "#f0600000"/>

  <stroke
       android: width = "10dP"
       android: color = "#00FF00"/>
```

```
<corners
        android : radius = "15dp" />

<Padding
        android : left = "10dp"
        android : top = "10dp"
        android : right = "10dp"
        android : bottom = "10dp" />

</shape>
```



where · <Solid> => define solid background color

<Stroke> => Specify an outline for the Shapes using width and color.

<corners> => uses <radius> attribute to create a rounded rectang

<Padding> ⇒ to offset the positioning

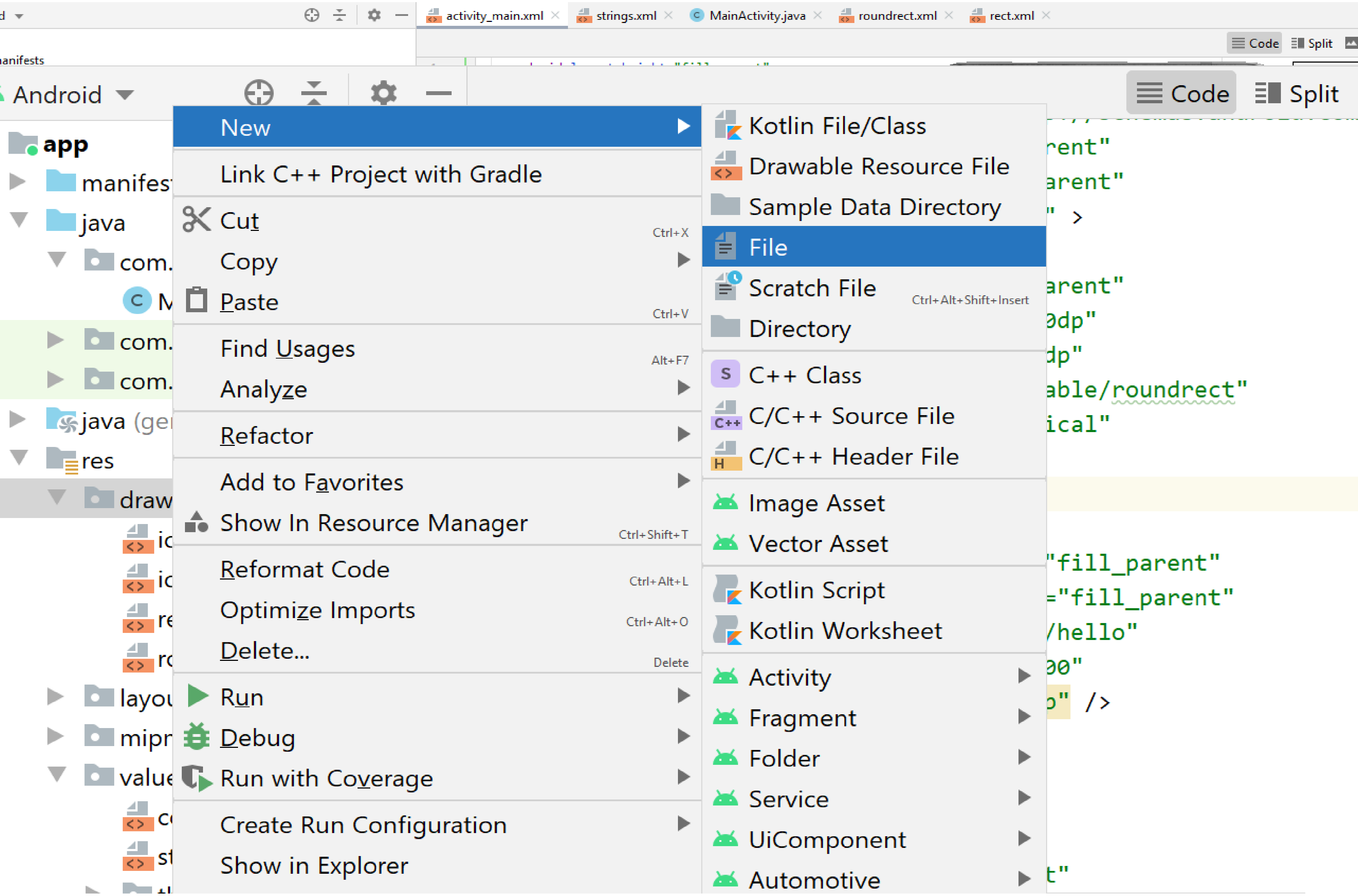
your shape on the canvas.

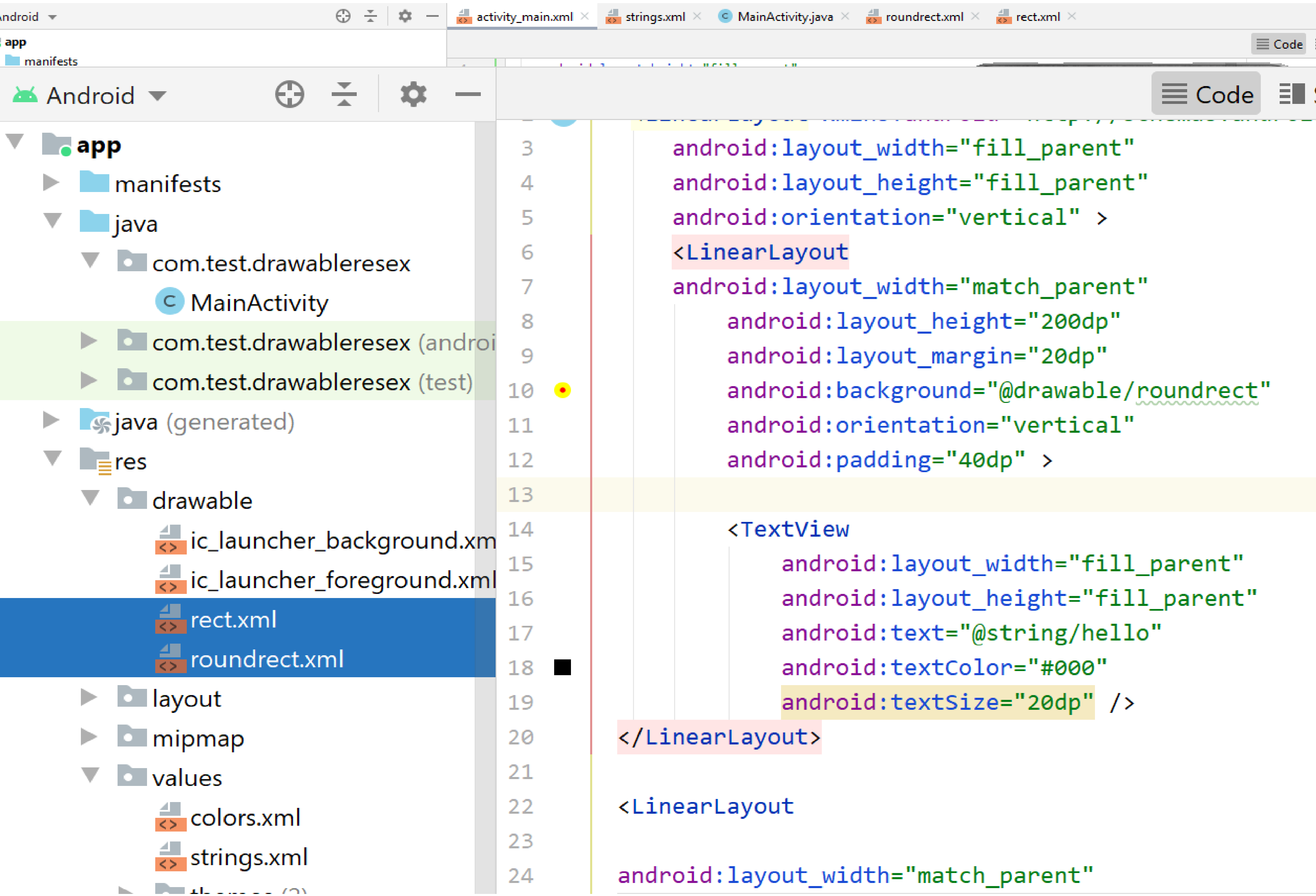
⇓

Here this is rectan

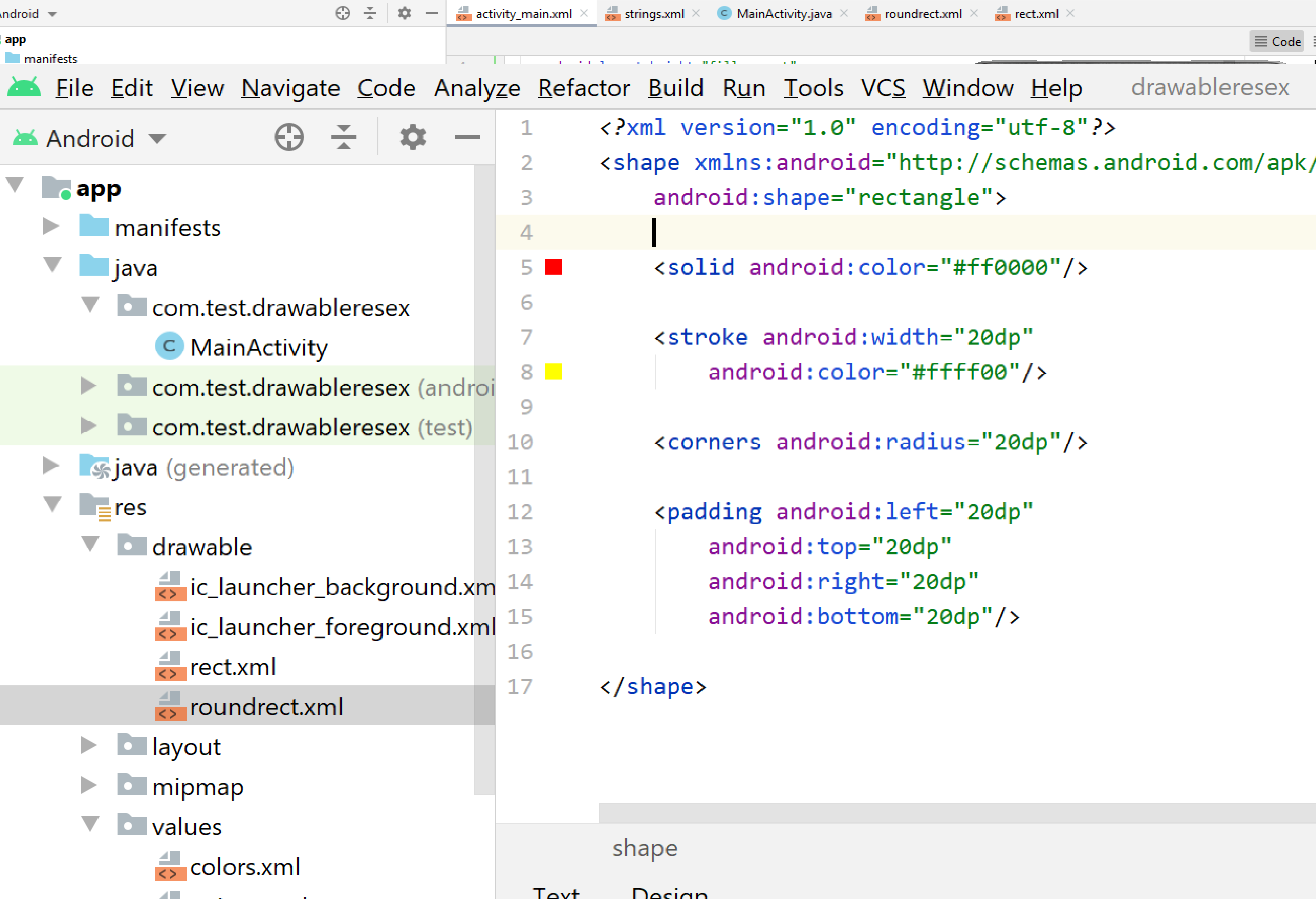
2) Transformative:

- Scale and rotate existing drawable resources using the ~~apply~~ aptly named

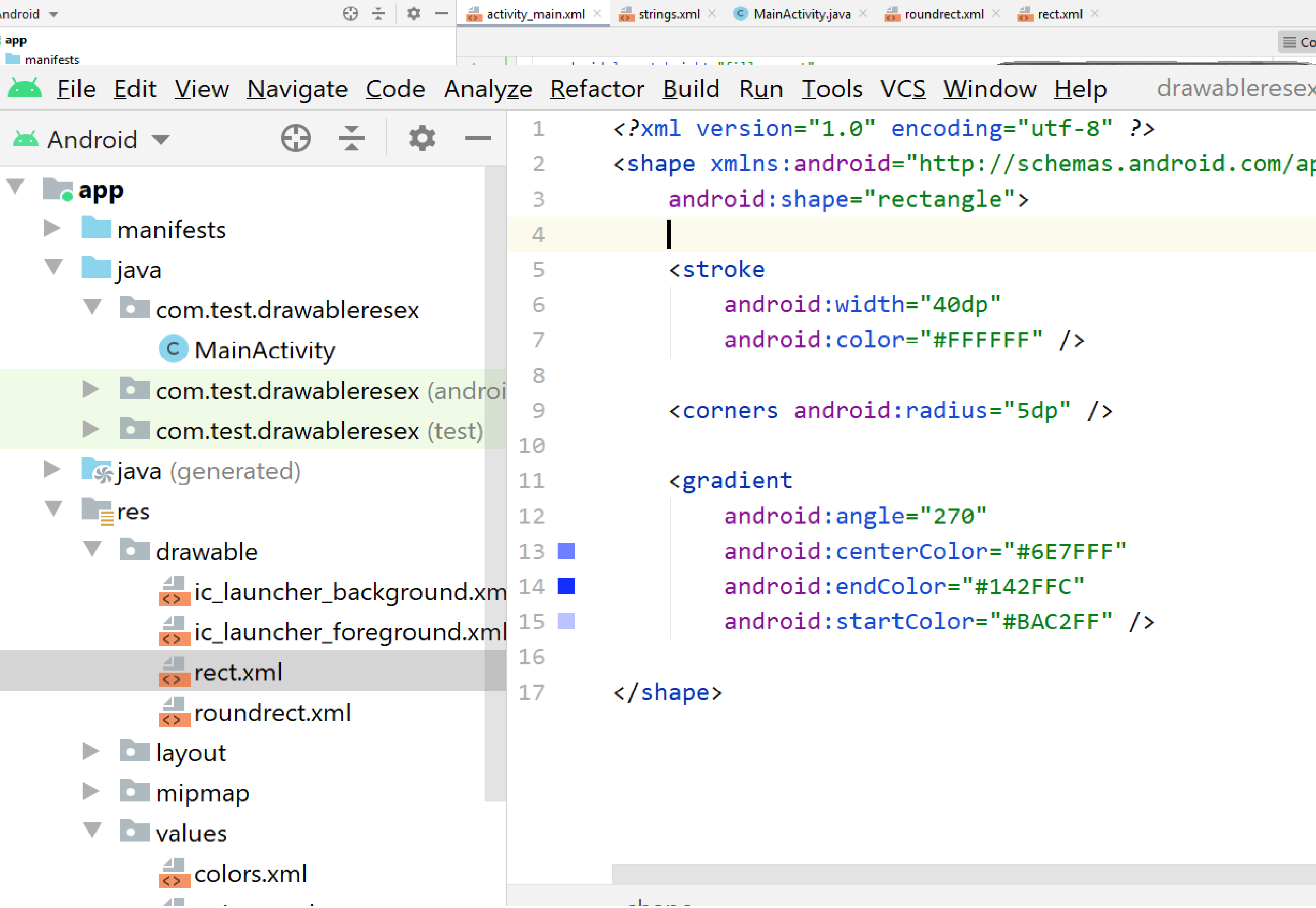

activity_main.xml
strings.xml
MainActivity.java
roundrect.xml
rect.xml
Code
Split
Android
app
manifests
java
com.
com.
com.
java (ge
res
draw
layou
mipr
value
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Show In Resource Manager Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run
Debug
Run with Coverage
Create Run Configuration
Show in Explorer
Kotlin File/Class
Drawable Resource File
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Kotlin Worksheet
Activity
Fragment
Folder
Service
UiComponent
Automotive



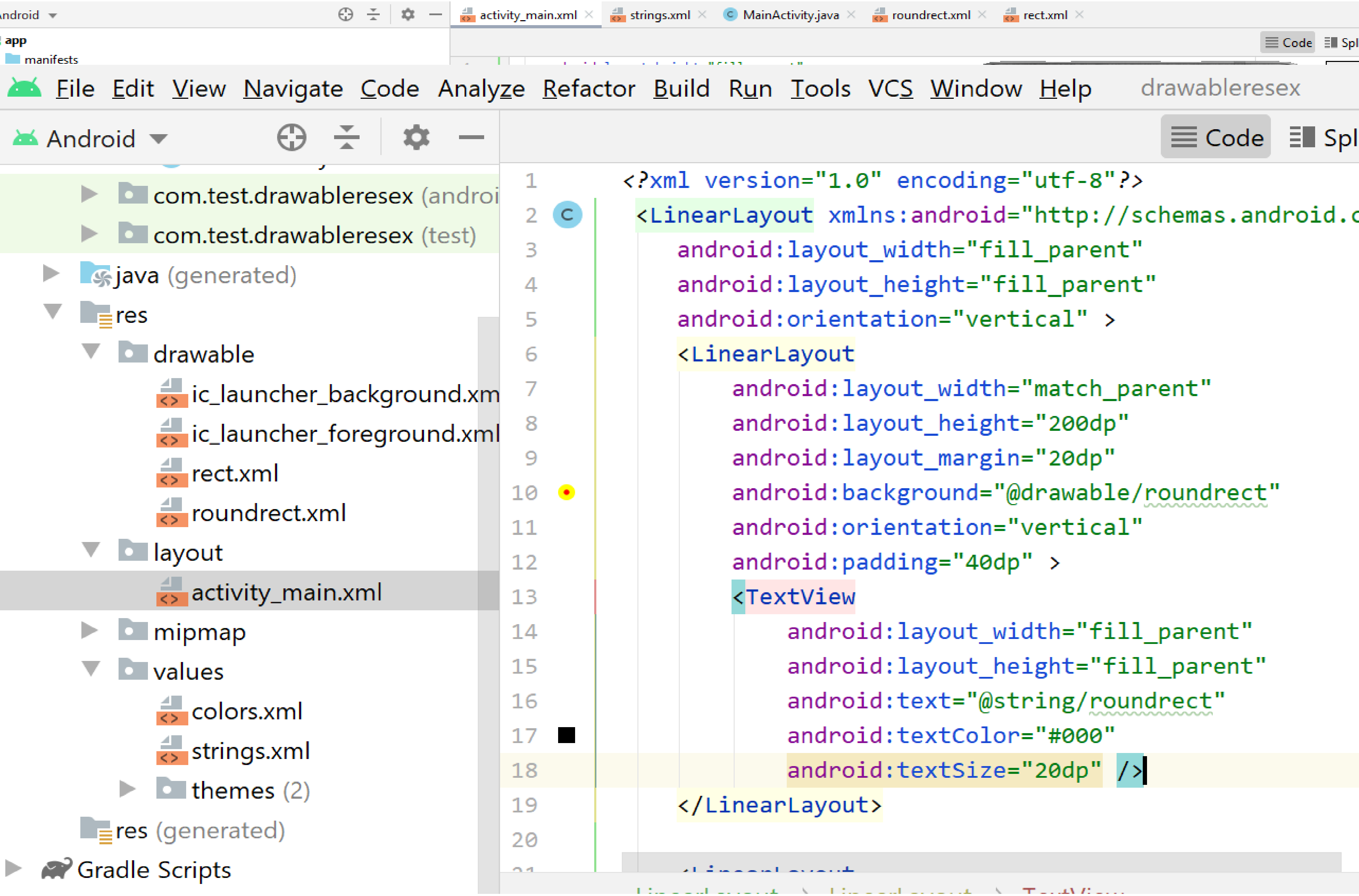
```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout
    android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_margin="20dp"
        android:background="@drawable/roundrect"
        android:orientation="vertical"
        android:padding="40dp" >

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="@string/hello"
            android:textColor="#000"
            android:textSize="20dp" />
</LinearLayout>

<LinearLayout

android:layout_width="match_parent"
```



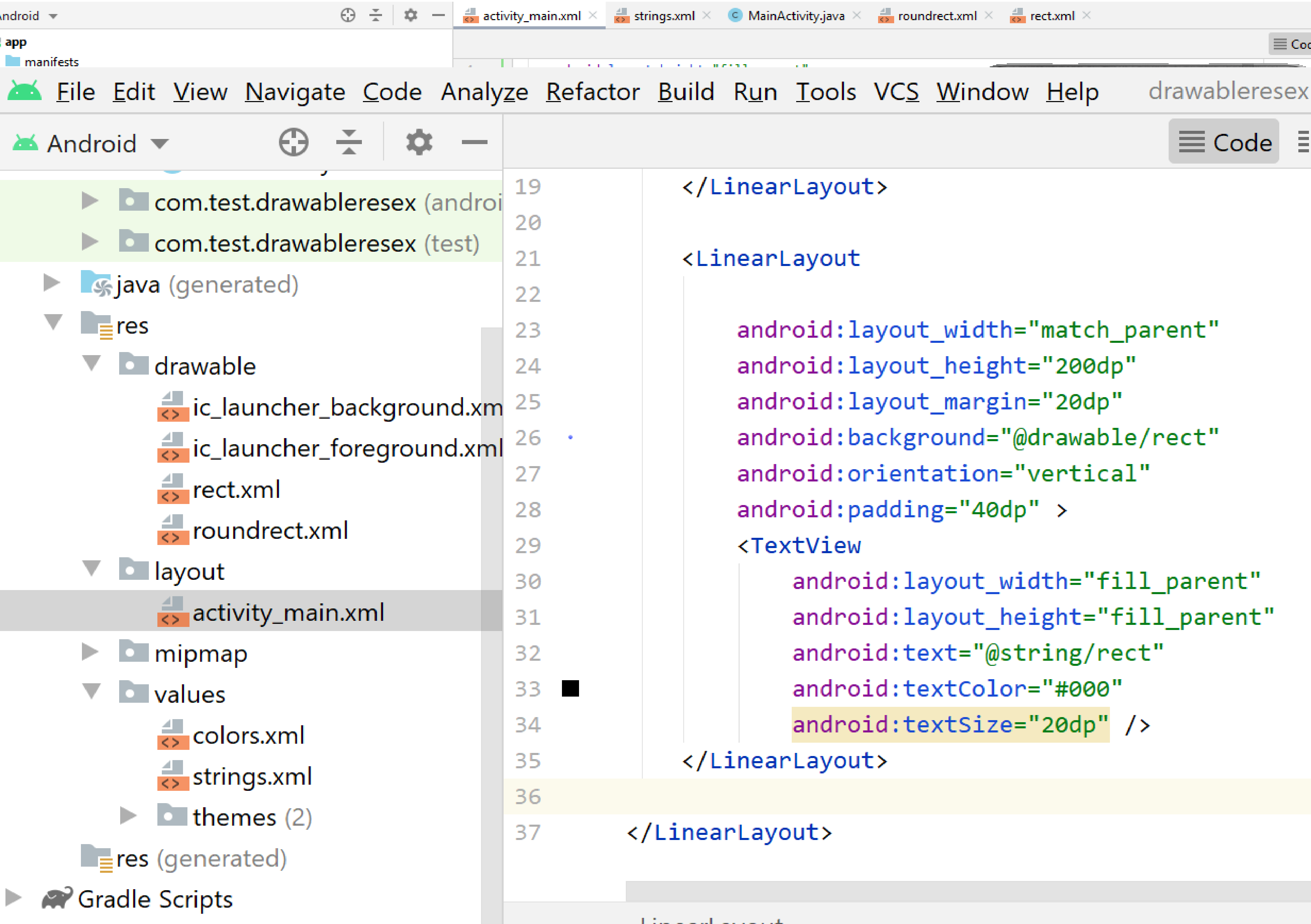
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/
    android:shape="rectangle">

    <solid android:color="#ff0000"/>

    <stroke android:width="20dp"
        android:color="#ffff00"/>

    <corners android:radius="20dp"/>

    <padding android:left="20dp"
        android:top="20dp"
        android:right="20dp"
        android:bottom="20dp"/>

</shape>
```



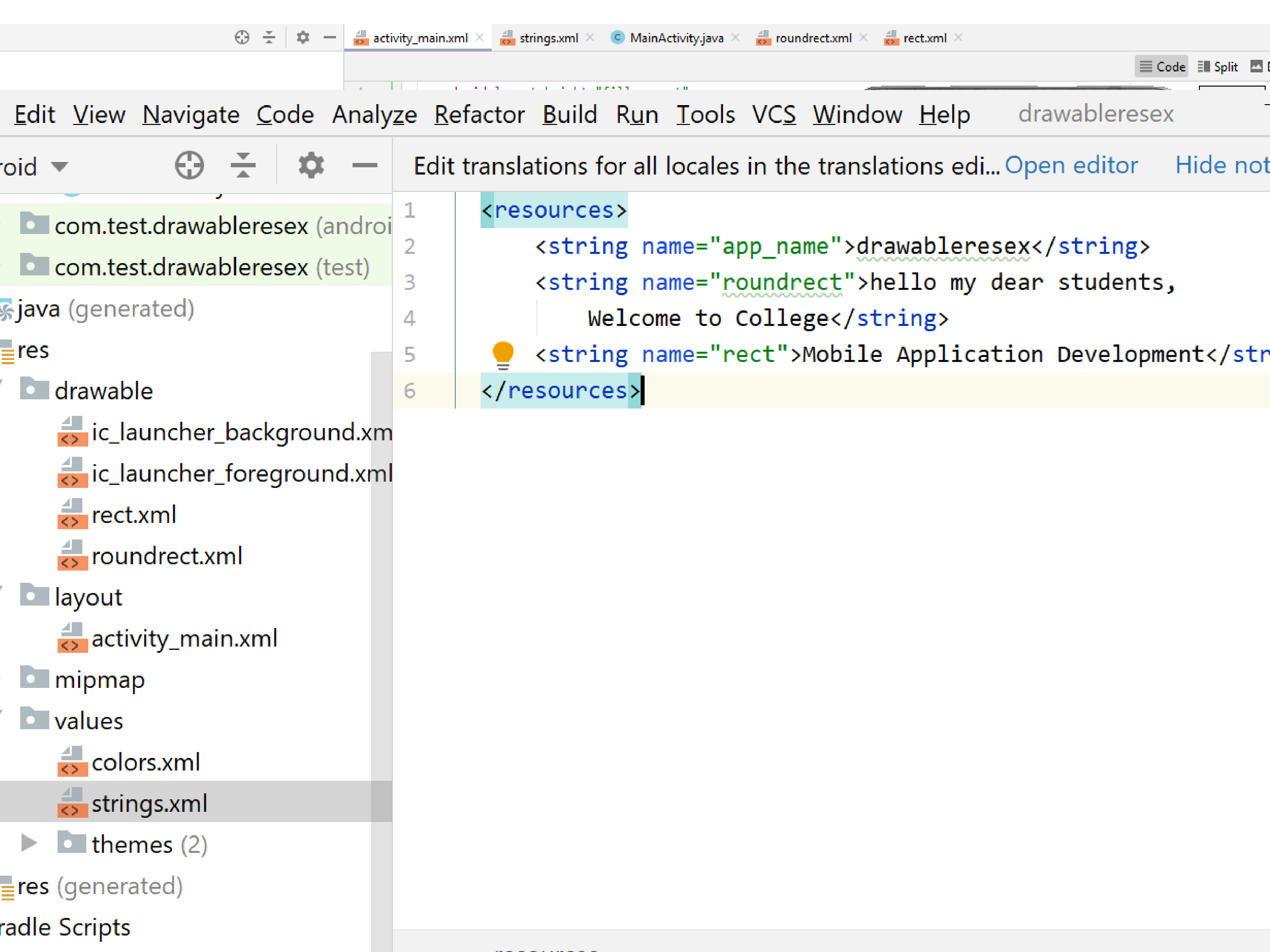
```
<?xml version="1.0" encoding="utf-8" ?>
<shape xmlns:android="http://schemas.android.com/ap
    android:shape="rectangle">

    <stroke
        android:width="40dp"
        android:color="#FFFFFF" />

    <corners android:radius="5dp" />

    <gradient
        android:angle="270"
        android:centerColor="#6E7FFF"
        android:endColor="#142FFC"
        android:startColor="#BAC2FF" />

</shape>
```



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.c
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_margin="20dp"
        android:background="@drawable/roundrect"
        android:orientation="vertical"
        android:padding="40dp" >
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="@string/roundrect"
            android:textColor="#000"
            android:textSize="20dp" />
    </LinearLayout>
```

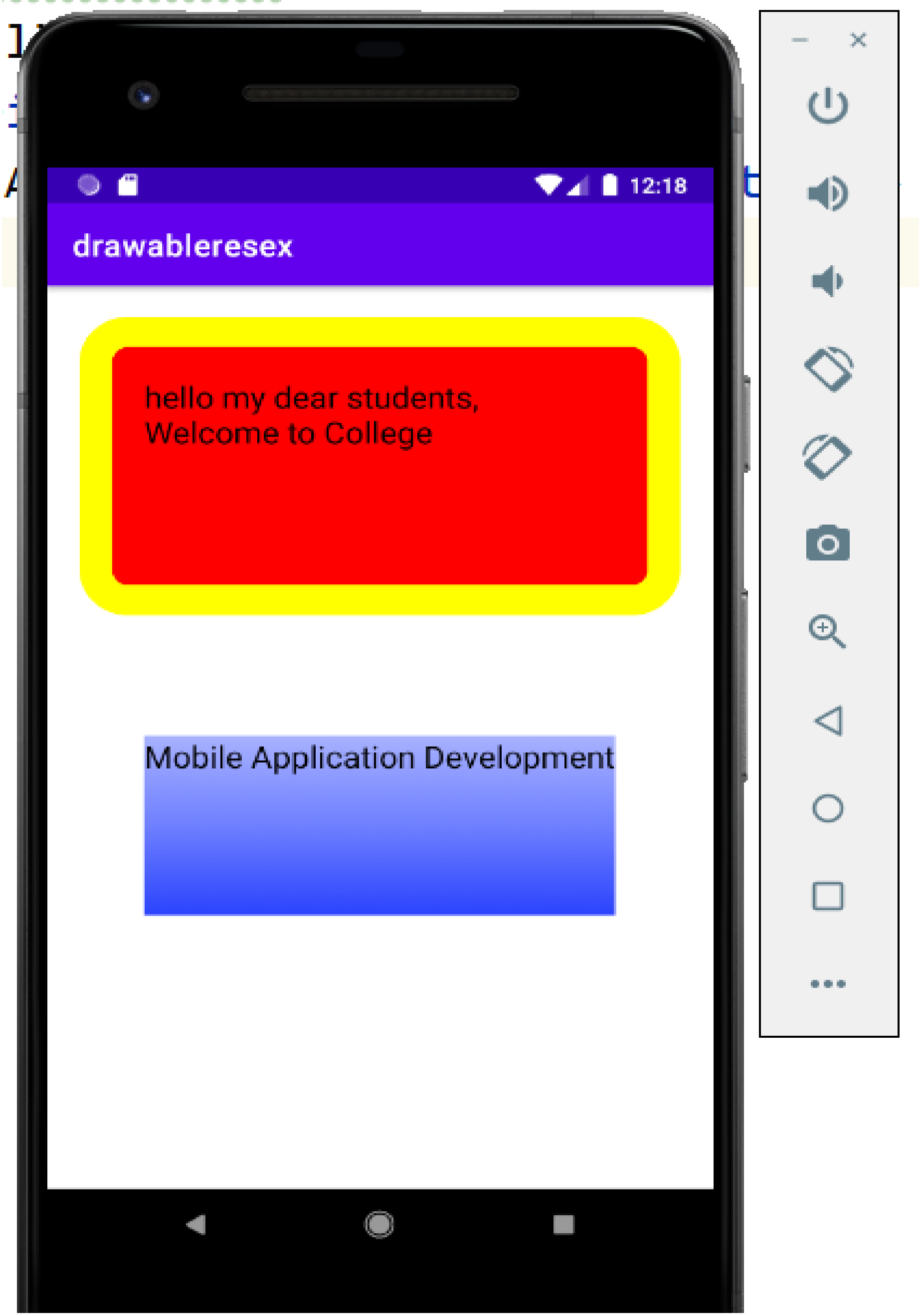


```
    </LinearLayout>

    <LinearLayout

        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_margin="20dp"
        android:background="@drawable/rect"
        android:orientation="vertical"
        android:padding="40dp" >
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="@string/rect"
            android:textColor="#000"
            android:textSize="20dp" />
    </LinearLayout>

</LinearLayout>
```



Edit translations for all locales in the translations edi...

```
<resources>
    <string name="app_name">drawableresex</string>
    <string name="roundrect">hello my dear students,
        Welcome to College</string>
    <string name="rect">Mobile Application Development</str
</resources>
```



hello my dear students,
Welcome to College

Mobile Application Development

- Scale Drawable
- Rotate Drawable classes.

- This transformative drawables are Particularly useful for creating Progress bars or animating Views.

Scale Drawable => within the <scale> tag, use the scale Height and scale width attributes to define the target height and width relative to the bounding box of the original drawable.

original

- use the scalegravity attribute to contro
the anchor point of the scaled image.

Rotate Drawable => within the <rotate> tag,
use fromDegrees and toDegrees to
define the start and end rotation angle
around a pivot point.

- Define the Pivot using the PivotX and PivotY attributes, specifying a percentage of the Drawable's width and height, respectively, using nn% notation.

To apply the scaling and rotation at run time, use the setLevel method on the view object hosting the drawable to move between the Start and finish values. (0 to 10,000)

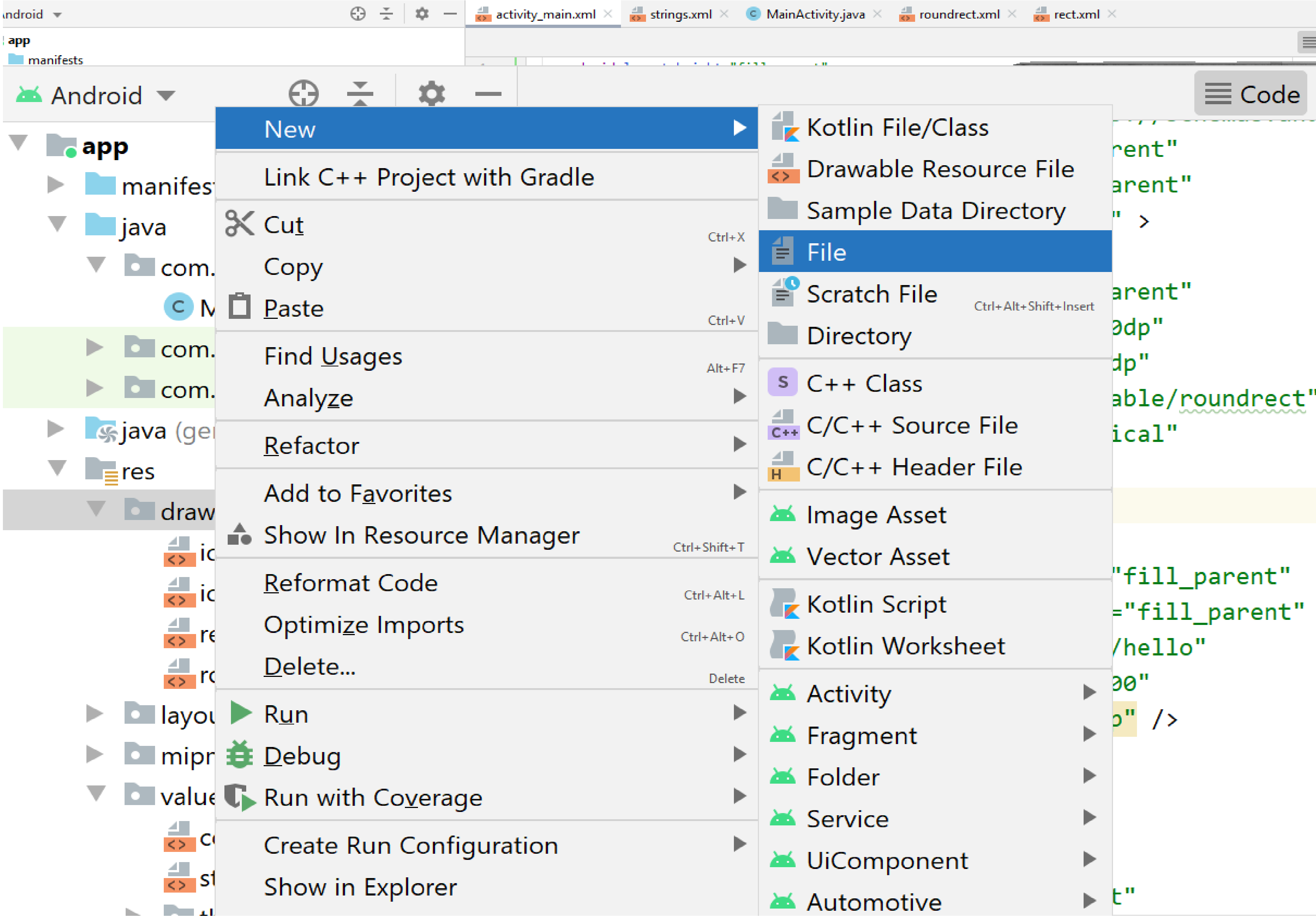
Level 0 => represents the start angle (smallest scale result)

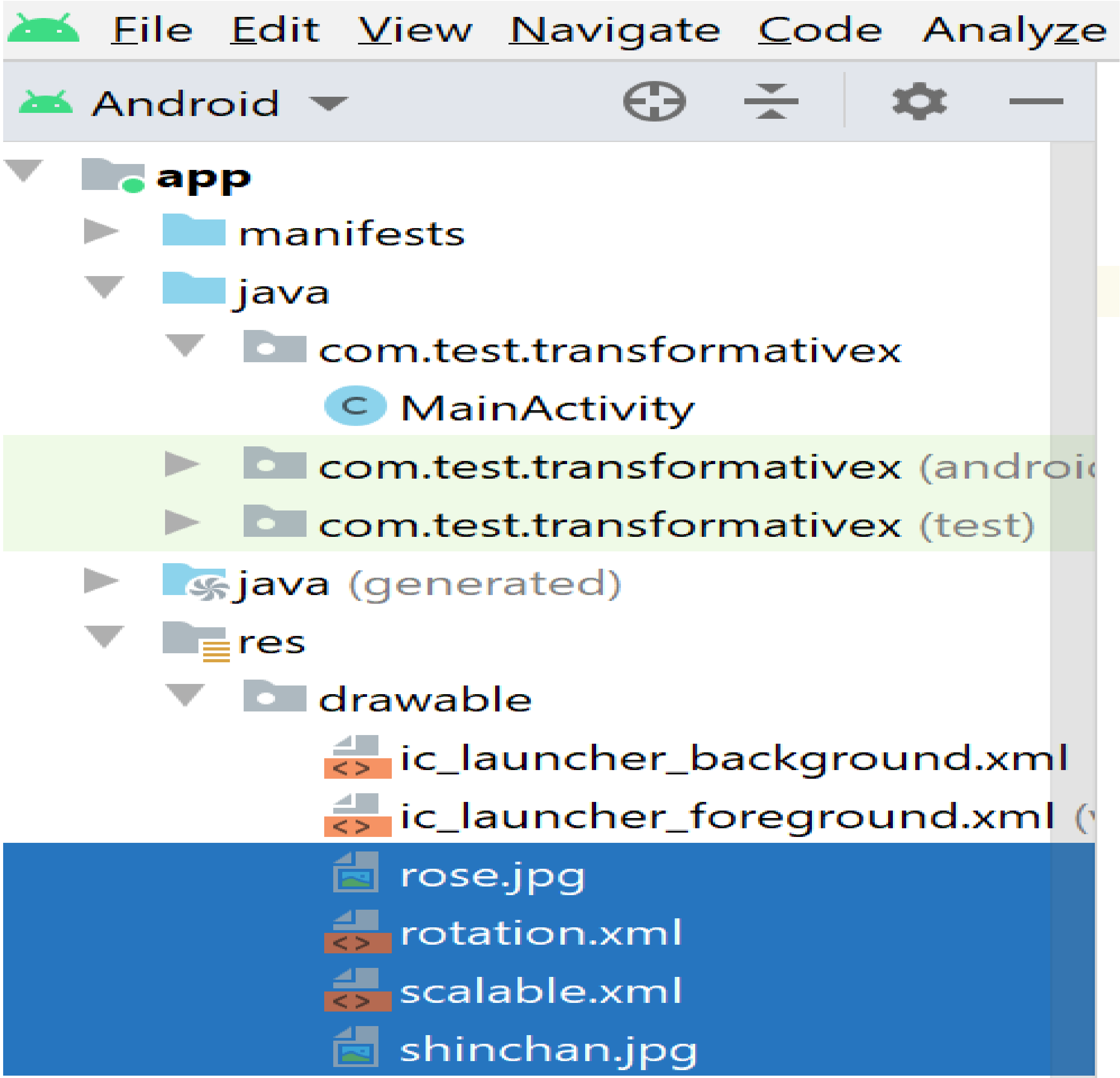
Level 10,000 => represents the end of the transformation (the finish angle highest scaling).

```
<!-- Rotation Drawable Resource -->
 <?xml version = "1.0" encoding = "utf-8"?>
 <rotate xmlns : android = "http :// schemas .
          android. com / apk / res/android"
   android : drawable = @ drawable/icon"
   android : fromDegrees = "0"
   android : toDegrees = "90"
```

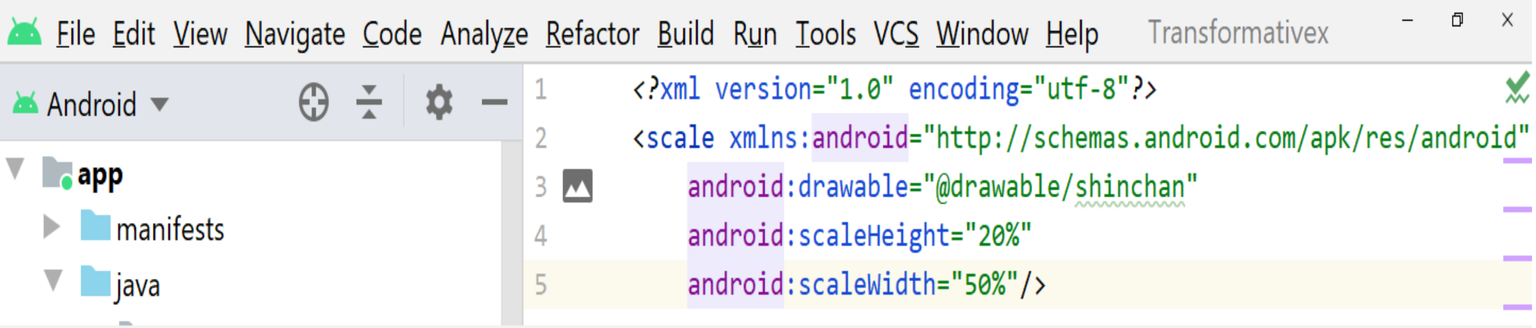
```
android: pivotX = "50%"
android: pivotY = "50%" />

<!-- Scale Drawable Resource -->

  <?xml version="1.0" encoding = "utf-8
  <rotate xmlns:android = " http://schema
          android.com/ apk / res/ android"
     android : drawable = "@drawable/ico
     android : scaleHeight = "100%"
     android : scaleWidth = "100%"
```

```
// Apply rotation and scale drawable transformati
    in code.

  ImageView rotatingImage = (ImageView) findView
              (R.id.RotatingImageView);
  ImageView ScalingImage = (ImageView) findView
              (R.id.ScalingImageView);
  // Rotate the image 50% of the way to its fir
     orientation.
     rotatingImage.setImageLevel(5000);
```

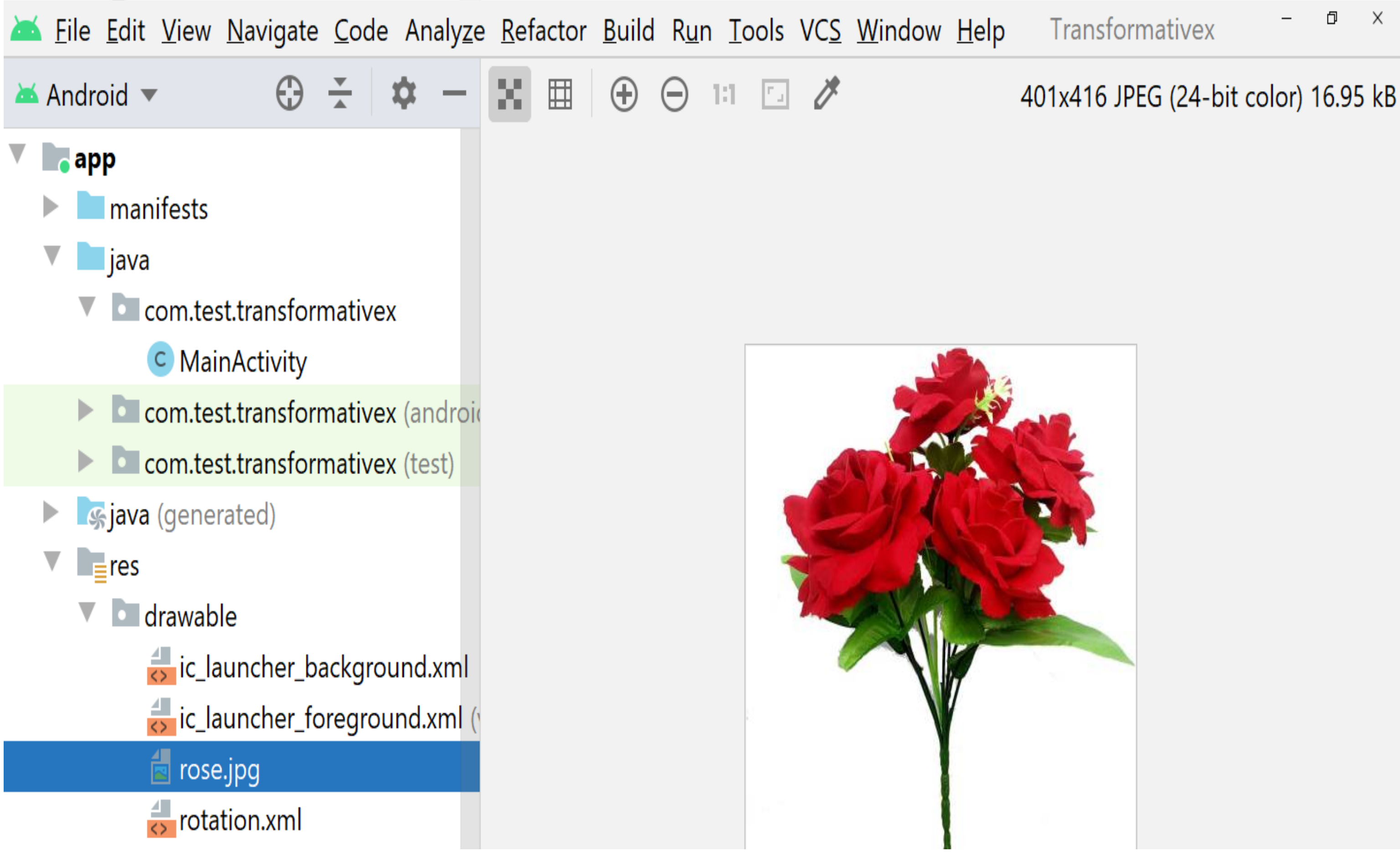

activity_main.xml
strings.xml
MainActivity.java
roundrect.xml
rect.xml
Android
Code
app
manifests
java
res
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Show In Resource Manager Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run
Debug
Run with Coverage
Create Run Configuration
Show in Explorer
Kotlin File/Class
Drawable Resource File
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Directory
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Kotlin Script
Kotlin Worksheet
Activity
Fragment
Folder
Service
UiComponent
Automotive



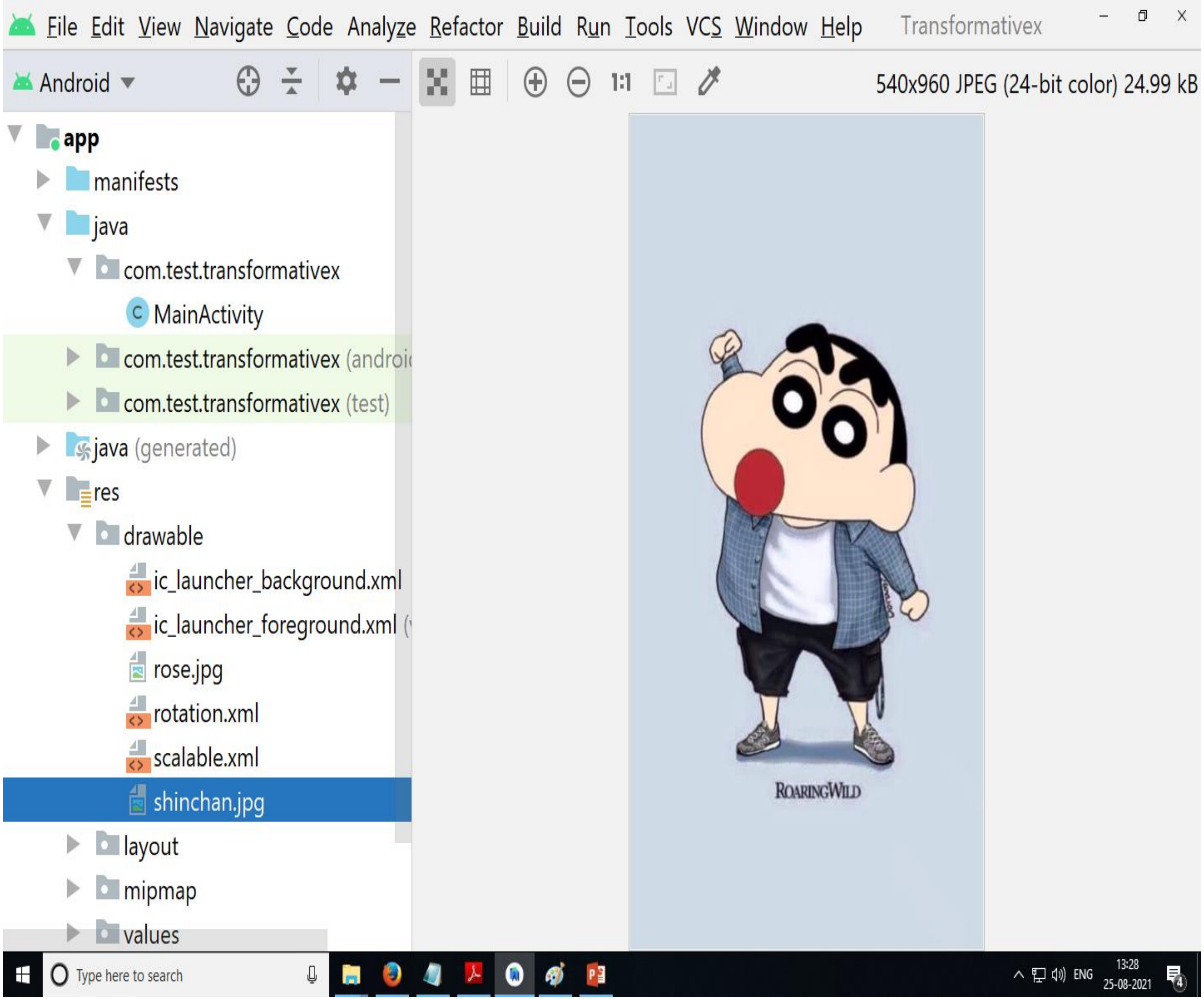
```
app
    manifests
    java
        com.test.transformativex
            MainActivity
        com.test.transformativex (androi
        com.test.transformativex (test)
    java (generated)
    res
        drawable
            ic_launcher_background.xml
            ic_launcher_foreground.xml (
            rose.jpg
            rotation.xml
            scalable.xml
            shinchan.jpg
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Transformativex
Android
app
manifests
java

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/shinchan"
    android:scaleHeight="20%"
    android:scaleWidth="50%"/>
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Transformativex
Android
app
manifests
java
com.test.transformativex
MainActivity
com.test.transformativex (androi
com.test.transformativex (test)
java (generated)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (
rose.jpg
rotation.xml

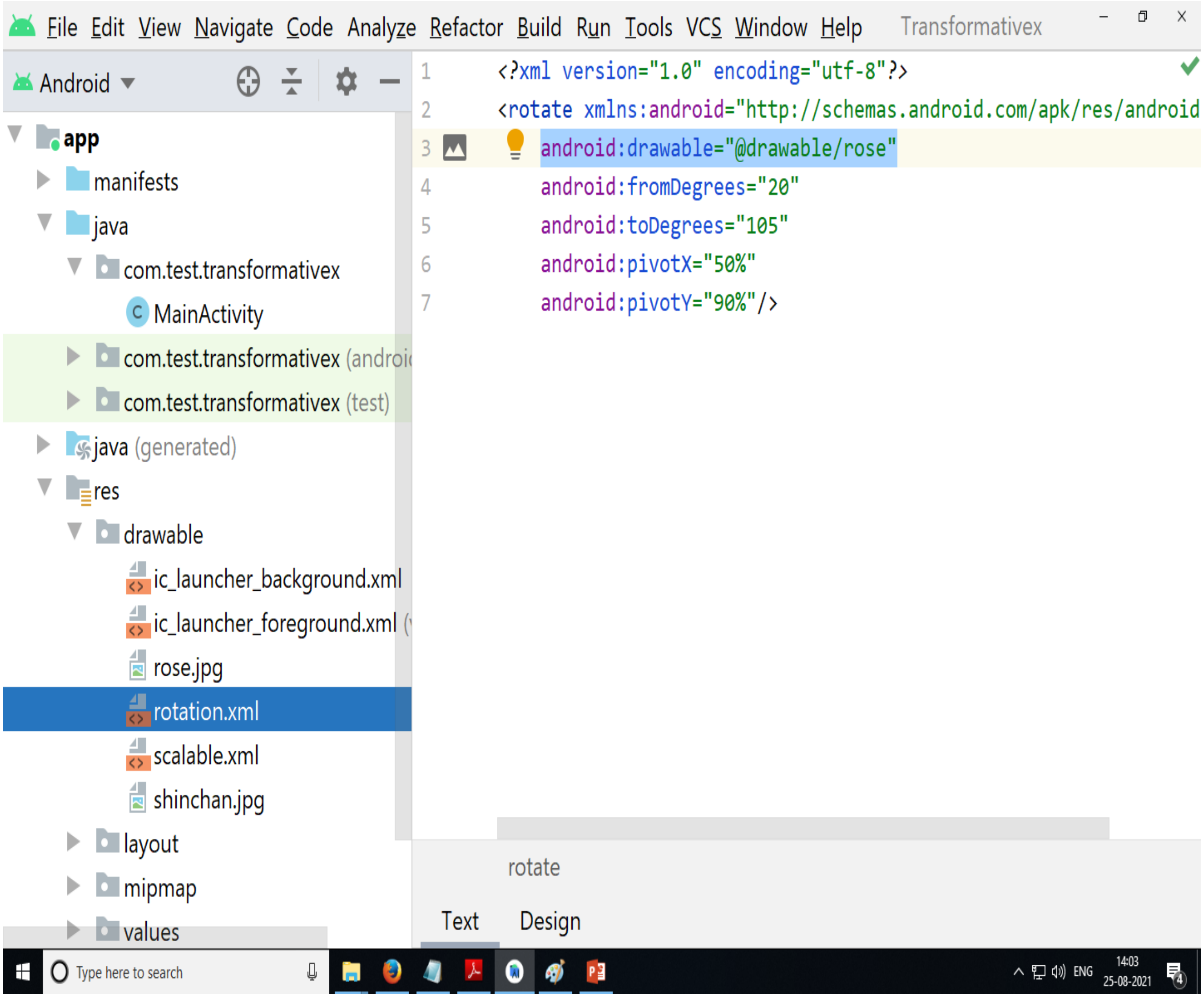
401x416 JPEG (24-bit color) 16.95 kB



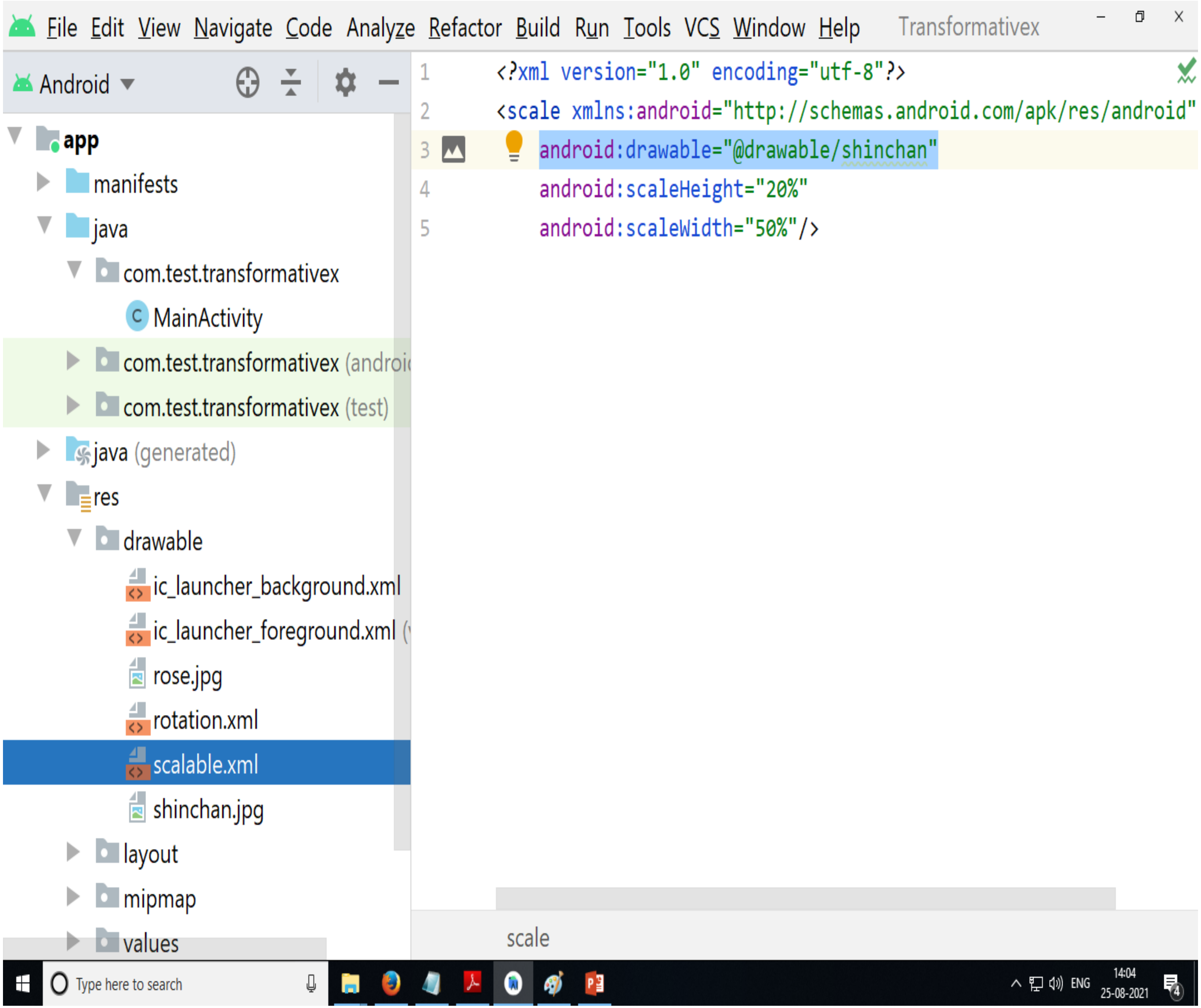
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Transformativex
Android
1:1
ROARINGWILD
Type here to search
ENG
13:28
25-08-2021

540x960 JPEG (24-bit color) 24.99 kB

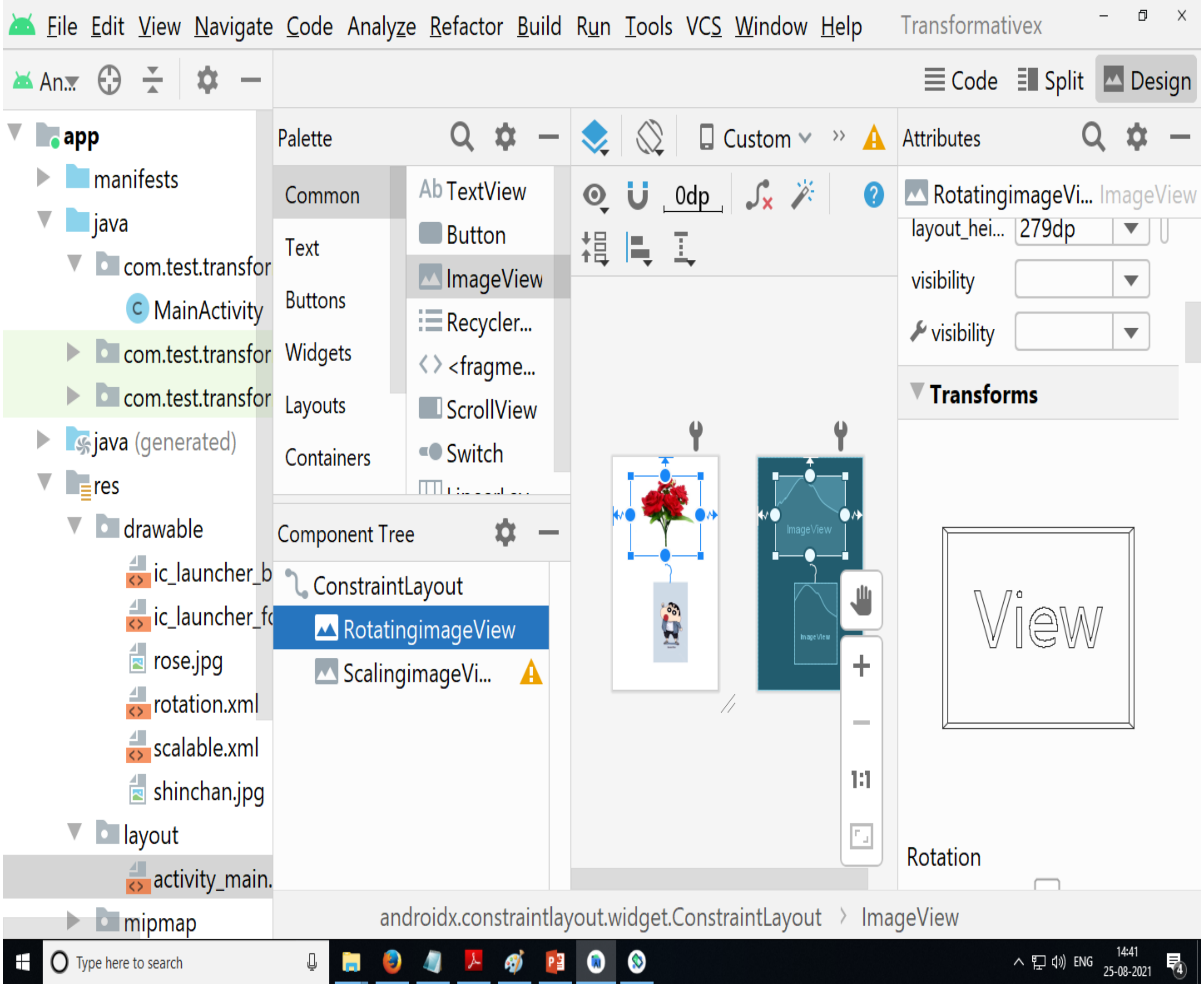
```
app
  manifests
  java
    com.test.transformativex
      MainActivity
    com.test.transformativex (androi
    com.test.transformativex (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (
      rose.jpg
      rotation.xml
      scalable.xml
      shinchan.jpg
    layout
    mipmap
    values
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android
    android:drawable="@drawable/rose"
    android:fromDegrees="20"
    android:toDegrees="105"
    android:pivotX="50%"
    android:pivotY="90%"/>
```



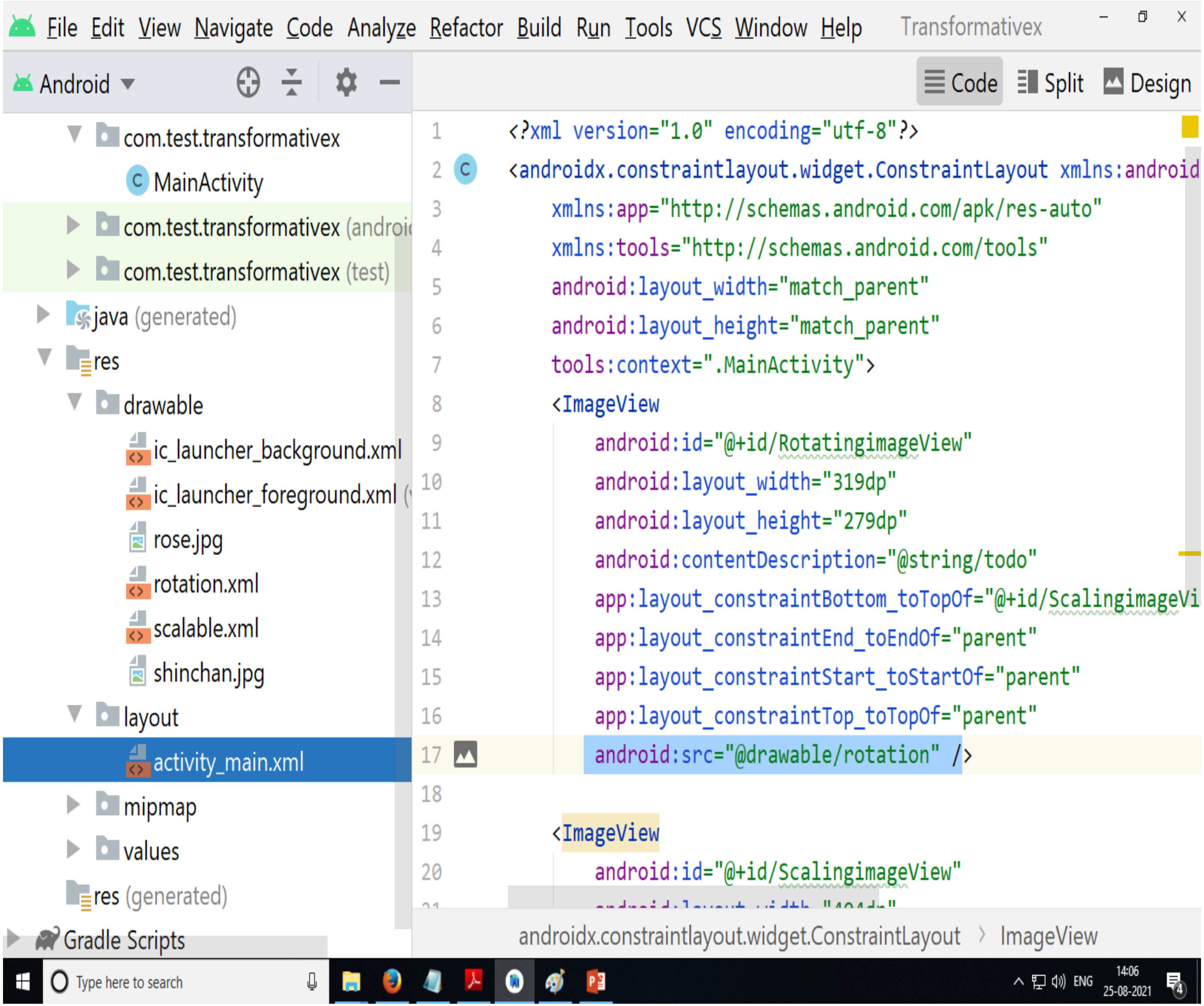
```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/shinchan"
    android:scaleHeight="20%"
    android:scaleWidth="50%"/>
```



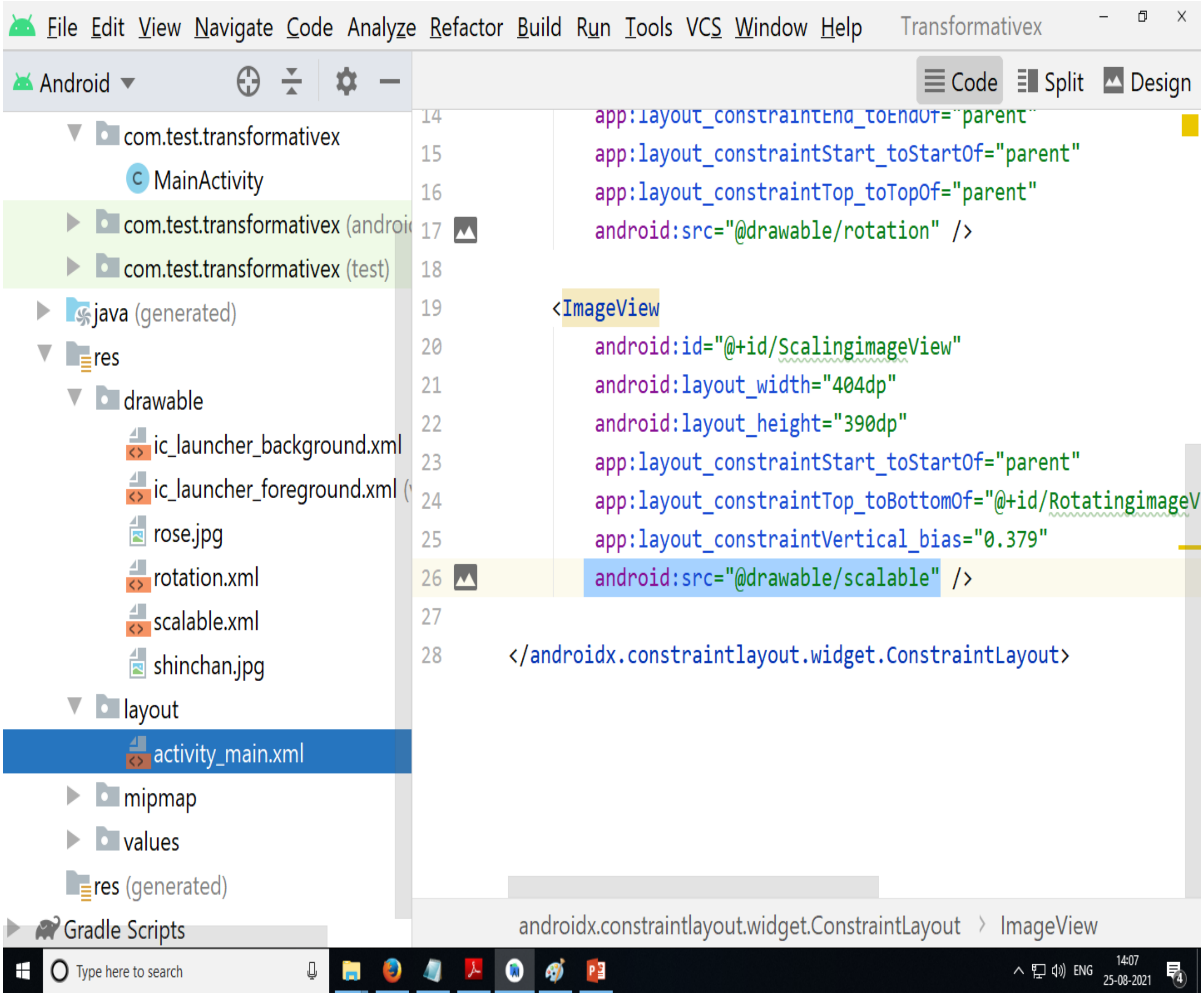
layout_hei... 279dp

visibility

visibility

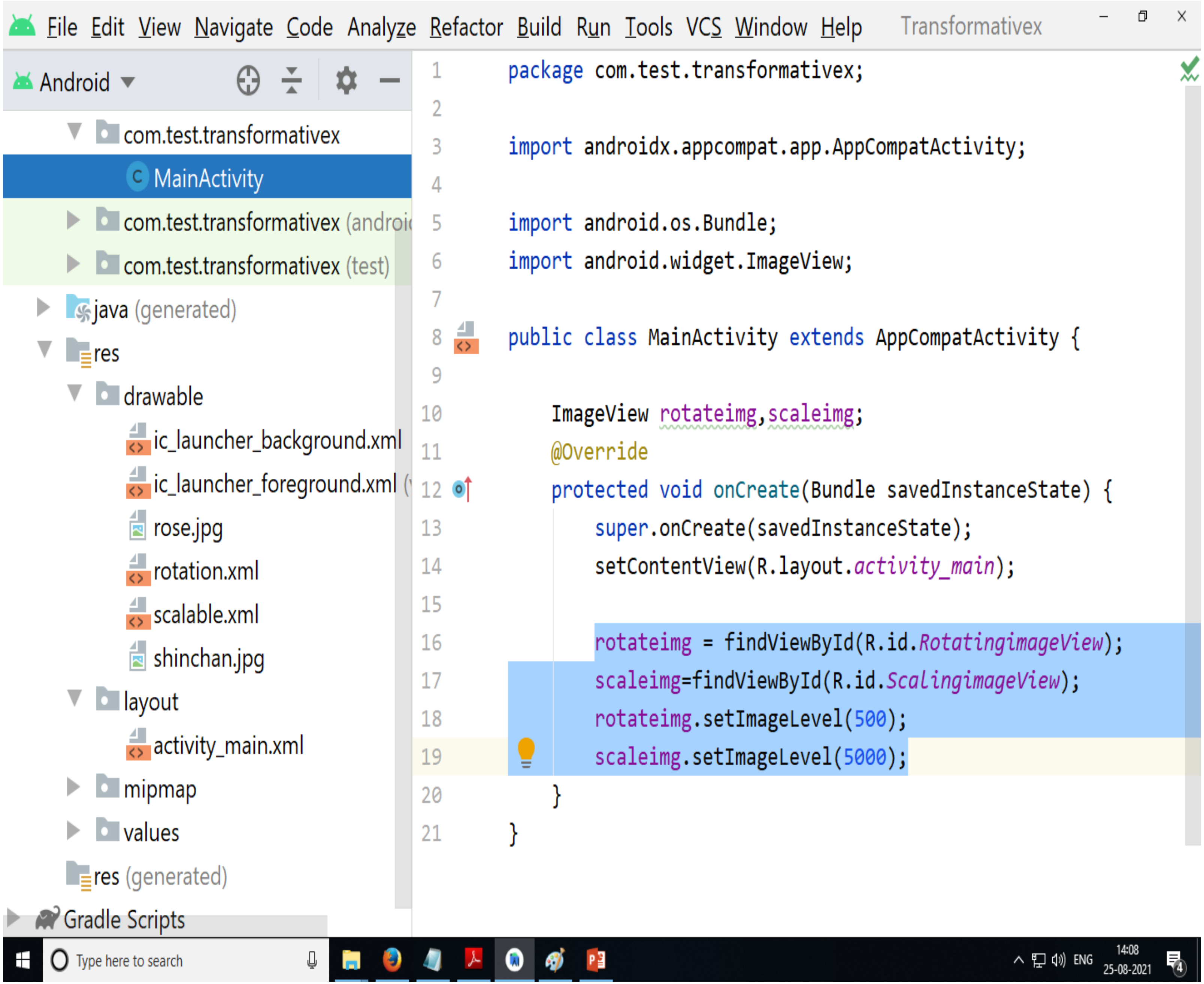

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Transformativex
Android
Code
Split
Design
com.test.transformativex
MainActivity
com.test.transformativex (androi
com.test.transformativex (test)
java (generated)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml (
rose.jpg
rotation.xml
scalable.xml
shinchan.jpg
layout
activity_main.xml
mipmap
values
res (generated)
Gradle Scripts
androidx.constraintlayout.widget.ConstraintLayout › ImageView
Type here to search
ENG
14:06
25-08-2021

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns: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/RotatingimageView"
        android:layout_width="319dp"
        android:layout_height="279dp"
        android:contentDescription="@string/todo"
        app:layout_constraintBottom_toTopOf="@+id/ScalingimageVi
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:src="@drawable/rotation" />

    <ImageView
        android:id="@+id/ScalingimageView"
```

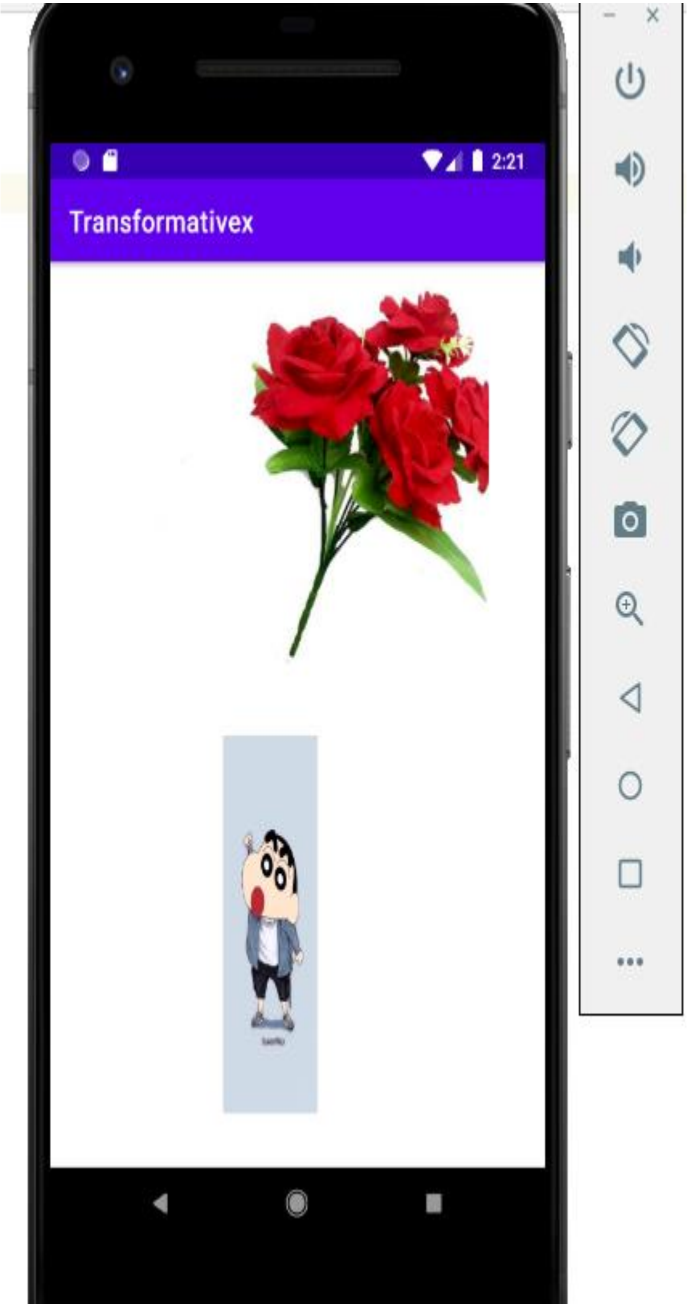


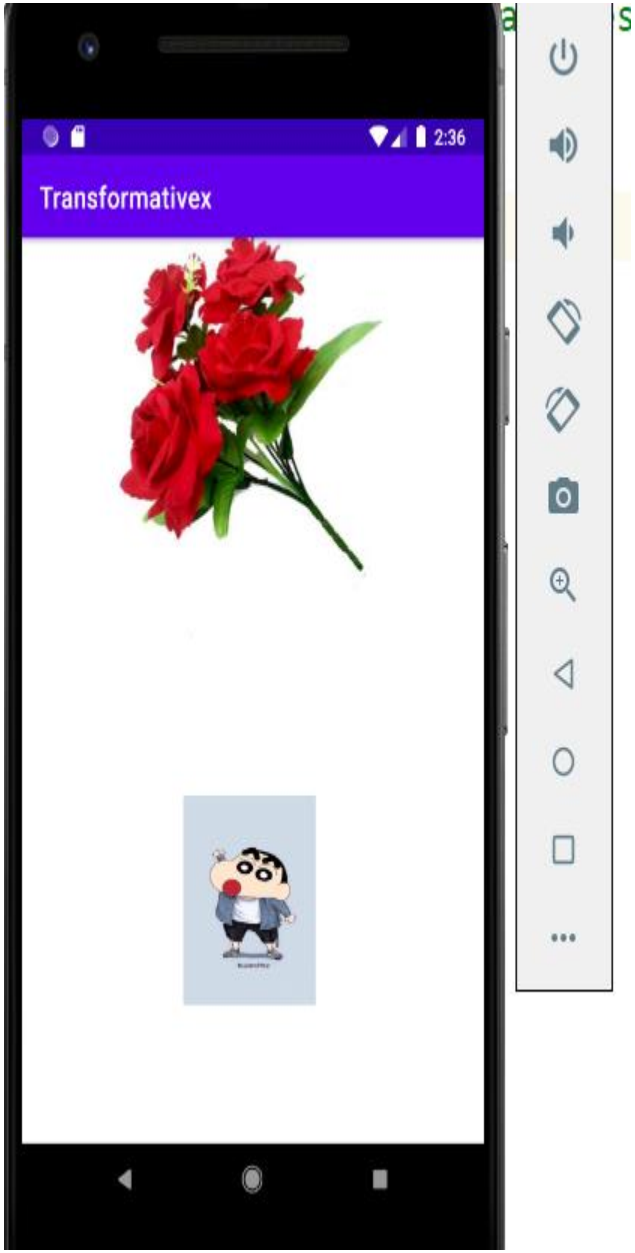
```
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            android:src="@drawable/rotation" />

        <ImageView
            android:id="@+id/ScalingimageView"
            android:layout_width="404dp"
            android:layout_height="390dp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/RotatingimageV
            app:layout_constraintVertical_bias="0.379"
            android:src="@drawable/scalable" />

    </androidx.constraintlayout.widget.ConstraintLayout>
```

androidx.constraintlayout.widget.ConstraintLayout › ImageView



```java
package com.test.transformativex;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

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

        rotateimg = findViewById(R.id.RotatingimageView);
        scaleimg=findViewById(R.id.ScalingimageView);
        rotateimg.setImageLevel(500);
        scaleimg.setImageLevel(5000);
    }
}
```


2:21
Transformativex


a
s
2:36
Transformativex

// s

3) (

Topic 6: Resolution and density indepen

- include alternative resources different screen sizes, pixel densi and aspect ratios.

- Screen size: The size of the screen to a "standard" sm
    - small => A screen smaller th standard 3.2"
    - medium => Typical smartphone

the same way as a

- large ⇒ A screen significantly larger than that of a typical smart phone, such as the screen of a tablet or notebook.

- Pixel density : Refers to the density of pixels on the display. Typically measured in dots per inch (dpi), this is calculated as a function of the physical screen size and resolution.

- ldpi ⇒ used to store low density resources for screens with pixel density in the range of 100 to 140 dpi.

density screens

range of 100 to 14

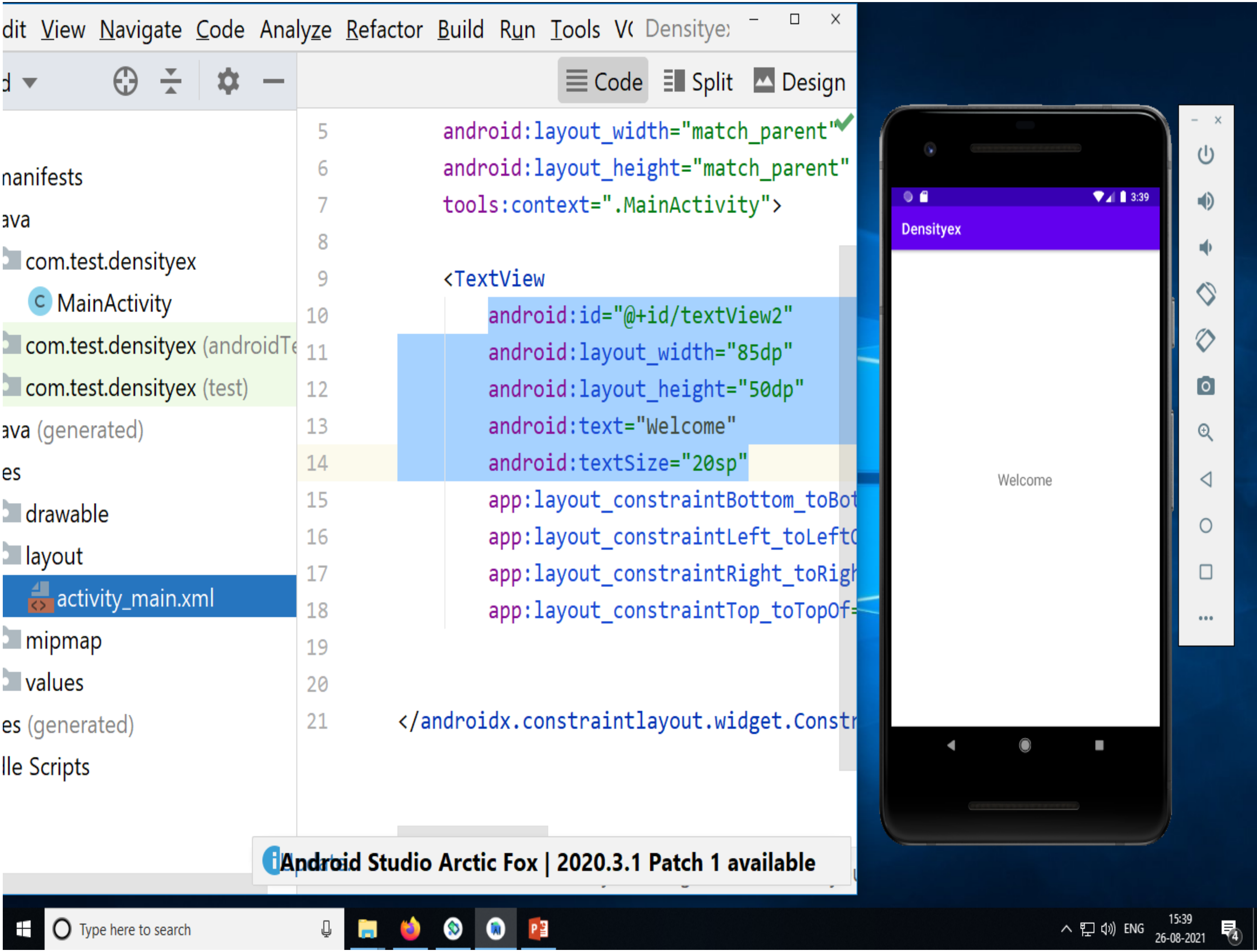
- mdpi ⇒ used for medium-density screens with 140 to 180 dpi

- hdpi ⇒ used for high-density screens featuring 190 to 250 dpi

- nodpi ⇒ used for resources that must not be scaled regardless of host screen's density.

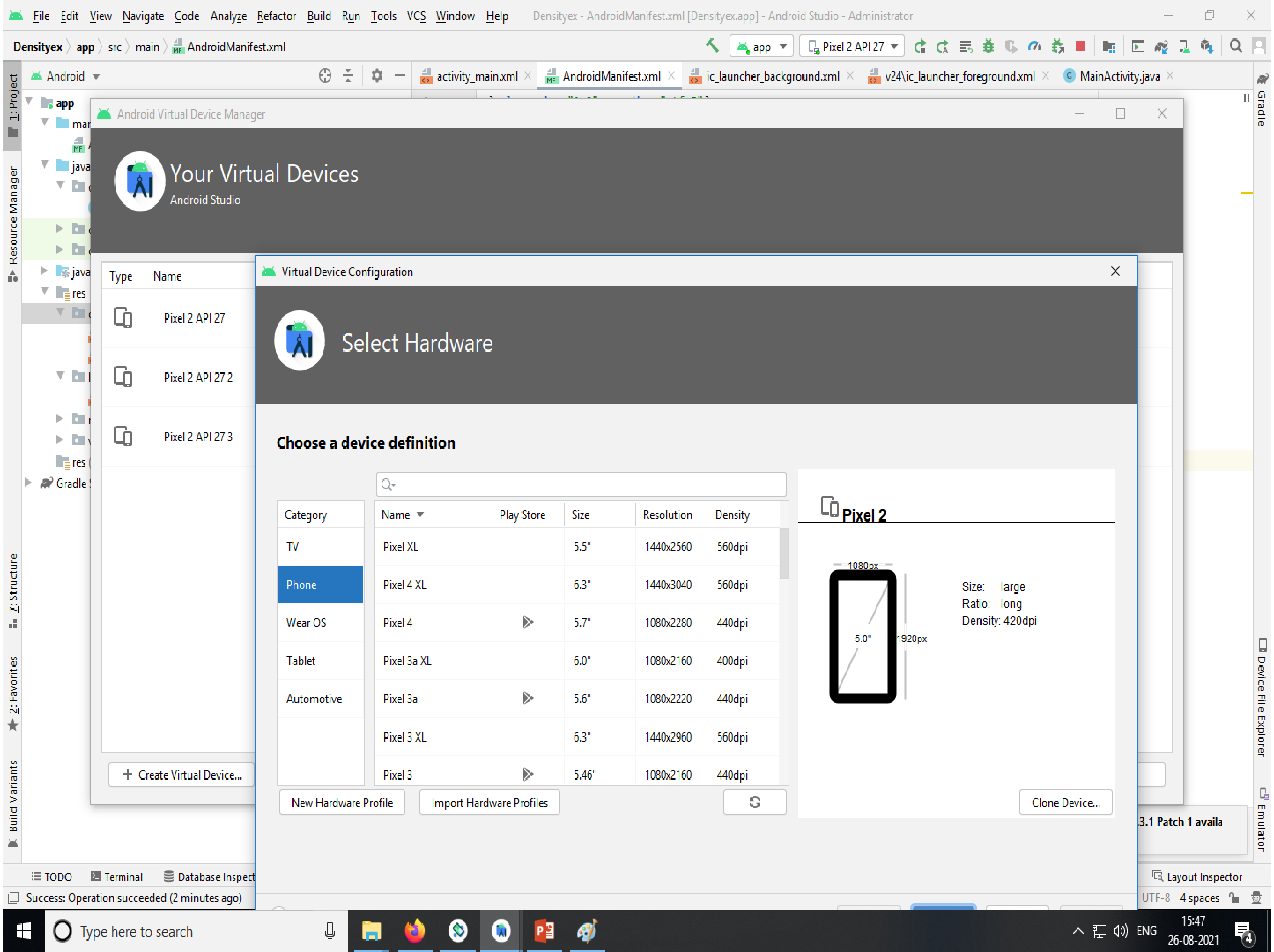
- Aspect ratio :- is the ratio of its height to its width.

- long ⇒ used for screens that are significantly wider in landscape mode than those of standard smartphones.



```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="85dp"
        android:layout_height="50dp"
        android:text="Welcome"
        android:textSize="20sp"
        app:layout_constraintBottom_toBot
        app:layout_constraintLeft_toLeftO
        app:layout_constraintRight_toRigh
        app:layout_constraintTop_toTopOf=


</androidx.constraintlayout.widget.Constr
```

Android Studio Arctic Fox | 2020.3.1 Patch 1 available


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Densityex - AndroidManifest.xml [Densityex.app] - Android Studio - Administrator
Densityex 〉 app 〉 src 〉 main 〉 AndroidManifest.xml
app
Pixel 2 API 27
Android
activity_main.xml
AndroidManifest.xml
ic_launcher_background.xml
v24\ic_launcher_foreground.xml
MainActivity.java
app
java
res
Gradle
1: Project
Resource Manager
7: Structure
2: Favorites
Build Variants
Gradle
Device File Explorer
Emulator
Android Virtual Device Manager
Your Virtual Devices
Android Studio
Type
Name
Pixel 2 API 27
Pixel 2 API 27 2
Pixel 2 API 27 3
+ Create Virtual Device...
Virtual Device Configuration

Select Hardware

Choose a device definition

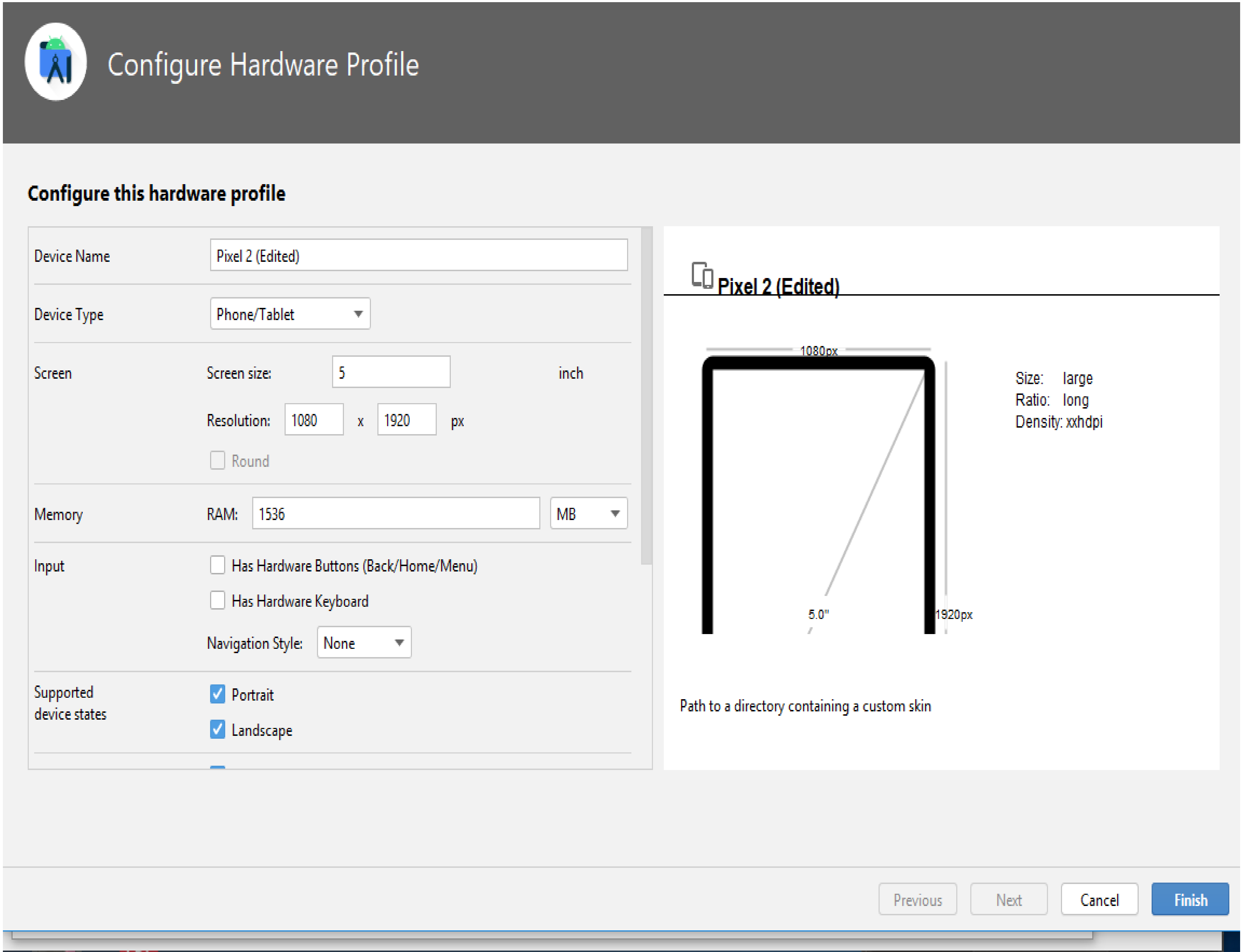
Category
TV
Phone
Wear OS
Tablet
Automotive

Name ▼	Play Store	Size	Resolution	Density
Pixel XL		5.5"	1440x2560	560dpi
Pixel 4 XL		6.3"	1440x3040	560dpi
Pixel 4	▶	5.7"	1080x2280	440dpi
Pixel 3a XL		6.0"	1080x2160	400dpi
Pixel 3a	▶	5.6"	1080x2220	440dpi
Pixel 3 XL		6.3"	1440x2960	560dpi
Pixel 3	▶	5.46"	1080x2160	440dpi

Pixel 2

Size: large
Ratio: long
Density: 420dpi

Success: Operation succeeded (2 minutes ago)



Configure Hardware Profile

Configure this hardware profile

Device Name: Pixel 2 (Edited)

Device Type: Phone/Tablet

Screen
Screen size: 5 inch
Resolution: 1080 x 1920 px
□ Round

Memory
RAM: 1536 MB

Input
□ Has Hardware Buttons (Back/Home/Menu)
□ Has Hardware Keyboard
Navigation Style: None

Supported device states
☑ Portrait
☑ Landscape

Pixel 2 (Edited)

Size: large
Ratio: long
Density: xxhdpi

Path to a directory containing a custom skin

App Basics
Introduction
Build your first app
App fundamentals
App resources
Overview
Handle configuration changes
Localization
Complex XML resources
Resource types
App manifest file

Devices
Device compatibility
Tablets, Large screens, and Foldables
Wear
Android TV
Android for Cars

Providing alternative resources

Almost every app should provide alternative resources to support specific device configurations. For instance, you should include alternative drawable resources for different screen densities and alternative string resources for different languages. At runtime, Android detects the current device configuration and loads the appropriate resources for your app.

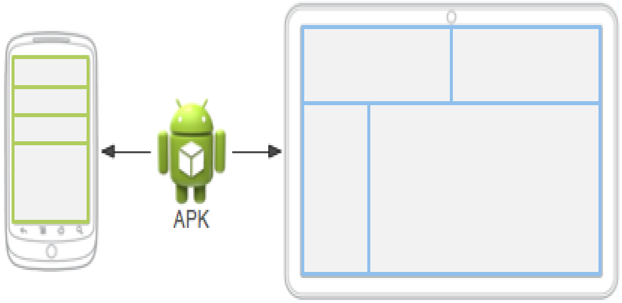
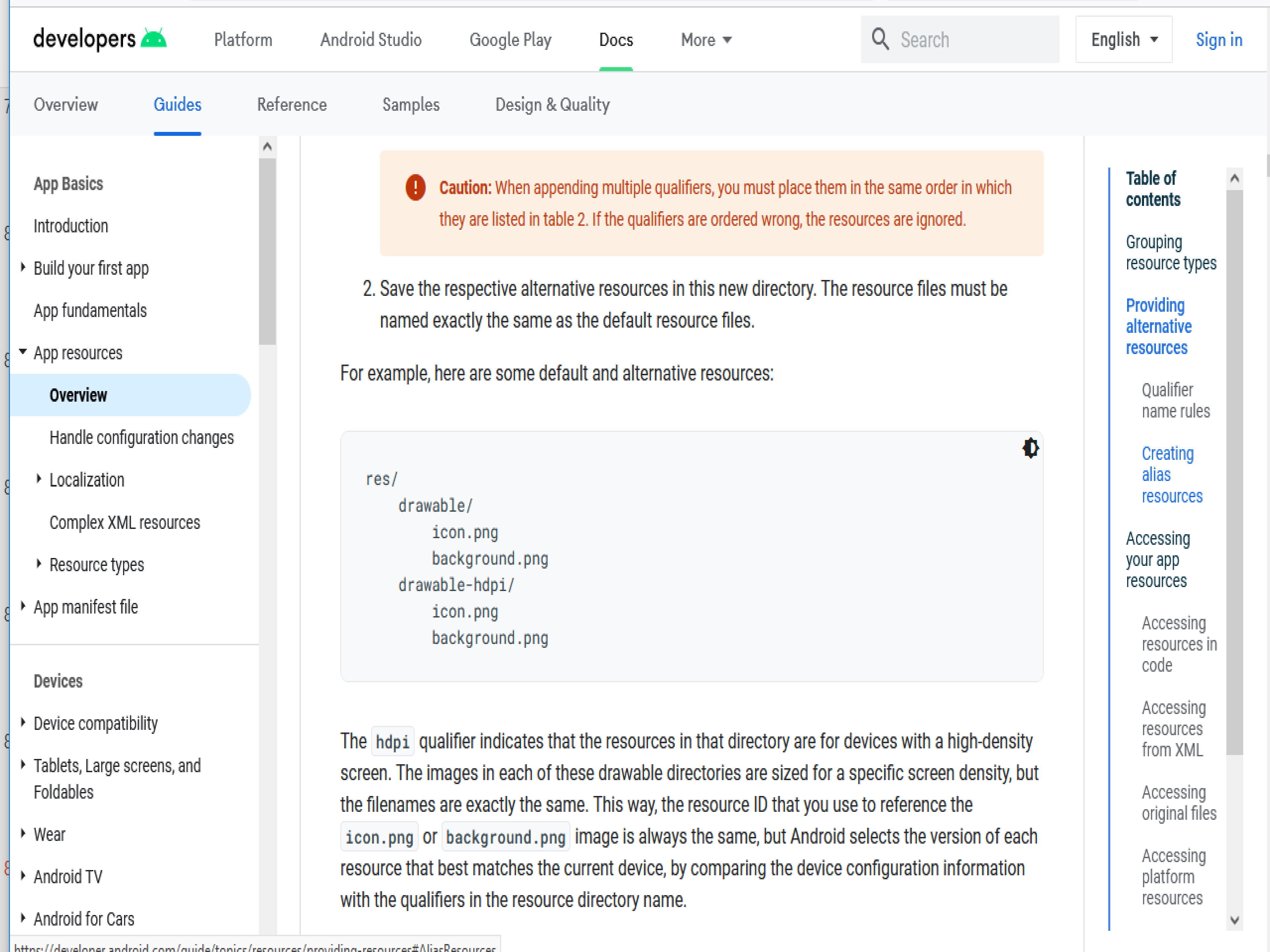


Figure 1. Two different devices, each using different layout resources.

To specify configuration-specific alternatives for a set of resources:

1. Create a new directory in `res/` named in the form *`<resources_name>-<qualifier>`*.
   - `<resources_name>` is the directory name of the corresponding default resources (defined in table 1).
   - `<qualifier>` is a name that specifies an individual configuration for which these resources are to be used (defined in table 2).

   You can append more than one `<qualifier>`. Separate each one with a dash.



developers
Platform Android Studio Google Play Docs More
Search English Sign in
Overview Guides Reference Samples Design & Quality

App Basics
Introduction
Build your first app
App fundamentals
App resources
Overview
Handle configuration changes
Localization
Complex XML resources
Resource types
App manifest file
Devices
Device compatibility
Tablets, Large screens, and Foldables
Wear
Android TV
Android for Cars

Caution: When appending multiple qualifiers, you must place them in the same order in which they are listed in table 2. If the qualifiers are ordered wrong, the resources are ignored.

2. Save the respective alternative resources in this new directory. The resource files must be named exactly the same as the default resource files.

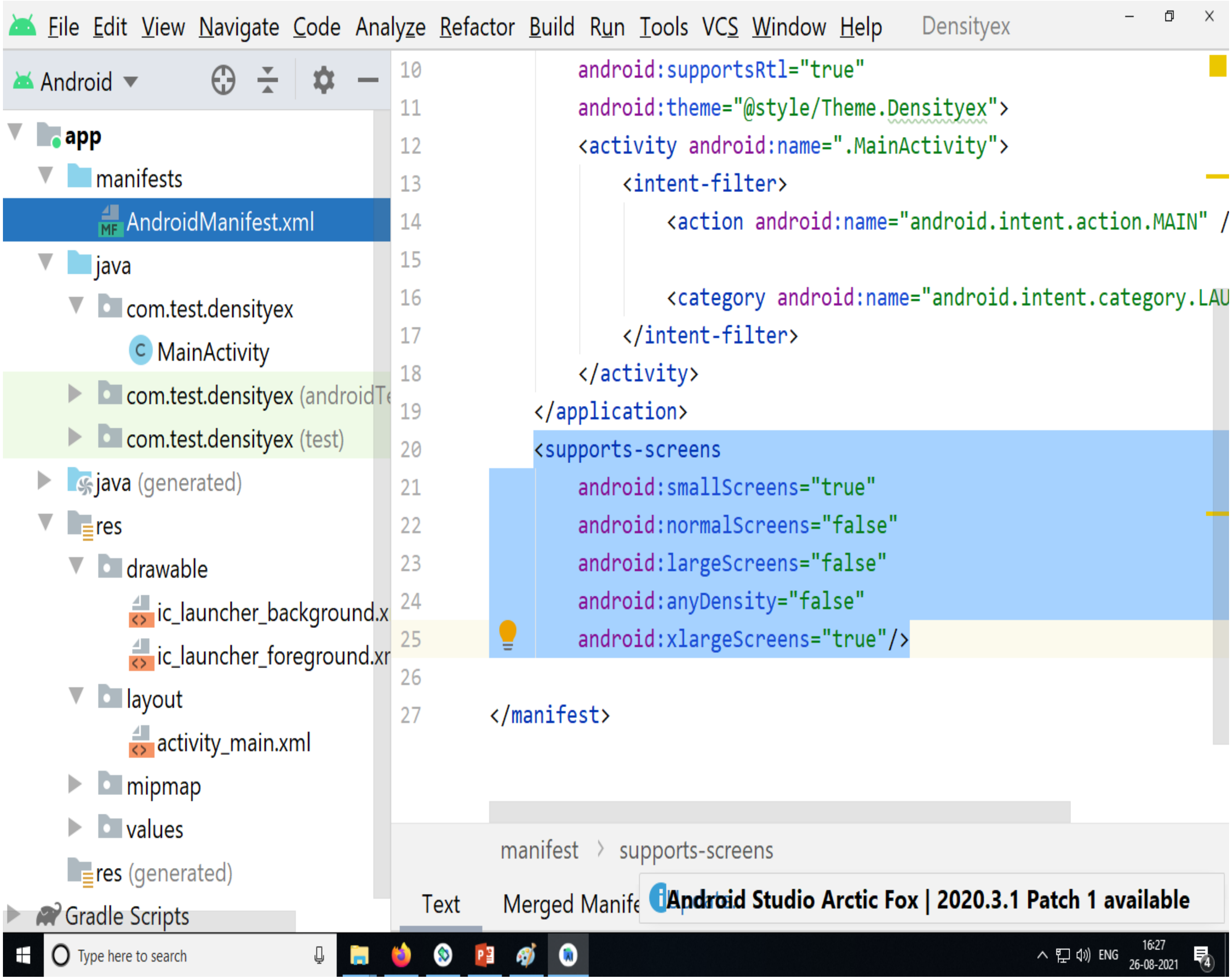
For example, here are some default and alternative resources:

```
res/
    drawable/
        icon.png
        background.png
    drawable-hdpi/
        icon.png
        background.png
```

The `hdpi` qualifier indicates that the resources in that directory are for devices with a high-density screen. The images in each of these drawable directories are sized for a specific screen density, but the filenames are exactly the same. This way, the resource ID that you use to reference the `icon.png` or `background.png` image is always the same, but Android selects the version of each resource that best matches the current device, by comparing the device configuration information with the qualifiers in the resource directory name.

Table of contents

https://developer.android.com/guide/topics/resources/providing-resources#AliasResources



```xml
        android:supportsRtl="true"
        android:theme="@style/Theme.Densityex">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>
    <supports-screens
        android:smallScreens="true"
        android:normalScreens="false"
        android:largeScreens="false"
        android:anyDensity="false"
        android:xlargeScreens="true"/>

</manifest>
```

- notlong => used for screens with a typical Smart phone aspect ratio.

Each of these qualifiers is independent and can be used independently, or in combination with each other. ie within the layout, [illegible]

the height and width of Views using wrap_content or fill_Parent where appropriate, and density-independent pixels (dp) or scale-independent pixels (sp) ~~atte~~ as required for view and font sizes.

i.e

sp for fonts
dp for layout Views

- One density independent pixel (dp)

  = One Pixel on a 160 dpi screen

- Two density independent pixel (dp)

  = Three Pixels on a 240 dpi screen

UNIT-3

FILES AND DATABASES

Saving Simple Application Data - creating and saving preferences - preferences activity -saving activity state - loading files - file management tools-sending emails through application - Introducing Android databases -SQLite - Web Database- Firebase, MySQL-PHP

Saving Simple Application Data

There are two lightweight techniques for saving simple application data for Android applications — Shared Preferences and a pair of event handlers used for saving Activity instance details. Both mechanisms use a name/value pair (NVP) mechanism to store simple primitive values.

Using SharedPreferences, you can create named maps of key/value pairs within your application that can be shared between application components running in the same Context.

Shared Preferences support the primitive types Boolean, string, float, long, and integer, making them an ideal way to quickly store default values, class instance variables, the current UI state, and user preferences. They are most commonly used to persist data across user sessions and to share settings between application components.

Alternatively, Activities offer the onSaveInstanceState handler. It's designed specifically to persist the UI state when the Activity becomes eligible for termination by a resource-hungry run time.

The handler works like the Shared Preference mechanism. It offers a Bundle parameter that represents a key/value map of primitive types that can be used to save the Activity's instance values. This Bundle is then made available as a parameter passed in to the onCreate and onRestoreInstanceState method handlers.

This UI state Bundle is used to record the values needed for an Activity to provide an identical UI following unexpected restarts.

Creating and Saving Preferences

To create or modify a Shared Preference, call getSharedPreferences on the application Context, passing in the name of the Shared Preferences to change. Shared Preferences are shared across an application's components but aren't available to other applications.

To modify a Shared Preference, use the SharedPreferences.Editor class. Get the Editor object by calling edit on the Shared Preferences object you want to change. To save edits, call commit on the Editor, as shown in the code snippet below.

```
public static final String MYPREFS = “mySharedPreferences”;
protected void savePreferences(){
// Create or retrieve the shared preference object.
int mode = Activity.MODE_PRIVATE;
SharedPreferences mySharedPreferences = getSharedPreferences(MYPREFS, mode);
// Retrieve an editor to modify the shared preferences.
SharedPreferences.Editor editor = mySharedPreferences.edit();
// Store new primitive types in the shared preferences object.
editor.putBoolean(“isTrue”, true);
editor.putFloat(“lastFloat”, 1f);
editor.putInt(“wholeNumber”, 2);
editor.putLong(“aNumber”, 3l);
editor.putString(“textEntryValue”, “Not Empty”);
// Commit the changes.
editor.commit();
}
```

Retrieving Shared Preferences

Accessing saved Shared Preferences is also done with the getSharedPreferences method. Pass in the name of the Shared Preference you want to access, and use the type-safe get<type> methods to extract saved values.

Each getter takes a key and a default value (used when no value is available for that key), as shown in the skeleton code below:

```
public void loadPreferences() {
// Get the stored preferences
int mode = Activity.MODE_PRIVATE;
SharedPreferences mySharedPreferences = getSharedPreferences(MYPREFS,
mode);
// Retrieve the saved values.
boolean isTrue = mySharedPreferences.getBoolean(“isTrue”, false);
float lastFloat = mySharedPreferences.getFloat(“lastFloat”, 0f);
int wholeNumber = mySharedPreferences.getInt(“wholeNumber”, 1);
long aNumber = mySharedPreferences.getLong(“aNumber”, 0);
String stringPreference;
stringPreference = mySharedPreferences.getString(“textEntryValue”,
“”);
}
```

Saving the Activity State

If you want to save Activity information that doesn’t need to be shared with other components (e.g., class instance variables), you can call Activity.getPreferences() without specifying a preferences name. Access to the Shared Preferences map returned is restricted to the calling Activity; each Activity supports a single unnamed SharedPreferences object.

The following skeleton code shows how to use the Activity’s private Shared Preferences:

```
protected void saveActivityPreferences(){
// Create or retrieve the activity preferences object.
SharedPreferences activityPreferences =
getPreferences(Activity.MODE_PRIVATE);
// Retrieve an editor to modify the shared preferences.
SharedPreferences.Editor editor = activityPreferences.edit();
// Retrieve the View
TextView myTextView = (TextView)findViewById(R.id.myTextView);
// Store new primitive types in the shared preferences object.
editor.putString(“currentTextValue”,
myTextView.getText().toString());
// Commit changes.
editor.commit();
}
```

Saving and Restoring Instance State

To save Activity instance variables, Android offers a specialized alternative to Shared Preferences. By overriding an Activity’s onSaveInstanceState event handler, you can use its Bundle parameter to save instance values. Store values using the same get and put methods as shown for Shared Preferences, before passing the modified Bundle into the superclass’s handler, as shown in the following code snippet:

```
private static final String TEXTVIEW_STATE_KEY = “TEXTVIEW_STATE_KEY”;
@Override
public void onSaveInstanceState(Bundle outState) {
// Retrieve the View
TextView myTextView = (TextView)findViewById(R.id.myTextView);
// Save its state
outState.putString(TEXTVIEW_STATE_KEY,
myTextView.getText().toString());
super.onSaveInstanceState(outState);
}
```

This handler will be triggered whenever an Activity completes its Active life cycle, but only when it’s not being explicitly finished. As a result, it’s used to ensure a consistent Activity state between active life cycles of a single user session.

The saved Bundle is passed in to the onRestoreInstanceState and onCreate methods if the application is forced to restart during a session. The following snippet shows how to extract values from the Bundle and use them to update the Activity instance state:

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
TextView myTextView = (TextView)findViewById(R.id.myTextView);
String text = “”;
if (icicle != null && icicle.containsKey(TEXTVIEW_STATE_KEY))
text = icicle.getString(TEXTVIEW_STATE_KEY);
myTextView.setText(text);
}
```

It’s important to remember that onSaveInstanceState is called only when an Activity becomes inactive, but not when it is being closed by a call to finish or by the user pressing the Back button.

Loading File

Android provides many kinds of storage for applications to store their data. These storage places are

- shared preferences
- internal and external storage
- SQLite storage
- storage via network connection(on Cloud)

Android Internal Storage

Android Internal storage is the storage of the private data on the device memory. By default, saving and loading files to the internal storage are private to the application and other applications will not have access to these files. When the user uninstalls the applications the internal stored files associated with the application are also removed. However, note that some users root their Android phones, gaining superuser access. These users will be able to read and write whatever files they wish.

Reading and Writing Text File in Android Internal Storage

Android offers openFileInput and openFileOutput from the Java I/O classes to modify reading and writing streams from and to local files.

- openFileOutput(): This method is used to create and save a file. Its syntax is given below:

- FileOutputStream fOut = openFileOutput("file name",Context.MODE_PRIVATE);

The method openFileOutput() returns an instance of FileOutputStream. After that we can call write method to write data on the file. Its syntax is given below:

```
String str = "test data";

fOut.write(str.getBytes());

fOut.close();
```

- **openFileInput()**: This method is used to open a file and read it. It returns an instance of FileInputStream. Its syntax is given below:

- FileInputStream fin = openFileInput(file);

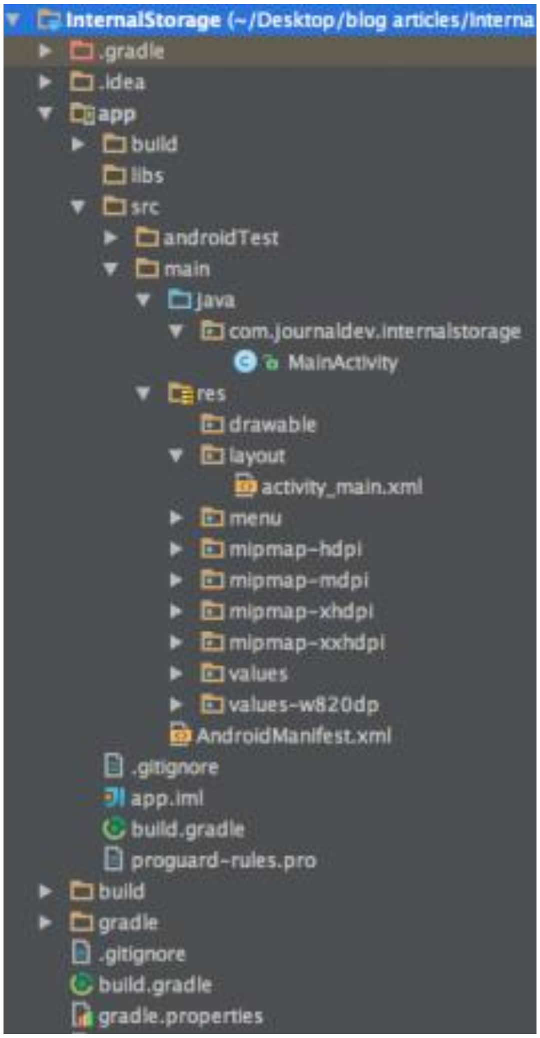
After that, we call read method to read one character at a time from the file and then print it. Its syntax is given below:

```
int c;

String temp="";
while( (c = fin.read()) != -1){
   temp = temp + Character.toString((char)c);
}


fin.close();
```

In the above code, string temp contains all the data of the file.

- Note that these methods do not accept file paths (e.g. path/to/file.txt), they just take simple file names.

Android Internal Storage Project Structure



```
▼ InternalStorage (~/Desktop/blog articles/Interna
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▶ build
      libs
    ▼ src
      ▶ androidTest
      ▼ main
        ▼ java
          ▼ com.journaldev.internalstorage
              MainActivity
        ▼ res
            drawable
          ▼ layout
              activity_main.xml
          ▶ menu
          ▶ mipmap-hdpi
          ▶ mipmap-mdpi
          ▶ mipmap-xhdpi
          ▶ mipmap-xxhdpi
          ▶ values
          ▶ values-w820dp
          AndroidManifest.xml
      .gitignore
      app.iml
      build.gradle
      proguard-rules.pro
  ▶ build
  ▶ gradle
    .gitignore
    build.gradle
    gradle.properties
```

Android Internal Storage Example Code

The xml layout contains an EditText to write data to the file and a Write Button and Read Button. Note that the onClick methods are defined in the xml file only as shown below: activity_main.xml

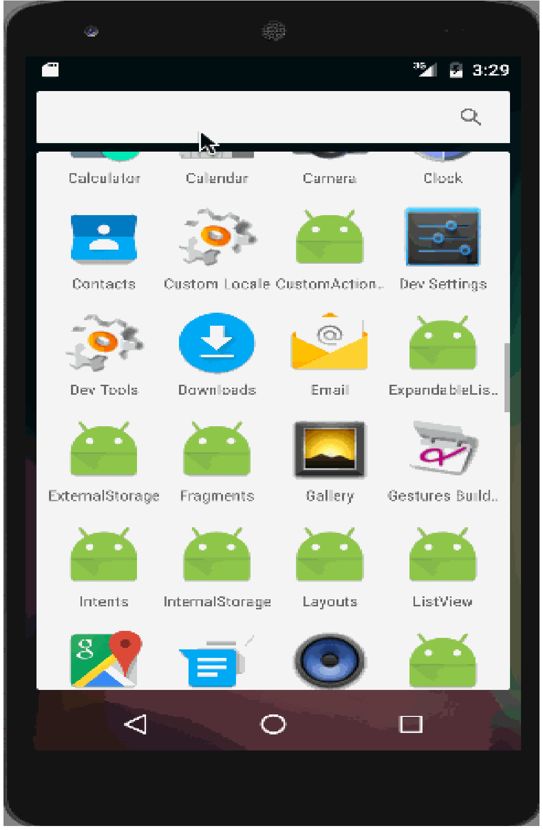
```
<RelativeLayout xmlns:android="https://schemas.android.com/apk/res/android"
   xmlns:tools="https://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:padding="5dp"
        android:text="Android Read and Write Text from/to a File"
        android:textStyle="bold"
        android:textSize="28sp" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="22dp"
        android:minLines="5"
        android:layout_margin="5dp">

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Write Text into File"
        android:onClick="WriteBtn"
        android:layout_alignTop="@+id/button2"
        android:layout_alignRight="@+id/editText1"
        android:layout_alignEnd="@+id/editText1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Read Text From file"
        android:onClick="ReadBtn"
        android:layout_centerVertical="true"
        android:layout_alignLeft="@+id/editText1"
        android:layout_alignStart="@+id/editText1" />

</RelativeLayout>
```

The MainActivity contains the implementation of the reading and writing to files as it was explained above.

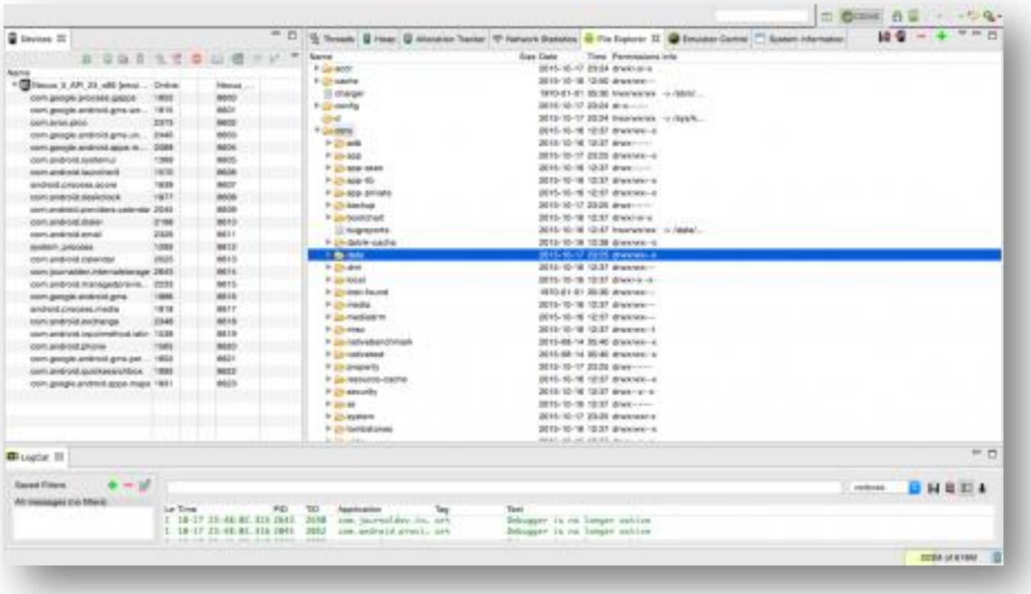
```
package com.journaldev.internalstorage;
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends Activity {

    EditText textmsg;
    static final int READ_BLOCK_SIZE = 100;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textmsg=(EditText)findViewById(R.id.editText1);
    }

    // write text to file
    public void WriteBtn(View v) {
        // add-write text into file
        try {
            FileOutputStream fileout=openFileOutput("mytextfile.txt", MODE_PRIVATE);
            OutputStreamWriter outputWriter=new OutputStreamWriter(fileout);
            outputWriter.write(textmsg.getText().toString());
            outputWriter.close();
            //display file saved message
            Toast.makeText(getBaseContext(), "File saved successfully!",
                    Toast.LENGTH_SHORT).show();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // Read text from file
    public void ReadBtn(View v) {
        //reading text from file
        try {
            FileInputStream fileIn=openFileInput("mytextfile.txt");
            InputStreamReader InputRead= new InputStreamReader(fileIn);

            char[] inputBuffer= new char[READ_BLOCK_SIZE];
            String s="";
            int charRead;

            while ((charRead=InputRead.read(inputBuffer))>0) {
                // char to string conversion
                String readstring=String.copyValueOf(inputBuffer,0,charRead);
                s +=readstring;
            }
            InputRead.close();
            textmsg.setText(s);


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Here, a toast is displayed when data is successfully written into the internal storage and the data is displayed in the EditText itself on reading the data from the file. The image shown below is the output of the project. The image depicts text being written to the internal storage and on clicking Read it displays back the text in the same EditText.

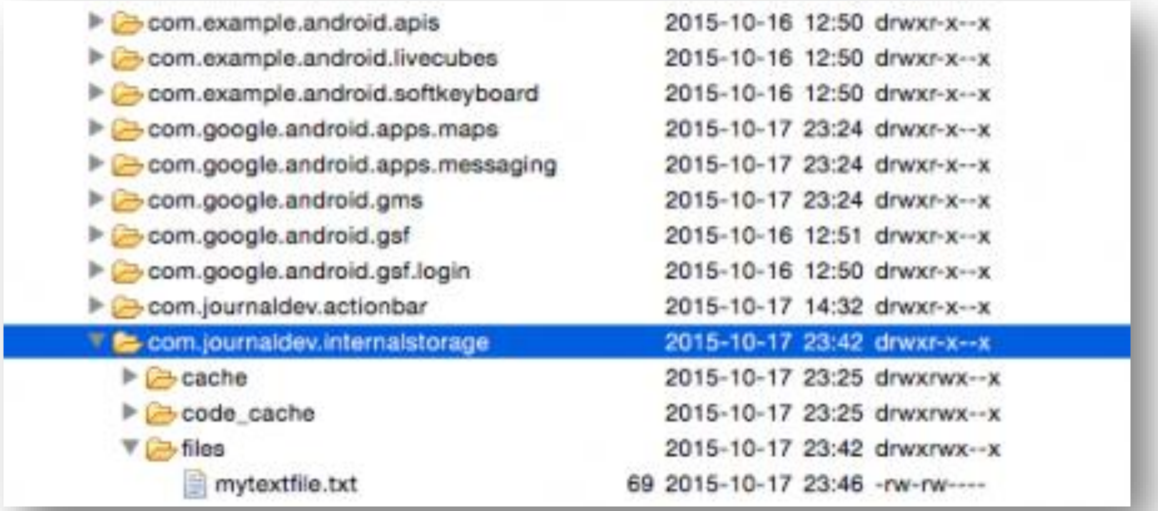


Where is the file located?

To actually view the file open the Android Device Monitor from Tools->Android->Android Device Monitor. The file is present in the folder data->data->{package name}->files as shown in the



images below: The file "mytextfile.txt" is found in the package name of the project i.e. **com.journaldev.internalstorage** as shown below:



```
▶ com.example.android.apis                 2015-10-16 12:50 drwxr-x--x
▶ com.example.android.livecubes            2015-10-16 12:50 drwxr-x--x
▶ com.example.android.softkeyboard         2015-10-16 12:50 drwxr-x--x
▶ com.google.android.apps.maps             2015-10-17 23:24 drwxr-x--x
▶ com.google.android.apps.messaging        2015-10-17 23:24 drwxr-x--x
▶ com.google.android.gms                   2015-10-17 23:24 drwxr-x--x
▶ com.google.android.gsf                   2015-10-16 12:51 drwxr-x--x
▶ com.google.android.gsf.login             2015-10-16 12:50 drwxr-x--x
▶ com.journaldev.actionbar                 2015-10-17 14:32 drwxr-x--x
▼ com.journaldev.internalstorage           2015-10-17 23:42 drwxr-x--x
   ▶ cache                                 2015-10-17 23:25 drwxrwx--x
   ▶ code_cache                            2015-10-17 23:25 drwxrwx--x
   ▼ files                                 2015-10-17 23:42 drwxrwx--x
        mytextfile.txt                  69 2015-10-17 23:46 -rw-rw----
```

Android External Storage Example

Like internal storage, we are able to save or read data from the device external memory such as sdcard. The FileInputStream and FileOutputStream classes are used to read and write data into the file.

Example of reading and writing data in the android external storage

activity_main.xml

Drag the 2 edittexts, 2 textviews and 2 buttons from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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="example.javatpoint.com.externalstorage.MainActivity">

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="20dp"
        android:layout_marginTop="24dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/editText1"
        android:layout_below="@+id/editText1"
        android:layout_marginTop="24dp"
        android:ems="10" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText1"
        android:layout_alignBottom="@+id/editText1"
        android:layout_alignParentLeft="true"
        android:text="File Name:" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/editText2"
        android:layout_alignBottom="@+id/editText2"
        android:layout_alignParentLeft="true"
        android:text="Data:" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText2"
        android:layout_below="@+id/editText2"
        android:layout_marginLeft="70dp"
        android:layout_marginTop="16dp"
        android:text="save" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/button1"
        android:layout_alignBottom="@+id/button1"
        android:layout_toRightOf="@+id/button1"
        android:text="read" />
</RelativeLayout>
```

Provide permission for the external storage

You need to provide the WRITE_EXTERNAL_STORAGE permission.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

File: Activity_Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.javatpoint.com.externalstorage">
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <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/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>
```

Activity class

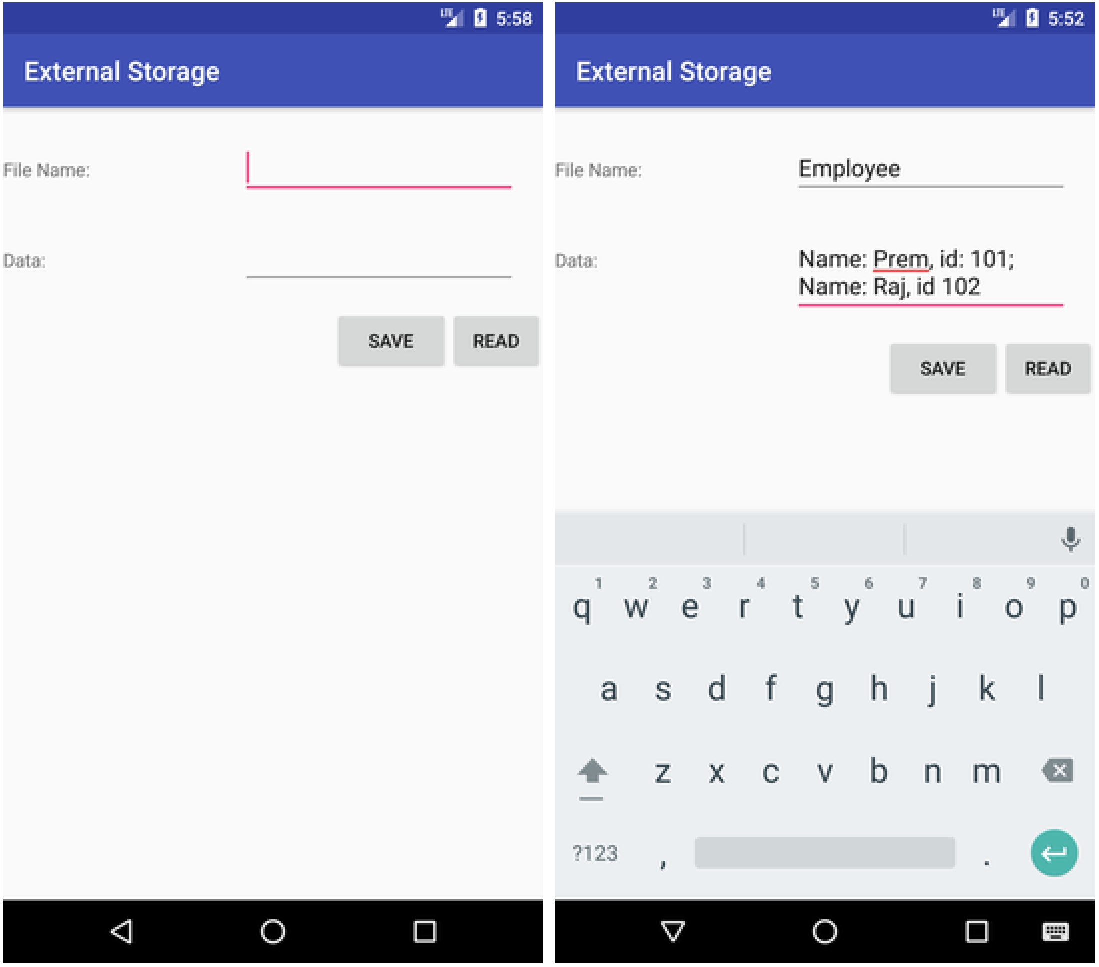
Let's write the code to write and read data from the android external storage.

File: MainActivity.java

```
package example.javatpoint.com.externalstorage;

```

```java
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {
    EditText editTextFileName,editTextData;
    Button saveButton,readButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editTextFileName=findViewById(R.id.editText1);
        editTextData=findViewById(R.id.editText2);
        saveButton=findViewById(R.id.button1);
        readButton=findViewById(R.id.button2);

        //Performing action on save button
        saveButton.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View arg0) {
                String filename=editTextFileName.getText().toString();
                String data=editTextData.getText().toString();

                FileOutputStream fos;
                try {
                    File myFile = new File("/sdcard/"+filename);
                    myFile.createNewFile();
                    FileOutputStream fOut = new FileOutputStream(myFile);
                    OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
                    myOutWriter.append(data);
                    myOutWriter.close();
                    fOut.close();
                    Toast.makeText(getApplicationContext(),filename + "saved",Toast.LENGTH_LONG).show();
                } catch (FileNotFoundException e) {e.printStackTrace();}
                catch (IOException e) {e.printStackTrace();}
            }
        });

        //Performing action on Read Button
        readButton.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View arg0) {
                String filename=editTextFileName.getText().toString();
                StringBuffer stringBuffer = new StringBuffer();
                String aDataRow = "";
                String aBuffer = "";
                try {
                    File myFile = new File("/sdcard/"+filename);
                    FileInputStream fIn = new FileInputStream(myFile);
                    BufferedReader myReader = new BufferedReader(
                            new InputStreamReader(fIn));
                    while ((aDataRow = myReader.readLine()) != null) {
                        aBuffer += aDataRow + "\n";
                    }
                    myReader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                Toast.makeText(getApplicationContext(),aBuffer,Toast.LENGTH_LONG).show();
            }
        });
```

```
    }
}
```



File Name:

Data:

File Name: Employee

Data: Name: Prem, id: 101;
Name: Raj, id 102

Android File Manager

Android allows every user to access their device's file system using a file manager. They can access Android files and transfer them between a computer and a smartphone or tablet. This is performed by simply plug in a USB cable in an Android device and fire up your desktop file manager. Android's file manager app reacts as an important part of the software on a device. It provides facility to browse files, manage storage space, downloads, move files around, and a lot of other activities. However, several Android device manufacturers provide a pre-install Android file manager app on their devices. The pre-install file manager contains fewer features as compared to other third-party file managers.

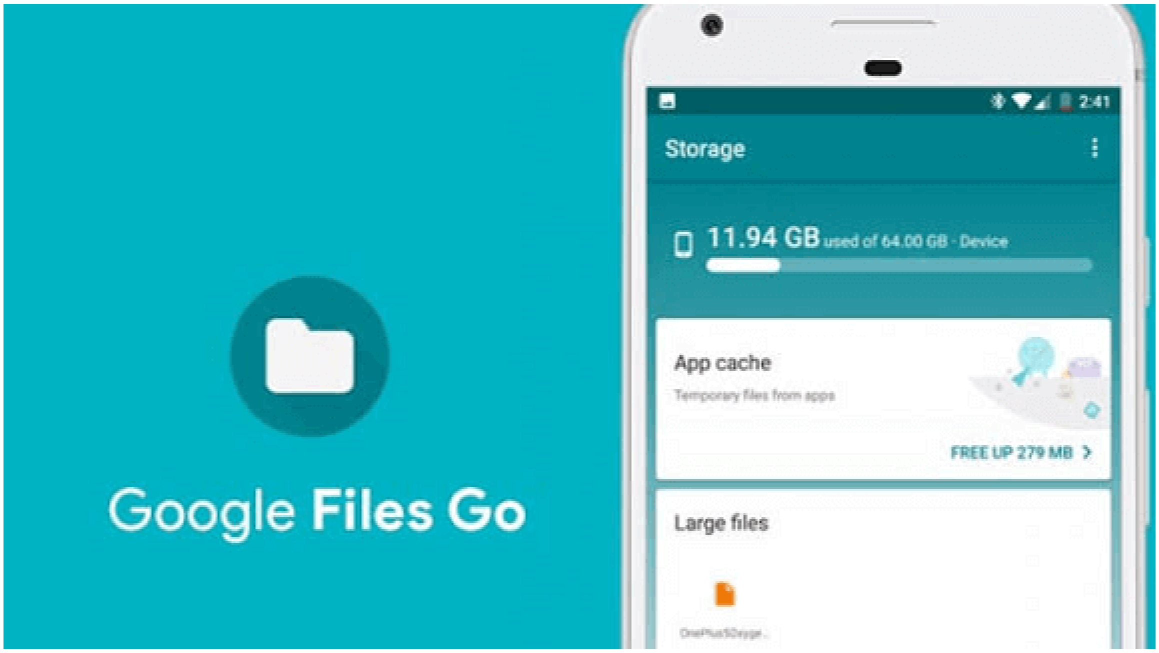
Some of the best Android File Managers:

1. Files Go (free)
2. Mixplorer Silver ($4.79)

3. Astro Cloud & File Manager (free)
4. Solid Explorer ($1.99 after free trial)
5. DiskUsage (free)
6. File Commander (free, in-app purchase)

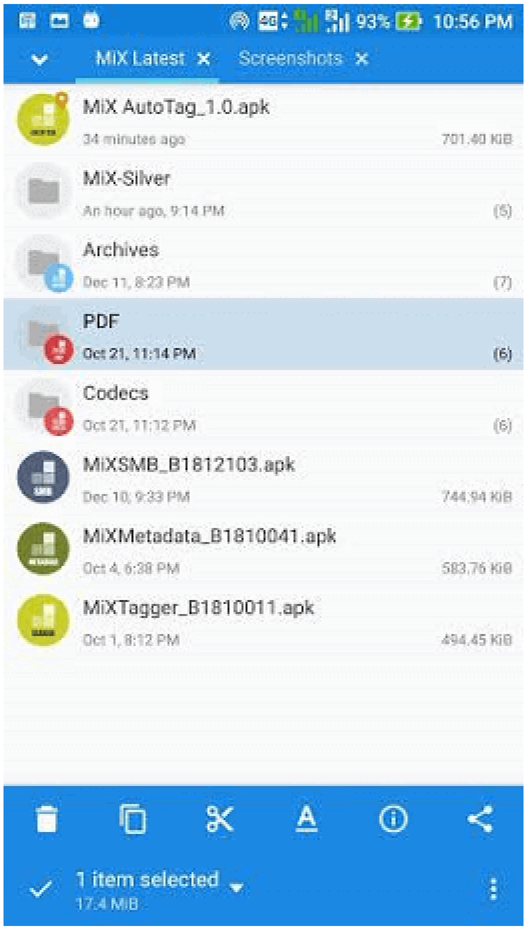
Files Go

Files Go file manager is a product of Google, which behaves less like a file manager app and more as a device assistant. This file manager helps you to manage your device's storage space by deleting rarely used applications, files, documents, large size media, and folders. It also facilitates a cache clearing functionality and allows you to view files based on their categories. The Files Go supports local encrypted file sharing and backup features that provide a facility to send files to the cloud.



MiXplorer Silver

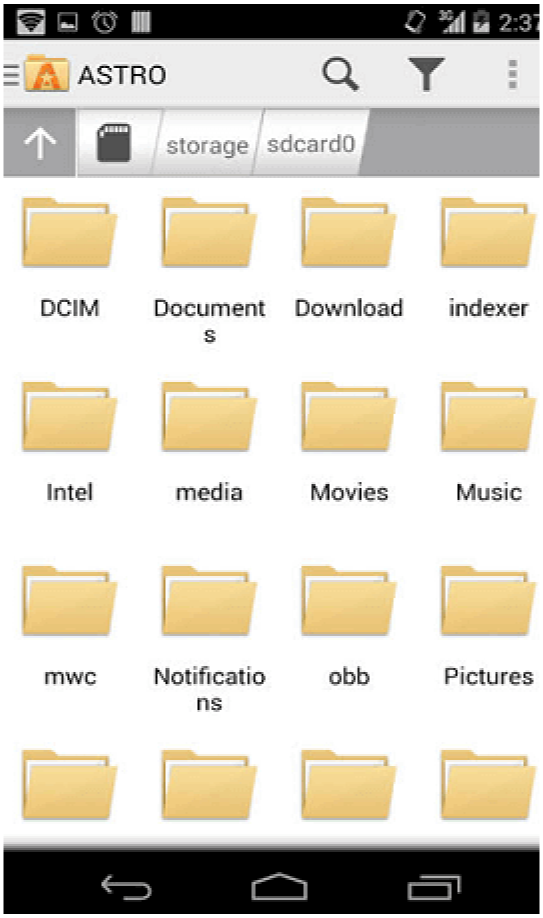
MiXplorer Silver has a strong file manager in this era now, but it is now recently placed in the Google Play store in a premium version. This version bunches together MiXplorer with several premium add-ons like Mix archive, tagger, and metadata. It supports multiple archive formats, including RAR and ZIP. The MiXplorer Silver contains a built-in media player and image viewer. It supports for both networked and cloud-based storage device.



MiX AutoTag_1.0.apk	34 minutes ago	701.40 KiB
MiX-Silver	An hour ago, 9:14 PM	(5)
Archives	Dec 11, 8:23 PM	(7)
PDF	Oct 21, 11:14 PM	(6)
Codecs	Oct 21, 11:12 PM	(6)
MiXSMB_B1812103.apk	Dec 10, 9:33 PM	744.94 KiB
MiXMetadata_B1810041.apk	Oct 4, 6:38 PM	583.76 KiB
MiXTagger_B1810011.apk	Oct 1, 8:12 PM	494.45 KiB

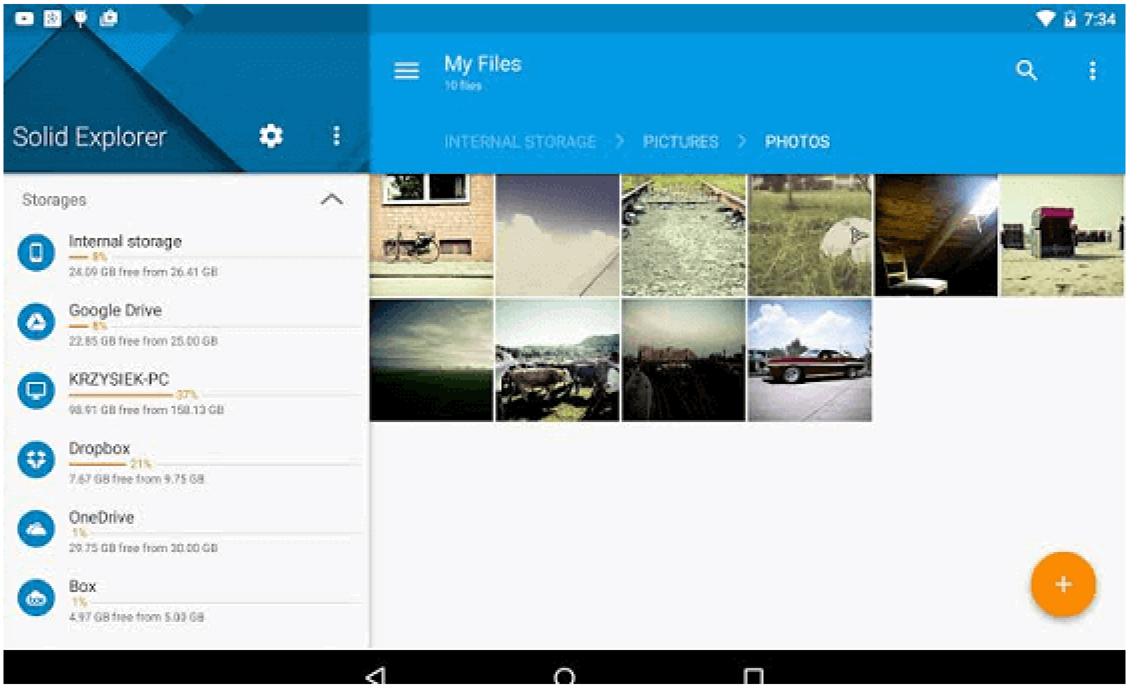
Astro Cloud & File Manager

ASTRO file manager is another important and popular choice for Android file management. This file manager contains both cloud and local storage management components in a single package. It offers you to manage your internal and SD card storage space. Using Astro Cloud file manager, you can move your files and media to and from cloud services like One Drive, Dropbox, and Google Drive. Its left side drawer contains convenient shortcuts such as cloud service and network storage, bookmarked locations, and shortcuts to media files. Other than file management functionality, this app also contains an app manager, quick viewer to SD card usage space, and a task killer.



Solid Explorer

This is an attractive file manager app which provides two-pane user interface over the Android phones and tablets. It also supports for archive files like TAR, RAR, and ZIP, and indexed search function. Other than an impressive feature, Solid Explorer supports a wide range of plugins that add new features such as FTP server plugin, USB OTG support, and lots more. It has two separate explorer panes, which are combined with drag-and-drop support, which offers copying and moving files from one location to another.



Internal storage
8%
24.09 GB free from 26.41 GB

Google Drive
8%
22.85 GB free from 25.00 GB

KRZYSIEK-PC
37%
98.91 GB free from 158.13 GB

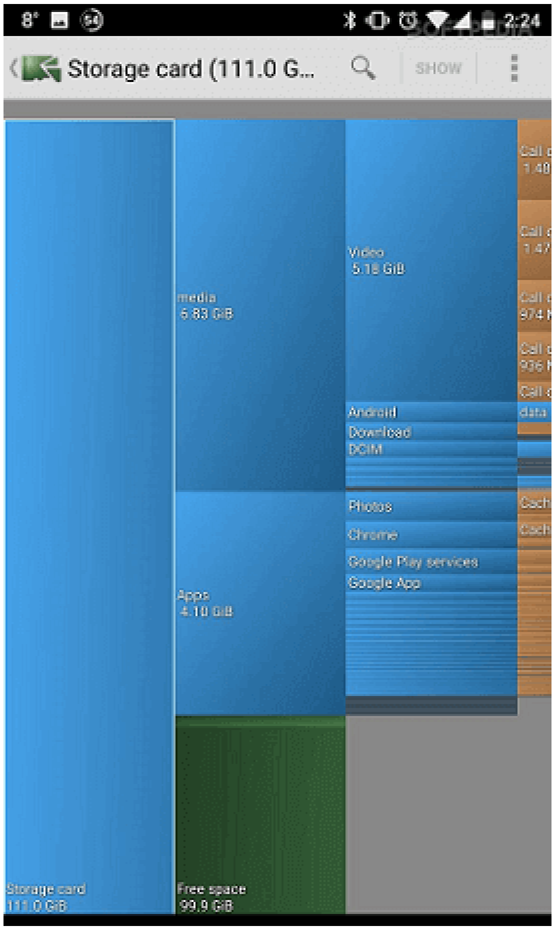
Dropbox
21%
7.67 GB free from 9.75 GB

OneDrive
1%
29.75 GB free from 30.00 GB

Box
1%
4.97 GB free from 5.03 GB

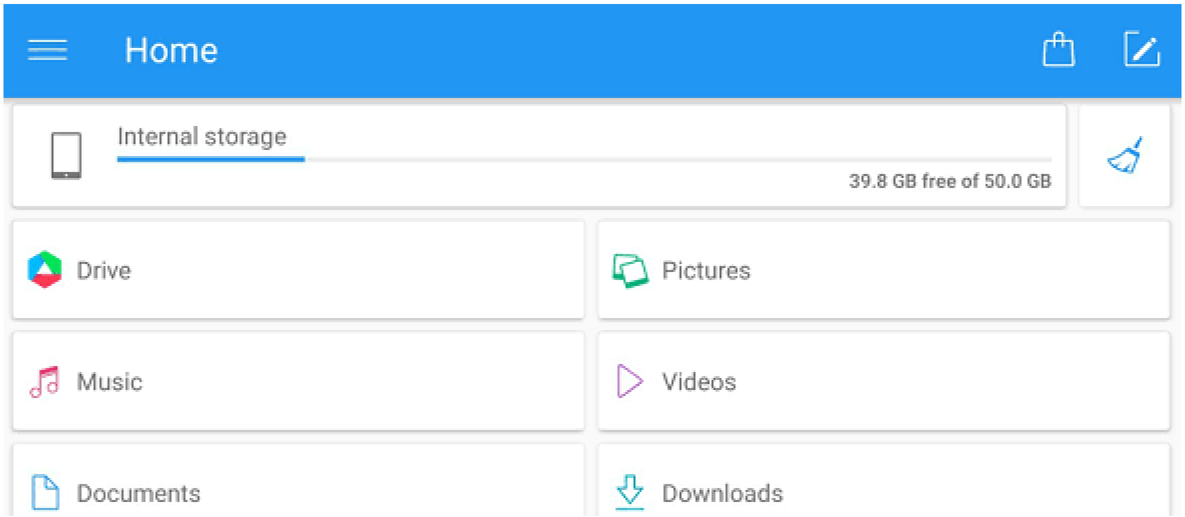
DiskUsage

DiskUsage is not a really complete file manager app; however, it is still an attractive tool that helps you to manage your file system. This helps you to instantly find the files and folders that occupy more storage space in the device. It performs as a separate tool to view and delete folders.



File Commander

File Commander is an excellent file manager that provides extra features and utilities in-app purchases. Instead of opening a file and directory view, the app's home screen gives users a series of classified "libraries" such as document, audio, and video. This file manager also supports remote access, file sharing, and cloud storage. It contains additions tools in-app purchases.



sending emails through application

Email is messages distributed by electronic means from one system user to one or more recipients via a network.

Before starting Email Activity, You must know Email functionality with intent, Intent is carrying data from one component to another component with-in the application or outside the application.

To send an email from your application, you don’t have to implement an email client from the beginning, but you can use an existing one like the default Email app provided from Android, Gmail, Outlook, K-9 Mail etc. For this purpose, we need to write an Activity that launches an email client, using an implicit Intent with the right action and data. In this example, we are going to send an email from our app by using an Intent object that launches existing email clients.

Following section explains different parts of our Intent object required to send an email.

Intent Object - Action to send Email

You will use **ACTION_SEND** action to launch an email client installed on your Android device. Following is simple syntax to create an intent with ACTION_SEND action.

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

Intent Object - Data/Type to send Email

To send an email you need to specify **mailto:** as URI using setData() method and data type will be to **text/plain** using setType() method as follows −

```
emailIntent.setData(Uri.parse("mailto:"));
emailIntent.setType("text/plain");
```

Intent Object - Extra to send Email

Android has built-in support to add TO, SUBJECT, CC, TEXT etc. fields which can be attached to the intent before sending the intent to a target email client. You can use following extra fields in your email −

Sr.No.	Extra Data & Description
1	**EXTRA_BCC** A String[] holding e-mail addresses that should be blind carbon copied.
2	**EXTRA_CC** A String[] holding e-mail addresses that should be carbon copied.
3	**EXTRA_EMAIL** A String[] holding e-mail addresses that should be delivered to.
4	**EXTRA_HTML_TEXT** A constant String that is associated with the Intent, used with ACTION_SEND to supply an alternative to EXTRA_TEXT as HTML formatted text.
5	**EXTRA_SUBJECT** A constant string holding the desired subject line of a message.
6	**EXTRA_TEXT** A constant CharSequence that is associated with the Intent, used with ACTION_SEND to supply the literal data to be sent.
7	**EXTRA_TITLE** A CharSequence dialog title to provide to the user when used with a ACTION_CHOOSER.

Here is an example showing you how to assign extra data to your intent −

```
emailIntent.putExtra(Intent.EXTRA_EMAIL  , new String[]{"Recipient"});
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "subject");
emailIntent.putExtra(Intent.EXTRA_TEXT   , "Message Body");
```

The out-put of above code is as below shown an image

Email Example

Example

Following example shows you in practical how to use Intent object to launch Email client to send an Email to the given recipients.

To Email experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you might get struggle with emulator which may not work properly. Second you will need to have an Email client like GMail(By default every android version having Gmail client App) or K9mail installed on your device.

Step	Description
1	You will use Android studio to create an Android application and name it as *Tutorialspoint* under a package *com.example.tutorialspoint*.
2	Modify *src/MainActivity.java* file and add required code to take care of sending email.
3	Modify layout XML file *res/layout/activity_main.xml* add any GUI component if required. I'm adding a simple button to launch Email Client.
4	Modify *res/values/strings.xml* to define required constant values
5	Modify *AndroidManifest.xml* as shown below
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.Tutorialspoint/MainActivity.java**.

```
package com.example.tutorialspoint;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBtn = (Button) findViewById(R.id.sendEmail);
      startBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            sendEmail();
         }
      });
   }

   protected void sendEmail() {
      Log.i("Send email", "");
      String[] TO = {""};
      String[] CC = {""};
      Intent emailIntent = new Intent(Intent.ACTION_SEND);

      emailIntent.setData(Uri.parse("mailto:"));
      emailIntent.setType("text/plain");
      emailIntent.putExtra(Intent.EXTRA_EMAIL, TO);
      emailIntent.putExtra(Intent.EXTRA_CC, CC);
      emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Your subject");
      emailIntent.putExtra(Intent.EXTRA_TEXT, "Email message goes here");

      try {
         startActivity(Intent.createChooser(emailIntent, "Send mail..."));
         finish();
         Log.i("Finished sending email...", "");
      } catch (android.content.ActivityNotFoundException ex) {
         Toast.makeText(MainActivity.this, "There is no email client installed.",
Toast.LENGTH_SHORT).show();
      }
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

Here abc indicates about tutorialspoint logo

```
<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/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Sending Mail Example"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point "
      android:textColor="#ff87ff09"
      android:textSize="30dp"
      android:layout_above="@+id/imageButton"
      android:layout_alignRight="@+id/imageButton"
      android:layout_alignEnd="@+id/imageButton" />

   <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:src="@drawable/abc"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true" />

   <Button
      android:id="@+id/sendEmail"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/compose_email"/>

</LinearLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

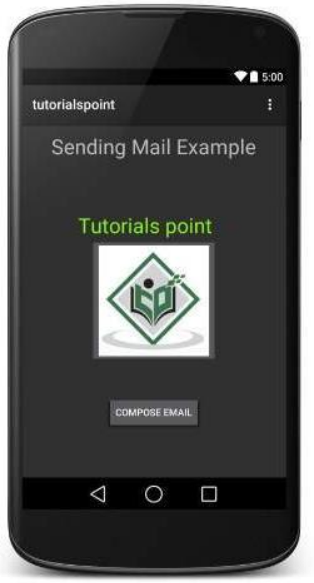
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">Tutorialspoint</string>
   <string name="compose_email">Compose Email</string>
</resources>
```

Following is the default content of **AndroidManifest.xml** −

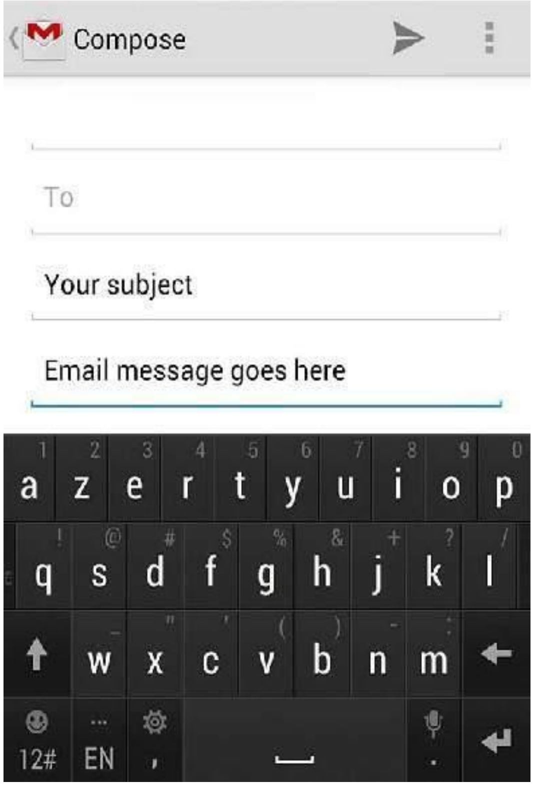
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.Tutorialspoint" >

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name="com.example.tutorialspoint.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>
```

Let's try to run your **tutorialspoint** application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Android Studio, open one of

your project's activity files and click Run icon from the toolbar. Before starting your application, Android studio installer will display following window to select an option where you want to run your Android application.Select your mobile device as an option and then check your mobile device which will display following screen −



Now use **Compose Email** button to list down all the installed email clients. From the list, you can choose one of email clients to send your email. I'm going to use Gmail client to send my email which will have all the provided defaults fields available as shown below. Here **From:** will be default email ID you have registered for your Android device.



To

Your subject

Email message goes here

You can modify either of the given default fields and finally use send email button to send your email to the mentioned recipients.

Android databases –SQLite

Database - Package

The main package is android.database.sqlite that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object.Its syntax is given below

SQLiteDatabase mydatabase = openOrCreateDatabase("your database name",MODE_PRIVATE,null);

Apart from this , there are other functions available in the database package , that does this job. They are listed below

Sr.No	Method & Description
1	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler)** This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY
2	**openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)** It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases
3	**openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)** It not only opens but create the database if it not exists. This method is equivalent to openDatabase method.
4	**openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)** This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath()

Database - Insertion

we can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username
VARCHAR,Password VARCHAR);");
mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin','admin');");
```

This will insert some values into our table in our database. Another method that also does the same job but take some additional parameter is given below

Sr.No	Method & Description
1	**execSQL(String sql, Object[] bindArgs)** This method not only insert data , but also used to update or modify already existing data in database using bind arguments

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

```
Cursor resultSet = mydatbase.rawQuery("Select * from TutorialsPoint",null);
resultSet.moveToFirst();
String username = resultSet.getString(0);
String password = resultSet.getString(1);
```

There are other functions available in the Cursor class that allows us to effectively retrieve the data. That includes

Sr.No	Method & Description
1	**getColumnCount()** This method return the total number of columns of the table.
2	**getColumnIndex(String columnName)** This method returns the index number of a column by specifying the name of the column
3	**getColumnName(int columnIndex)** This method returns the name of the column by specifying the index of the column

4	**getColumnNames()** This method returns the array of all the column names of the table.
5	**getCount()** This method returns the total number of rows in the cursor
6	**getPosition()** This method returns the current position of the cursor in the table
7	**isClosed()** This method returns true if the cursor is closed and return false otherwise

Database - Helper class

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database. Its syntax is given below

```
public class DBHelper extends SQLiteOpenHelper {
   public DBHelper(){
      super(context,DATABASE_NAME,null,1);
   }
   public void onCreate(SQLiteDatabase db) {}
   public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}
}
```

onCreate(SQLiteDatabase sqLiteDatabase) method is called only once throughout the application lifecycle. It will be called whenever there is a first call to getReadableDatabase() or getWritableDatabase() function available in super SQLiteOpenHelper class. So SQLiteOpenHelper class call the onCreate() method after creating database and instantiate SQLiteDatabase object. Database name is passed in constructor call.

onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) is only called whenever there is a updation in existing version. So to update a version we have to increment the value of version variable passed in the superclass constructor.

In onUpgrade method we can write queries to perform whatever action is required. In most example you will see that existing table(s) are being dropped and again onCreate() method is being called to create tables again. But it’s not mandatory to do so and it all depends upon your requirements.

We have to change database version if we have added a new row in the database table. If we have requirement that we don’t want to lose existing data in the table then we can write alter table query in the onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) method.

For more details read: Insert, Read, Delete & Update Operation In SQLite

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.sqliteoperations.MainActivity"
    android:background="@android:color/holo_blue_dark">

    <TextView
        android:text="@string/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="12dp"
        android:id="@+id/textView"
        android:textSize="18sp"
        android:textStyle="bold|italic"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:gravity="center" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/editName"
        android:textStyle="bold|italic"
        android:layout_below="@+id/textView"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:hint="Enter Name"
        android:gravity="center_vertical|center" />

    <TextView
        android:text="@string/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="13dp"
        android:id="@+id/textView2"
        android:textStyle="bold|italic"
        android:textSize="18sp"
        android:layout_below="@+id/editName"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:gravity="center"
        android:hint="Enter Password" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:layout_marginTop="11dp"
        android:id="@+id/editPass"
        android:hint="Enter Password"
        android:gravity="center_vertical|center"
        android:textSize="18sp"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textAllCaps="false"
        android:textStyle="normal|bold" />

    <Button
        android:text="@string/add_user"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:textStyle="bold|italic"
        android:textSize="18sp"
        android:onClick="addUser"
        android:layout_marginLeft="28dp"
        android:layout_marginStart="28dp"
        android:layout_below="@+id/editPass"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="23dp" />

</RelativeLayout>
```

Now open app -> java -> package -> MainActivity.java and add the below code.

```
package com.example.sqliteoperations;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    EditText Name, Pass , updateold, updatenew, delete;
    myDbAdapter helper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Name= (EditText) findViewById(R.id.editName);
        Pass= (EditText) findViewById(R.id.editPass);

        helper = new myDbAdapter(this);
    }
    public void addUser(View view)
    {
        String t1 = Name.getText().toString();
        String t2 = Pass.getText().toString();
        if(t1.isEmpty() || t2.isEmpty())
        {
            Message.message(getApplicationContext(),"Enter Both Name and Password");
        }
        else
        {
            long id = helper.insertData(t1,t2);
            if(id<=0)
            {
                Message.message(getApplicationContext(),"Insertion Unsuccessful");
                Name.setText("");
                Pass.setText("");
            } else
            {
                Message.message(getApplicationContext(),"Insertion Successful");
                Name.setText("");
                Pass.setText("");
            }
        }
    }
}
```

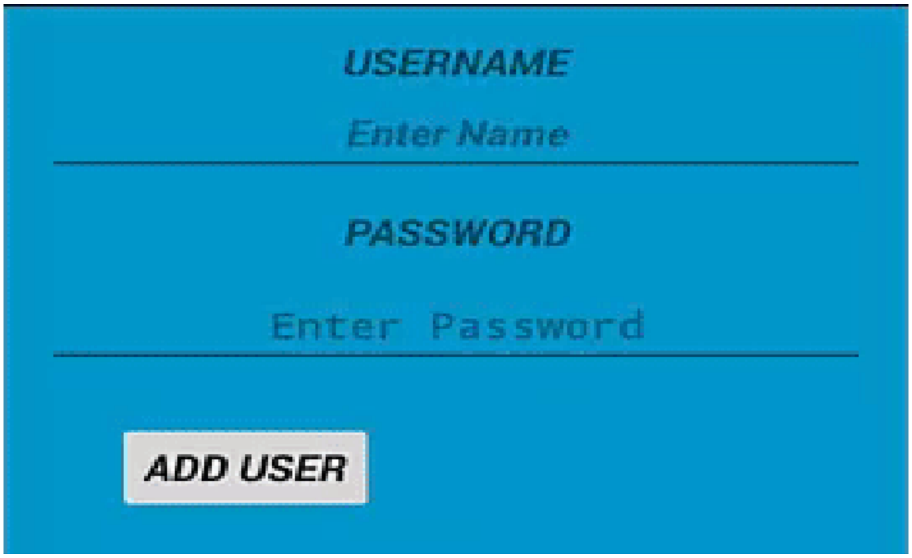
Step 4: In this step create a java class myDbAdapter. java.

```
package com.example.sqliteoperations;

import android.content.Context;
import android.widget.Toast;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class myDbAdapter {
    myDbHelper myhelper;
    public myDbAdapter(Context context)
    {
        myhelper = new myDbHelper(context);
    }

    public long insertData(String name, String pass)
    {
        SQLiteDatabase dbb = myhelper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(myDbHelper.NAME, name);
        contentValues.put(myDbHelper.MyPASSWORD, pass);
        long id = dbb.insert(myDbHelper.TABLE_NAME, null , contentValues);
        return id;
    }
static class myDbHelper extends SQLiteOpenHelper
    {
        private static final String DATABASE_NAME = "myDatabase";    // Database Name
        private static final String TABLE_NAME = "myTable";   // Table Name
        private static final int DATABASE_Version = 1;.    // Database Version
        private static final String UID="_id";     // Column I (Primary Key)
        private static final String NAME = "Name";    //Column II
        private static final String MyPASSWORD= "Password";    // Column III
        private static final String CREATE_TABLE = "CREATE TABLE "+TABLE_NAME+
                " ("+UID+" INTEGER PRIMARY KEY AUTOINCREMENT, "+NAME+"
VARCHAR(255) ,"+ MyPASSWORD+" VARCHAR(225));";
        private static final String DROP_TABLE ="DROP TABLE IF EXISTS "+TABLE_NAME;
        private Context context;

        public myDbHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_Version);
            this.context=context;
        }

        public void onCreate(SQLiteDatabase db) {

            try {
                db.execSQL(CREATE_TABLE);
            } catch (Exception e) {
                Message.message(context,""+e);
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            try {
                Message.message(context,"OnUpgrade");
                db.execSQL(DROP_TABLE);
                onCreate(db);
            }catch (Exception e) {
                Message.message(context,""+e);
            }
        }
    }
}
```

Step 5: In this step create another java class Message.class

In this just simply add toast for displaying message. This is optional, it is just added to again and again defining toast in the example.

```
package com.example.sqliteoperations;


import android.content.Context;
import android.widget.Toast;


public class Message {
    public static void message(Context context, String message) {
        Toast.makeText(context, message, Toast.LENGTH_LONG).show();
    }
}
```

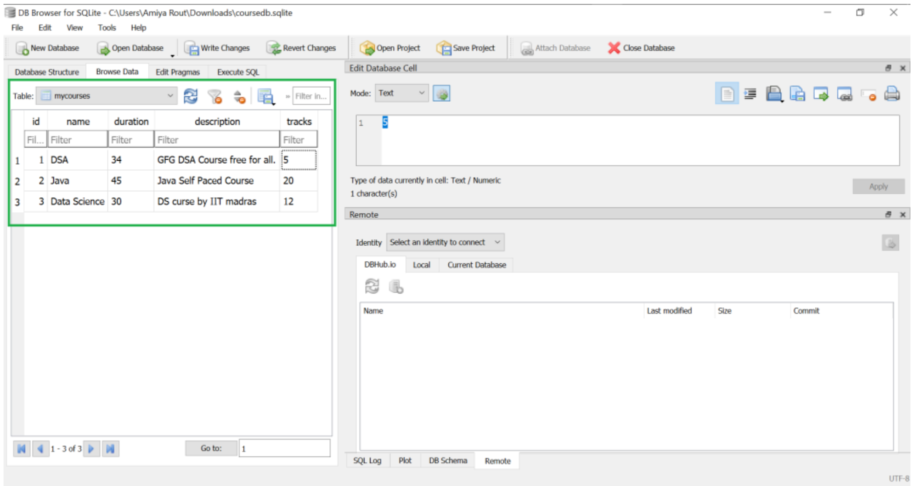


USERNAME

Enter Name

PASSWORD

Enter Password



DB Browser for SQLite - C:\Users\Amiya Rout\Downloads\coursedb.sqlite

Table: mycourses

	id	name	duration	description	tracks
1	1	DSA	34	GFG DSA Course free for all.	5
2	2	Java	45	Java Self Paced Course	20
3	3	Data Science	30	DS curse by IIT madras	12

1 - 3 of 3

Mode: Text

Type of data currently in cell: Text / Numeric
1 character(s)

Identity: Select an identity to connect

Name	Last modified	Size	Commit

Firebase: Realtime Database

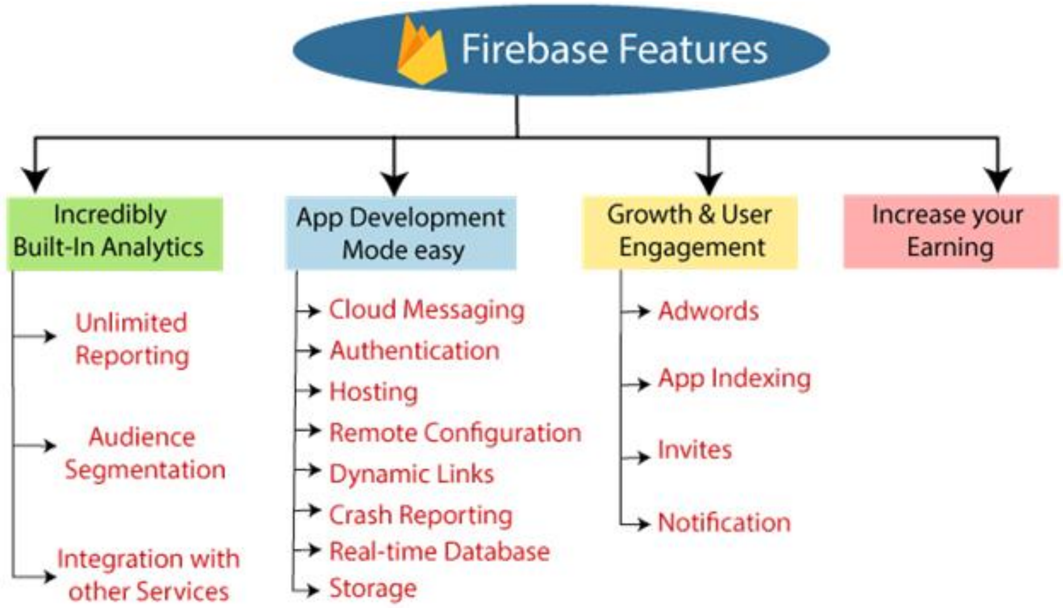
The Firebase Realtime Database is a cloud-hosted database in which data is stored as JSON. The data is synchronized in real-time to every connected client. All of our clients share one Realtime Database instances and automatically receive updates with the newest data, when we build cross-platform applications with our iOS, and JavaScript SDKs.

The Firebase Realtime Database is a NoSQL database from which we can store and sync the data between our users in real-time. It is a big JSON object which the developers can manage in real-time. By using a single API, the Firebase database provides the application with the current value of the data and updates to that data. Real-time syncing makes it easy for our users to access their data from any device, be it web or mobile.

The Realtime database helps our users collaborate with one another. It ships with mobile and web SDKs, which allow us to build our app without the need for servers. When our users go offline, the Real-time Database SDKs use local cache on the device for serving and storing changes. The local data is automatically synchronized, when the device comes online.

Features of Firebase

Firebase has several features that make this platform essential. These features include unlimited reporting, cloud messaging, authentication and hosting, etc. Let's take a look at these features to understand how these features make Firebase essential:



Incredibly Built-In Analytics

The analytics dashboard is one of the best features of Firebase, which is equipped with. It is free and can report 500 event types, each with 25 attributes. The dashboard is top-notch for observing user behavior and measuring various user characteristics. Ultimately it helps us to understand how people use our app so that we can better optimize it in the future.

Key features

Unlimited Reporting

It allows for reporting of 500 distinct events.

Audience Segmentation

We can identify custom audiences in the Firebase console based on device data, custom events, or user properties. After that, we can use these audiences that we specified with other Firebase attributes when targeting new features or notifications.

Integration with Other Services

We can integrate Firebase with other services that can utilize our business apps such as Big Query, Firebase Notifications, Firebase Remote Configuration, Firebase CrashReporting, and Google Tag Manager.

App Development Made Easy

With Firebase, we can focus our time and attention on developing the best possible applications for our business. The operation and internal functions are very solid. They have taken care of the Firebase Interface. We can spend more time in developing high-quality apps that users want to use.

Cloud Messaging

Firebase allows us to deliver and receive messages in a more reliable way across platforms.

Authentication

Firebase has little friction with acclaimed authentication.

Test Lab

Test in the lab instead on your users.

Hosting

Firebase delivers web content faster.

Remote Configuration

It allows us to customize our app on the go.

Dynamic Links

Dynamic Links are smart URLs which dynamically change behavior for providing the best experience across different platforms. These links allow app users to take directly to the content of their interest after installing the app - no matter whether they are completely new or lifetime customers.

Crash Reporting

It keeps our app stable.

Real-time Database

It can store and sync app data in real-time.

Storage

We can easily store the file in the database.

Growth and User Engagement

One of the most important aspects of application development is being able to develop and engage with users over time. Firebase has a lot of built-in features, which ensures that it is exactly what we do. With the platform leading to commercial apps, it is really at the center of what makes Firebase so great.

Here are some user interaction aspects which make development a piece of cake:

AdWords

Linking AdWords is very easy, and with it, we can segment and define our user base using Firebase Analytics. Also, it is easy to improve our targeting in marketing advertising campaigns. Some other benefits include conversion tracking, cross-network, attribution networks, and LTV (Calculating Customer Lifetime Value).

App Indexing

With app indexing, we can work on aspects like re-engaging with our app, especially by surfing the in-app content within Google search results. It will also help in ranking our application in Google search results.

Invites

It is a perfect tool for referrals and sharing. Get the help of our users to develop our app easily via email or SMS, allowing their existing users to share our app or in-app content. If we use this feature in combination with promotions, then we can also work towards acquiring new customers and retaining our existing customers.

Notifications

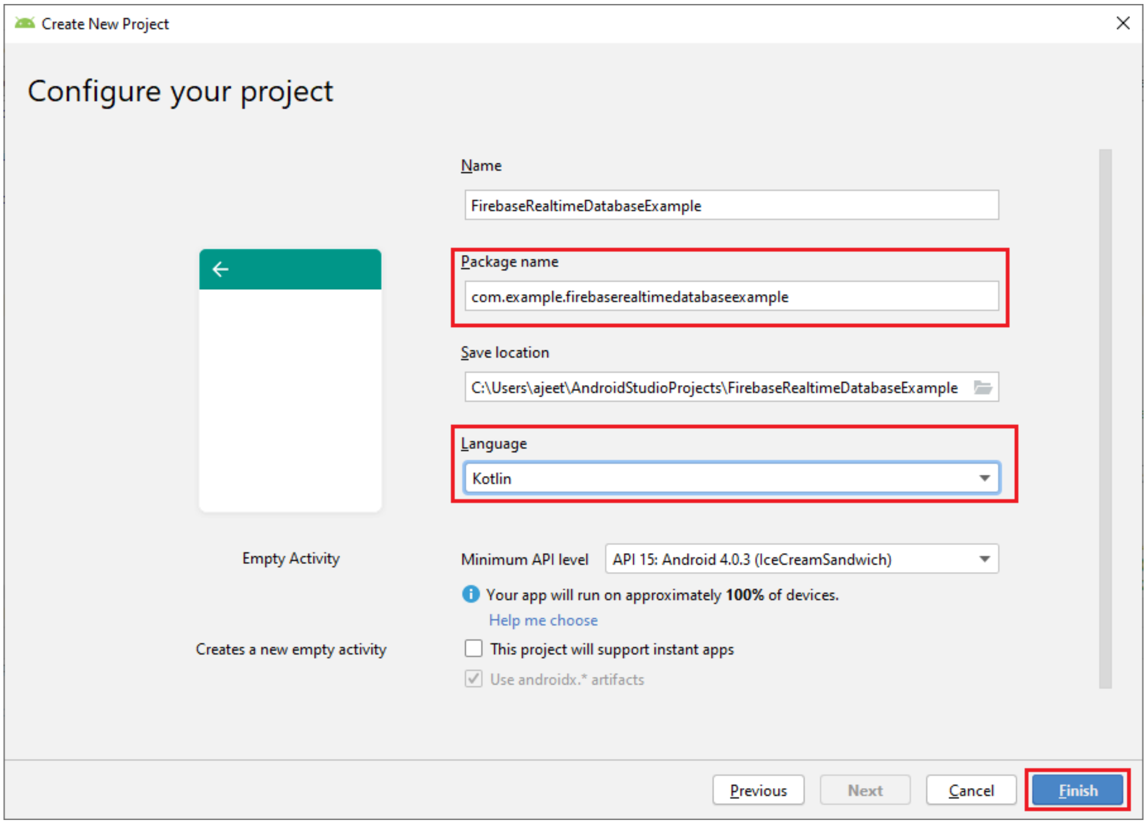
We can manage information campaigns very easily, including the ability to set and schedule messages to engage users at the right time of day. These notifications are completely free. These are unlimited for both iOS and Android. There is only one dashboard to worry about, and if we integrate with Firebase Analytics, we can use various user segmentation features.

Increase Your Earnings

Of course, the thing about having an app or any other business strategy is that we can increase our earnings. With the feature of AdMob, we can monetize our app, considering the best possible experience for our users. Showing real-time ads to millions of Google advertisers, choosing a format which suits our app, and working with over 40 top ads networks using AdMob Mediation, we can make app development well worth it, while speaking silently.

Step 1:

In the first step, we will create a new Android Studio project with an empty activity and Kotlin language and give it name **FirebaseRealtimeDatabaseExample**.



Configure your project

Name: FirebaseRealtimeDatabaseExample

Package name: com.example.firebaserealtimedatabaseexample

Save location: C:\Users\ajeet\AndroidStudioProjects\FirebaseRealtimeDatabaseExample

Language: Kotlin

Minimum API level: API 15: Android 4.0.3 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

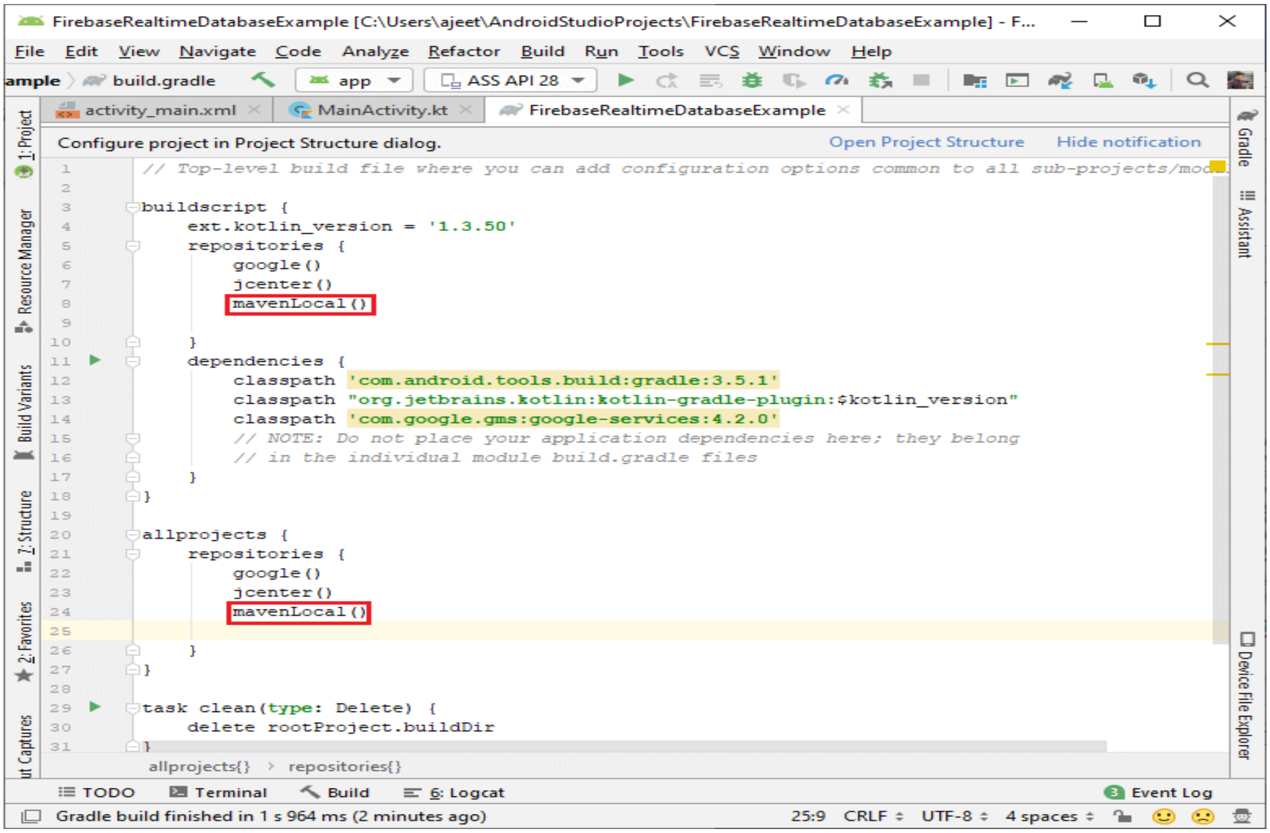
Help me choose

☐ This project will support instant apps

☑ Use androidx.* artifacts

Step 2:

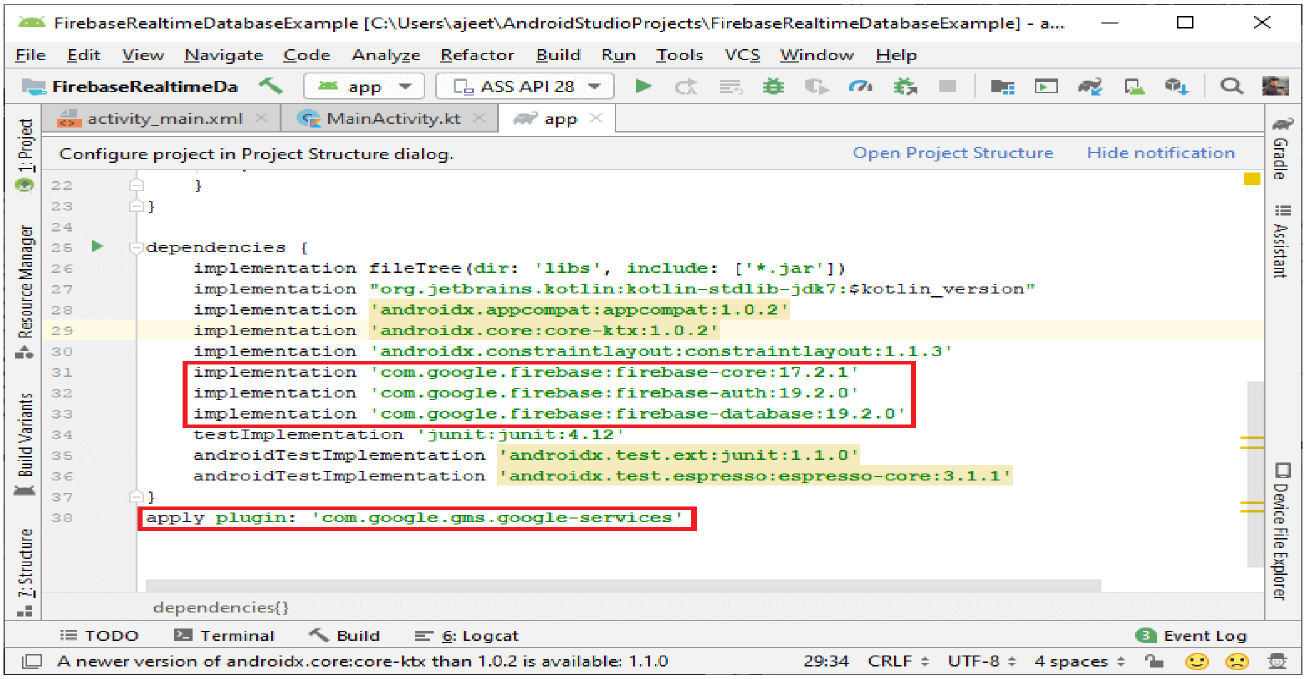
In the next step, we will connect our Android Application with the Firebase either from Firebase Assistant or manually using console. After that, we will add all the required libraries and plugin to our app.gradle file. And we will also add mavenLocal() as our repository and all projects.



Configure project in Project Structure dialog.

```
// Top-level build file where you can add configuration options common to all sub-projects/mod

buildscript {
    ext.kotlin_version = '1.3.50'
    repositories {
        google()
        jcenter()
        mavenLocal()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.1'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
        classpath 'com.google.gms:google-services:4.2.0'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        mavenLocal()

    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Gradle build finished in 1 s 964 ms (2 minutes ago)



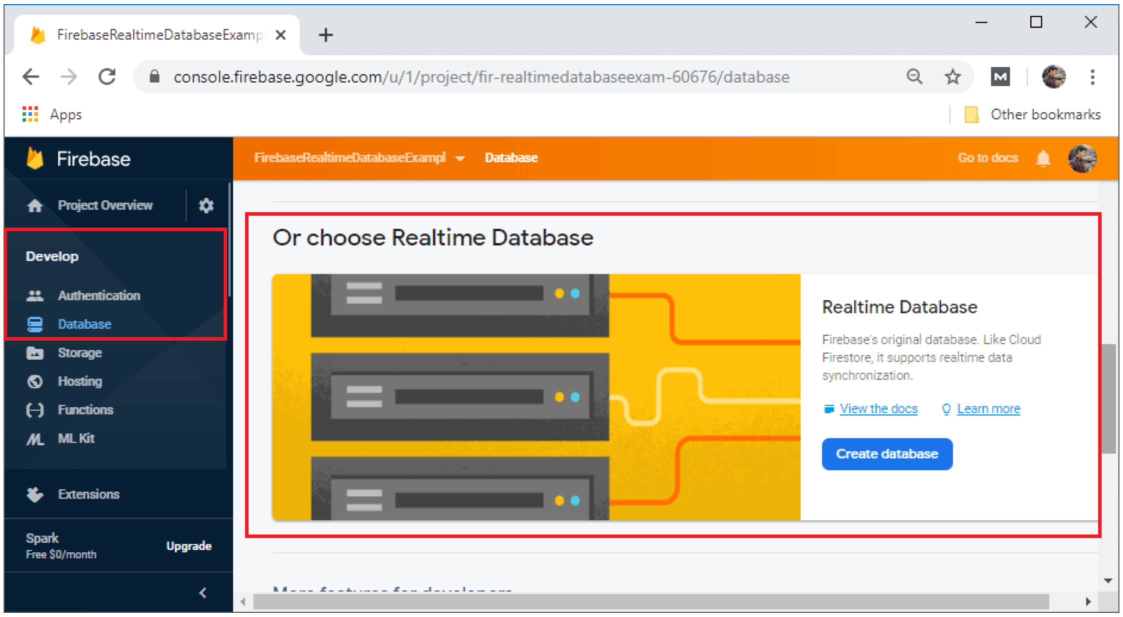
Configure project in Project Structure dialog.

```
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.core:core-ktx:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.firebase:firebase-core:17.2.1'
    implementation 'com.google.firebase:firebase-auth:19.2.0'
    implementation 'com.google.firebase:firebase-database:19.2.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
}
apply plugin: 'com.google.gms.google-services'
```

A newer version of androidx.core:core-ktx than 1.0.2 is available: 1.1.0

Step 3:

In the next step, we will go to the Firebase console and look at the Real-time database. In Developers-> Database, there will be two options, i.e., cloud Firestore and Real-time database.



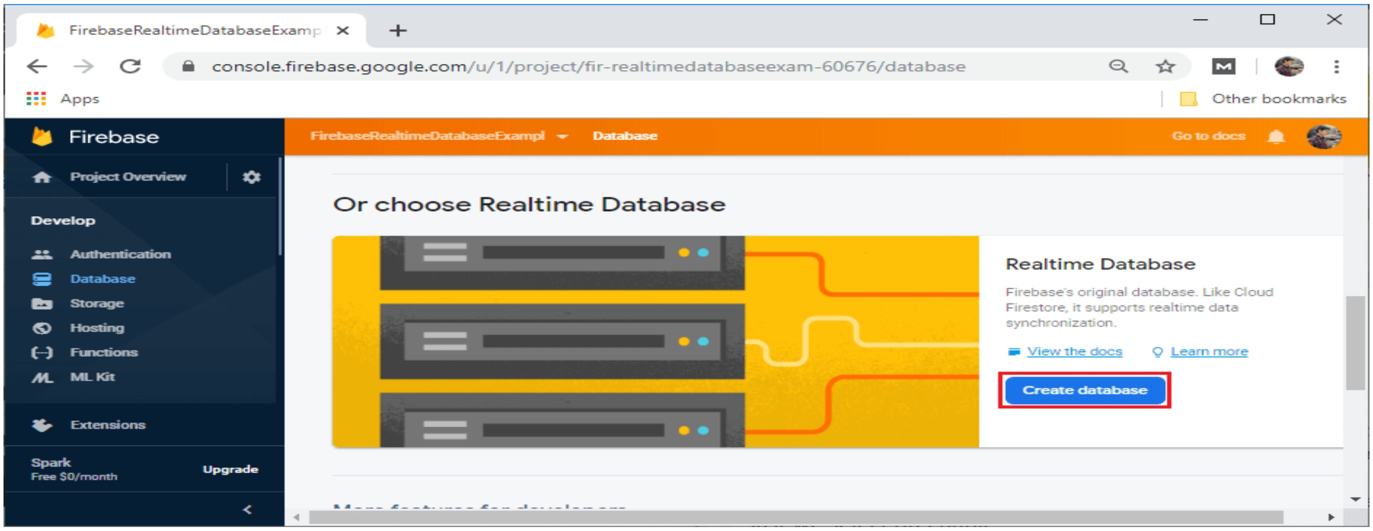
Or choose Realtime Database

Realtime Database

Firebase's original database. Like Cloud Firestore, it supports realtime data synchronization.

Step 4:

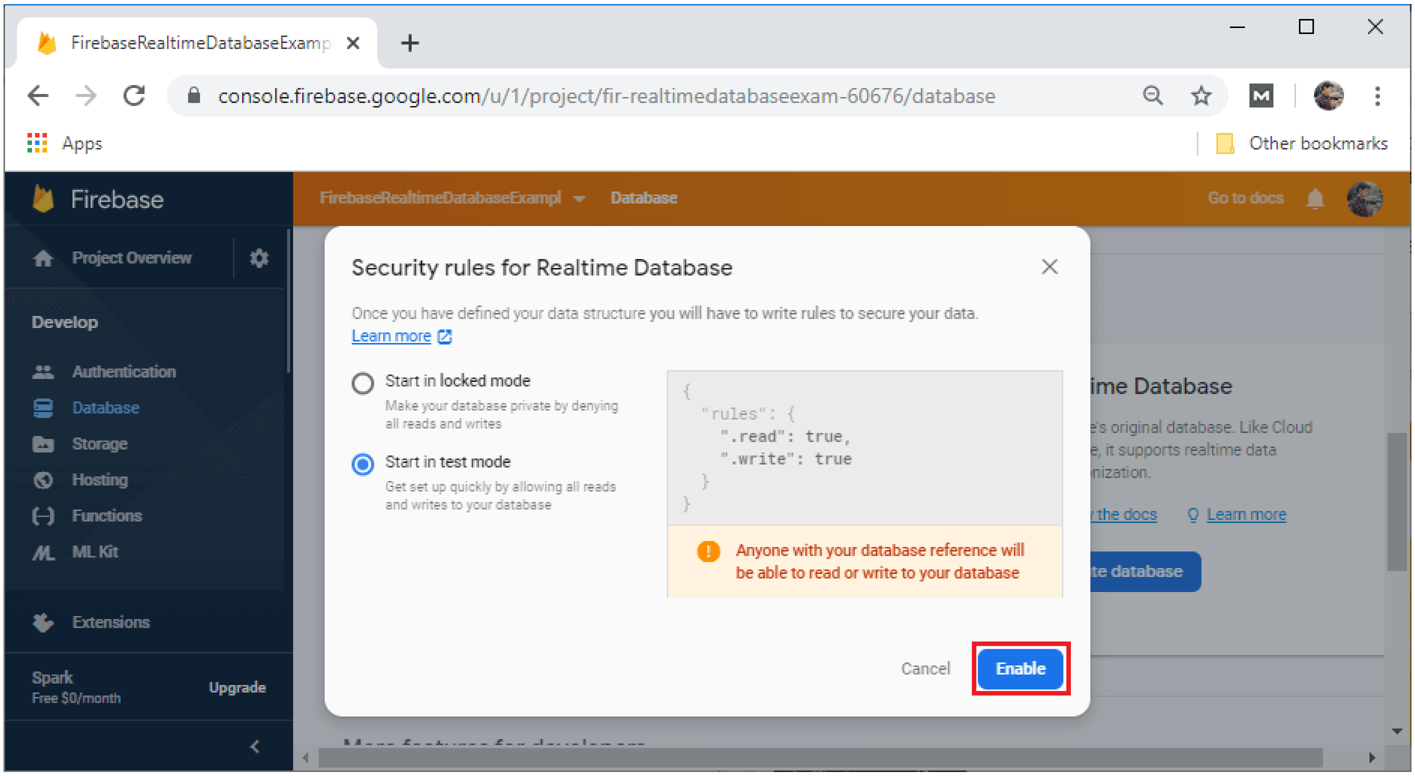
In the next step, we will create a database by clicking on the Create database. After clicking on Create database, a popup box is open where we actually create a database with specific rules. We will talk about these rules later in this section. But for now, we will select start in test mode where anybody can access our data, and later we change these rules. And last we select on Enable.



Or choose Realtime Database

Realtime Database

Firebase's original database. Like Cloud Firestore, it supports realtime data synchronization.



Security rules for Realtime Database

Once you have defined your data structure you will have to write rules to secure your data. Learn more

○ Start in locked mode
Make your database private by denying all reads and writes

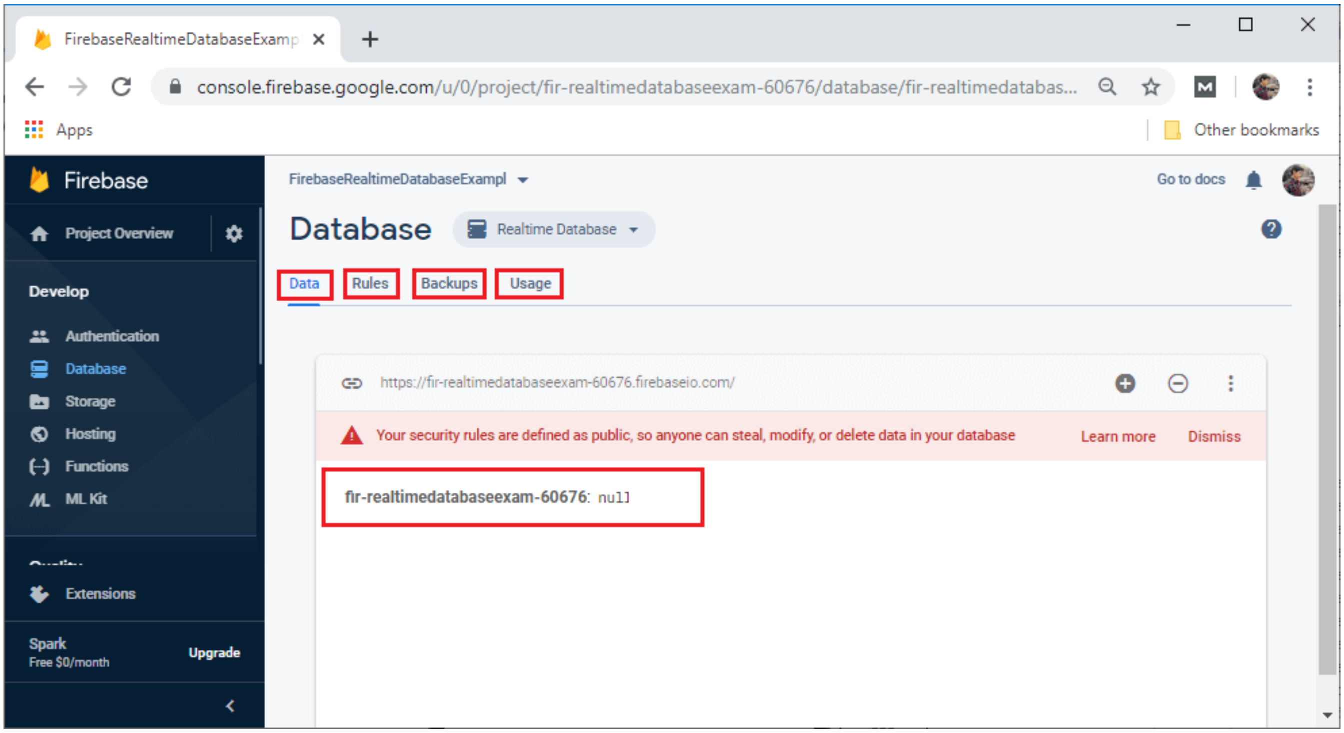
● Start in test mode
Get set up quickly by allowing all reads and writes to your database

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Anyone with your database reference will be able to read or write to your database

Step 5:

After clicking on Enable, the real-time database will be enabled with a database by default. Here, we have Data, Rules, Backups, and, Usage for data storing, security rules, backups, and usage, respectively.



https://fir-realtimedatabaseexam-60676.firebaseio.com/

Your security rules are defined as public, so anyone can steal, modify, or delete data in your database

fir-realtimedatabaseexam-60676: null

Before understanding the next steps, we will talk about the Firebase Database Rules.

The Real-time database provides a declarative rules language. It defines how our data should be structured, how it should be indexed, and when our data can be read from and written to. By default, read and write access to our database is restricted, so only authenticated users can read or write data.

To get started without setting up Authentication, we can configure our rules for public access. These rules make our database open to anyone, even people not using our app, read and write access to our database.

```
{
 "rules": {
   ".read": true,
   ".write": true
  }
}
```

If we want to allow authenticated users for accessing read and write to our database, then we will use the following rules:

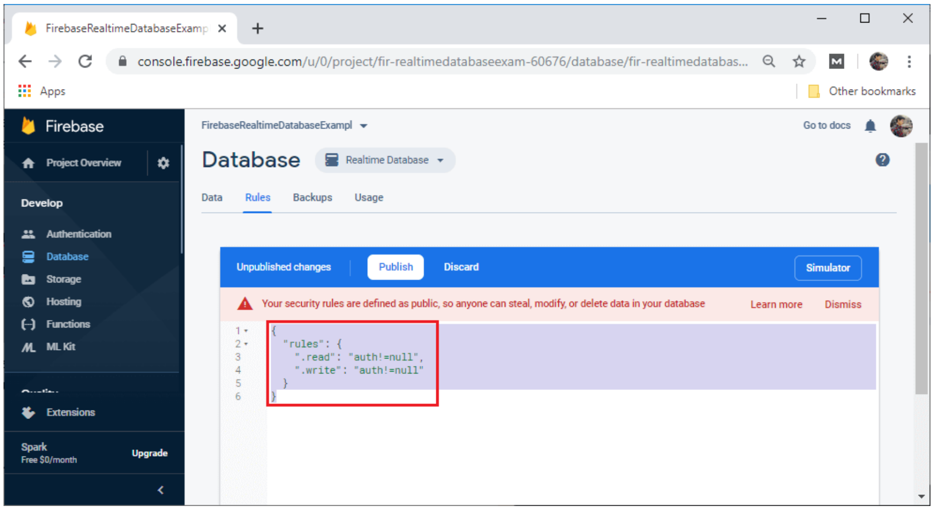
```
{
 "rules": {
   ".read": "auth!=null",
   ".write": "auth!=null"
```

```
  }
}
```

This will make sure that user only who have been authenticated using firebase can read and write to our database.

Step 6:

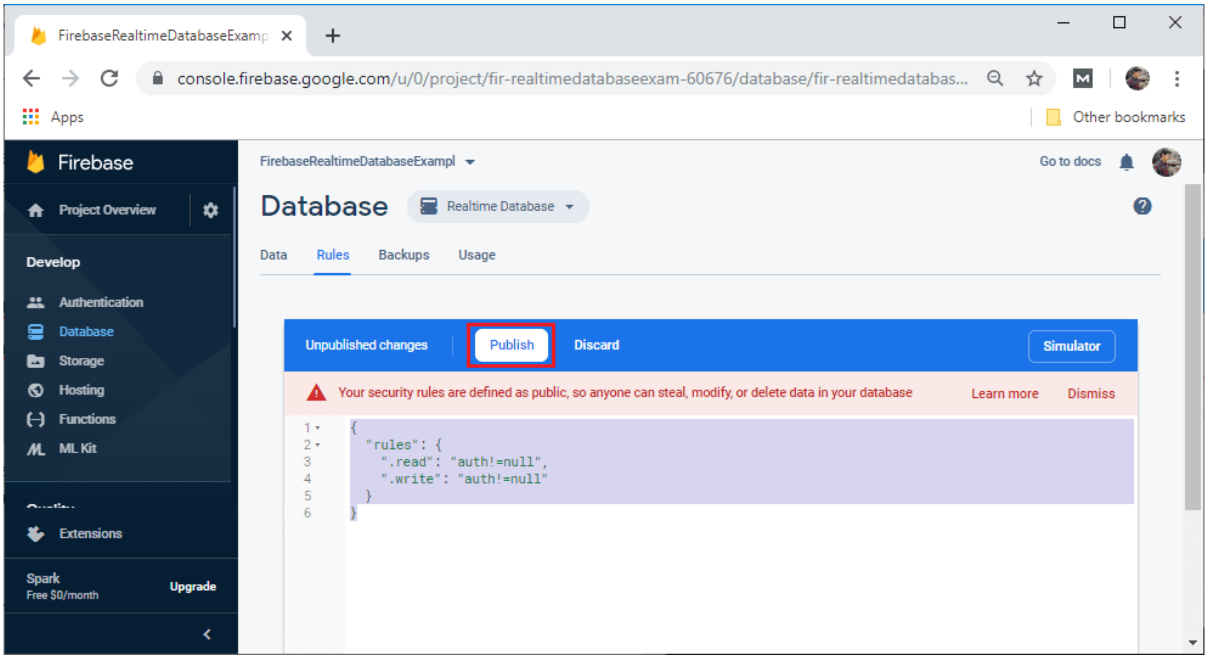
In the next step, we will go to the console and go to database rules and modify these rules to authenticated users.



Your security rules are defined as public, so anyone can steal, modify, or delete data in your database

```
{
  "rules": {
    ".read": "auth!=null",
    ".write": "auth!=null"
  }
}
```

After performing the required changes in the rules, we will publish them.



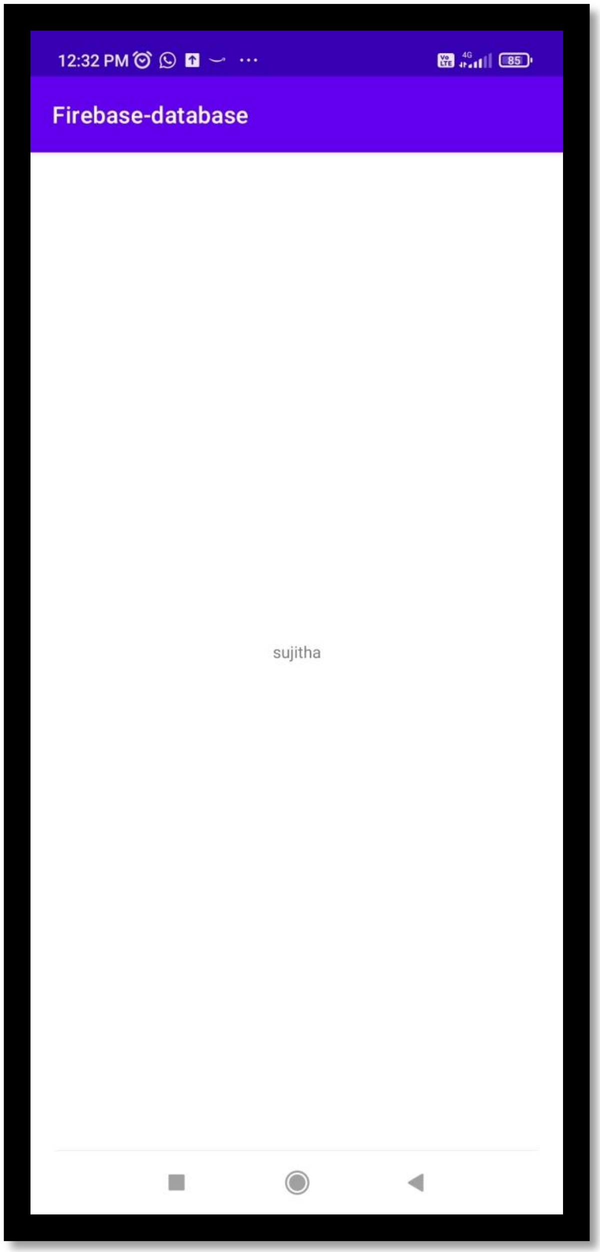
Your security rules are defined as public, so anyone can steal, modify, or delete data in your database

```
{
  "rules": {
    ".read": "auth!=null",
    ".write": "auth!=null"
  }
}
```

Now, our database is set with specific rules, and we can use it now. In the next section, we will learn how we perform read and write operations in a Real-time database.

Example:

```
package com.example.firebase_database;

import android.os.Bundle;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {
    DatabaseReference ref;
    TextView txt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txt=(TextView)findViewById(R.id.tv1);
        ref= FirebaseDatabase.getInstance().getReference().child("student").child("name");
        ref.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                String str;
                str=dataSnapshot.getValue(String.class);
                //Toast.makeText(getApplicationContext(),str,Toast.LENGTH_LONG).show();
                txt.setText(str);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }
}
```



sujitha

MySQL-PHP

mysql

MySQL is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation.

- The data in a MySQL database are stored in tables which consists of columns and rows.
- MySQL is a database system that runs on a server.

- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, and easy to use database system.It uses standard SQL
- MySQL compiles on a number of platforms.

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

Open a Connection to MySQL

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Object-Oriented:

```
$conn->close();
```

Create a MySQL Database Using MySQLi and PDO

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}
$conn->close();
?>
```

Create a MySQL Table Using MySQLi

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

Insert Data Into MySQL Using MySQLi and PDO

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

UNIT-IV

SMALL COMPUTING TECNOLOGY AND SENSORS 9

Audio, Video Using the Camera - Telephony And SMS - Bluetooth Networks - Managing network connectivity - WI-FI - **Sensors**-Sensors and the Sensor Manager - Interpreting sensor values-Using the compass, Accelerometer and Orientation sensor.

Android MediaPlayer Class

In android, by using **MediaPlayer** class we can access audio or video files from application (raw) resources, standalone files in file system or from a data stream arriving over a network connection and play audio or video files with the multiple playback options such as play, pause, forward, backward, etc.

Audio algorithm:-

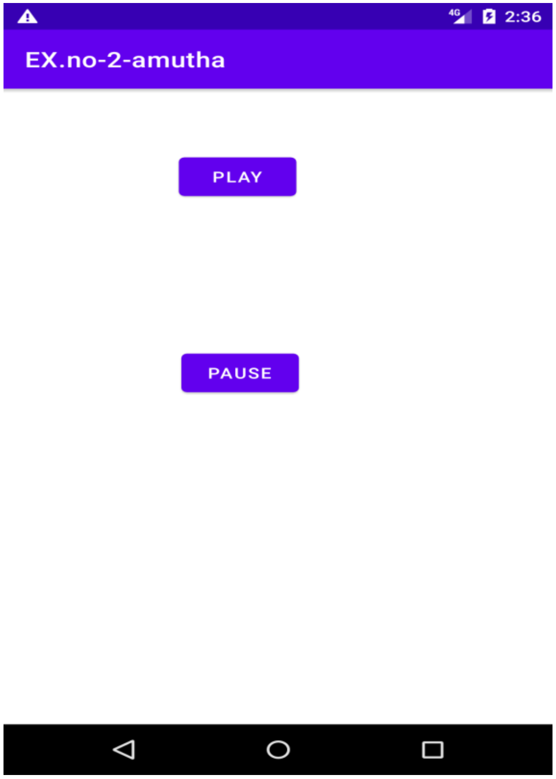
Step 1: Open an empty activity and give a name(Musicplayer)then click finish.
Step 2: In project, right click the res folder(directory),select new directory &give a name raw.
Step 3: Download any mp3 (music) file and copy the file.
Step 4: In raw directory , right click and paste the mp3 file, then click refactor.
Step 5: In activity-main.xml, drag and drop 2 buttons in design view ,and change the text of 2 buttons ,play and pause respectively.
Step 6: Add the code in mainactivity.java ,for play and pause button.
Step7: Run the app ,check the output in emulator, the audio will play ,when click the play button.
Step 8: Stop or pause the audio when click the pause button.
Step 9: Save and exit the application

Activity_Main.XML

```
<?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">

    <Button
        android:id="@+id/start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="131dp"
        android:layout_marginTop="158dp"
        android:layout_marginEnd="192dp"
        android:layout_marginBottom="69dp"

        android:text="@string/play"
        app:layout_constraintBottom_toTopOf="@+id/stop"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="133dp"
        android:layout_marginTop="69dp"
        android:layout_marginEnd="190dp"
        android:layout_marginBottom="408dp"

        android:text="@string/pause"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/start" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
package com.example.exno_2_amutha;

import androidx.appcompat.app.AppCompatActivity;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private MediaPlayer mediaPlayer;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mediaPlayer=MediaPlayer.create(this,R.raw.sound);
        Button playbutton=(Button) findViewById(R.id.start);
        playbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mediaPlayer.start();
                mediaPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
                    @Override
                    public void onCompletion(MediaPlayer mediaPlayer) {
                        Toast.makeText(MainActivity.this,"The Song is
Over",Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
        Button pausebutton=(Button) findViewById(R.id.stop);
        pausebutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mediaPlayer.pause();
            }
        });

    }


}
```



Video

Step 1: Create a new project ,open an empty activity named by videoplayer , then click finish.

Step 2: Remove text view in the code view of activity-main.xml

Step 3: Drag and drop the relative layout.
Step 4: In designview ,drag and drop the button and give a text as play , and give android:onclick =”videoplay”, and change the id.
Step 5:Click widgets in palette , drag and drop the videoview and change the id.
Step 6:In project right click yes ,select new. Select resource->click new resource folder and click change the folder location. src/ main/res/raw
Step 7: Download any mp4, or webm file, copy that file.
Step 8: Right click raw folder and paste it then click refactor.
Step 9: Write the code in mainactivity.java
Step 10: Run the app

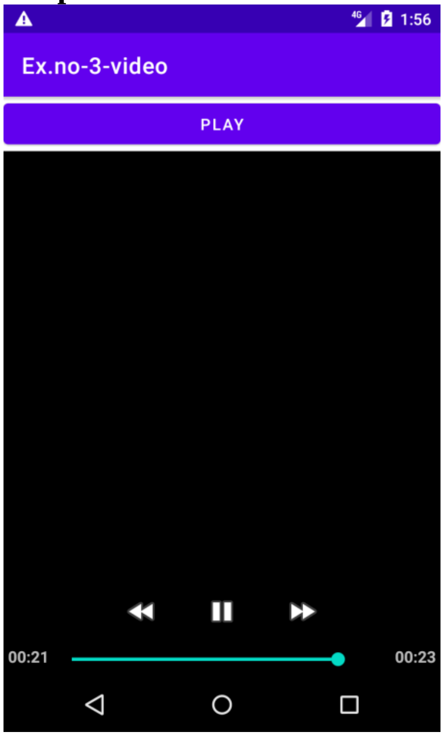
Acivity_main.xml

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:onClick="videoplay"
        android:layout_height="wrap_content"
        android:text="play" />

    <VideoView
        android:id="@+id/videoView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Main_acivity.java

```
package com.example.exno_3_video;
import androidx.appcompat.app.AppCompatActivity;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.MediaController;
import android.widget.VideoView;
public class MainActivity extends AppCompatActivity {
    Button clk;
    VideoView videov;
    MediaController mediaC;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        clk=(Button)findViewById(R.id.button);
        videov=(VideoView)findViewById(R.id.videoView);
        mediaC=new MediaController(this);
    }
    public void videoplay(View view)
    {

        String videopath="android.resource://com.example.exno_3_video/"+R.raw.cloud;
        Uri uri=Uri.parse(videopath);
        videov.setVideoURI(uri);
        videov.setMediaController(mediaC);
        mediaC.setAnchorView(videov);
        videov.start();



    }
}
```

Output:



Camera

Camera is a hardware device that allows capturing pictures and videos in your applications. Follow this tutorial to easily understand how to use a camera in your own Android App.

The Android framework provides the facility of working with Camera in two ways:

1.By using existing camera application(camera intent)

2.By using Camera Api

#1 Using Camera By Using Camera Application

We can capture pictures without using the instance of Camera class. Here you will use an intent action type of MediaStore.ACTION_IMAGE_CAPTURE to launch an existing Camera application on your phone. In Android MediaStore is a type of DataBase which stores pictures and videos in android.

```
Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
```

#2 Using Camera By using Camera Api

This class is used for controlling device cameras. It can be used to take pictures when you are building a camera application.

Camera API works in following ways:

1.**Camera Manager**: This is used to get all the cameras available in the device like front camera back camera each having the camera id.

2.**CameraDevice**: You can get it from Camera Manager class by its id.

3.**CaptureRequest**: You can create a capture request from camera device to capture images.

4.**CameraCaptureSession**: To get capture request's from Camera Device create a CameraCaptureSession.

5.**CameraCaptureSession.CaptureCallback**: This is going to provide the Capture session results.

Camera Permission Declarations In Manifest

First, you should declare the Camera requirement in your Manifest file if Camera is compulsory for your application and you don't want your application to be installed on a device that does not support Camera.

Before you start development on your application you have to make sure that your Manifest has appropriate declarations in it that will allow you to use Camera feature in your Application.

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Camera Example in Android Studio By Using Camera Intent and storing the camera Image in Local DataBase

In this Camera in Android example, I will show you how to capture the image from device camera using Intent and store that camera image in the local database and display a toast when an image is stored successfully or when some error occurred while storing an image.

activity_main.xml

Drag one imageview and one button from the pallete, now the xml file will look like this:

File: activity_main.xml

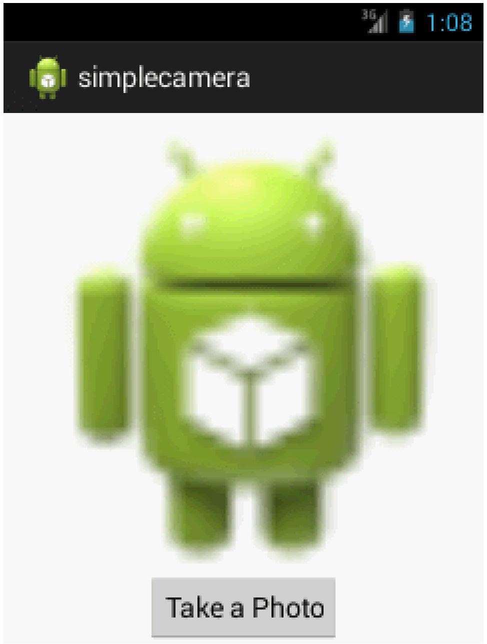
```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

   <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="Take a Photo" >
    </Button>

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_above="@+id/button1"
        android:layout_alignParentTop="true"
        android:src="@drawable/ic_launcher" >
    </ImageView>
</RelativeLayout>
```

Activity class

Let's write the code to capture image using camera and displaying it on the image view.

File: MainActivity.java

```java
package com.example.simplecamera;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends Activity {
    private static final int CAMERA_REQUEST = 1888;
    ImageView imageView;
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageView = (ImageView) this.findViewById(R.id.imageView1);
        Button photoButton = (Button) this.findViewById(R.id.button1);

        photoButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
  Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, CAMERA_REQUEST);
        }
       });
      }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
     if (requestCode == CAMERA_REQUEST) {
      Bitmap photo = (Bitmap) data.getExtras().get("data");
      imageView.setImageBitmap(photo);
     }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
       // Inflate the menu; this adds items to the action bar if it is present.
       getMenuInflater().inflate(R.menu.activity_main, menu);
       return true;
   }

}
```



Telephony and SMS

SMSManager class manages operations like sending a text message, data message, and multimedia messages (MMS). For sending a text message method **sendTextMessage()** is used likewise for multimedia message **sendMultimediaMessage()** and for data message **sendDataMessage()** method is used. The details of each function are:

sendTextMessage()	sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent, long messageId)
sendDataMessage()	sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)
sendMultimediaMessage()	sendMultimediaMessage(Context context, Uri contentUri, String locationUrl, Bundle configOverrides, PendingIntent sentIntent

Below is an example of a basic application that sends a text message.

Step 1: Create a new Android Application.
Step 2: Go to ***AndroidManifest.xml***.
app->Manifest->AndroidManifest.xml

Step 3: In ***AndroidManifest.xml*** add the permission to send SMS. It will permit an android application to send SMS.

```
<uses-permission android:name=” android.permission.SEND_SMS ” />
```

Step 4: Open ***activity_main.xml*** from the following address and add the below code. Here, in a Linear Layout, two edittext for taking phone number and a text message and a button for sending the message is added.
app->res->layout->activitymain.xml

```
<?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"

        xmlns:tools="http://schemas.android.com/tools"

        android:layout_width="match_parent"

        android:orientation="vertical"

        android:layout_marginTop="140dp"
```

```
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Enter number"
        android:inputType="textPersonName" />
    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Enter message"
        android:inputType="textPersonName" />
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="60dp"
        android:layout_marginRight="60dp"
        android:text="SEND" />
</LinearLayout>
```

Since the Send button is for sending the message so **onClickListener** is added with the button. Now create two string variables and store the value of editText phone number and message into them using method **getText()** (before assigning convert them to a string using **toString()** method). Now in try block create an instance of SMSManager class and get the SMSManager associated with the default subscription id. Now call method **sendTextMessage()** to send message.

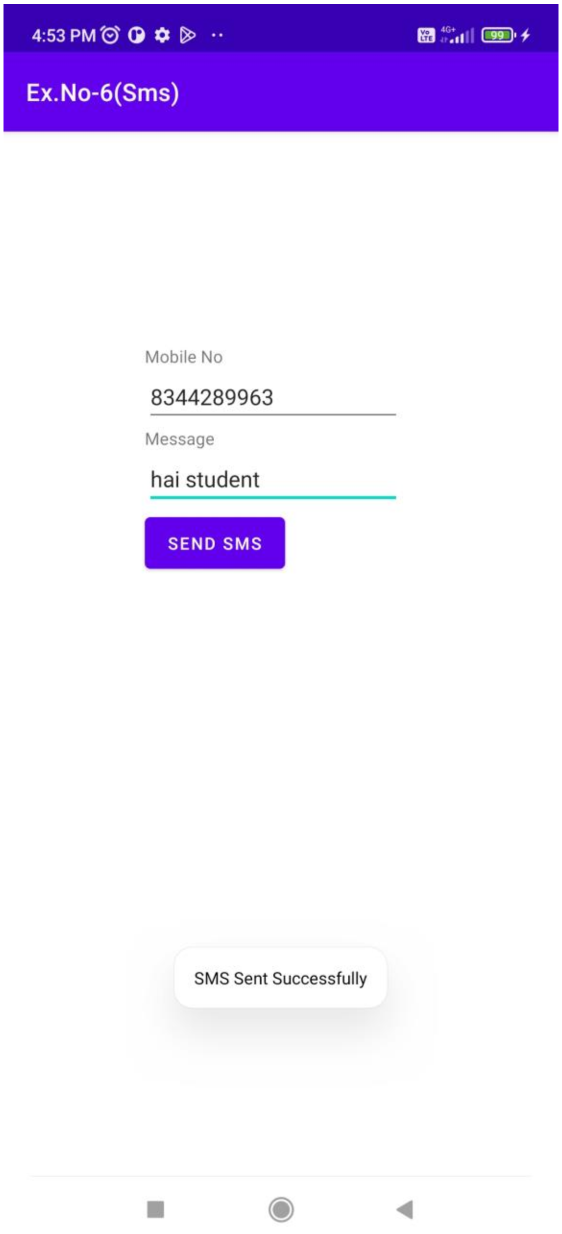
```
SmsManager smsManager=SmsManager.getDefault();
smsManager.sendTextMessage(number,null,msg,null,null);
```

```
package com.example.gfg;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText phonenumber,message;
    Button send;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        send=findViewById(R.id.button);
        phonenumber=findViewById(R.id.editText);
        message=findViewById(R.id.editText2);
        send.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                String number=phonenumber.getText().toString();
                String msg=message.getText().toString();
                try {
                    SmsManager smsManager=SmsManager.getDefault();
            smsManager.sendTextMessage(number,null,msg,null,null);
                    Toast.makeText(getApplicationContext(),"Message
Sent",Toast.LENGTH_LONG).show();
                }catch (Exception e)
                {
                    Toast.makeText(getApplicationContext(),"Some fiedls is
Empty",Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```

Note: *For running application in Android device enable the SMS permission for the app. Goto permissions->SMS->YourApp and enable permission.*

Output:

Sending the message



Mobile No
8344289963

Message
hai student

SMS Sent Successfully

sing Built-in Intent to send SMS

You can use Android Intent to send SMS by calling built-in SMS functionality of the Android. Following section explains different parts of our Intent object required to send an SMS.

Intent Object - Action to send SMS

You will use **ACTION_VIEW** action to launch an SMS client installed on your Android device. Following is simple syntax to create an intent with ACTION_VIEW action.

```
Intent smsIntent = new Intent(Intent.ACTION_VIEW);
```

Intent Object - Data/Type to send SMS

To send an SMS you need to specify **smsto:** as URI using setData() method and data type will be to **vnd.android-dir/mms-sms** using setType() method as follows −

```
smsIntent.setData(Uri.parse("smsto:"));
smsIntent.setType("vnd.android-dir/mms-sms");
```

Intent Object - Extra to send SMS

Android has built-in support to add phone number and text message to send an SMS as follows −

```
smsIntent.putExtra("address"  , new String("0123456789;3393993300"));
smsIntent.putExtra("sms_body"  , "Test SMS to Angilla");
```

Here address and sms_body are case sensitive and should be specified in small characters only. You can specify more than one number in single string but separated by semi-colon (;).

Example

Following example shows you in practical how to use Intent object to launch SMS client to send an SMS to the given recipients.

To experiment with this example, you will need actual Mobile device equipped with latest Android OS, otherwise you will have to struggle with emulator which may not work.

Step	Description
1	You will use Android studio IDE to create an Android application and name it as *tutorialspoint* under a package *com.example.tutorialspoint.*
2	Modify *src/MainActivity.java* file and add required code to take care of sending SMS.
3	Modify layout XML file *res/layout/activity_main.xml* add any GUI component if required. I'm adding a simple button to launch SMS Client.
4	No need to define default constants.Android studio takes care of default constants.
5	Modify *AndroidManifest.xml* as shown below
6	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.tutorialspoint/MainActivity.java**.

```
package com.example.tutorialspoint;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
```

```java
import android.widget.Toast;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBtn = (Button) findViewById(R.id.button);
      startBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            sendSMS();
         }
      });
   }

   protected void sendSMS() {
      Log.i("Send SMS", "");
      Intent smsIntent = new Intent(Intent.ACTION_VIEW);

      smsIntent.setData(Uri.parse("smsto:"));
      smsIntent.setType("vnd.android-dir/mms-sms");
      smsIntent.putExtra("address"  , new String ("01234"));
      smsIntent.putExtra("sms_body"  , "Test ");

      try {
         startActivity(smsIntent);
         finish();
         Log.i("Finished sending SMS...", "");
      } catch (android.content.ActivityNotFoundException ex) {
         Toast.makeText(MainActivity.this,
         "SMS faild, please try again later.", Toast.LENGTH_SHORT).show();
      }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is
present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

Here abc indicates about tutorialspoint logo

```xml
<?xml version="1.0" encoding="utf-8"?>
<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="@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">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Drag and Drop Example"
      android:id="@+id/textView"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials Point "
      android:id="@+id/textView2"
      android:layout_below="@+id/textView"
      android:layout_centerHorizontal="true"
      android:textSize="30dp"
      android:textColor="#ff14be3c" />

   <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageView"
      android:src="@drawable/abc"
      android:layout_marginTop="48dp"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Compose SMS"
      android:id="@+id/button"
      android:layout_below="@+id/imageView"
      android:layout_alignRight="@+id/textView2"
      android:layout_alignEnd="@+id/textView2"
      android:layout_marginTop="54dp"
      android:layout_alignLeft="@+id/imageView"
      android:layout_alignStart="@+id/imageView" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

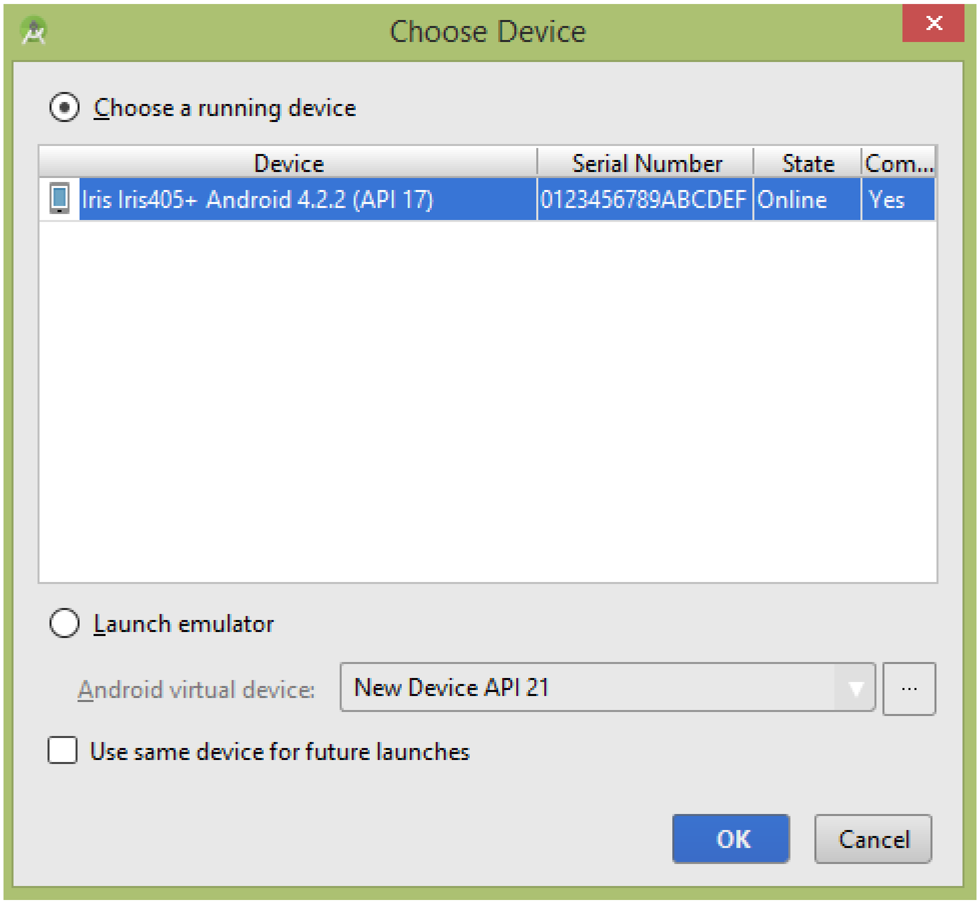
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">tutorialspoint</string>
</resources>
```

Following is the default content of **AndroidManifest.xml** −

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

```
   package="com.example.tutorialspoint" >

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >

      <activity
         android:name="com.example.tutorialspoint.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>
```

Let's try to run your **tutorialspoint** application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Android studio, open one of your project's activity files and click Run ▶icon from the toolbar. Before starting your application, Android studio will display following window to select an option where you want to run your Android application.



Choose Device

● Choose a running device

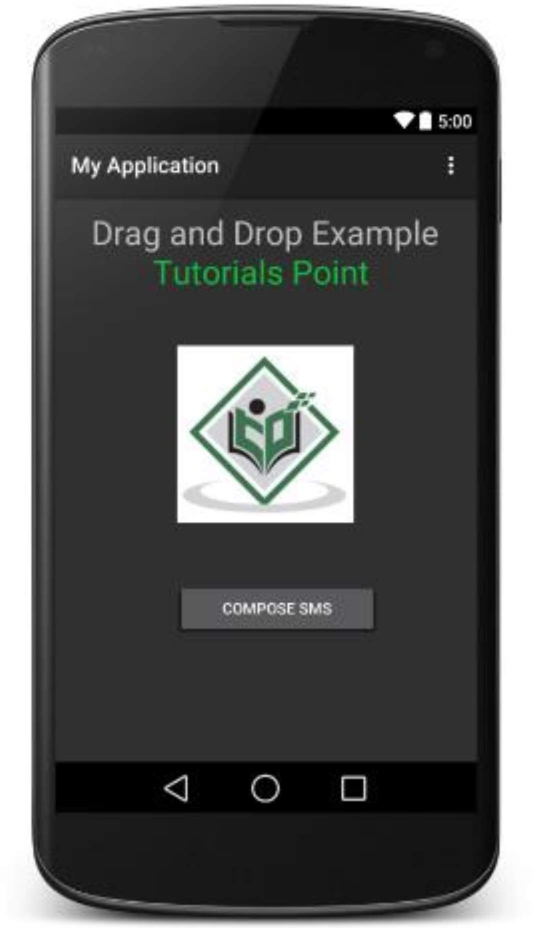
Device	Serial Number	State	Com...
Iris Iris405+ Android 4.2.2 (API 17)	0123456789ABCDEF	Online	Yes

○ Launch emulator

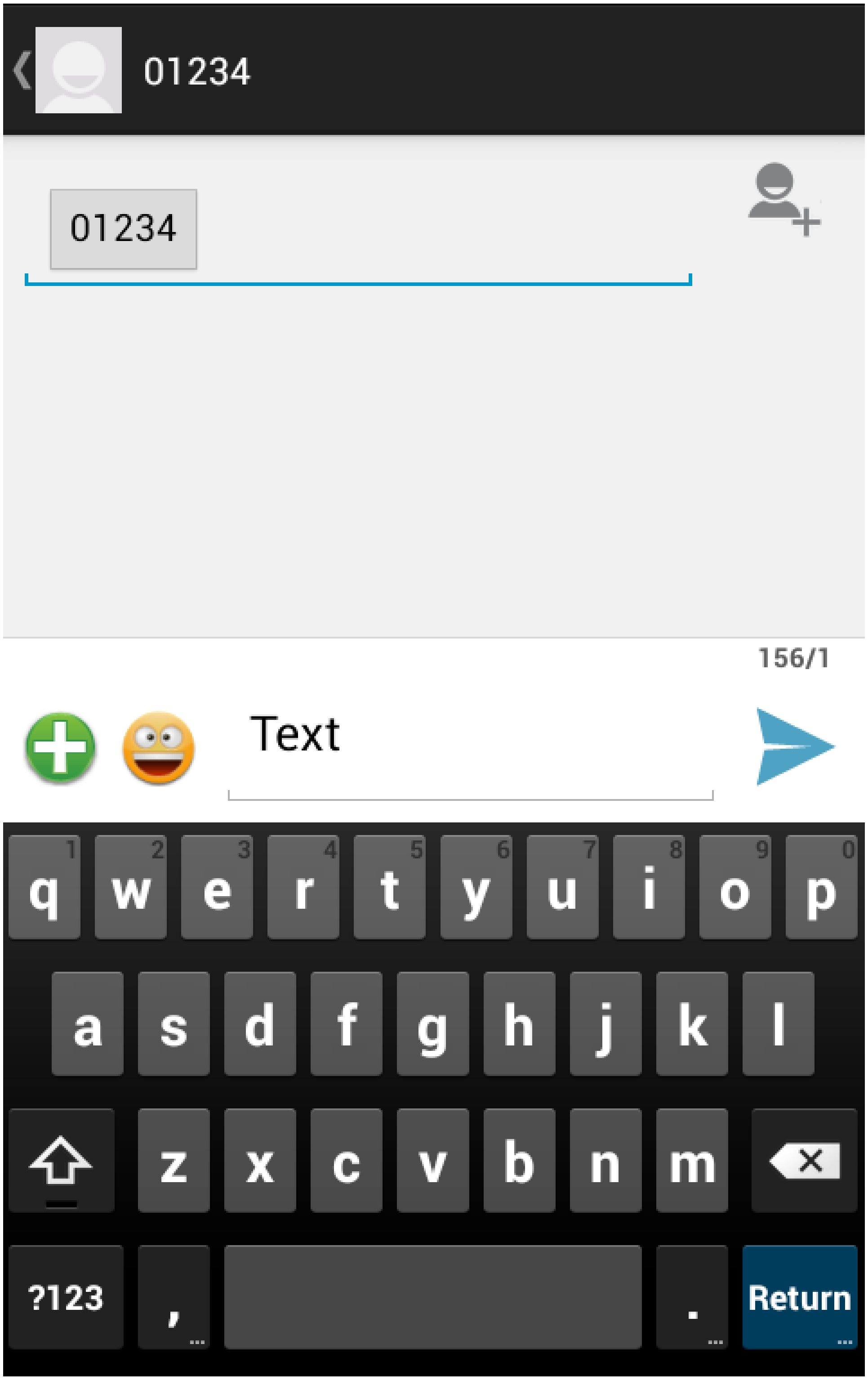
Android virtual device: New Device API 21

□ Use same device for future launches

Select your mobile device as an option and then check your mobile device which will display following screen –



Now use **Compose SMS** button to launch Android built-in SMS clients which is shown below −



You can modify either of the given default fields and finally use send SMS button to send your SMS to the mentioned recipient.

Bluetooth Networks

Bluetooth is a communication network protocol, which allows devices to connect wirelessly to exchange the data with other Bluetooth devices.

Generally, in android applications by using Bluetooth API's we can implement Bluetooth functionalities, such as searching for the available Bluetooth devices, connecting with the devices and managing the data transfer between devices within the range.

By using android Bluetooth API's in android applications, we can perform the following functionalities.

- Scan for the available Bluetooth devices within the range
- Use local Bluetooth adapter for paired Bluetooth devices
- Connect to other devices through service discovery
- Transfer data to and from other devices
- Manage multiple connections

To transfer the data between two Bluetooth devices first, they must establish a communication channel using the **pairing process**. The devices which we are going to pair must be discoverable and should accept the incoming connection requests. Generally, the devices will find discoverable devices using a **service discovery process**. Once the device accepts the pairing request, the two devices will exchange security keys to complete the bonding process and the devices will cache these security keys for later use.

Once the pairing and bonding process completes, the devices are ready to exchange the required information. When the session is complete, the device that initiated the pairing request will release the channel that linked to the discoverable device. The two devices remain bonded, so they can reconnect automatically during a future session as long as they're in the range of each other.

Android Set Bluetooth Permissions

To use Bluetooth features in our android applications, we must need to add multiple permissions, such as **BLUETOOTH** and **ACCESS_COARSE_LOCATION** or **ACCESS_FINE_LOCATION** in our manifest file.

Permission	Description
BLUETOOTH	We need this permission to perform any Bluetooth communication, such as requesting a connection, accepting a connection, and transferring data.

Permission	Description
LOCATION	We need this permission because the Bluetooth scans can be used to gather the information about the location of user.

In case, if we want to discover the available Bluetooth devices or manipulate Bluetooth settings from our app, we need to define **BLUETOOTH_ADMIN** permission.

Following is the example of defining the Bluetooth permissions in android manifest file.

```
<manifest ... >
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
...
</manifest>
```

Android BluetoothAdapter Class

In android, we can perform Bluetooth related activities by using **BluetoothAdapter** class in our applications.

By using **BluetoothAdapter** object, we can interact with device’s Bluetooth adapter to perform Bluetooth related operations. In case, if device does not contain any Bluetooth adapter, then it will return null.

Following is the code snippet to initialize **BluetoothAdapter** class and to know whether the Bluetooth is supported on the device or not.

```
BluetoothAdapter bAdapter = BluetoothAdapter.getDefaultAdapter();
if(bAdapter==null)
{
   // Device won't support Bluetooth
}
```

If you observe above code snippet, we used the **getDefaultAdapter()** method of **BluetoothAdapter** class, which will return whether the device contains Bluetooth adapter or not.

In case if **getDefaultAdapter()** method returns NULL, then the device does not support Bluetooth and we can disable all Bluetooth features.

Android Enable or Disable Bluetooth

If Bluetooth is supported but disabled, then the **isEnabled()** method will return **false** and we can request the user to enable Bluetooth without leaving our application by using **startActivityForResult()** method with **ACTION_REQUEST_ENABLE** intent action parameter.

Following is the code snippet to enable a Bluetooth by using **BluetoothAdapter** parameter **ACTION_REQUEST_ENABLE**.

```
if(!bAdapter.isEnabled())
{
   Intent eintent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
   startActivityForResult(eintent, intVal);
}
```

If you observe above code snippet, we used **startActivityForResult()** method with **ACTION_REQUEST_ENABLE** intent action parameter to enable a Bluetooth.

The second parameter **intVal** in **startActivityForResult()** method is a locally defined integer that must be greater than **0** and the system will return this parameter back to us during **onActivityResult()** implementation as a **requestCode** parameter.

To know more about to TURN ON / OFF Bluetooth in android applications, check this Android Bluetooth Turn ON / OFF with Examples.

Android Bluetooth Turn ON / OFF Example

activity_main.xml

```
<?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"
   android:paddingLeft="10dp"
   android:paddingRight="10dp">
   <Button
      android:id="@+id/btnOn"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Turn
On" android:layout_marginLeft="100dp" android:layout_marginTop="200dp" />
   <Button
      android:id="@+id/btnOFF"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btnOn"
        android:layout_toRightOf="@+id/btnOn"
        android:text="Turn OFF" />
</RelativeLayout>
```

Now open your main activity file **MainActivity.java** from **\java\com.tutlane.bluetoothexample** path and write the code like as shown below

MainActivity.java

```
package com.tutlane.bluetoothexample;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btntOn = (Button)findViewById(R.id.btnOn);
        Button btntOff = (Button)findViewById(R.id.btnOFF);
        final BluetoothAdapter bAdapter = BluetoothAdapter.getDefaultAdapter();
        btntOn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            if(bAdapter == null)
            {
                Toast.makeText(getApplicationContext(),"Bluetooth Not
Supported",Toast.LENGTH_SHORT).show();
            }
            else{
                if(!bAdapter.isEnabled()){

startActivityForResult(new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE),1);
                    Toast.makeText(getApplicationContext(),"Bluetooth Turned
ON",Toast.LENGTH_SHORT).show();
                }
            }
            }
        });
        btntOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                bAdapter.disable();
                Toast.makeText(getApplicationContext(),"Bluetooth Turned OFF",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

If you observe above code, we used **startActivityForResult()** method with BluetoothAdapter action parameter **ACTION_REQUEST_ENABLE** intent to enable or disable a Bluetooth in our application.

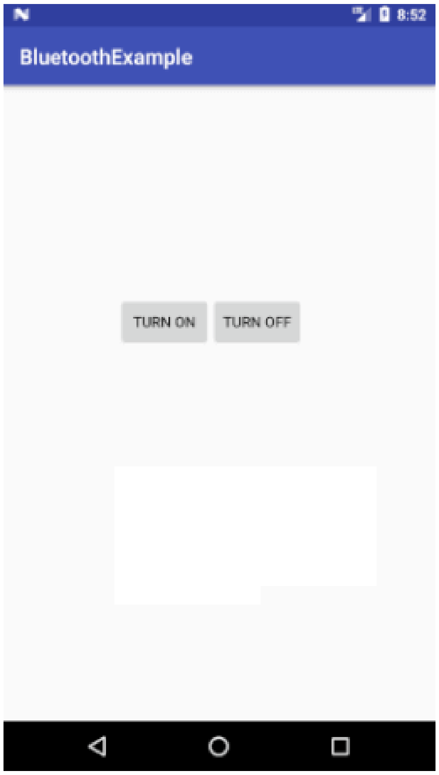
As discussed, we need to set Bluetooth permissions in android manifest file (**AndroidManifest.xml**) to access Bluetooth features in android applications. Now open android manifest file (**AndroidManifest.xml**) and write the code like as shown below

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tutlane.bluetoothexample">
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <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/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>
```

Output of Android Bluetooth Turn ON / OFF Example

When we run the above program in the android studio we will get the result as shown below.



Sensors:

Most of the android devices have built-in sensors that measure motion, orientation, and various environmental condition. The android platform supports three broad categories of sensors.

- Motion Sensors
- Environmental sensors
- Position sensors

- **Motion Sensors**
  These sensors track movement; they include accelerometers, gyroscopes and gravity sensors. They provide data on forces like acceleration and rotation that act on the sensor's three-dimensional axes.

- **Environmental Sensors**
  Barometers and thermometers are types of sensors that access environmental metrics. These sensors monitor environmental variables like air pressure and temperature.

- **Position Sensors**
  Magnetometer and orientation sensors help determine the physical position of a device.

- Each of these categories represents many specific sensors that are available on a device. You will go through them next.

Android Sensor API

We can collect raw sensor data by using Android Sensor API. Android sensor API provides many classes and interfaces. Some of the important classes and interfaces are:

1. **SensorManager Class:** Sensor manager is used to accessing various sensors present in the device.
2. **Sensor Class:** The sensor class is used to get information about the sensor such as sensor name, sensor type, sensor resolution, sensor type, etc.
3. **SensorEvent class:** This class is used to find information about the sensor.
4. **SensorEventListener interface:** This is used to perform some action when sensor accuracy changes.

Some of the sensors are hardware based and some are software based sensors. Whatever the sensor is, android allows us to get the raw data from these sensors and use it in our application. For this android provides us with some classes.

Android provides SensorManager and Sensor classes to use the sensors in our application. In order to use sensors, first thing you need to do is to instantiate the object of SensorManager class. It can be achieved as follows.

```
SensorManager sMgr;
sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);
```

The next thing you need to do is to instantiate the object of Sensor class by calling the getDefaultSensor() method of the SensorManager class. Its syntax is given below −

```
Sensor light;
light = sMgr.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Once that sensor is declared , you need to register its listener and override two methods which are onAccuracyChanged and onSensorChanged. Its syntax is as follows −

```
sMgr.registerListener(this, light,SensorManager.SENSOR_DELAY_NORMAL);
public void onAccuracyChanged(Sensor sensor, int accuracy) {
}

public void onSensorChanged(SensorEvent event) {
}
```

Example: Light Sensor App

This app will show us light intensity in our room with the help of a light sensor present in our phone.

Step by Step Implementation

Step 1: Create a New Project in your android studio

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select **Kotlin** as the programming language.

Step 2: Working with the XML file

Navigate to the **app > res > layout > activity_main.xml** and add the below code to that file.

XML

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

    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">



      <!-- Textview to show light sensor reading -->

    <TextView

        android:id="@+id/tv_text"

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="Light Sensor"

        android:textSize="20sp"

        android:textColor="@color/black"

        android:layout_centerInParent="true" />



</RelativeLayout>
```

Step 3: Working With the MainActivity.kt

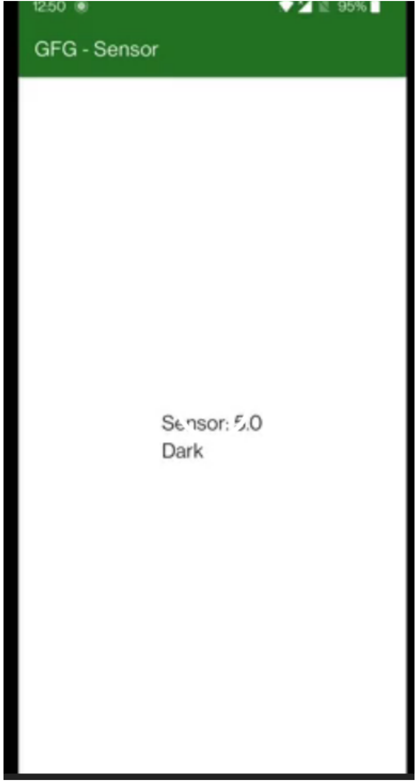
Go to the **MainActivity.kt** file and refer to the following code. Below is the code for the **MainActivity.kt** file. Comments are added inside the code to understand the code in more detail.

```
package com.mrtechy.gfg_sensor

import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatDelegate

class MainActivity : AppCompatActivity(), SensorEventListener {

      // Initialised sensorManager & two variables
      // for storing brightness value
    private lateinit var sensorManager: SensorManager
    private var brightness: Sensor? = null
    private lateinit var text: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Set default nightmode
        AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_N
O)
```

```
        // searched our textview id and stored it
        text = findViewById(R.id.tv_text)

        // setupSensor Called
        setUpSensor()
    }

    // Declared setupSensor function
    private fun setUpSensor() {
      sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
      brightness = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
    }

    // These are two methods from sensorEventListner Interface
    override fun onSensorChanged(event: SensorEvent?) {
        if (event?.sensor?.type == Sensor.TYPE_LIGHT) {
            val light1 = event.values[0]

            text.text = "Sensor: $light1\n${brightness(light1)}"
        }
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {
        return
    }

    // Created a function to show messages according to the brightness
    private fun brightness(brightness: Float): String {

        return when (brightness.toInt()) {
            0 -> "Pitch black"
            in 1..10 -> "Dark"
            in 11..50 -> "Grey"
            in 51..5000 -> "Normal"
            in 5001..25000 -> "Incredibly bright"
            else -> "This light will blind you"
        }
    }

    // This is onResume function of our app
    override fun onResume() {
        super.onResume()
        sensorManager.registerListener(this, brightness,
SensorManager.SENSOR_DELAY_NORMAL)
    }

    // This is onPause function of our app
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
}
```

Output:

Note: *App those usage sensors will only work on physical Android devices, not on any emulators.*



```
Sensor: 5.0
Dark
```

UNIT-V

ADVANCED TECHNOLOGY

Paranoid Android - Using Wake Locks - AIDL to Support IPC for Services -General API's- Payment-gateway, Android jetpack-Technology II-IOS-Introduction to Objective C-IOS features.

Paranoid Android

Paranoid Android is a custom ROM that targets to extend or enhance the system and working of existing Android.

Features of Paranoid Android

The two most popular and notable features of this operating system

are Halo and Pie. The Halo feature allows users to see notifications without leaving their current screen. On the other hand, Pie replaces the onscreen navigation button that stays away from the screen and allows users to swipe sideways to view the button. It also provides the features of accidental touch rejection and pocket lock that avoids pressing any buttons in a user's pocket. It also allows users to take a screenshot by simply swiping three fingers.

Other Features

- Built-in screen recorder.
- Fingerprint on display.
- Off-screen gestures.
- Pocket mode detects that your device is in your pocket and prevents it from working.
- Fast charging indicator.
- Navigation bar customization.
- Improve lock-screen security.
- VPN, WiFi restrictions, and per-app cellular data.
- Swipe down three fingers for a screenshot.
- Long-press the volume to skip tracks.

UNIT-V

ADVANCED ANDROID DEVELOPMENT

Topic1: Paranoid Android

It deals with

- Linux Kernel Security
- Permissions
- Declaring and Enforcing Permissions
- Enforcing Permissions for Broadcast Intents.

- Linux Kernel Security:

- Each Android Package has a unique Linux user ID assigned to it during installation.

- So that it can't affect (or be affected by) other applications.

- Because of this kernel-level security, you need to take additional steps to communicate between applications.

- Enter

  - content providers
  - broadcast intents
  - AIDL interfaces.

- Each of these mechanisms opens a tunnel through which information can flow between applications.
- Android permissions act as border guards at either and to control the traffic allowed through.

- Introducing Permissions:

- Permissions are an application-level security mechanism that restrict access to application components.

- Permissions are used to prevent 1) malicious applications from corrupting data.
  2) gaining access to sensitive information
  3) making excessive (unauthorized) use of hardware resources or external communication channels.

- The native Permission strings used by native Android activities and Services can be found as static constants in the android.Manifest.Permission class.

- Eg: Location Provider. (Find current location).

when an application package is installed, the Permissions requested in its manifest are analyzed and granted (or denied) by checks with trusted authorities and user feedback.

- unlike many existing mobile platforms, all android Permission checks are done at installation.

- Once an application is installed, the user will not be Prompted to reevaluate those Permissions.

- **Declaring and Enforcing Permissions:**

Before you can assign a Permission to an application component,

you need to define it within your manifest using the <Permission> tag.

// Declaring Permission

```
<Permission
      android:name = "com.Paad.DETONATE_DEVICE"
      android:protectionlevel = "dangerous"
       android:label = "self Destruct"
       android:description = "@string/
                               detonate_description">
</Permission>
```

- Those explains the risks of granting this Permission.

// Enforcing Permission

```
<activity
        android:name = ". MyActivity"
        android:label = " @string/app_name"
        android:Permission = " com. Paad.
                         DETONATE_DEVICE " >
</activity >
```

- Enforcing Permissions for Broadcast Intents :

As well as requiring Permissions for intents to be received by Your Broadcast Receivers , you can also attach a Permission requirement to each intent you broadcast.

when calling sendIntent , you can supply a Permission string required by Broadcast Receivers before they can receive the Intent .

```
sendBroadcast (myIntent , REQUIRED -
                              PERMISSION);
```

Topic 2: USING WAKE LOCKS

- In order to prolong battery life, over time Android devices will first dim, then turn off the screen, before turning off the CPU.

- Wake locks are a power manager system service feature, available to Android applications to control the power state of the host device.

- Wake locks can be used to ~~keep the~~

    - keep the CPU running
    - prevent the screen from dimming
    - prevent the screen from turning off &
    - prevent the keyboard backlight from turning off.

Types:

- screen wake locks
- CPU wake locks

Screen wake locks:

- used to prevent the screen from dimming during applications that are likely to involve little user interactions while users observe the screen (E.g: Playing videos)

CPU wake locks:

- used to prevent the device from going to sleep until an action is performed.
- Commonly used for

Services started within Intent Receivers, which may receive Intents while the device is asleep.

To create a wake lock, call new wakelock on the Power manager, specifying one of the following wake lock types:

► FULL_WAKE_LOCK:

Keeps the screen at full brightness, the keyboard backlight illuminated and the CPU running.

- **SCREEN_BRIGHT_WAKE_LOCK:**

  Keeps the screen at full brightness, and the CPU running.

- **SCREEN_DIM_WAKE_LOCK:**

  Keeps the screen on (but lets it dim) and the CPU running.

- **PARTAL_WAKE_LOCK:**

  Keeps the CPU running.

Pattern for creating, acquiring and releasing a wake lock.

// create

```
PowerManager pm = (PowerManager)
        getSystemService (context. POWER_SERVICE];

WakeLock wakeLock = pm. newWakeLock

 ( PowerManager. ~~PARTIAL~~ PARTIAL_WAKE_LOCK,
                            "MyWakeLock");
```

```
// acquire
```

```
wakeLock.acquire();
```

```
// release
```

```
wakeLock.release();
```

WakeLockActivity.java

```
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.os.PowerManager;
import android.os.PowerManager.WakeLock;

public class WakeLockActivity extends Activity {
 /** Called when the activity is first created. */
 protected static WakeLock wakeLock = null;

 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
  wakeLock = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK, "DoNotSleep");
  wakeLock.acquire();
 }

 @Override
 protected void onDestroy() {
  // TODO Auto-generated method stub
  super.onDestroy();
  wakeLock.release();
 }
}
```

Topic 3: AIDL to Support IPC for Services

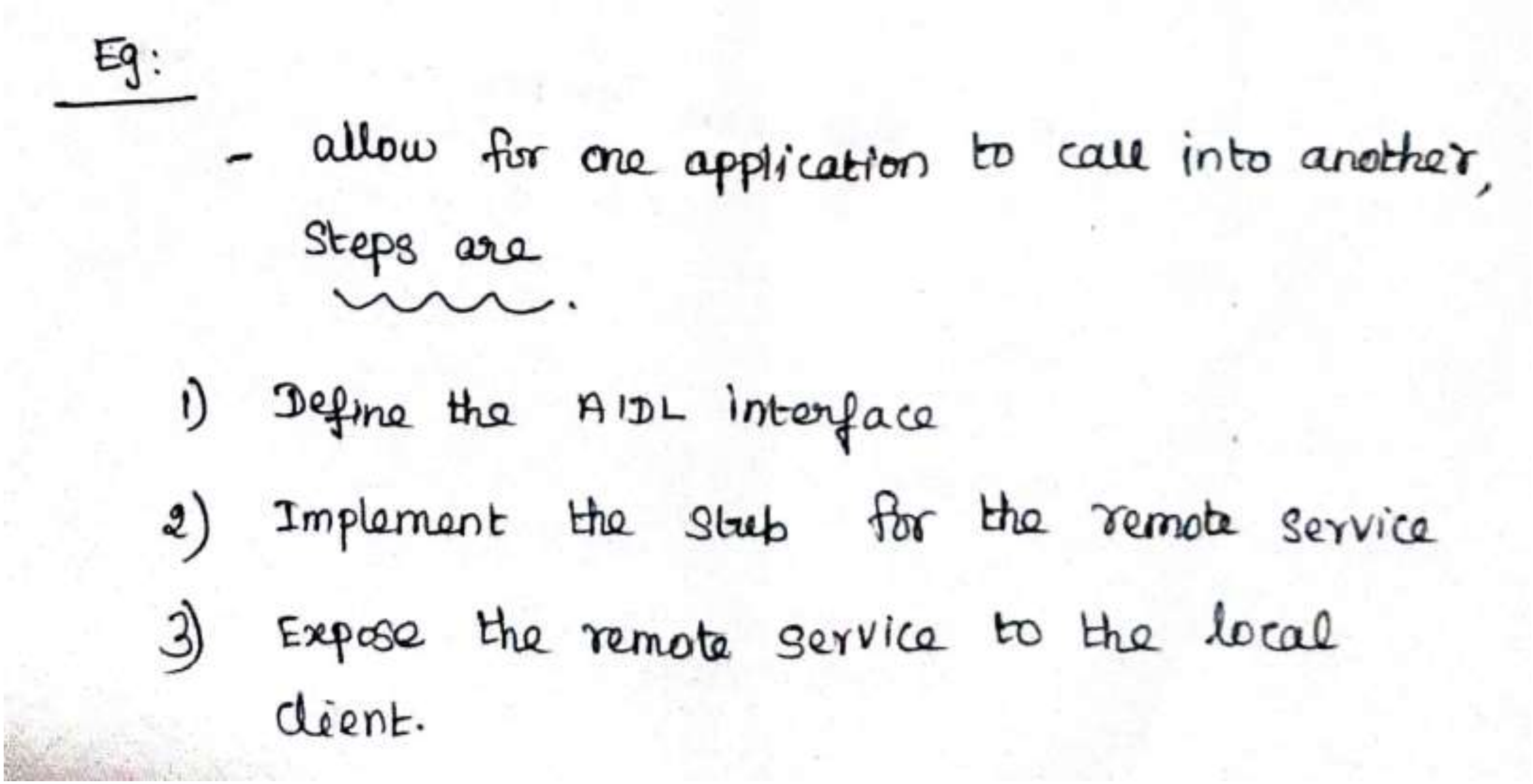
- AIDL (Android Interface Definition Language) to support IPC (Inter Process communication) between services and application components.

- AIDL give your services the ability to support multiple applications across process boundaries.

- AIDL is used to simplify the code that lets your processes exchange objects.

- It's similar to interfaces like com/corba

- create public methods within your services that can accept and return object parameters and return values between Processes.



Eg:

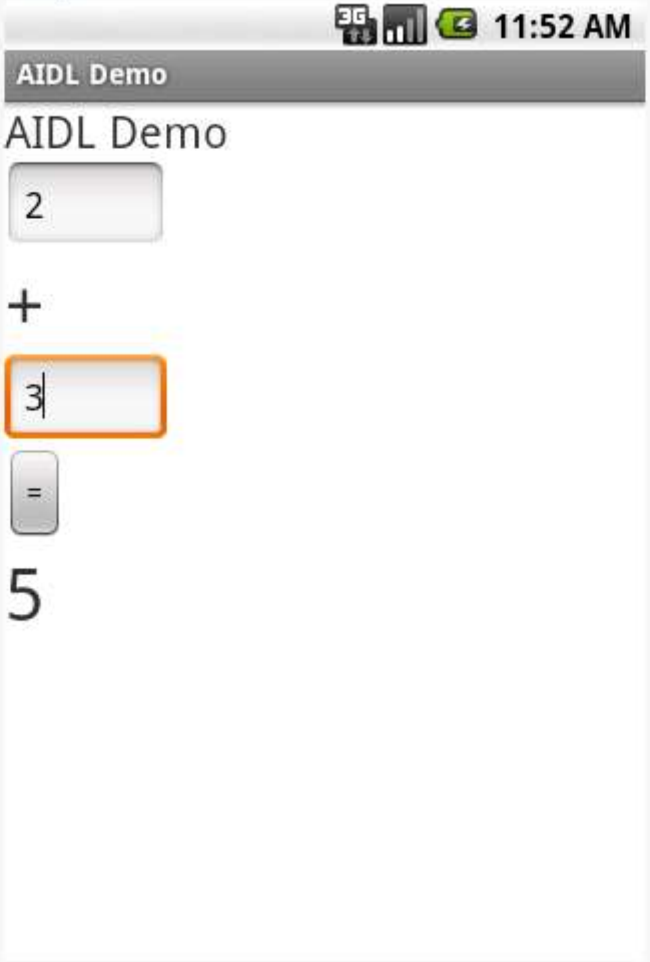
- allow for one application to call into another,

Steps are

1) Define the AIDL interface
2) Implement the Stub for the remote service
3) Expose the remote service to the local client.

https://www.protechtraining.com/blog/post/implementing-remote-interface-using-aidl-66

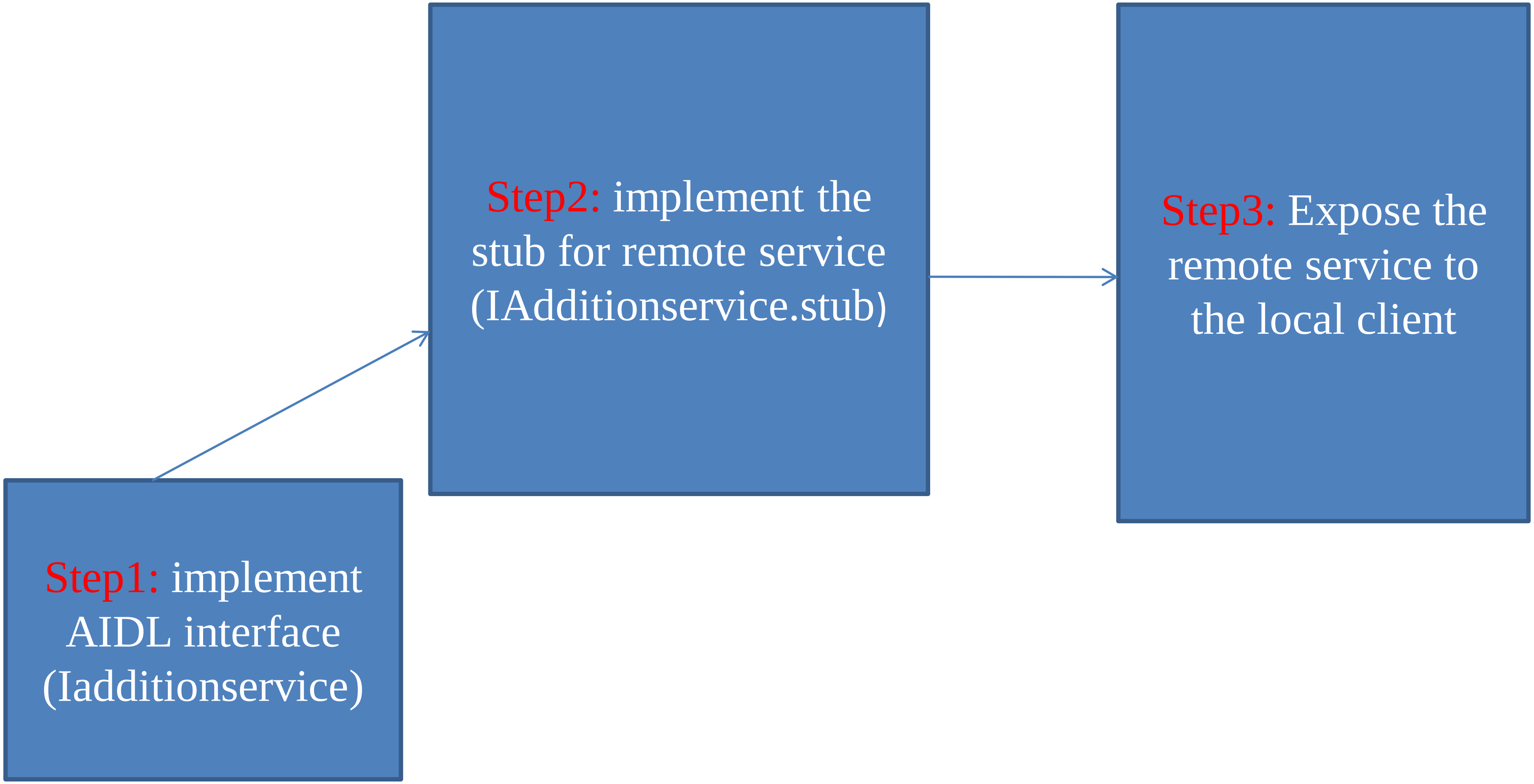
Output



2

3

5



Step1: implement AIDL interface (Iadditionservice)

Step2: implement the stub for remote service (IAdditionservice.stub)

Step3: Expose the remote service to the local client

Defining the AIDL

The AIDL syntax is very similar to regular Java interface.

You simply define the method signature.

The datatypes supported by AIDL are somewhat different than regular Java interfaces.

For one, all Java primitive datatypes are supported.

So are String, List, Map, and CharSequence classes.

Also, all other AIDL datatypes that you defined are supported.

In addition to that, all Parcelable classes are supported

/src/com.marakana/IAdditionService.aidl

Code:

```
package com.marakana;

// Declare the interface.

interface IAdditionService {

// You can pass values in, out, or inout.
// Primitive datatypes (such as int, boolean, etc.) can only be passed in.

int add(in int value1, in int value2);
}
```

Implementing the Remote Service

Once you create your AIDL file and put it in the right place, the Eclipse+AIDL tool will generate a file with the same name, but .java extension.

So I now have /gen/com.protechtraining.com/IAdditionService.java file. This is an auto-generated file so you don't want to edit it.

What is important is that it contains a Stub class that we'll want to implement for our remote service.

To implement our remote service, we'll return the IBinder from onBind() method in our service class, AdditionService. The IBinder represents the implementation of the remote service. To implement IBinder, we subclass IAddtionService.Stub class from the auto-generated Java code, and provide implementation for our AIDL-defined methods, in this case add().

/src/com.marakana/AdditionService.java

```
/**
* This class exposes the remote
service to the client
*/
public class AdditionService
extends Service {
private static final String TAG =
"AdditionService";

@Override
public void onCreate() {
super.onCreate();
Log.d(TAG, "onCreate()");
}
```

```
@Override
public IBinder onBind(Intent intent) {

return new IAdditionService.Stub() {
/**
* Implementation of the add() method
*/
public int add(int value1, int value2) throws
RemoteException {
Log.d(TAG,
String.format("AdditionService.add(%d,
%d)",value1, value2));
return value1 + value2;
}};}
```

```
@Override
public void onDestroy() {
super.onDestroy();
Log.d(TAG, "onDestroy()");
}
}
```

Exposing the Service Locally

Once we have the service implement the onBind() properly, we are ready to connect to that service from our client. In this case, we have AIDLDemo activity connecting to that service. To establish the connection, we need to implement ServiceConnection class. Our activity in this example provides this implementation in AdditionServiceConnection inner class by implementing onServiceConnected() and onServiceDiconnected() methods. Those callbacks will get the stub implementation of the remote service upon connection. We need to cast them from stubs to our AIDL service implementation. To do that, we use the IAdditionService.Stub.asInterface((IBinder) boundService) helper method.

From that point on, we have a local service object that we can use to make calls against the remote service.

/src/com.marakana/AIDLDemo.java

```
public class AIDLDemo extends Activity {
private static final String TAG = "AIDLDemo";
IAdditionService service;
AdditionServiceConnection connection;

/**
* This class represents the actual service connection. It casts the bound
* stub implementation of the service to the AIDL interface.
*/
class AdditionServiceConnection implements ServiceConnection {

public void onServiceConnected(ComponentName name, IBinder
boundService) {
service = IAdditionService.Stub.asInterface((IBinder) boundService);
Log.d(AIDLDemo.TAG, "onServiceConnected() connected");
Toast.makeText(AIDLDemo.this, "Service connected",
Toast.LENGTH_LONG)
.show();
}
public void onServiceDisconnected(ComponentName name) {
service = null;
Log.d(AIDLDemo.TAG, "onServiceDisconnected() disconnected");
Toast.makeText(AIDLDemo.this, "Service connected", Toast.LENGTH_LONG)
.show();
}
}
/** Binds this activity to the service. */
private void initService() {
connection = new AdditionServiceConnection();
Intent i = new Intent();
i.setClassName("com.marakana", com.marakana.AdditionService.class.getName());
boolean ret = bindService(i, connection, Context.BIND_AUTO_CREATE);
Log.d(TAG, "initService() bound with " + ret);
}
```

```
/** Unbinds this activity from the service. */
private void releaseService() {
unbindService(connection);
connection = null;
Log.d(TAG, "releaseService() unbound.");
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);

initService();

// Setup the UI
Button buttonCalc = (Button) findViewById(R.id.buttonCalc);

buttonCalc.setOnClickListener(new OnClickListener() {
TextView result = (TextView) findViewById(R.id.result);
EditText value1 = (EditText) findViewById(R.id.value1);
EditText value2 = (EditText) findViewById(R.id.value2);
public void onClick(View v) {
int v1, v2, res = -1;
v1 = Integer.parseInt(value1.getText().toString());
v2 = Integer.parseInt(value2.getText().toString());

try {
res = service.add(v1, v2);
} catch (RemoteException e) {
Log.d(AIDLDemo.TAG, "onClick failed with: " + e);
e.printStackTrace();
}
result.setText(new Integer(res).toString());
}
});
}

/** Called when the activity is about to be destroyed. */
@Override
protected void onDestroy() {
releaseService();
}
}
```

The UI in this case is very simple. There are couple of TextViews and EditText fields and a button The button handles its events in an anonymous inner class OnClickListener. The button simply invokes the add() method on the service as if it was a local call.

/res/layout/main.xml

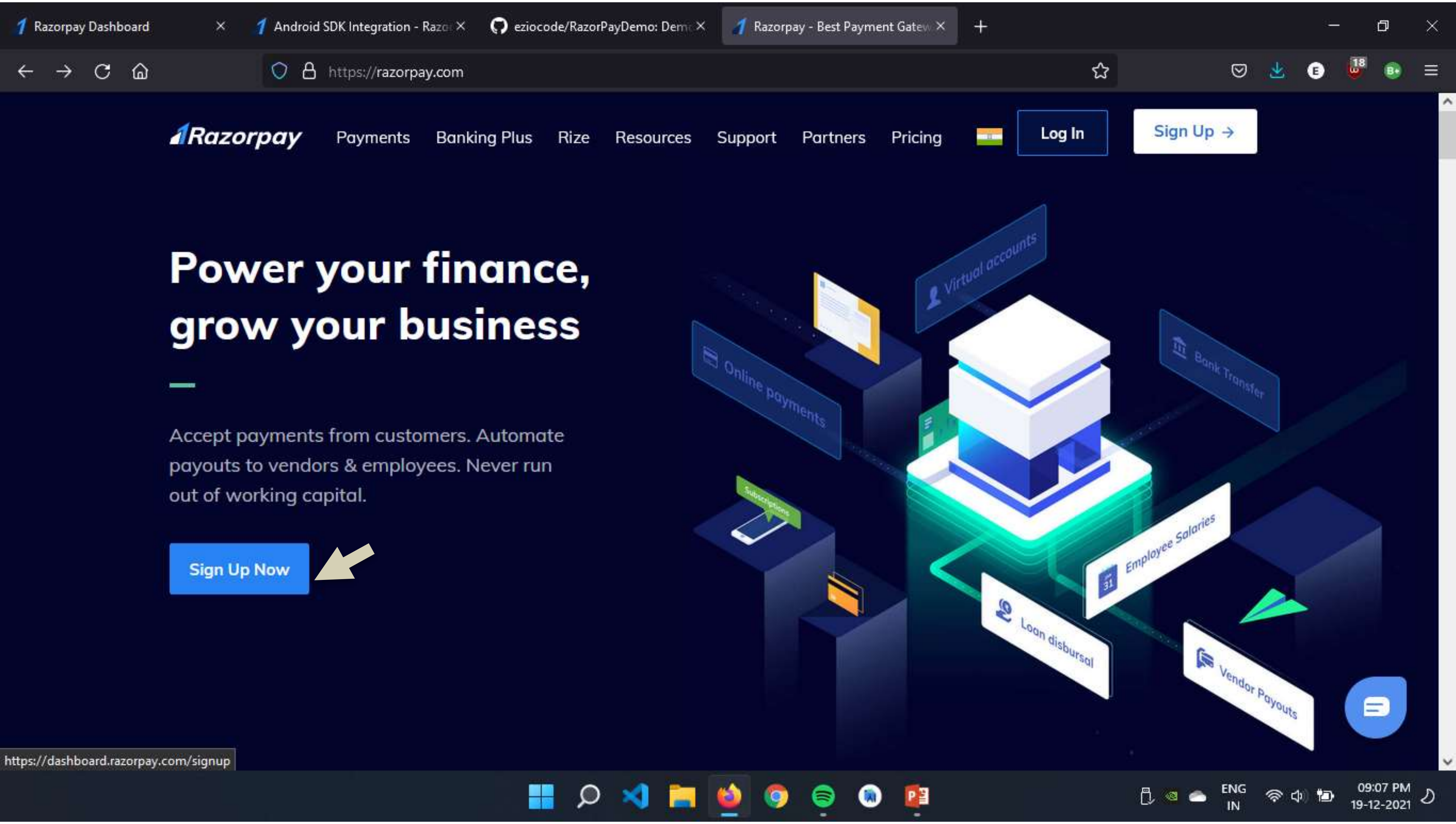
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
<TextView  android:text="AIDL Demo“ />

<EditText
android:id="@+id/value1"
android:hint="Value 1"></EditText>

<TextView android:id="@+id/TextView01“
android:text="+“></TextView>
```

```
<EditText
android:id="@+id/value2"
android:hint="Value 2"></EditText>

<Button
android:id="@+id/buttonCalc"
android:text="="></Button>

<TextView android:text="result“
android:id="@+id/result"></TextView>
</LinearLayout>
```

Topic 4 : Payment Gateway

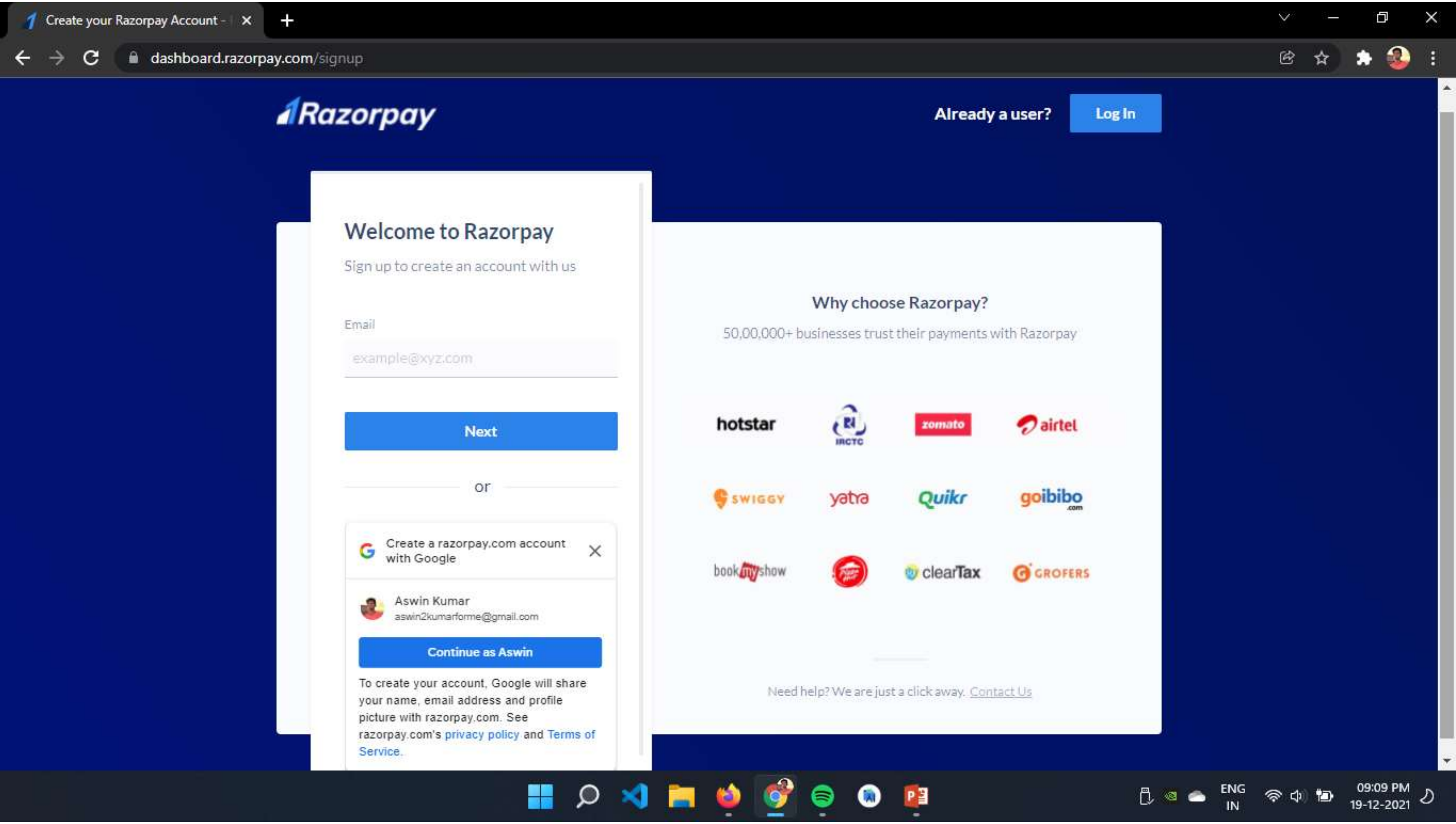
RAZORPAY INTEGRATION

ANDROID STUDIO



Power your finance, grow your business

Accept payments from customers. Automate payouts to vendors & employees. Never run out of working capital.



Welcome to Razorpay

Sign up to create an account with us

Email: example@xyz.com

Create a razorpay.com account with Google

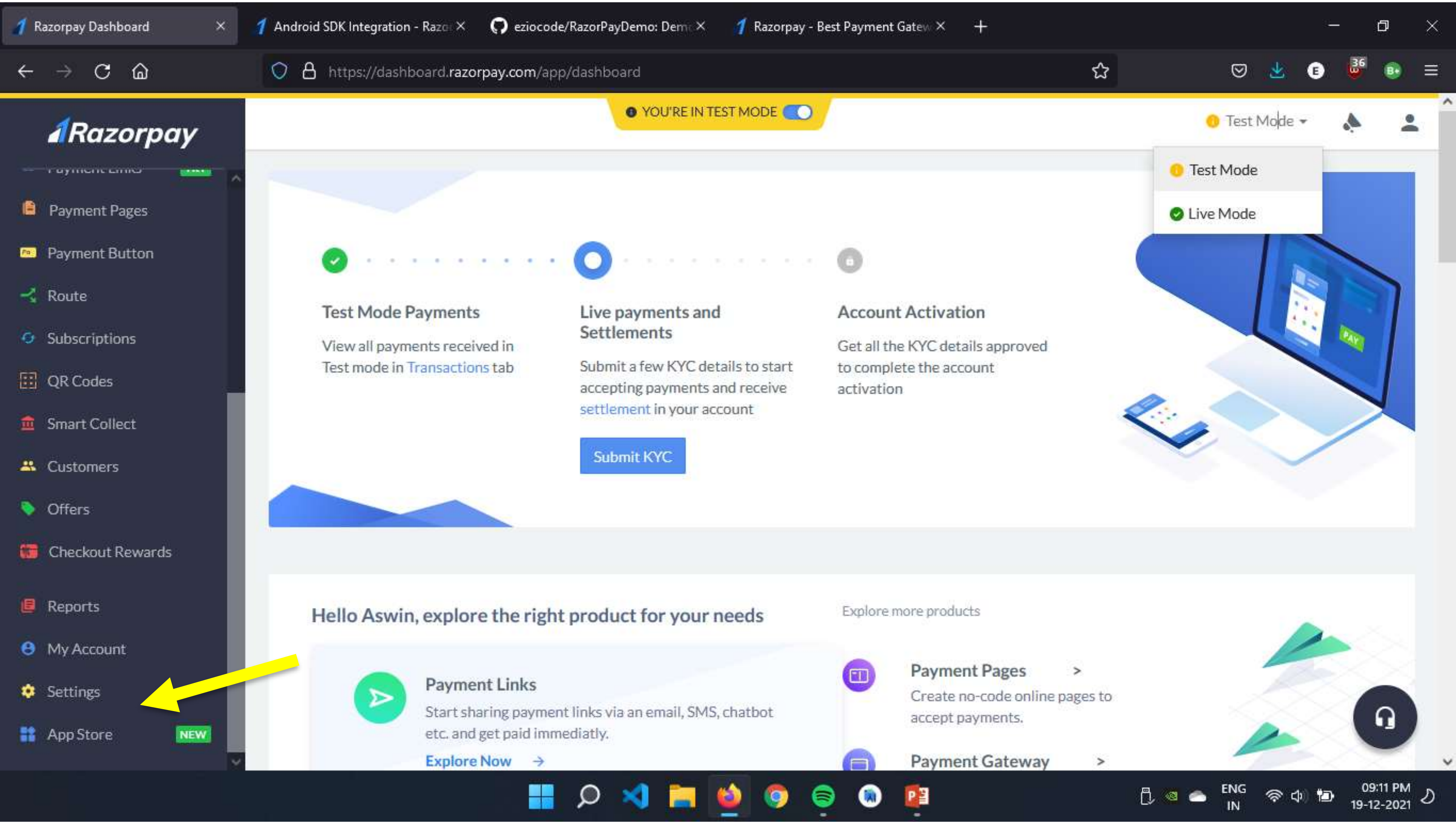
Aswin Kumar
aswin2kumarforme@gmail.com

To create your account, Google will share your name, email address and profile picture with razorpay.com. See razorpay.com's privacy policy and Terms of Service.

Why choose Razorpay?

50,00,000+ businesses trust their payments with Razorpay

Need help? We are just a click away. Contact Us



Test Mode Payments

View all payments received in Test mode in Transactions tab

Live payments and Settlements

Submit a few KYC details to start accepting payments and receive settlement in your account

Account Activation

Get all the KYC details approved to complete the account activation

Hello Aswin, explore the right product for your needs

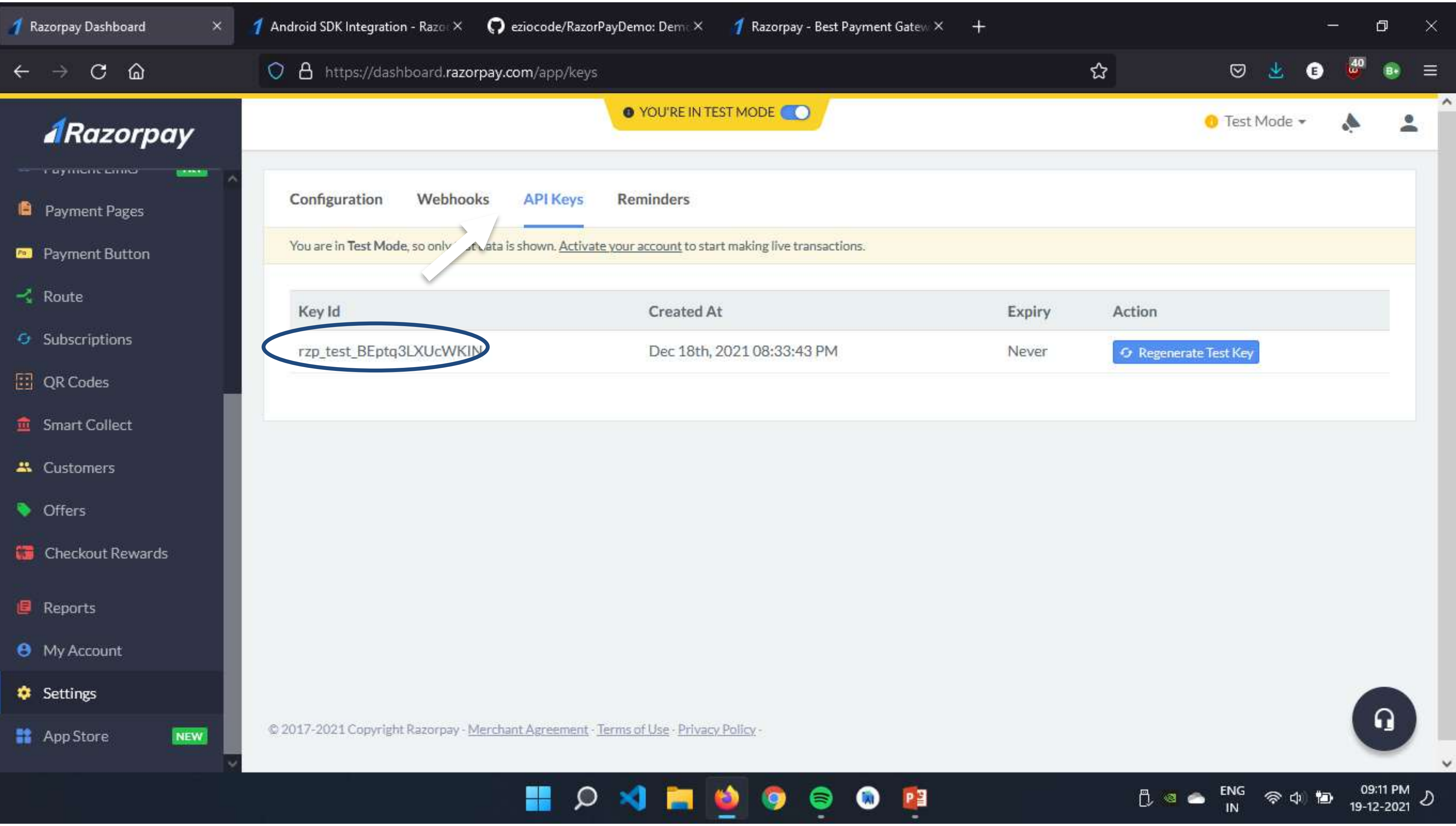
Payment Links

Start sharing payment links via an email, SMS, chatbot etc. and get paid immediatly.

Explore more products

Payment Pages

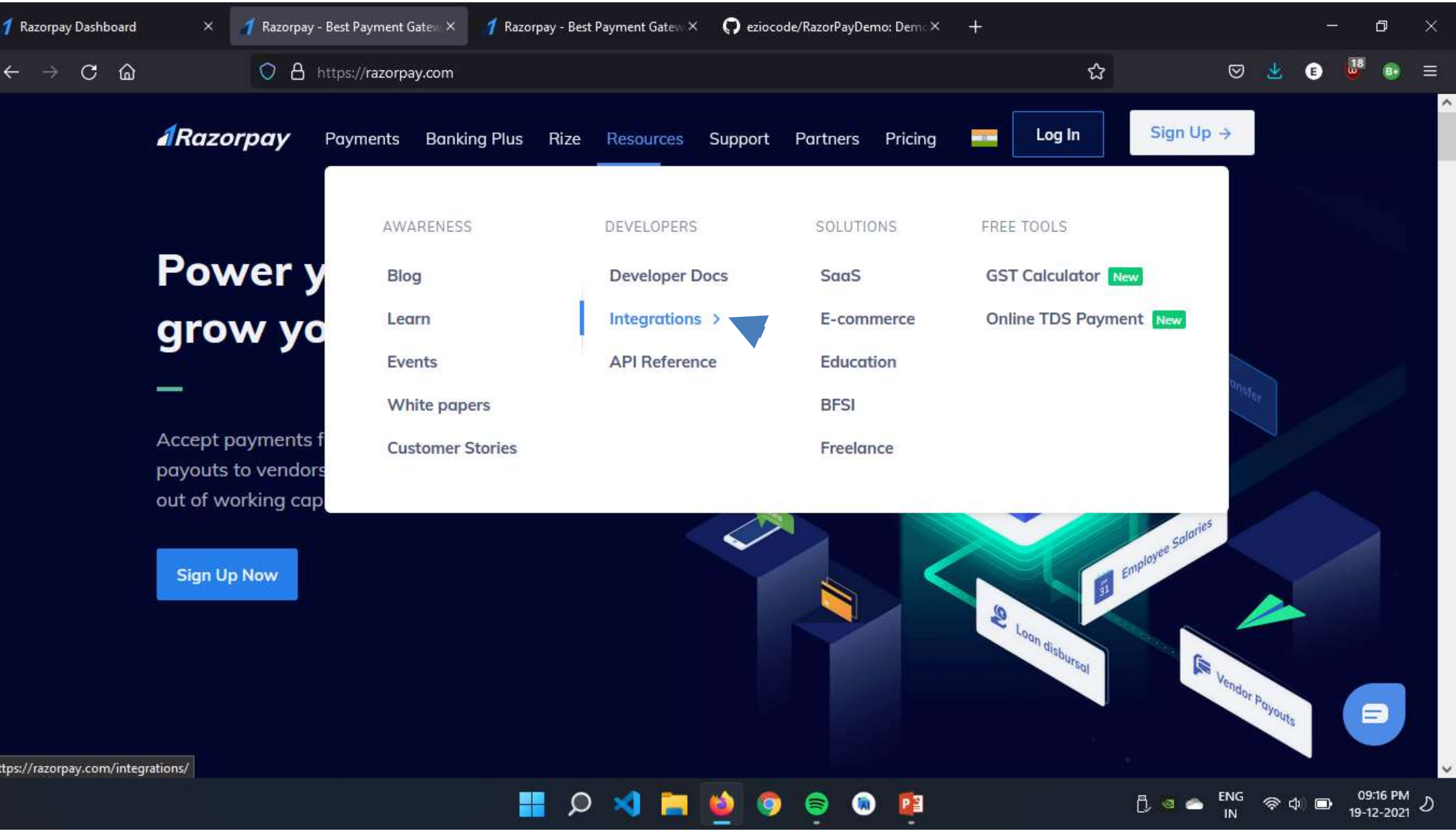
Create no-code online pages to accept payments.



You are in **Test Mode**, so only test data is shown. Activate your account to start making live transactions.

Key Id	Created At	Expiry	Action
rzp_test_BEptq3LXUcWKIN	Dec 18th, 2021 08:33:43 PM	Never	Regenerate Test Key

© 2017-2021 Copyright Razorpay · Merchant Agreement · Terms of Use · Privacy Policy ·

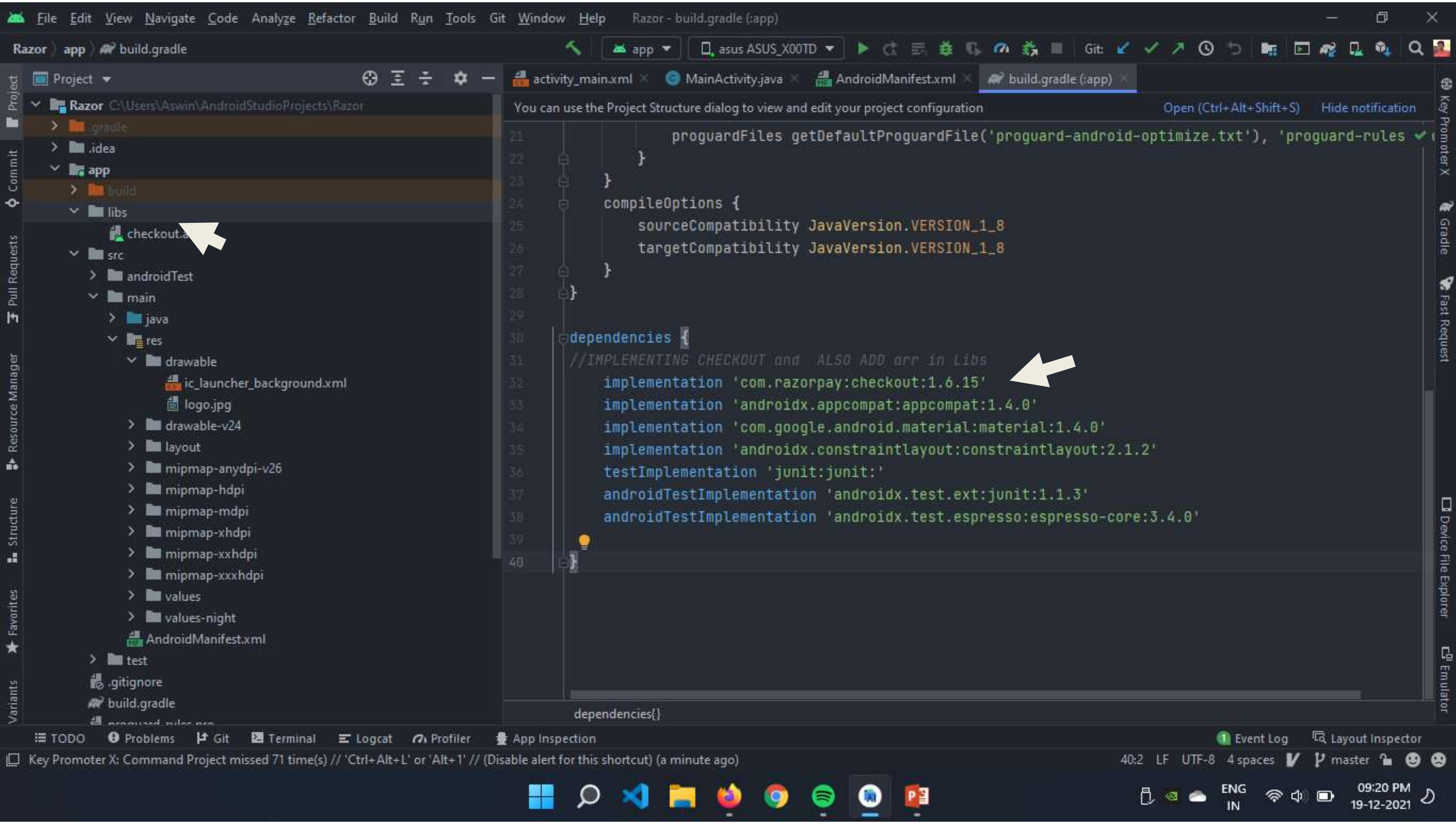


AWARENESS
- Blog
- Learn
- Events
- White papers
- Customer Stories

DEVELOPERS
- Developer Docs
- Integrations >
- API Reference

SOLUTIONS
- SaaS
- E-commerce
- Education
- BFSI
- Freelance

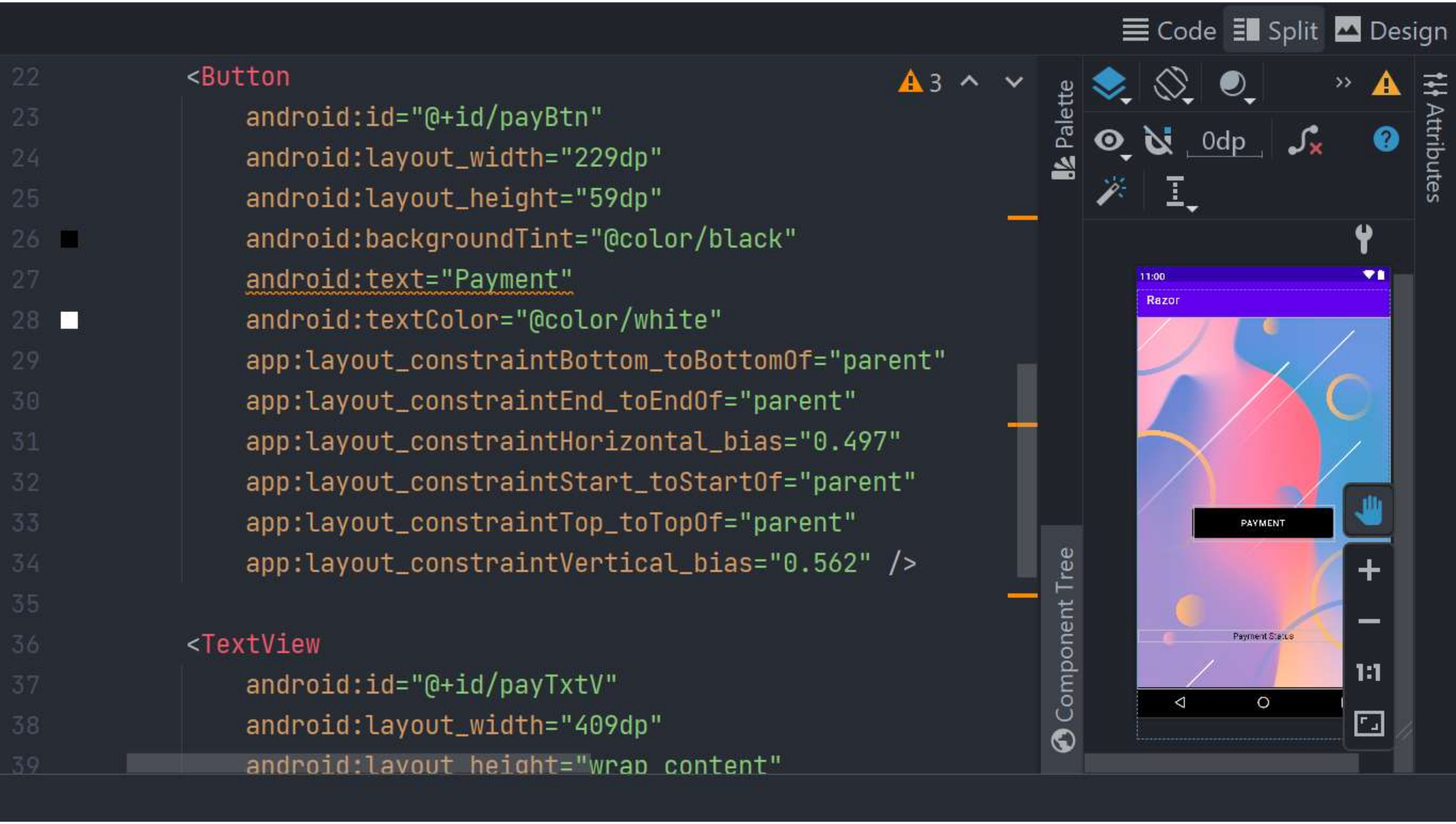
FREE TOOLS
- GST Calculator New
- Online TDS Payment New



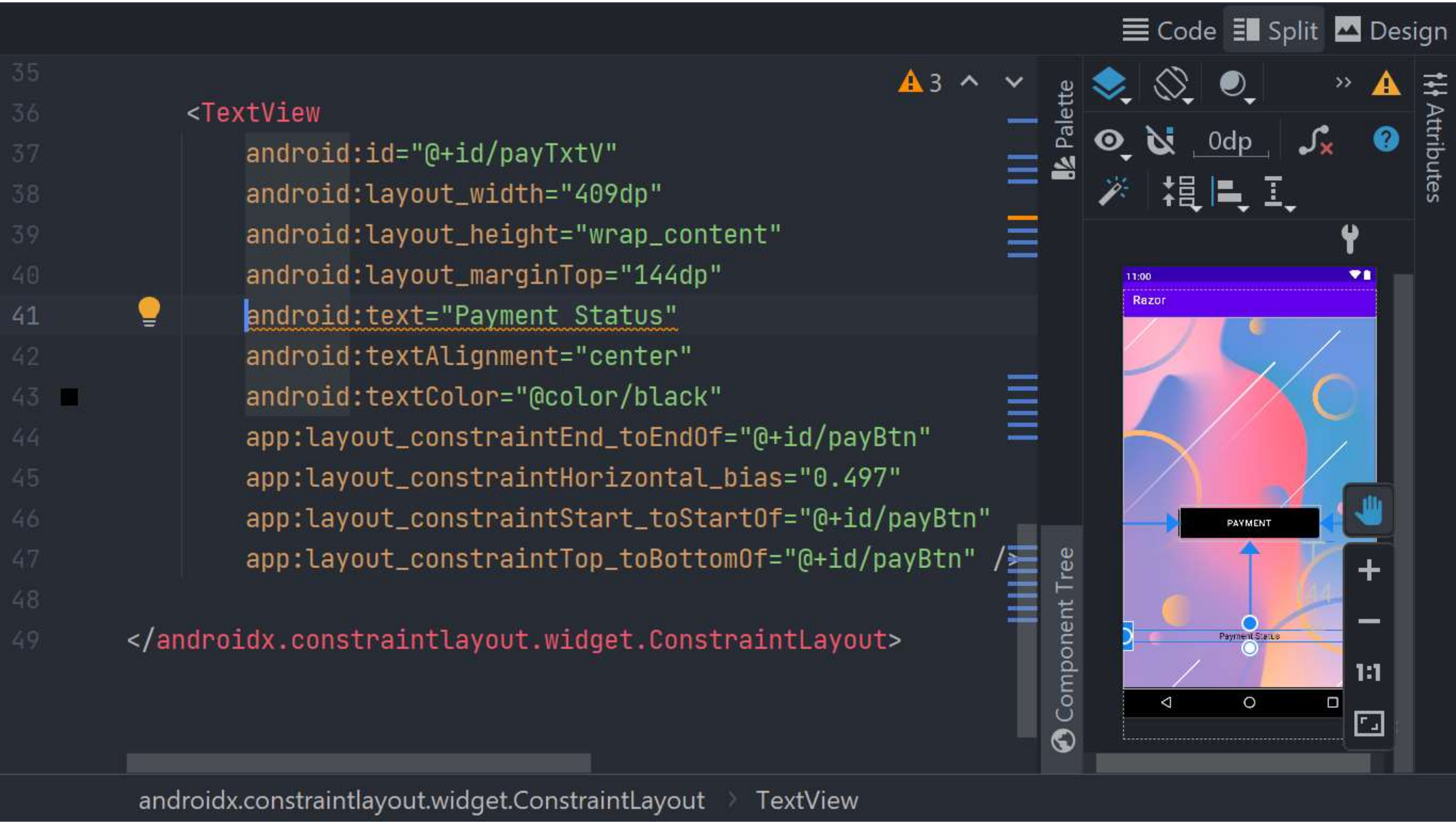
You can use the Project Structure dialog to view and edit your project configuration

```
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
//IMPLEMENTING CHECKOUT and  ALSO ADD arr in Libs
    implementation 'com.razorpay:checkout:1.6.15'
    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    testImplementation 'junit:junit:'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

}
```

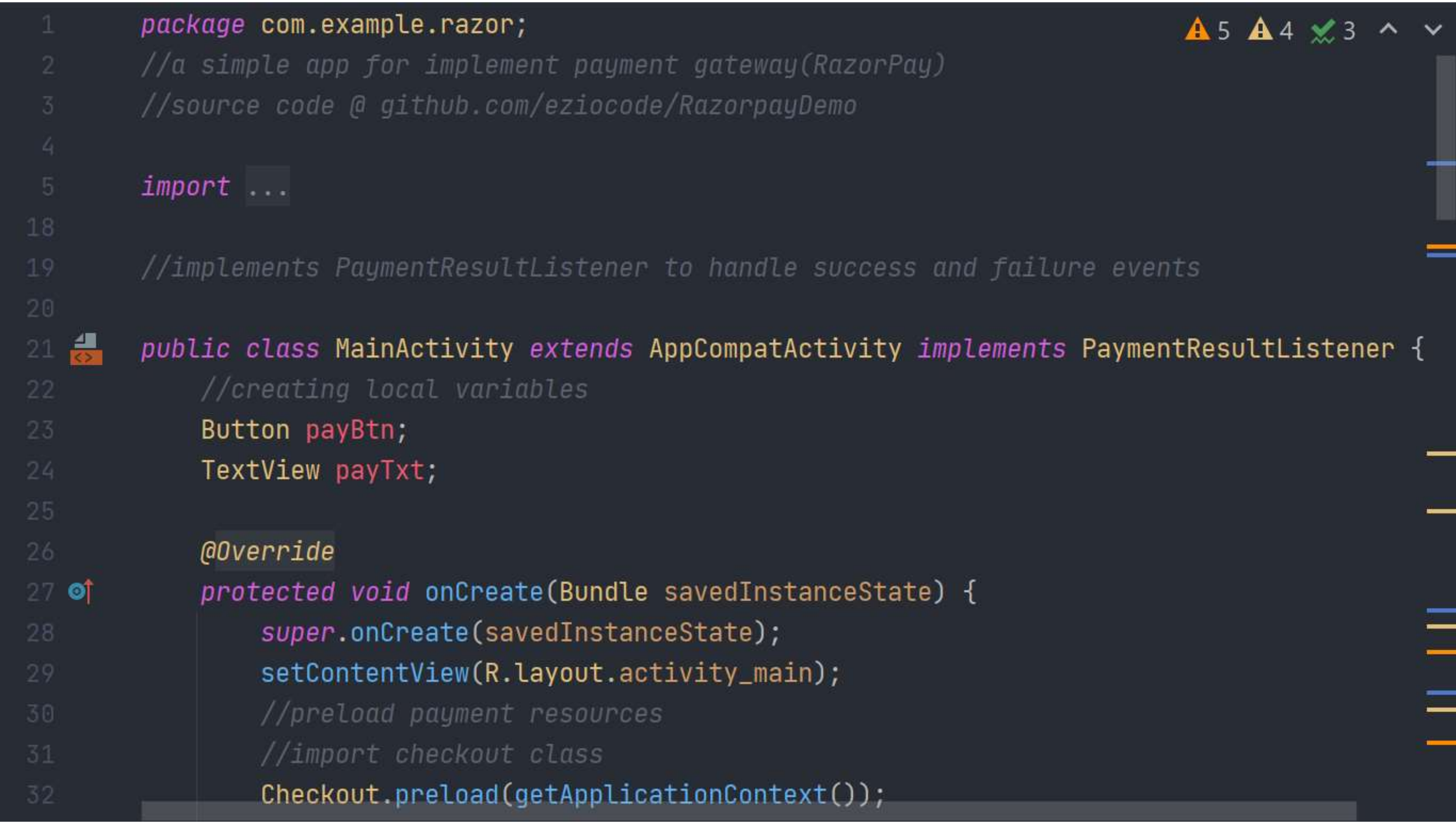
Key Promoter X: Command Project missed 71 time(s) // 'Ctrl+Alt+L' or 'Alt+1' // (Disable alert for this shortcut) (a minute ago)



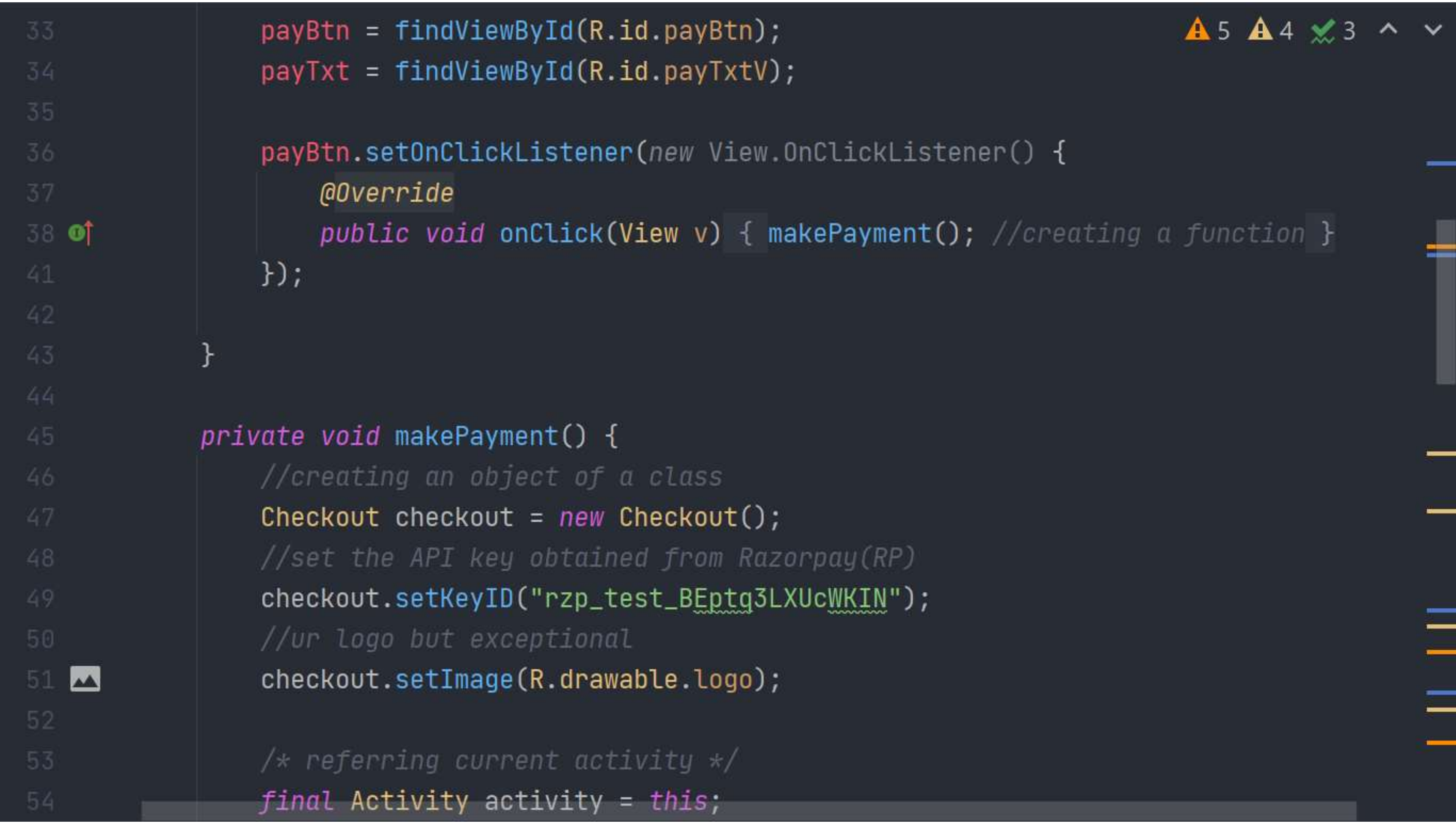
```
        <Button
            android:id="@+id/payBtn"
            android:layout_width="229dp"
            android:layout_height="59dp"
            android:backgroundTint="@color/black"
            android:text="Payment"
            android:textColor="@color/white"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.497"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.562" />

        <TextView
            android:id="@+id/payTxtV"
            android:layout_width="409dp"
            android:layout_height="wrap_content"
```



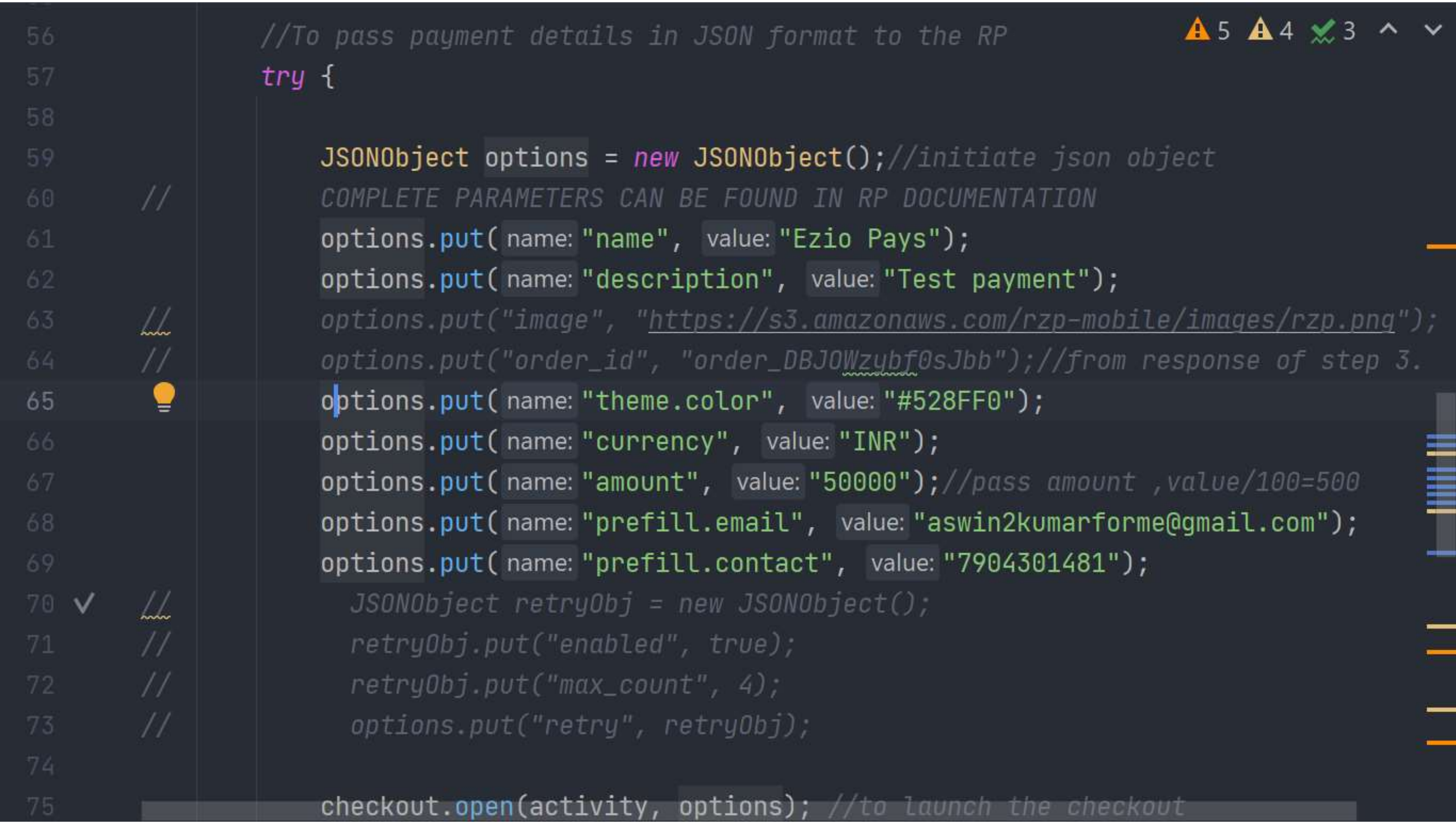
```
    <TextView
        android:id="@+id/payTxtV"
        android:layout_width="409dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="144dp"
        android:text="Payment Status"
        android:textAlignment="center"
        android:textColor="@color/black"
        app:layout_constraintEnd_toEndOf="@+id/payBtn"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="@+id/payBtn"
        app:layout_constraintTop_toBottomOf="@+id/payBtn" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



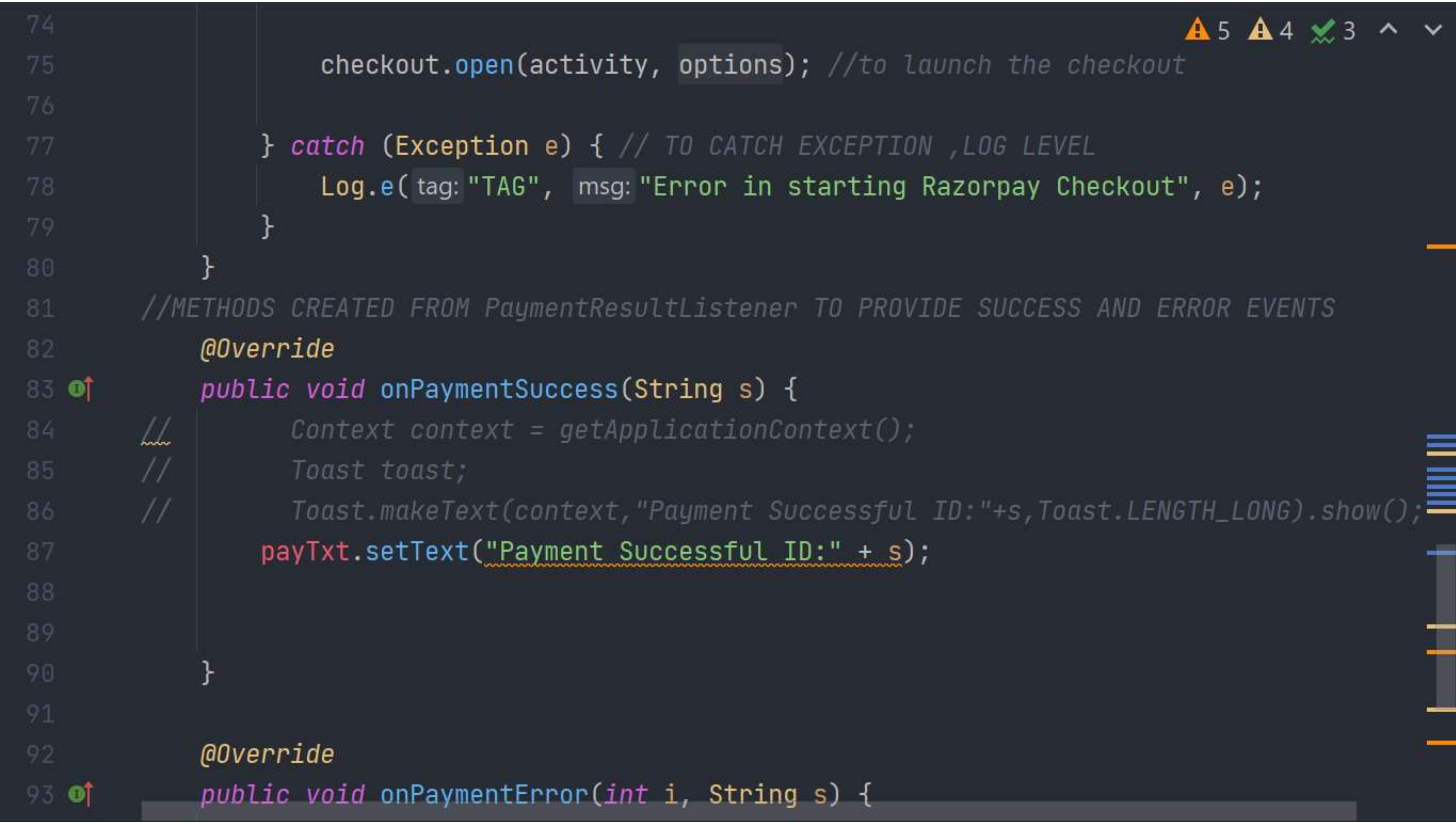
```java
package com.example.razor;
//a simple app for implement payment gateway(RazorPay)
//source code @ github.com/eziocode/RazorpayDemo

import ...

//implements PaymentResultListener to handle success and failure events

public class MainActivity extends AppCompatActivity implements PaymentResultListener {
    //creating local variables
    Button payBtn;
    TextView payTxt;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //preload payment resources
        //import checkout class
        Checkout.preload(getApplicationContext());
```



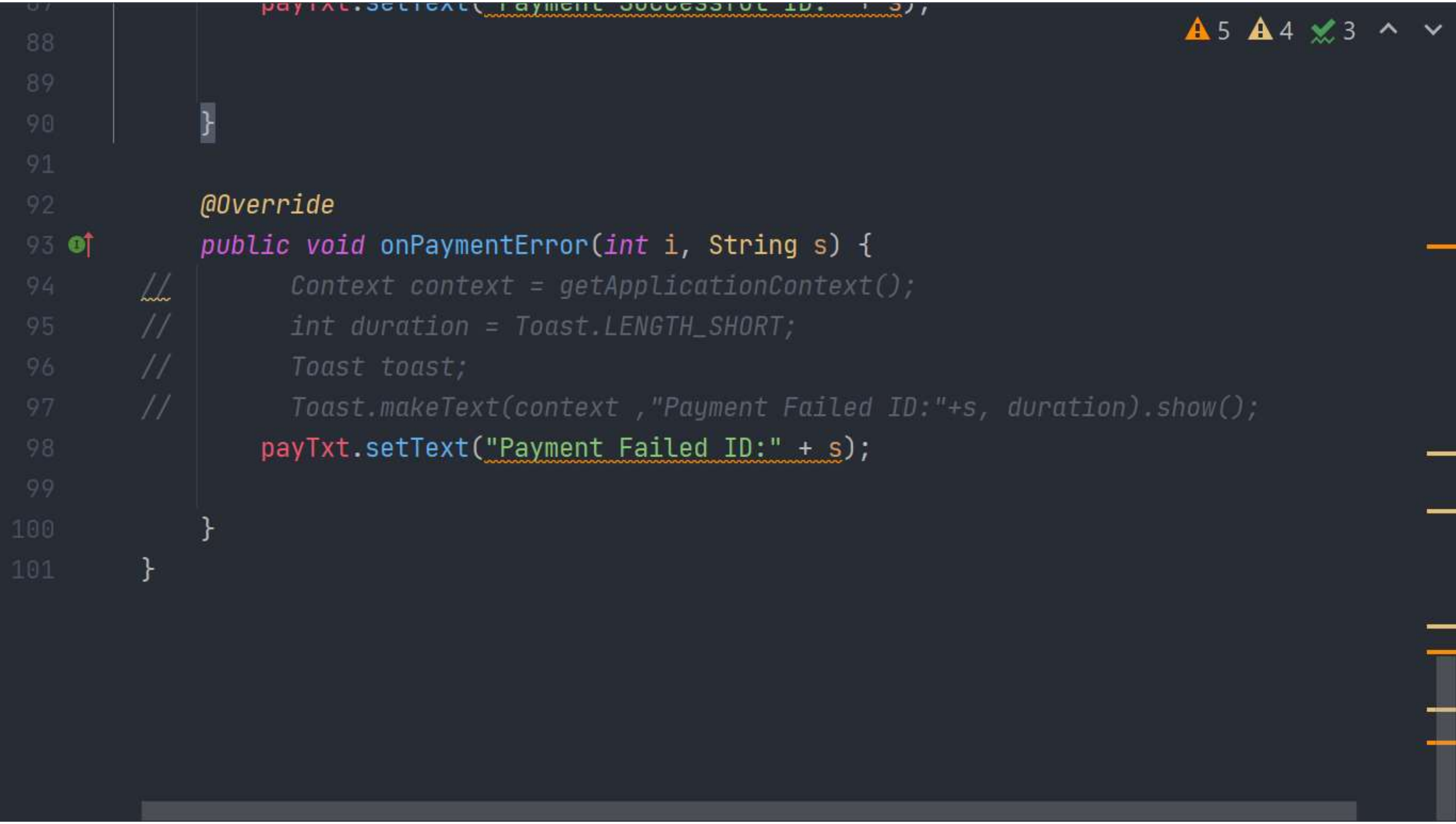
```
        payBtn = findViewById(R.id.payBtn);
        payTxt = findViewById(R.id.payTxtV);

        payBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { makePayment(); //creating a function }
        });

    }

    private void makePayment() {
        //creating an object of a class
        Checkout checkout = new Checkout();
        //set the API key obtained from Razorpay(RP)
        checkout.setKeyID("rzp_test_BEptq3LXUcWKIN");
        //ur logo but exceptional
        checkout.setImage(R.drawable.logo);

        /* referring current activity */
        final Activity activity = this;
```



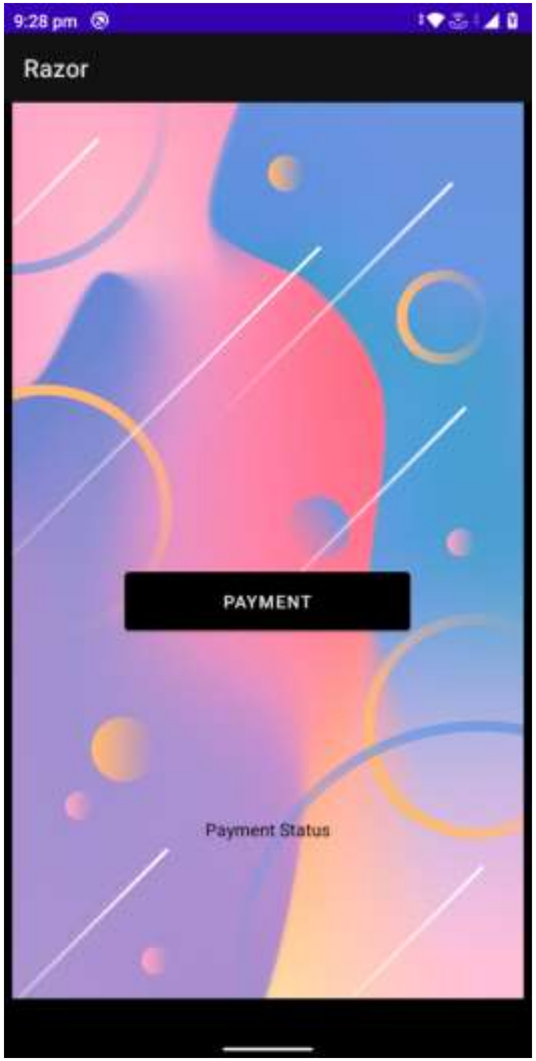
```
        //To pass payment details in JSON format to the RP
        try {

            JSONObject options = new JSONObject();//initiate json object
//          COMPLETE PARAMETERS CAN BE FOUND IN RP DOCUMENTATION
            options.put( name: "name",  value: "Ezio Pays");
            options.put( name: "description",  value: "Test payment");
//          options.put("image", "https://s3.amazonaws.com/rzp-mobile/images/rzp.png");
//          options.put("order_id", "order_DBJOWzybf0sJbb");//from response of step 3.
            options.put( name: "theme.color",  value: "#528FF0");
            options.put( name: "currency",  value: "INR");
            options.put( name: "amount",  value: "50000");//pass amount ,value/100=500
            options.put( name: "prefill.email",  value: "aswin2kumarforme@gmail.com");
            options.put( name: "prefill.contact",  value: "7904301481");
//            JSONObject retryObj = new JSONObject();
//            retryObj.put("enabled", true);
//            retryObj.put("max_count", 4);
//            options.put("retry", retryObj);

            checkout.open(activity, options); //to launch the checkout
```

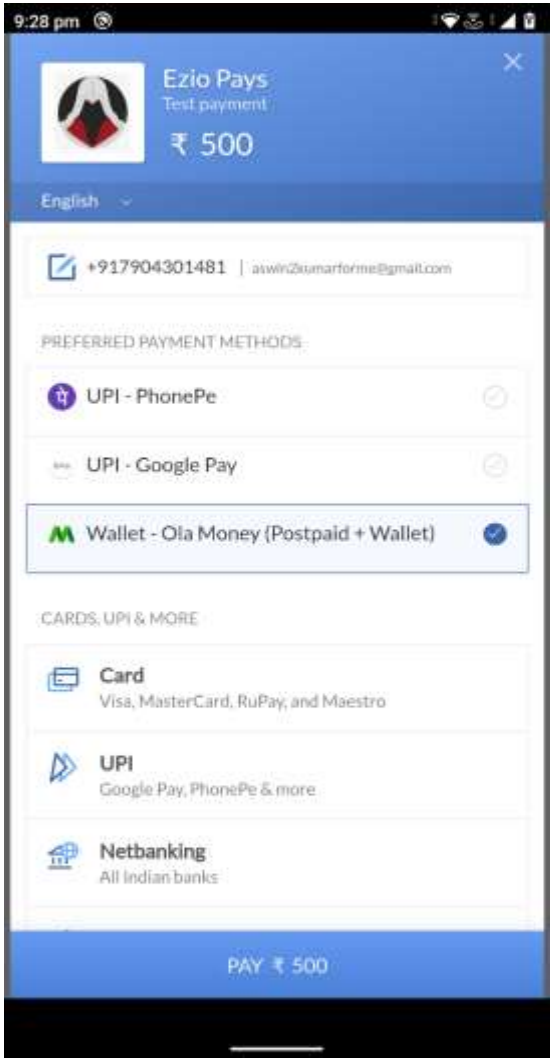


```
            checkout.open(activity, options); //to launch the checkout

        } catch (Exception e) { // TO CATCH EXCEPTION ,LOG LEVEL
            Log.e( tag: "TAG",  msg: "Error in starting Razorpay Checkout", e);
        }
    }
//METHODS CREATED FROM PaymentResultListener TO PROVIDE SUCCESS AND ERROR EVENTS
    @Override
    public void onPaymentSuccess(String s) {
//        Context context = getApplicationContext();
//        Toast toast;
//        Toast.makeText(context,"Payment Successful ID:"+s,Toast.LENGTH_LONG).show();
        payTxt.setText("Payment Successful ID:" + s);


    }

    @Override
    public void onPaymentError(int i, String s) {
```



```
        payTxt.setText("Payment Successful ID:" + s);


    }

    @Override
    public void onPaymentError(int i, String s) {
//          Context context = getApplicationContext();
//          int duration = Toast.LENGTH_SHORT;
//          Toast toast;
//          Toast.makeText(context ,"Payment Failed ID:"+s, duration).show();
        payTxt.setText("Payment Failed ID:" + s);

    }
}
```




9:28 pm

Ezio Pays

Test payment

₹ 500

English

+917904301481 | aswin2kumarforme@gmail.com

PREFERRED PAYMENT METHODS

UPI - PhonePe

UPI - Google Pay

Wallet - Ola Money (Postpaid + Wallet) ●

CARDS, UPI & MORE

Card

Visa, MasterCard, RuPay, and Maestro

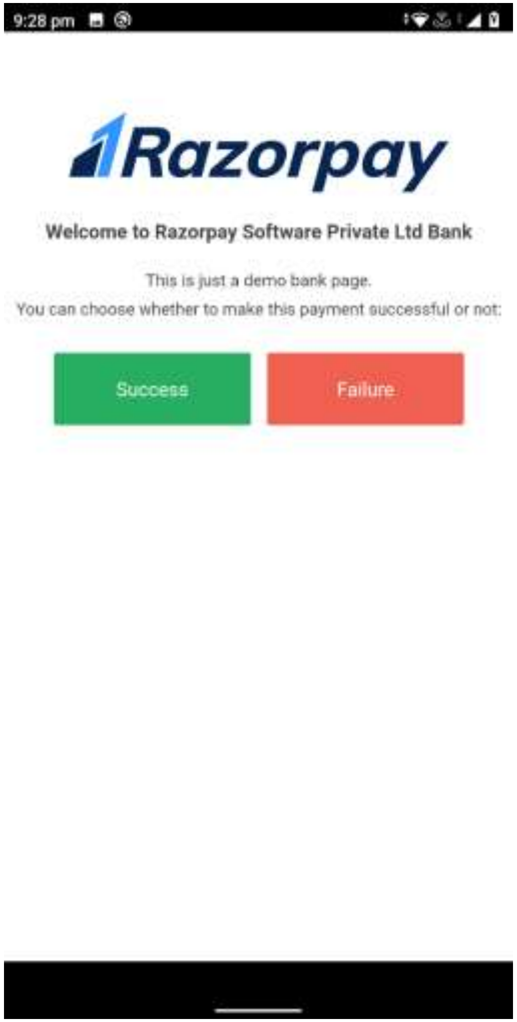
UPI

Google Pay, PhonePe & more

Netbanking

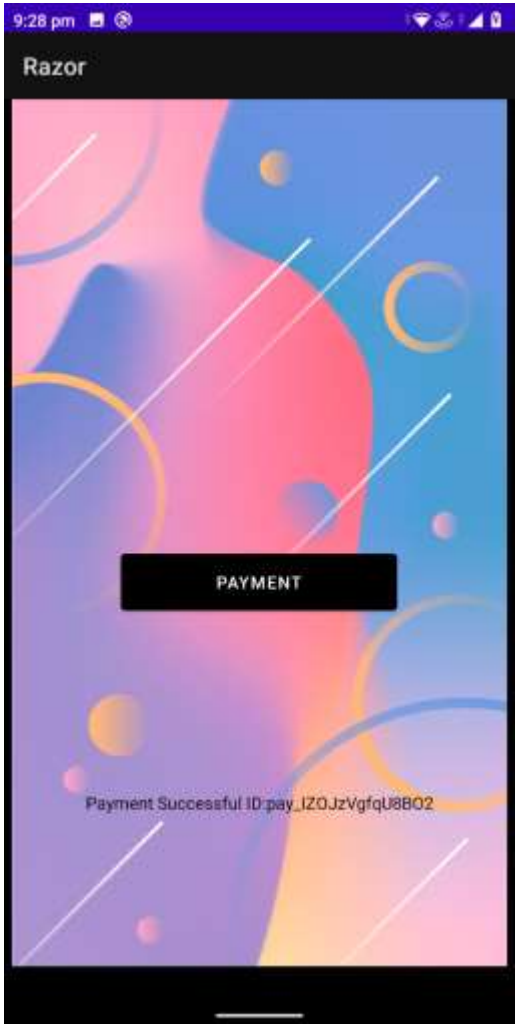
All Indian banks

PAY ₹ 500



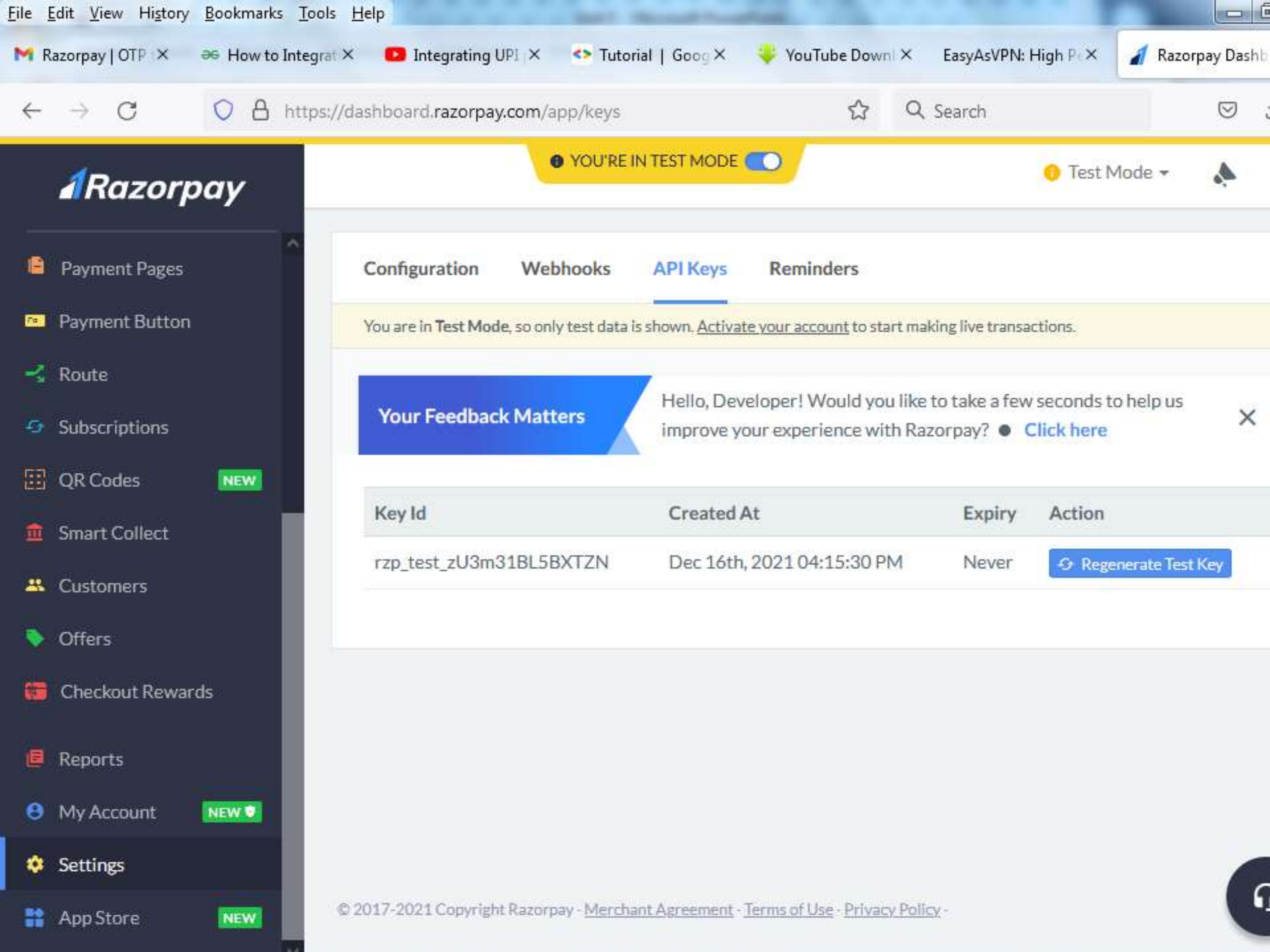
Welcome to Razorpay Software Private Ltd Bank

This is just a demo bank page.
You can choose whether to make this payment successful or not:



Payment Successful ID:pay_IZOJzVgfqU8BO2 https://www.geeksforgeeks.org/how-to-integrate-razorpay-payment-gateway-in-android/

https://dashboard.razorpay.com/app/keys

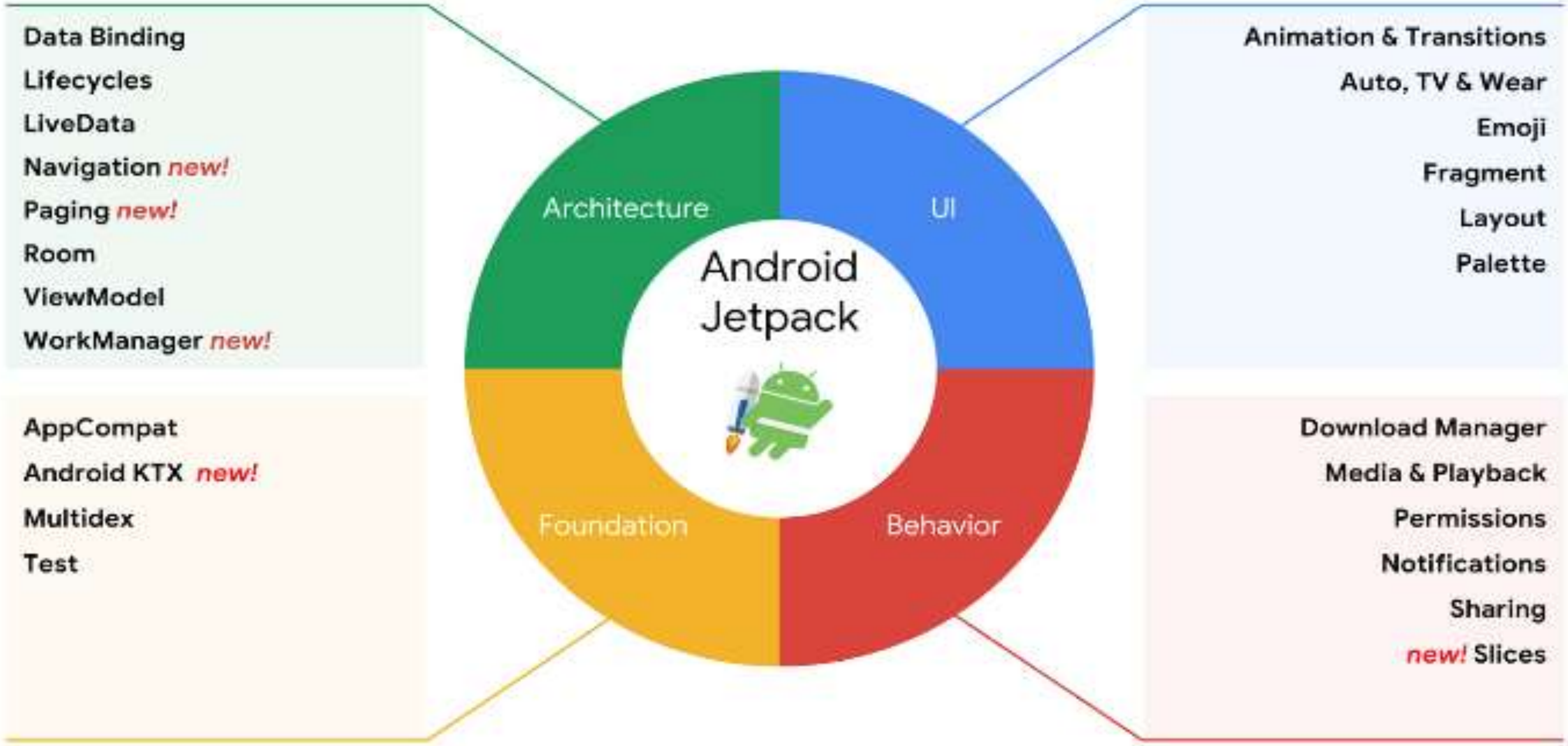


You are in **Test Mode**, so only test data is shown. Activate your account to start making live transactions.

Key Id	Created At	Expiry	Action
rzp_test_zU3m31BL5BXTZN	Dec 16th, 2021 04:15:30 PM	Never	Regenerate Test Key

Topic 5: Android Jetpack Technology - II

Android Jetpack is a set of components, tools and guidance to make great Android apps. The Android Jetpack components bring together the existing Support Library and Architecture Components and arranges them into four categories:



Android Jetpack

Architecture

- Data Binding
- Lifecycles
- LiveData
- Navigation *new!*
- Paging *new!*
- Room
- ViewModel
- WorkManager *new!*

UI

- Animation & Transitions
- Auto, TV & Wear
- Emoji
- Fragment
- Layout
- Palette

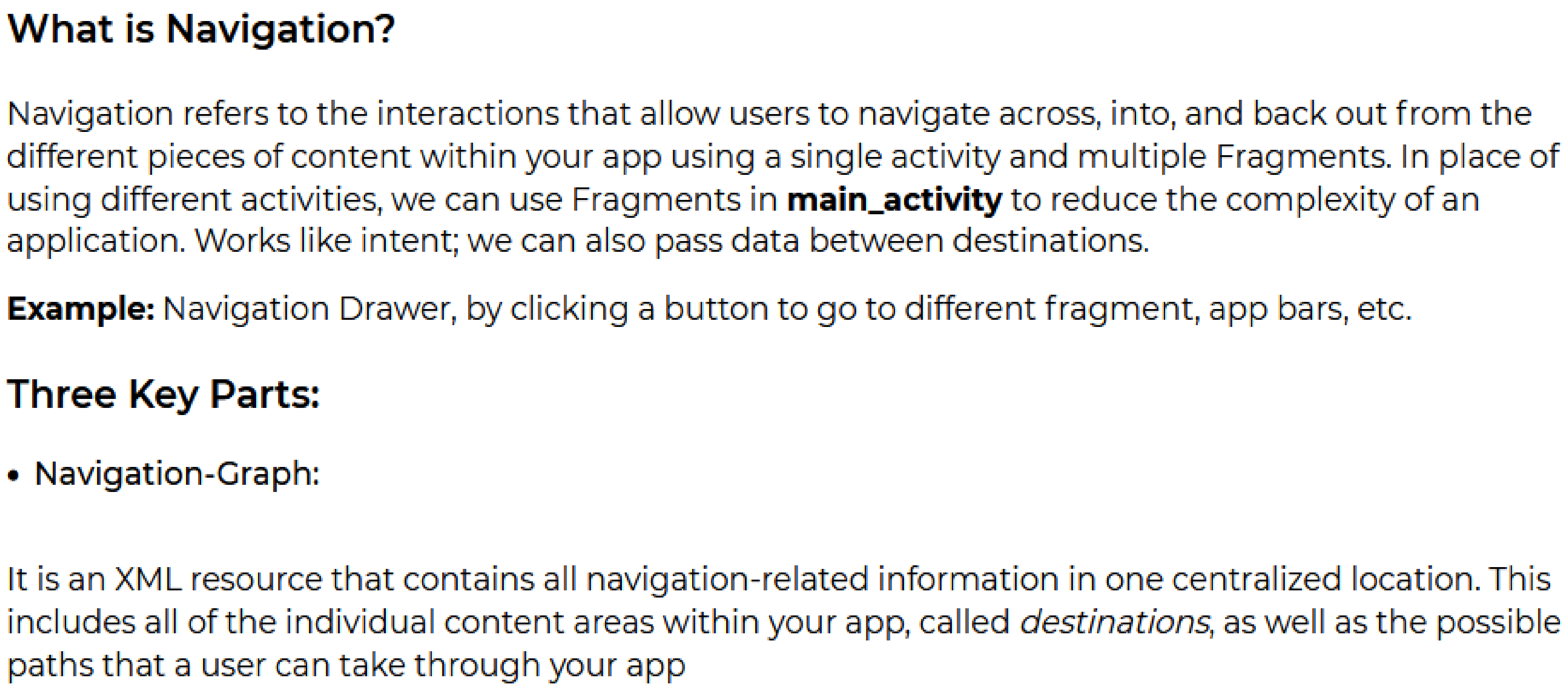
Foundation

- AppCompat
- Android KTX *new!*
- Multidex
- Test

Behavior

- Download Manager
- Media & Playback
- Permissions
- Notifications
- Sharing
- *new!* Slices

https://www.metizsoft.com/blog/android-jetpack-navigation-component



What is Navigation?

Navigation refers to the interactions that allow users to navigate across, into, and back out from the different pieces of content within your app using a single activity and multiple Fragments. In place of using different activities, we can use Fragments in **main_activity** to reduce the complexity of an application. Works like intent; we can also pass data between destinations.

Example: Navigation Drawer, by clicking a button to go to different fragment, app bars, etc.

Three Key Parts:

- Navigation-Graph:

It is an XML resource that contains all navigation-related information in one centralized location. This includes all of the individual content areas within your app, called *destinations*, as well as the possible paths that a user can take through your app

- NavHost:

It is an empty container that displays destinations from your navigation graph. The Navigation component contains a default NavHost implementation, **NavHostFragment**, that displays fragment destinations.

- NavController:

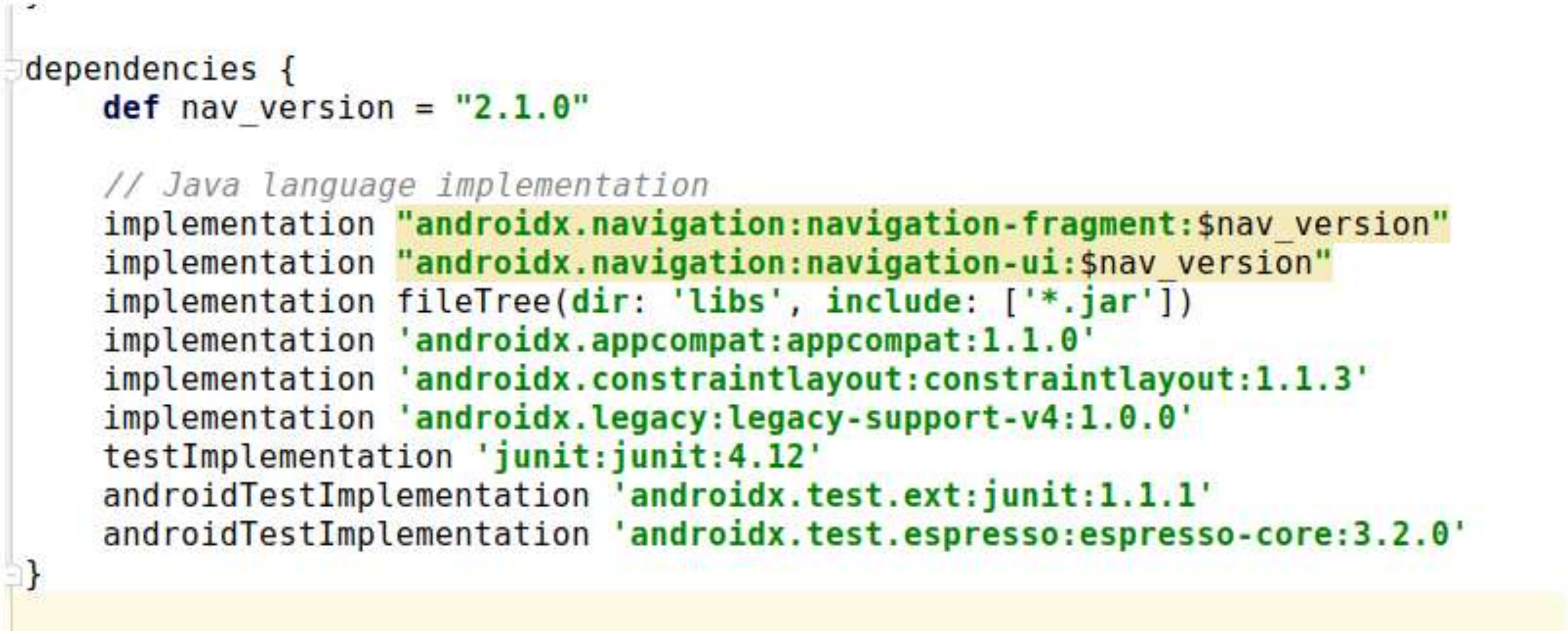
It is an object that manages app navigation within a **NavHost**. The **NavController** orchestrates the swapping of destination content in the NavHost as users move throughout your app.

Benefits of Navigation Component:

- Handling fragment transactions.
- Handling Up and Back actions correctly by default.
- Providing standardized resources for animations and transitions.
- Implementing and handling deep linking.
- Including Navigation UI patterns, such as navigation drawers and bottom navigation, with minimal additional work.

Here are the steps to Implement Navigation Component: (For J)

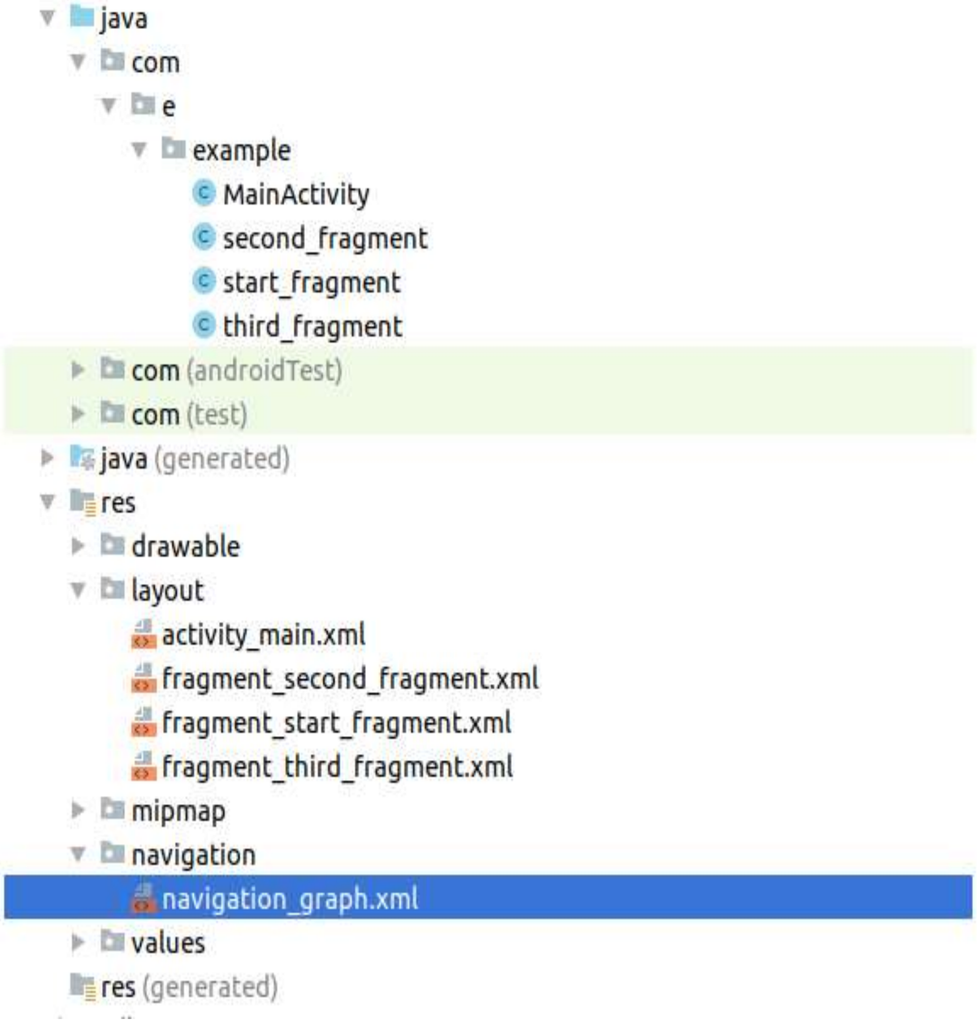
First of all, add the dependency in the application module Gradle file (**build.gradle**).



```
dependencies {
    def nav_version = "2.1.0"

    // Java language implementation
    implementation "androidx.navigation:navigation-fragment:$nav_version"
    implementation "androidx.navigation:navigation-ui:$nav_version"
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

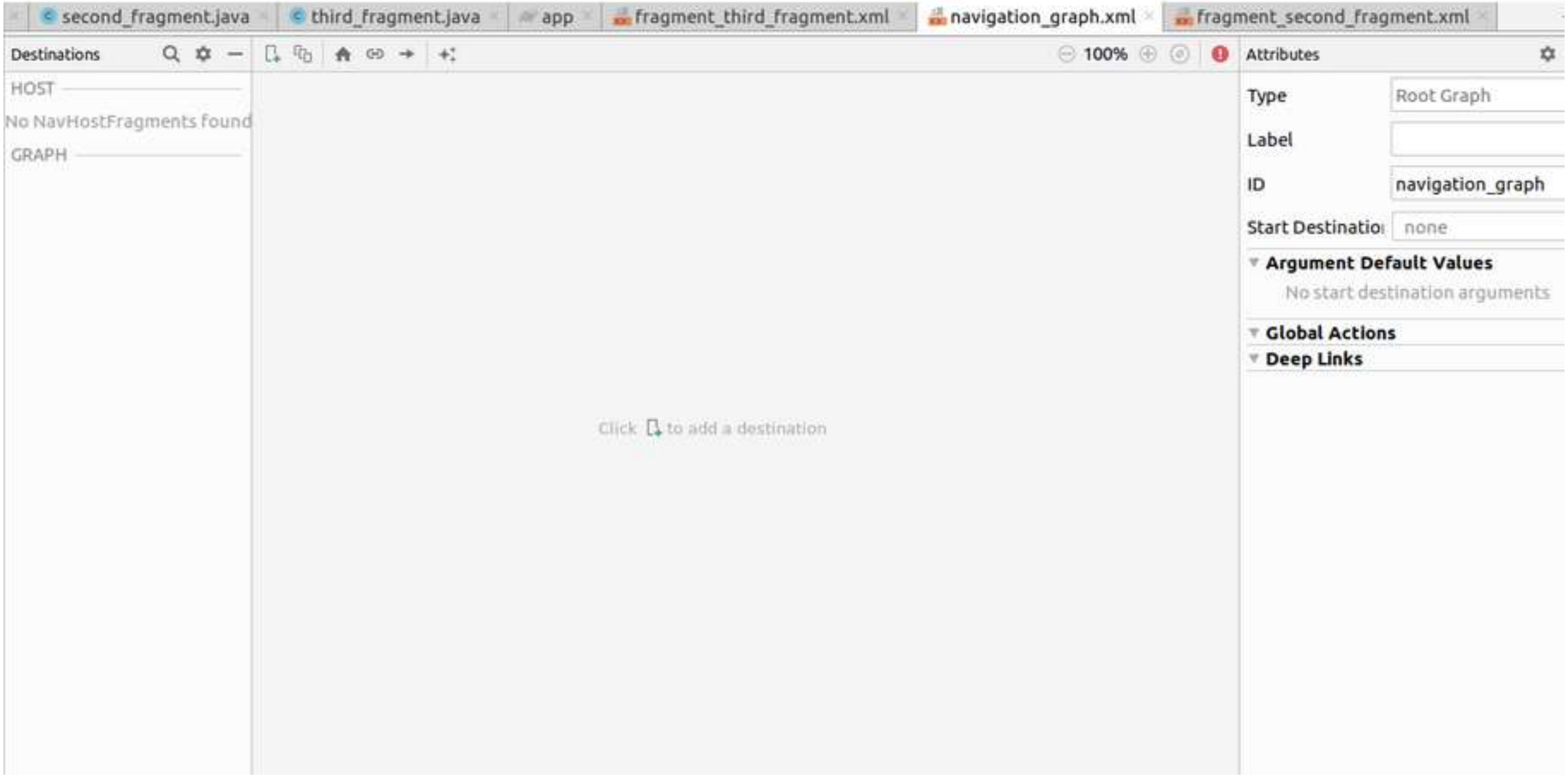
Create a Navigation Graph:

To create a navigation graph, go to App**>>**res, right-click on res and make new Android Resource Directory and set the File name as "navigation" and set resource type as navigation. Now there will be a new folder named navigation. Right-click on the folder; make a new Navigation Resource File named "**navigation_graph**". Now you can see in the screenshot that .xml file is generated.



```
▼ java
    ▼ com
        ▼ e
            ▼ example
                MainActivity
                second_fragment
                start_fragment
                third_fragment
    ▶ com (androidTest)
    ▶ com (test)
▶ java (generated)
▼ res
    ▶ drawable
    ▼ layout
        activity_main.xml
        fragment_second_fragment.xml
        fragment_start_fragment.xml
        fragment_third_fragment.xml
    ▶ mipmap
    ▼ navigation
        navigation_graph.xml
    ▶ values
  res (generated)
```

Open **nav_graph.xml**



second_fragment.java
third_fragment.java
app
fragment_third_fragment.xml
navigation_graph.xml
fragment_second_fragment.xml
Destinations
HOST
No NavHostFragments found
GRAPH
100%
Click to add a destination
Attributes

Type: Root Graph

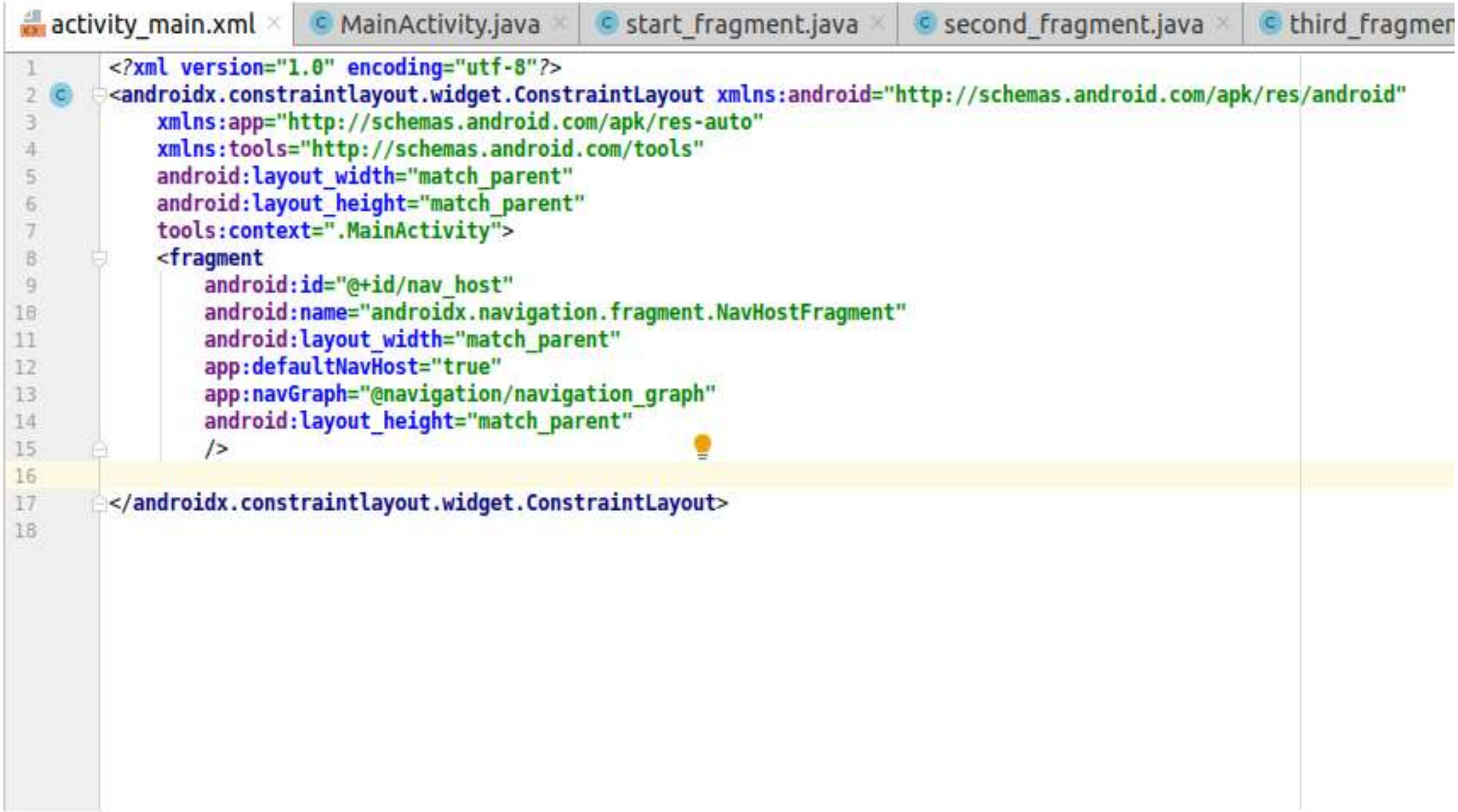
Label:

ID: navigation_graph

Start Destinatio: none

- Argument Default Values
  - No start destination arguments
- Global Actions
- Deep Links

Navigation Graph includes 3 sections – Destination, Graph and Attributes.now, **NavHost** main activity. Which will be a single activity. By adding this code into **activity_main.xml** we can host activity.

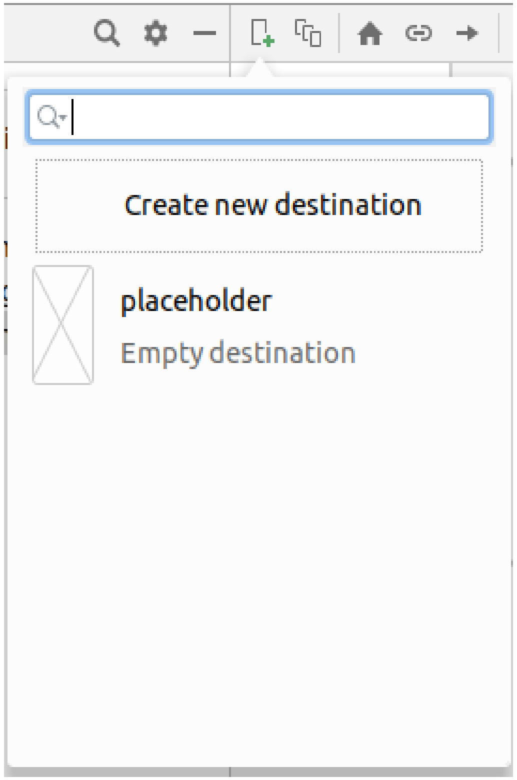


```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <fragment
        android:id="@+id/nav_host"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/navigation_graph"
        android:layout_height="match_parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

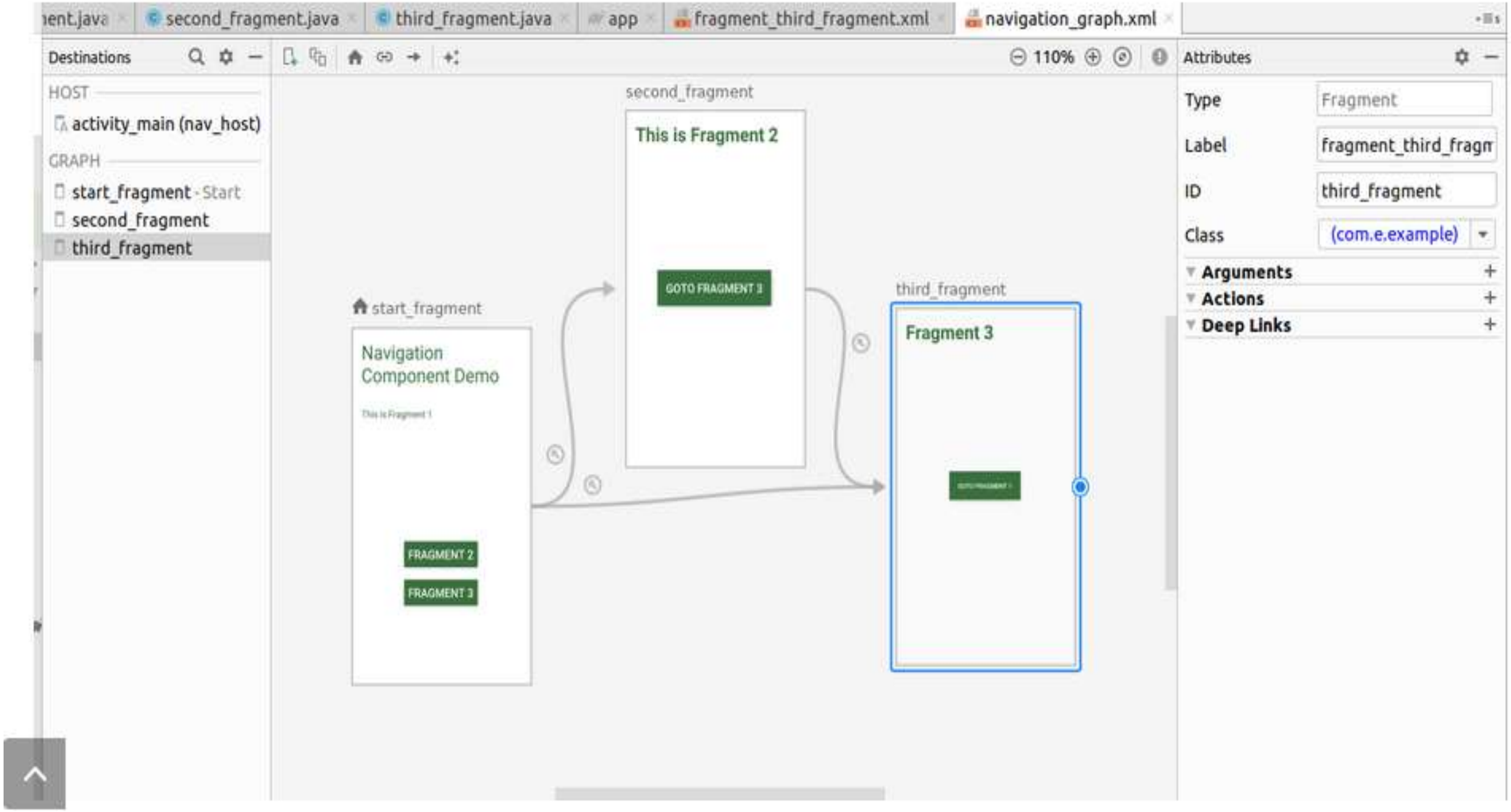
- **app:navGraph:** defines which Navigation Graph will be associated with the Navigation Host
- **app:defaultNavHost="true":** ensures that the Navigation Host intercepts the system back button when pressed.

Now, we can see the Top first icon in Graph toolbar using which we can add new fragments and placeholders.

Click on create a new destination and you can create new fragments.



Make at least two fragments. First Home Fragment will be Fragment and another will be the destination. Here we can see 3 fragments- Start fragment, Second fragment, and Third fragment.



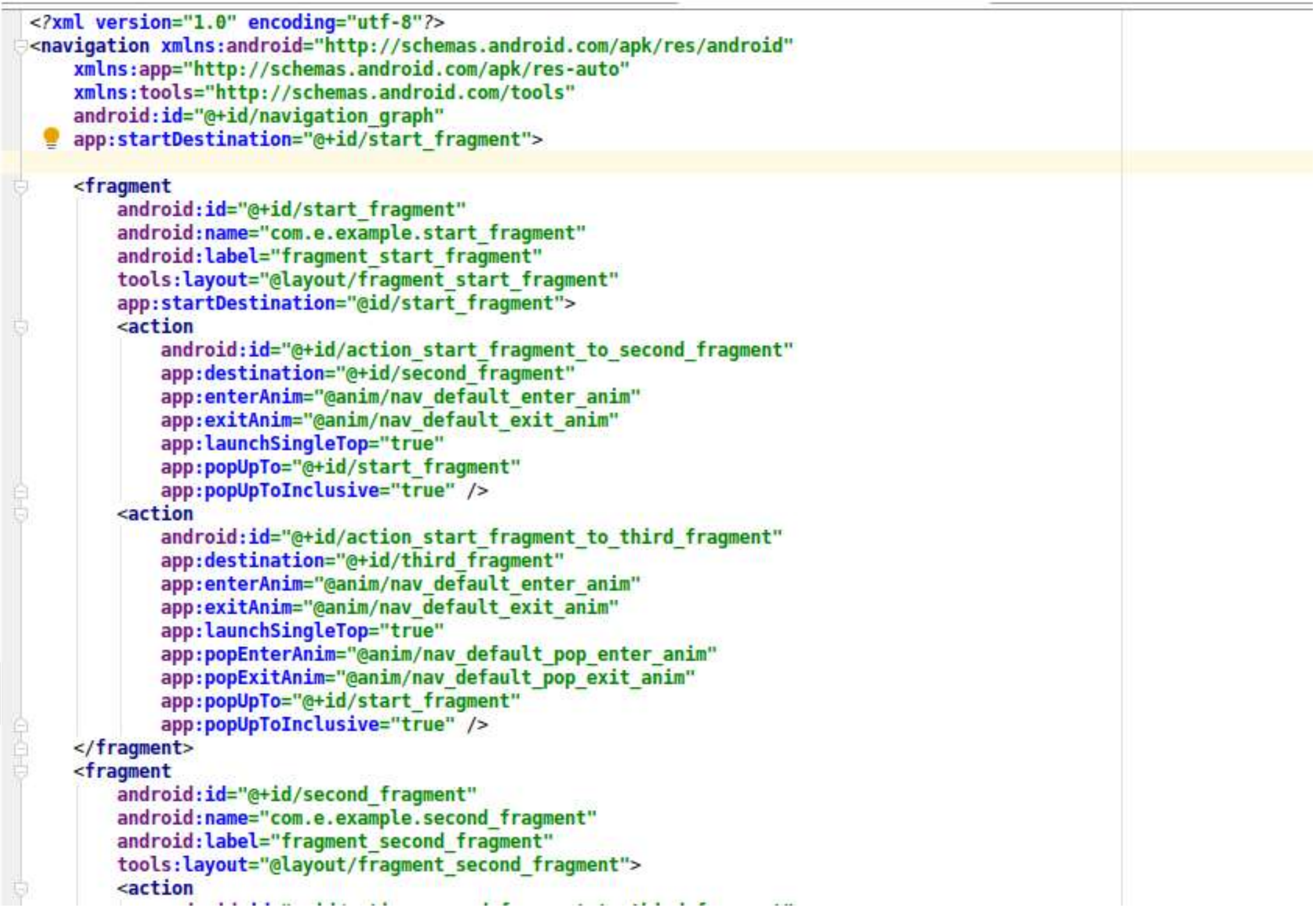
Type: Fragment

Label: fragment_third_fragm

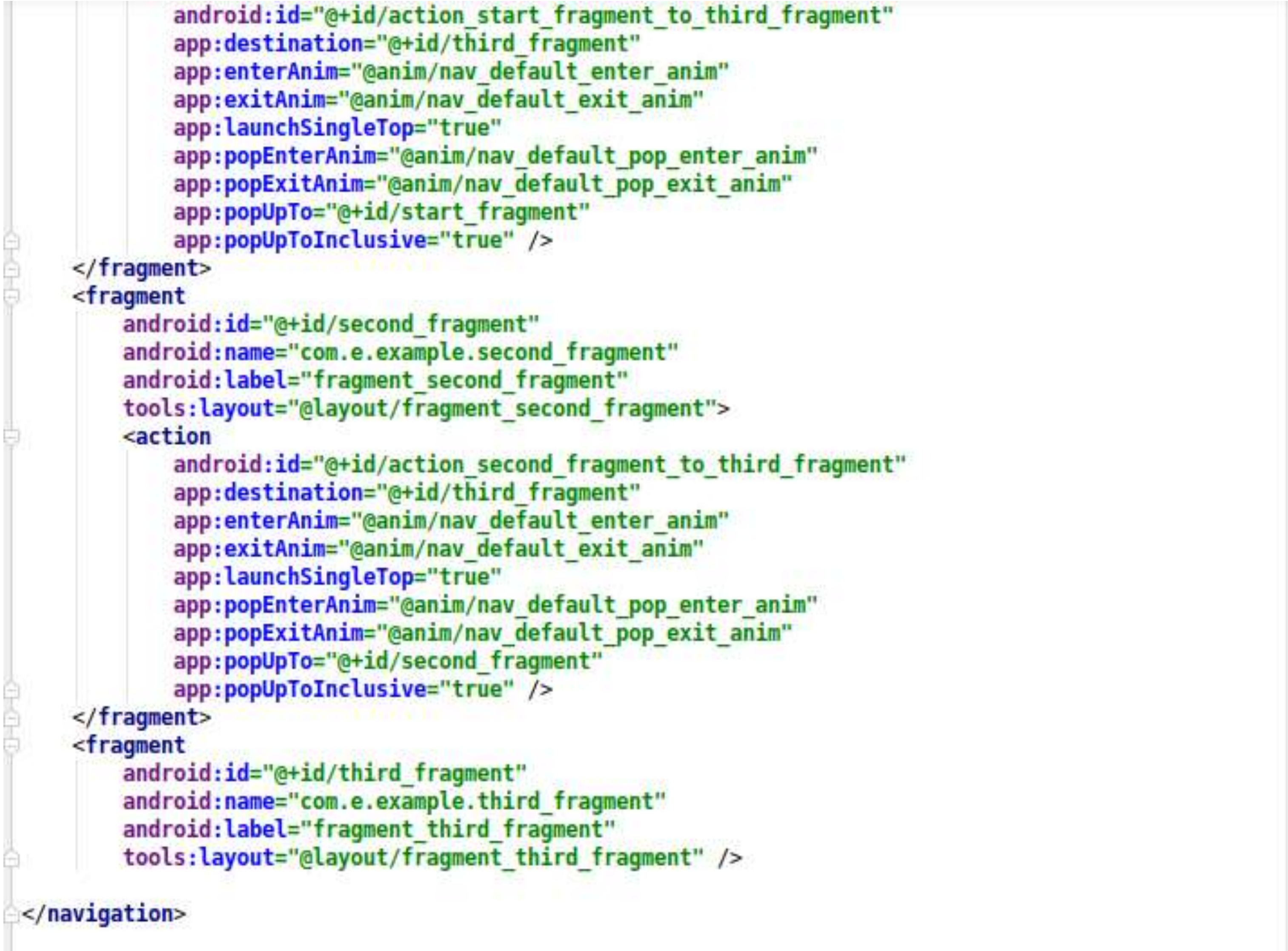
ID: third_fragment

Class: (com.e.example)

start_fragment.xml

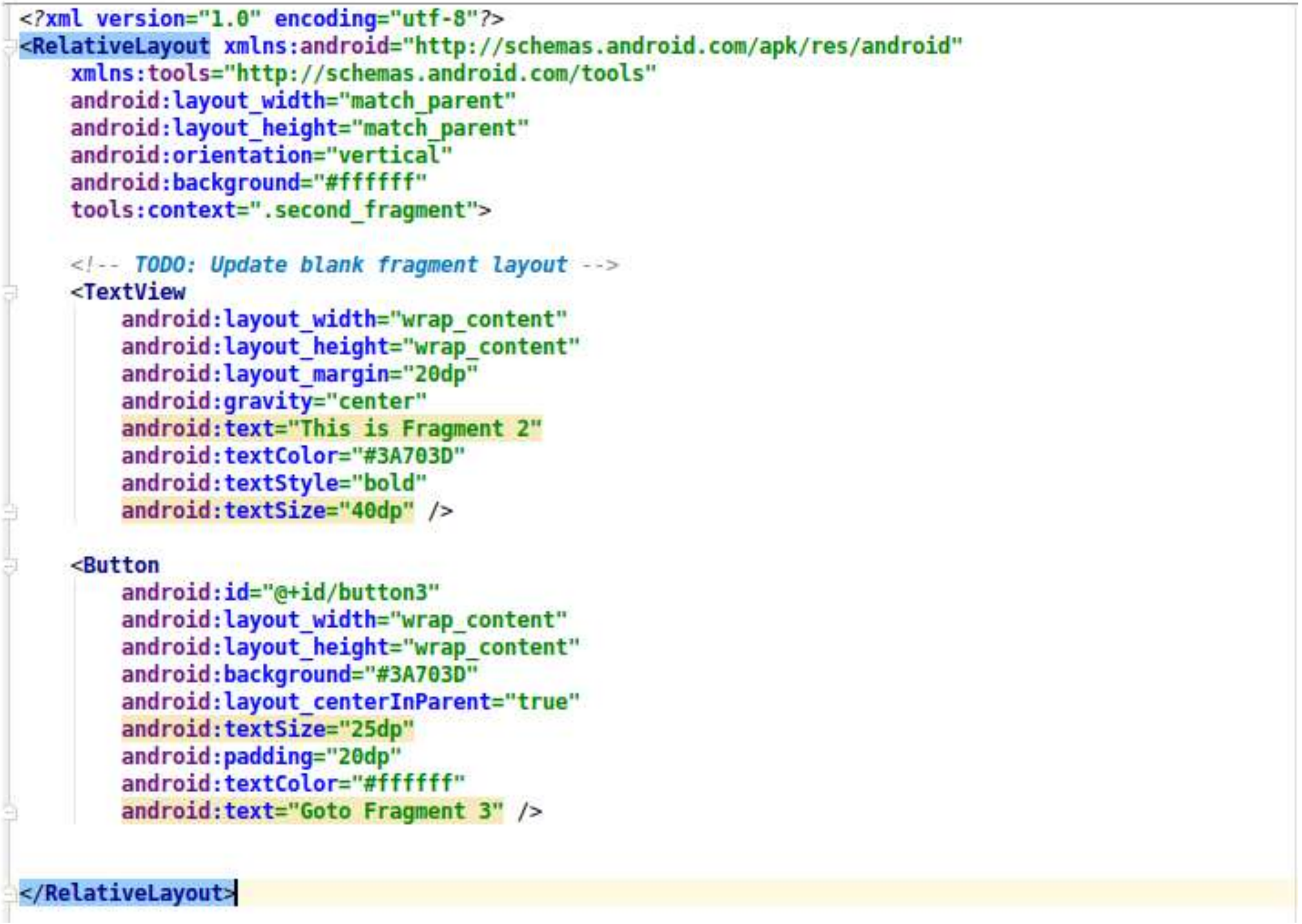


```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/navigation_graph"
    app:startDestination="@+id/start_fragment">

    <fragment
        android:id="@+id/start_fragment"
        android:name="com.e.example.start_fragment"
        android:label="fragment_start_fragment"
        tools:layout="@layout/fragment_start_fragment"
        app:startDestination="@id/start_fragment">
        <action
            android:id="@+id/action_start_fragment_to_second_fragment"
            app:destination="@+id/second_fragment"
            app:enterAnim="@anim/nav_default_enter_anim"
            app:exitAnim="@anim/nav_default_exit_anim"
            app:launchSingleTop="true"
            app:popUpTo="@+id/start_fragment"
            app:popUpToInclusive="true" />
        <action
            android:id="@+id/action_start_fragment_to_third_fragment"
            app:destination="@+id/third_fragment"
            app:enterAnim="@anim/nav_default_enter_anim"
            app:exitAnim="@anim/nav_default_exit_anim"
            app:launchSingleTop="true"
            app:popEnterAnim="@anim/nav_default_pop_enter_anim"
            app:popExitAnim="@anim/nav_default_pop_exit_anim"
            app:popUpTo="@+id/start_fragment"
            app:popUpToInclusive="true" />
    </fragment>
    <fragment
        android:id="@+id/second_fragment"
        android:name="com.e.example.second_fragment"
        android:label="fragment_second_fragment"
        tools:layout="@layout/fragment_second_fragment">
        <action
```



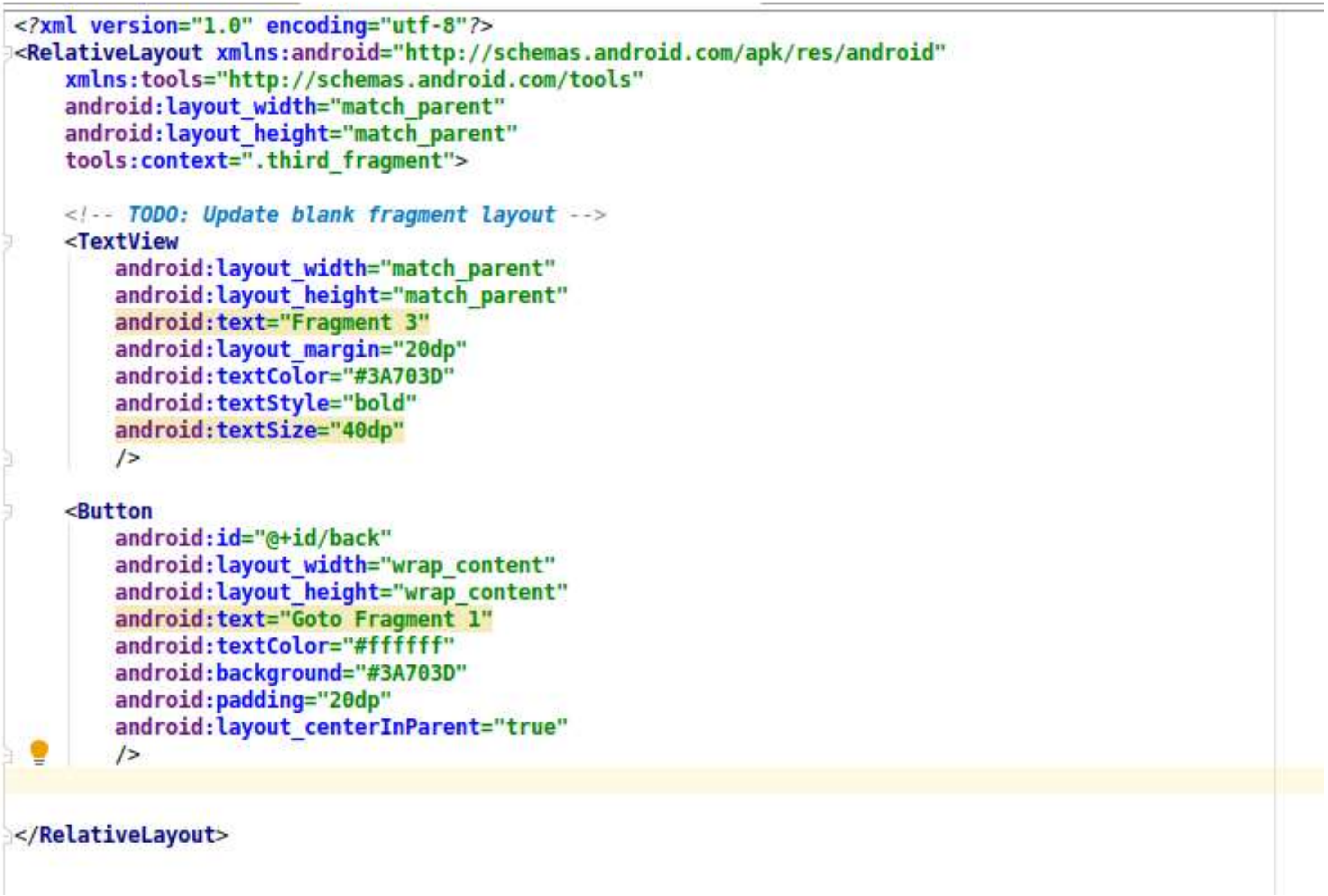
```
            android:id="@+id/action_start_fragment_to_third_fragment"
            app:destination="@+id/third_fragment"
            app:enterAnim="@anim/nav_default_enter_anim"
            app:exitAnim="@anim/nav_default_exit_anim"
            app:launchSingleTop="true"
            app:popEnterAnim="@anim/nav_default_pop_enter_anim"
            app:popExitAnim="@anim/nav_default_pop_exit_anim"
            app:popUpTo="@+id/start_fragment"
            app:popUpToInclusive="true" />
    </fragment>
    <fragment
        android:id="@+id/second_fragment"
        android:name="com.e.example.second_fragment"
        android:label="fragment_second_fragment"
        tools:layout="@layout/fragment_second_fragment">
        <action
            android:id="@+id/action_second_fragment_to_third_fragment"
            app:destination="@+id/third_fragment"
            app:enterAnim="@anim/nav_default_enter_anim"
            app:exitAnim="@anim/nav_default_exit_anim"
            app:launchSingleTop="true"
            app:popEnterAnim="@anim/nav_default_pop_enter_anim"
            app:popExitAnim="@anim/nav_default_pop_exit_anim"
            app:popUpTo="@+id/second_fragment"
            app:popUpToInclusive="true" />
    </fragment>
    <fragment
        android:id="@+id/third_fragment"
        android:name="com.e.example.third_fragment"
        android:label="fragment_third_fragment"
        tools:layout="@layout/fragment_third_fragment" />

</navigation>
```

Second_fragment.xml



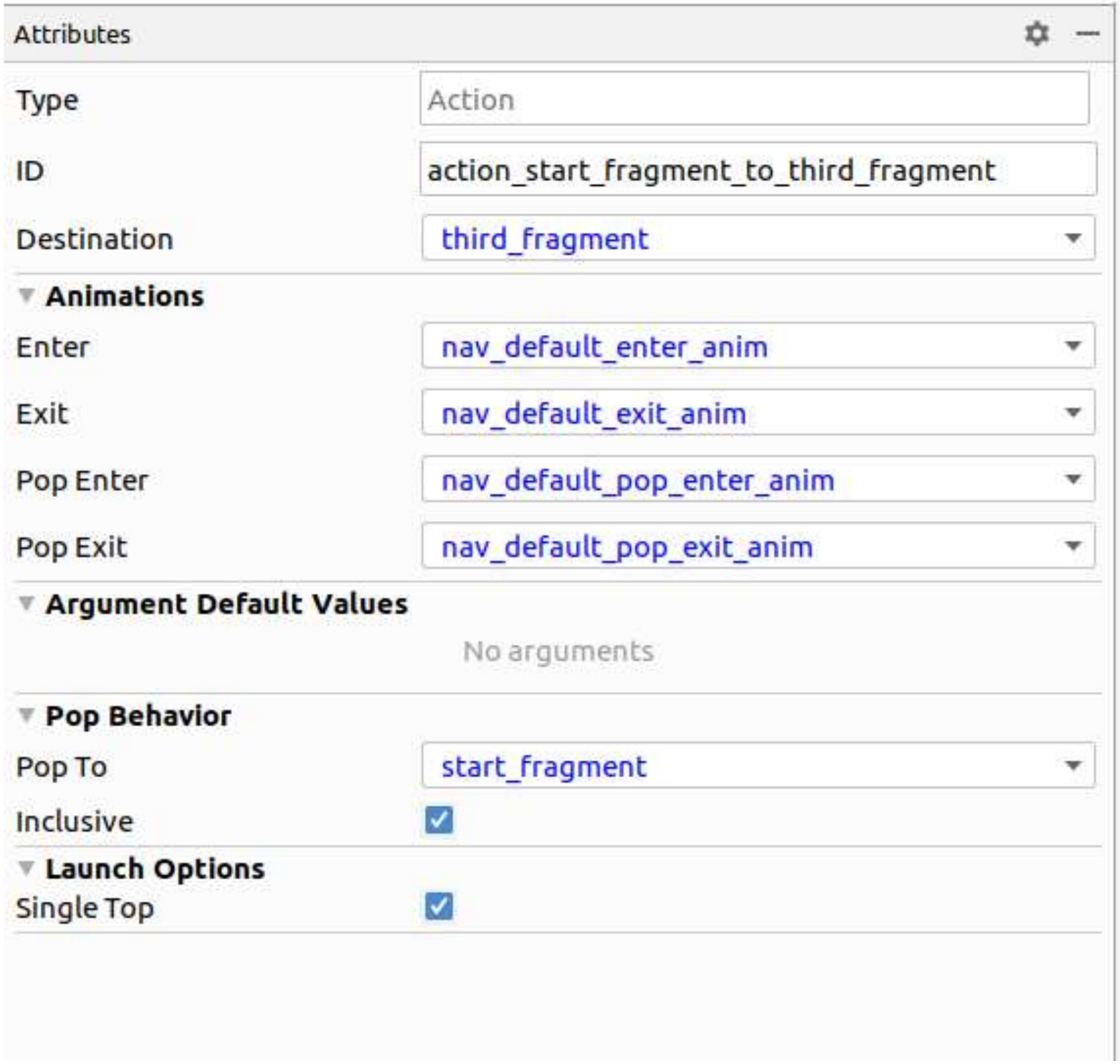
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#ffffff"
    tools:context=".second_fragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="This is Fragment 2"
        android:textColor="#3A703D"
        android:textStyle="bold"
        android:textSize="40dp" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#3A703D"
        android:layout_centerInParent="true"
        android:textSize="25dp"
        android:padding="20dp"
        android:textColor="#ffffff"
        android:text="Goto Fragment 3" />


</RelativeLayout>
```

Third_fragment.xml



```
<?xml version="1.0" encoding="utf-8"?>
<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=".third_fragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Fragment 3"
        android:layout_margin="20dp"
        android:textColor="#3A703D"
        android:textStyle="bold"
        android:textSize="40dp"
        />

    <Button
        android:id="@+id/back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Goto Fragment 1"
        android:textColor="#ffffff"
        android:background="#3A703D"
        android:padding="20dp"
        android:layout_centerInParent="true"
        />


</RelativeLayout>
```

Now add action between fragments for navigation by dragging an arrow from one fragment to another fragment.



Attributes

Type: Action

ID: action_start_fragment_to_third_fragment

Destination: third_fragment

Animations

Enter: nav_default_enter_anim

Exit: nav_default_exit_anim

Pop Enter: nav_default_pop_enter_anim

Pop Exit: nav_default_pop_exit_anim

Argument Default Values

No arguments

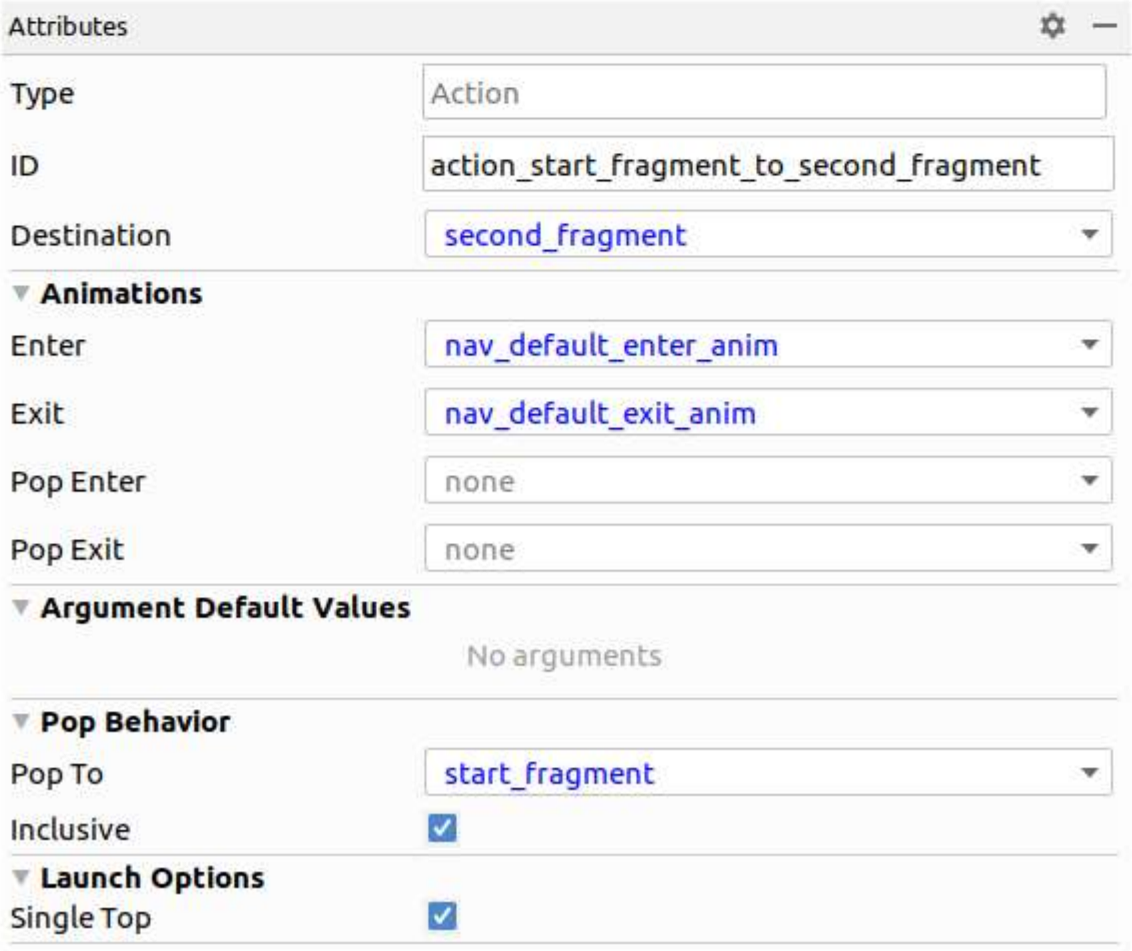
Pop Behavior

Pop To: start_fragment

Inclusive ☑

Launch Options

Single Top ☑



Attributes

Type Action

ID action_start_fragment_to_second_fragment

Destination second_fragment

Animations

Enter nav_default_enter_anim

Exit nav_default_exit_anim

Pop Enter none

Pop Exit none

Argument Default Values

No arguments

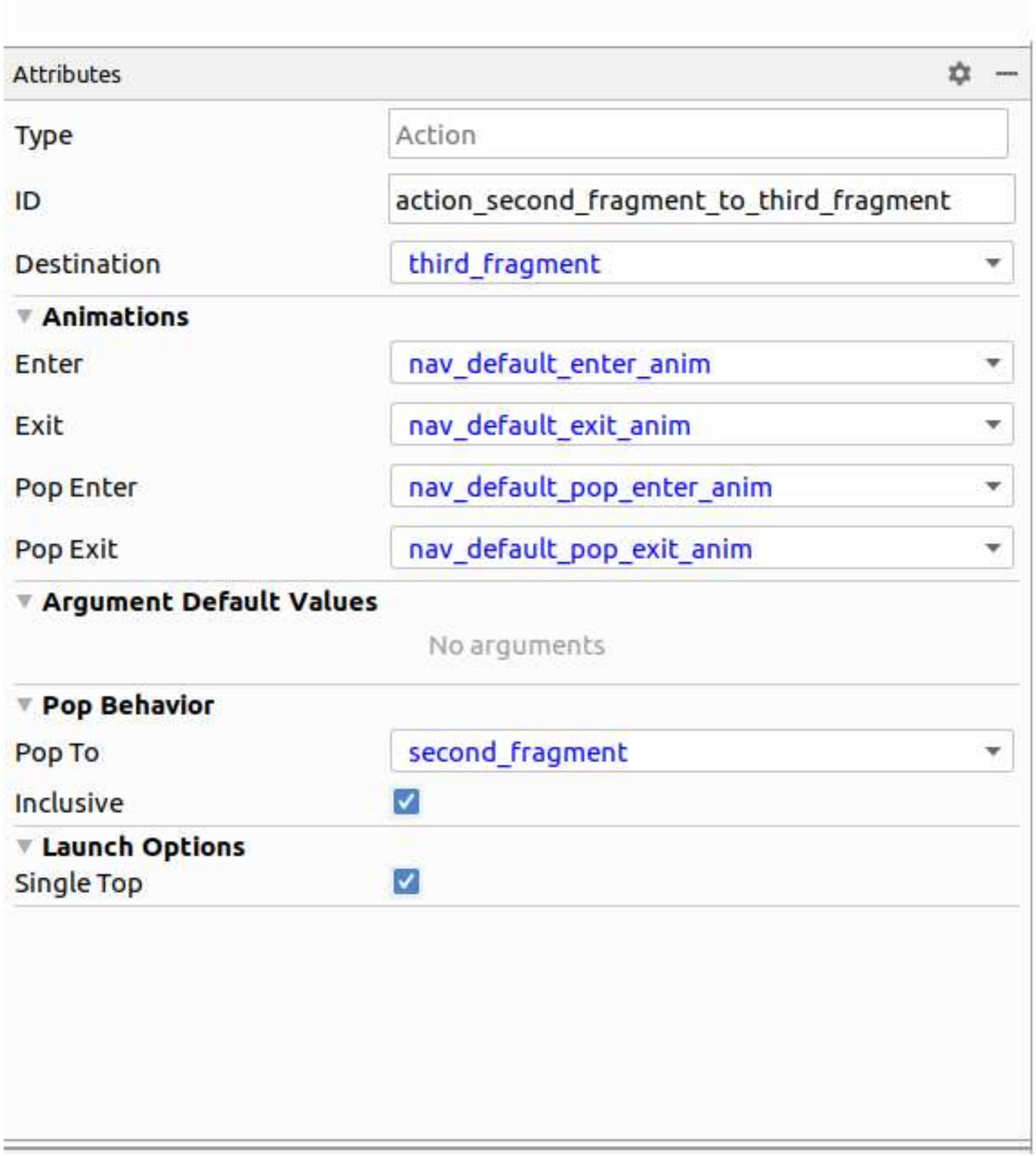
Pop Behavior

Pop To start_fragment

Inclusive ☑

Launch Options

Single Top ☑



Attributes

Type: Action

ID: action_second_fragment_to_third_fragment

Destination: third_fragment

▼ Animations

Enter: nav_default_enter_anim

Exit: nav_default_exit_anim

Pop Enter: nav_default_pop_enter_anim

Pop Exit: nav_default_pop_exit_anim

▼ Argument Default Values

No arguments

▼ Pop Behavior

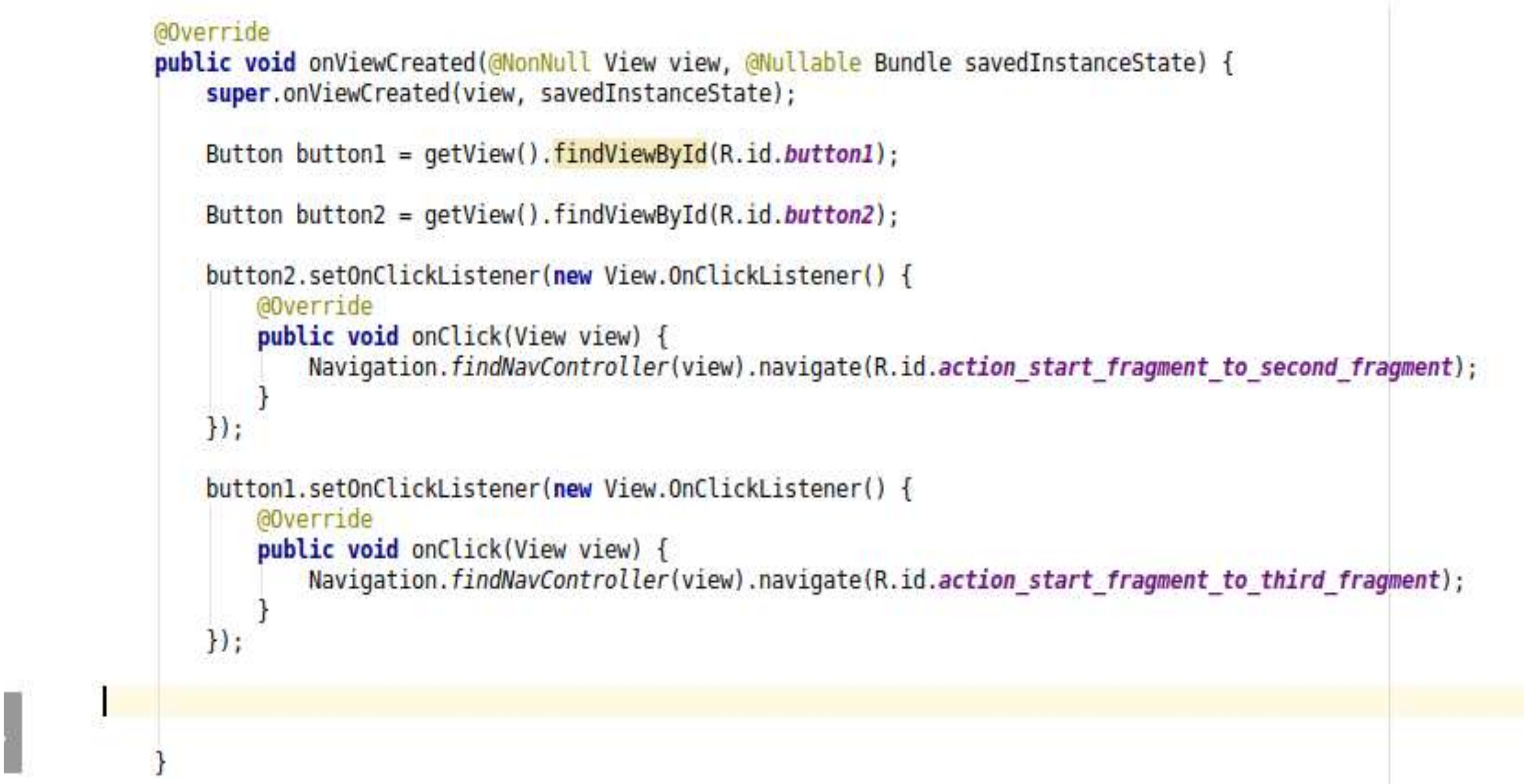
Pop To: second_fragment

Inclusive ☑

▼ Launch Options

Single Top ☑

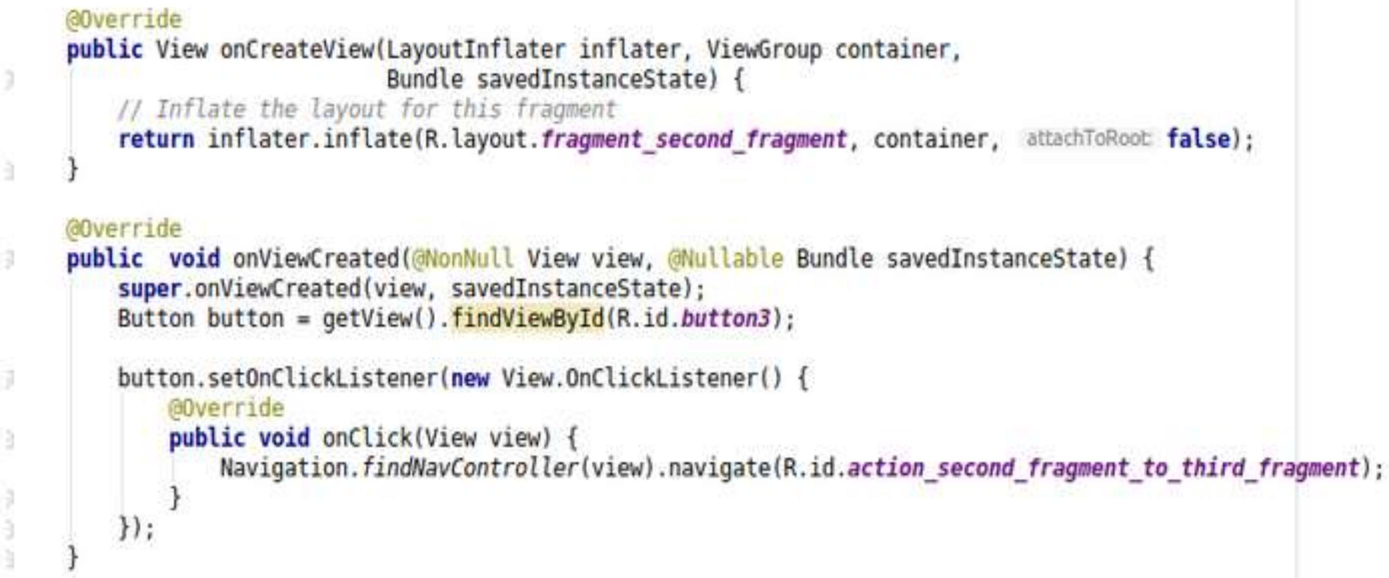
In **start_fragment**, write this java code to perform the navigation action



```
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    Button button1 = getView().findViewById(R.id.button1);

    Button button2 = getView().findViewById(R.id.button2);

    button2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Navigation.findNavController(view).navigate(R.id.action_start_fragment_to_second_fragment);
        }
    });

    button1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Navigation.findNavController(view).navigate(R.id.action_start_fragment_to_third_fragment);
        }
    });


}
```

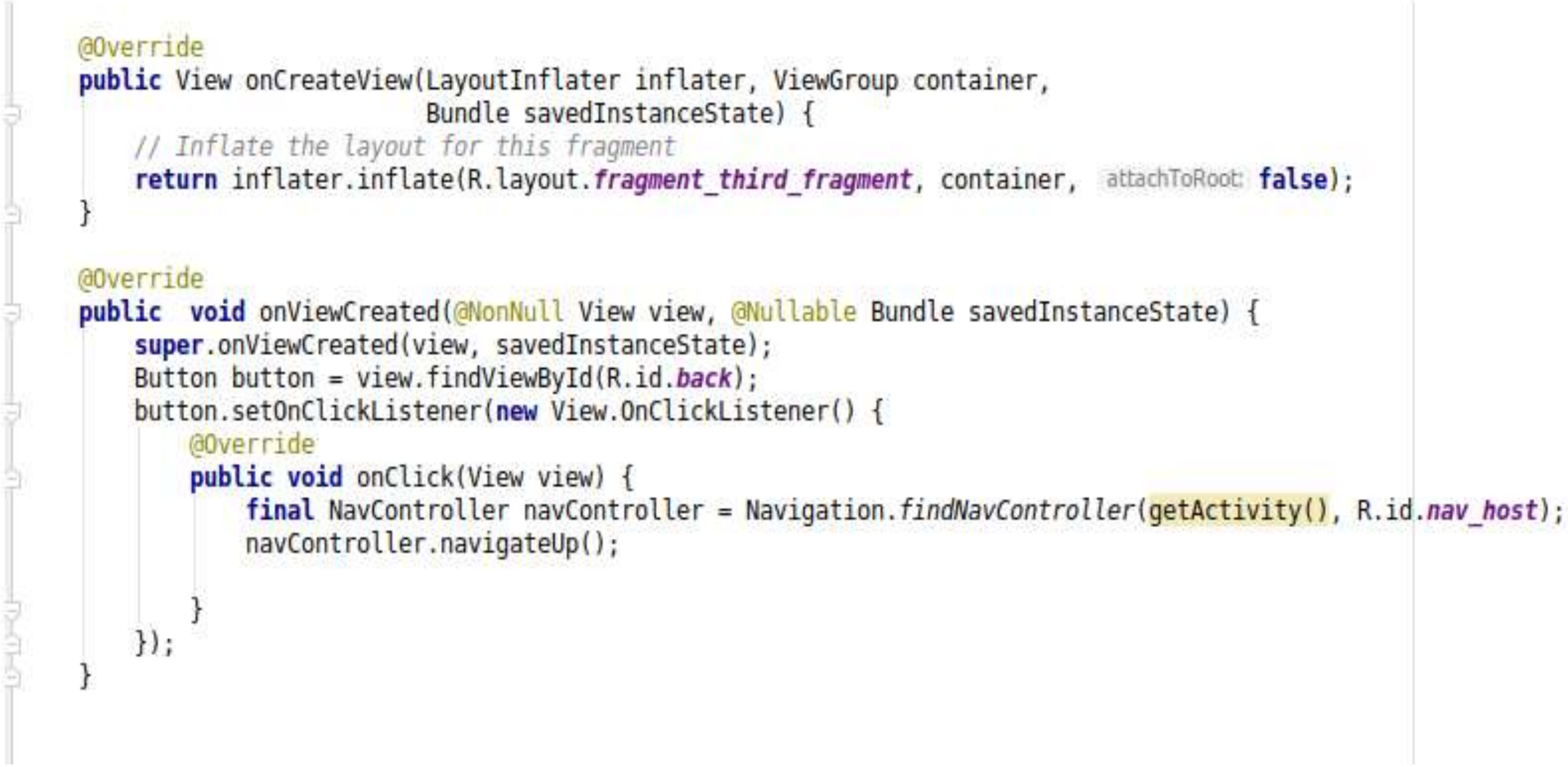
NavController manages app navigation within the NavHost. We pass the action id inside the **createNavigateOnClickListener** method.

Second_fragment.java



```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_second_fragment, container, attachToRoot: false);
}

@Override
public  void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    Button button = getView().findViewById(R.id.button3);

    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Navigation.findNavController(view).navigate(R.id.action_second_fragment_to_third_fragment);
        }
    });
}
```

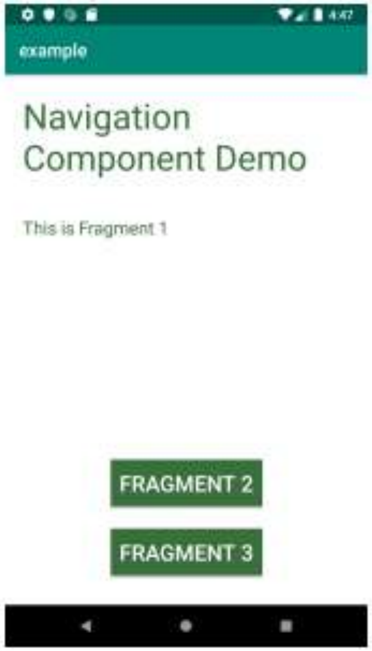
Third_fragment.java



```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_third_fragment, container,  attachToRoot: false);
}

@Override
public  void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    Button button = view.findViewById(R.id.back);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            final NavController navController = Navigation.findNavController(getActivity(), R.id.nav_host);
            navController.navigateUp();

        }
    });
}
```

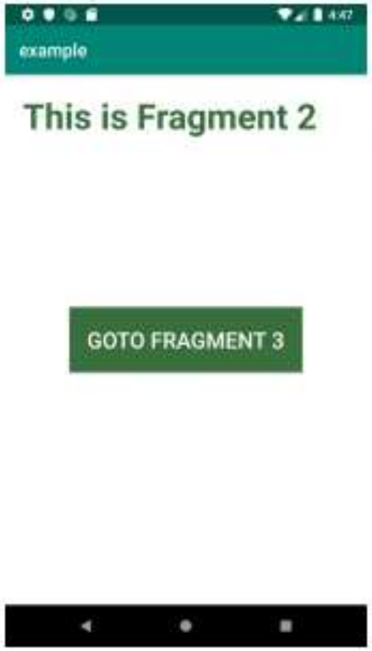
navigateUp() is used to return to the previous fragment.

Output:

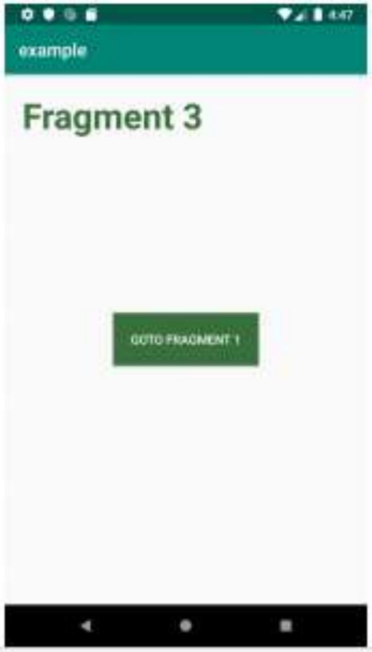


Navigation Component Demo

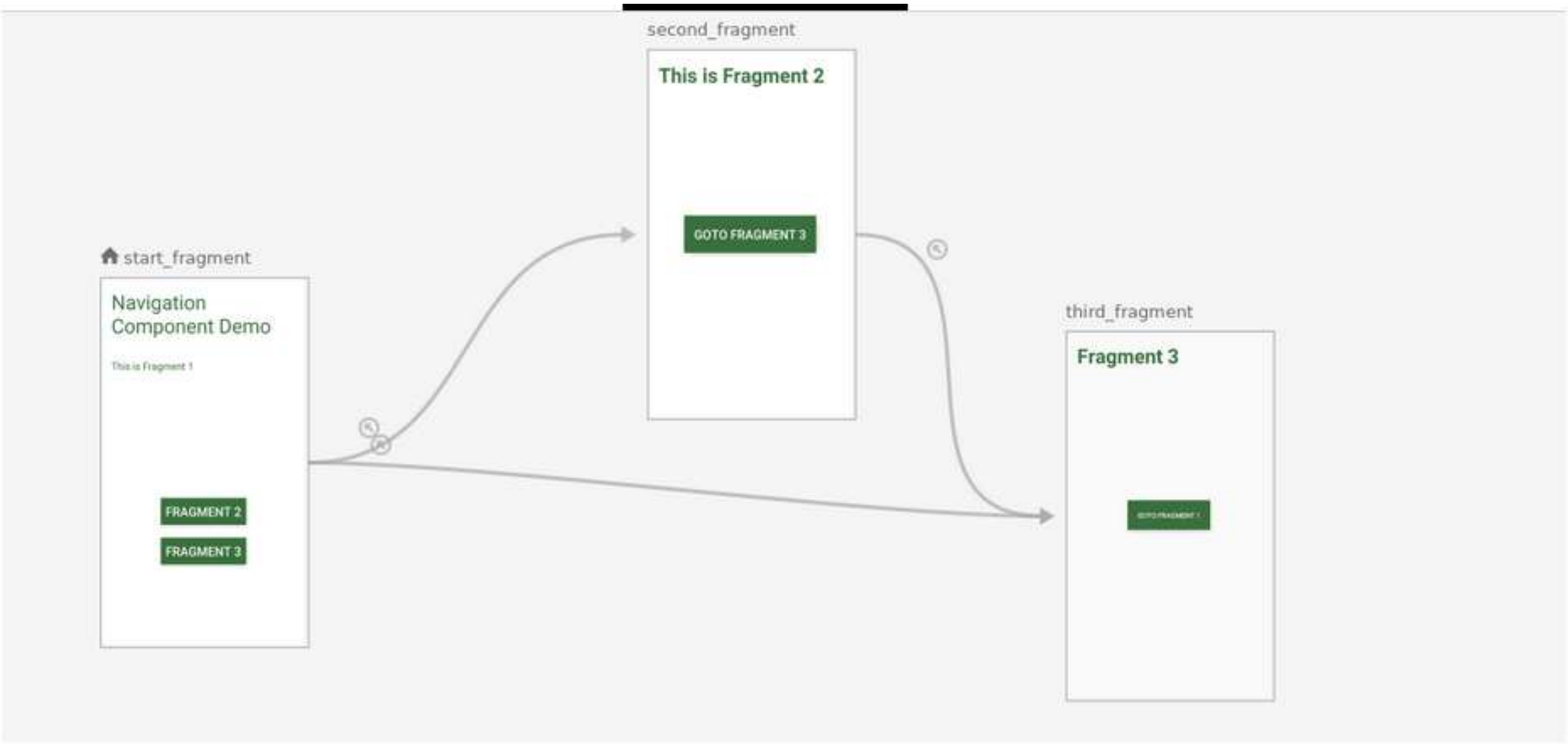
This is Fragment 1



This is Fragment 2



Fragment 3


second_fragment
This is Fragment 2
GOTO FRAGMENT 3
start_fragment
Navigation Component Demo
This is Fragment 1
FRAGMENT 2
FRAGMENT 3
third_fragment
Fragment 3
GOTO FRAGMENT 1

Topic 6: Introduction to IOS features and Objective C Features

iOS (formerly **iPhone OS**) is a mobile operating system created and developed by Apple Inc. exclusively for its hardware.

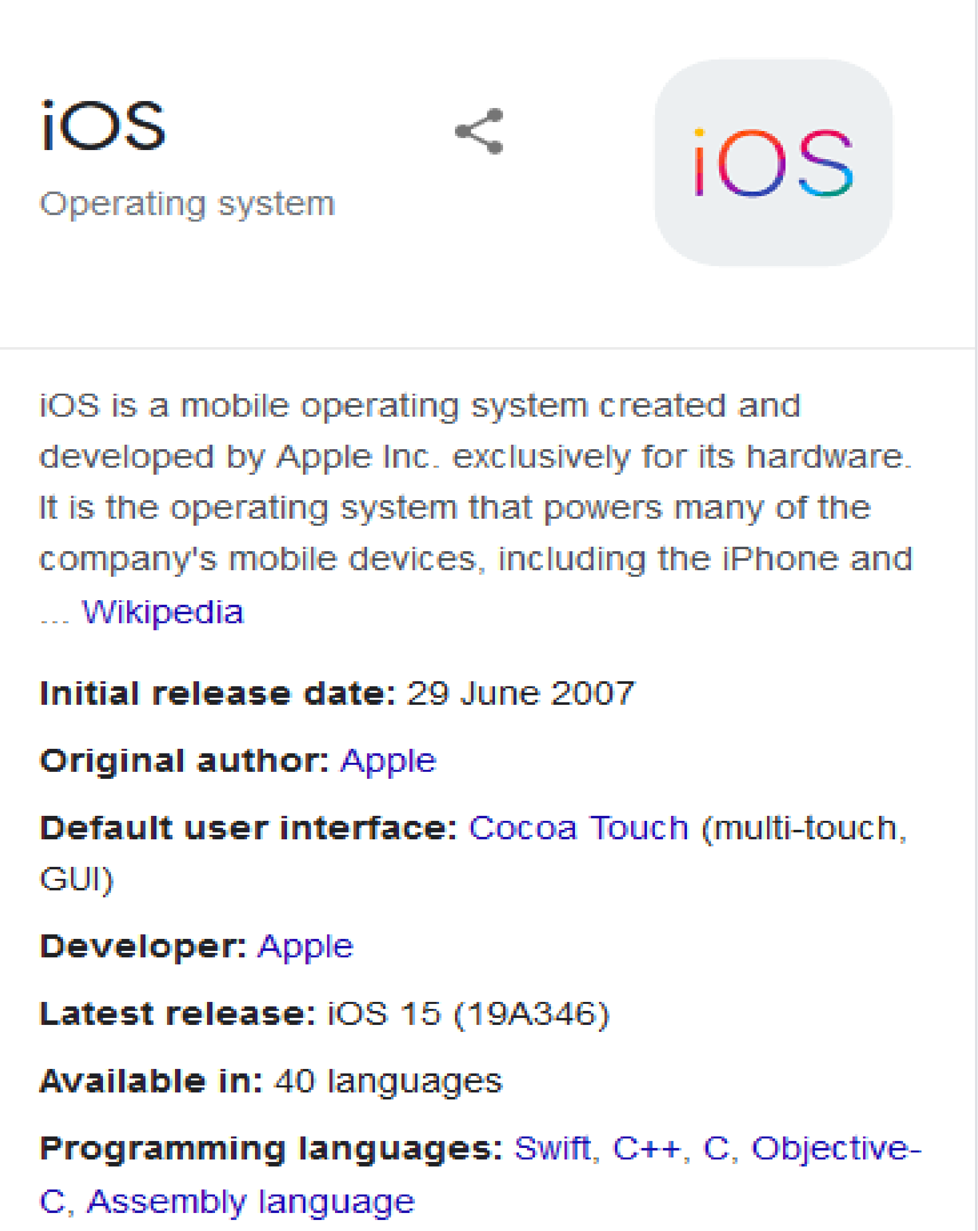
It is the operating system that powers many of the company's mobile devices, including the iPhone and iPod Touch; t

he term also included the versions running on iPads until the name *iPadOS* was introduced with version 13 in 2019.

It is the world's second-most widely installed mobile operating system, after Android.

It is the basis for three other operating systems made by Apple: iPadOS, tvOS, and watchOS.

It is proprietary software, although some parts of it are open source under the Apple Public Source License and other licenses.



iOS

Operating system

iOS is a mobile operating system created and developed by Apple Inc. exclusively for its hardware. It is the operating system that powers many of the company's mobile devices, including the iPhone and ... Wikipedia

Initial release date: 29 June 2007

Original author: Apple

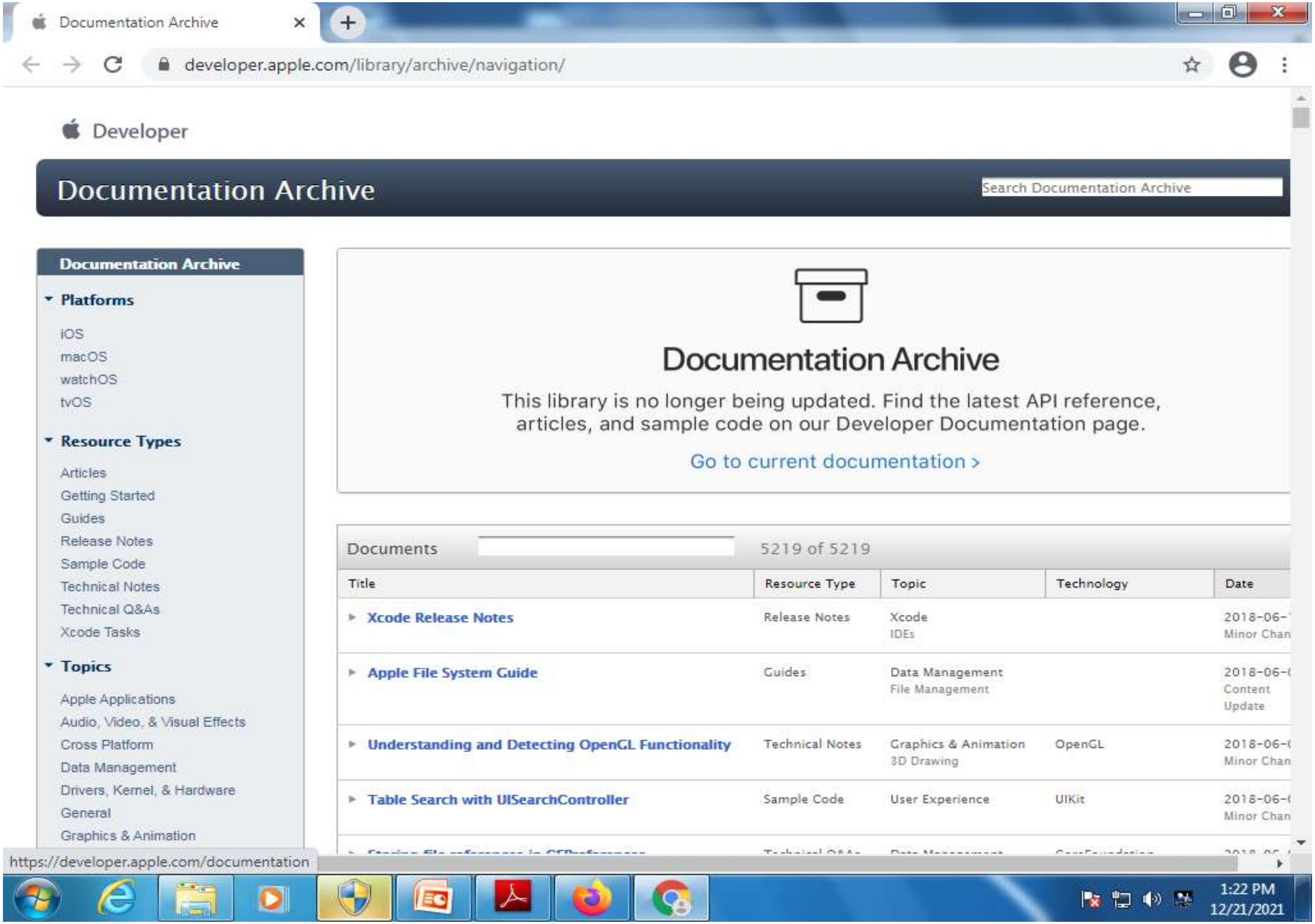
Default user interface: Cocoa Touch (multi-touch, GUI)

Developer: Apple

Latest release: iOS 15 (19A346)

Available in: 40 languages

Programming languages: Swift, C++, C, Objective-C, Assembly language https://developer.apple.com/library/archive/navigation/



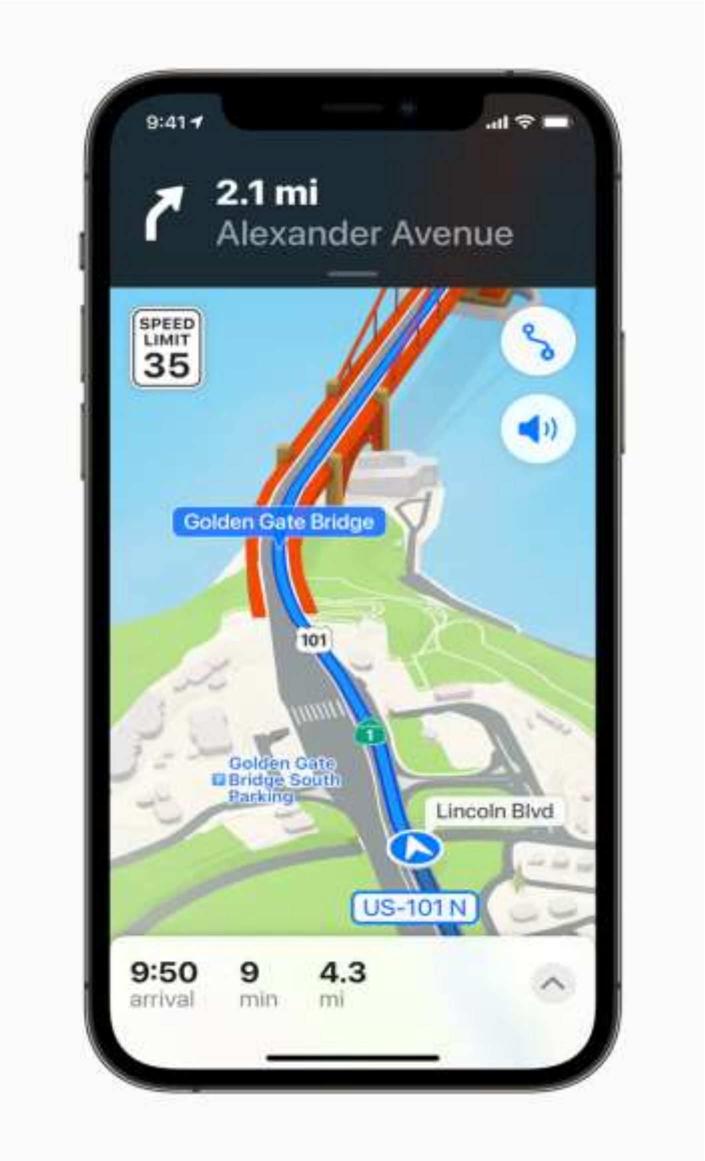
Documentation Archive

This library is no longer being updated. Find the latest API reference, articles, and sample code on our Developer Documentation page.

Go to current documentation ›

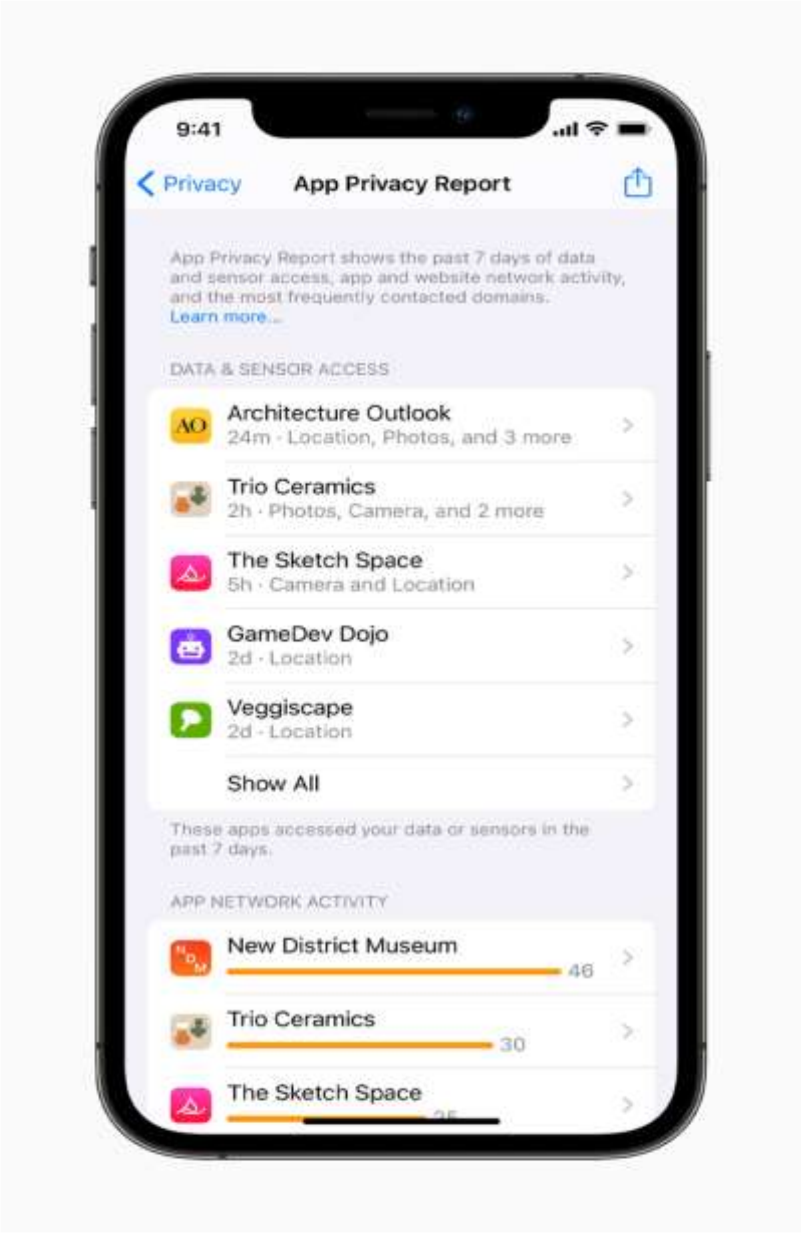
Documents 5219 of 5219

Title	Resource Type	Topic	Technology	Date
▸ Xcode Release Notes	Release Notes	Xcode IDEs		2018-06- Minor Chan
▸ Apple File System Guide	Guides	Data Management File Management		2018-06- Content Update
▸ Understanding and Detecting OpenGL Functionality	Technical Notes	Graphics & Animation 3D Drawing	OpenGL	2018-06- Minor Chan
▸ Table Search with UISearchController	Sample Code	User Experience	UIKit	2018-06- Minor Chan



Explore the World with Apple Maps

Apple is committed to building the world’s best map, and iOS takes Maps even further with brand new ways to navigate and explore. Users will experience significantly enhanced details in cities for neighborhoods, commercial districts, elevation, and buildings, new road colors and labels, custom-designed landmarks, and a new night-time mode with a moonlit glow. This is a whole new way of looking at the world through Maps.



9:41

< Privacy App Privacy Report

App Privacy Report shows the past 7 days of data and sensor access, app and website network activity, and the most frequently contacted domains. Learn more...

DATA & SENSOR ACCESS

- Architecture Outlook — 24m · Location, Photos, and 3 more
- Trio Ceramics — 2h · Photos, Camera, and 2 more
- The Sketch Space — 5h · Camera and Location
- GameDev Dojo — 2d · Location
- Veggiscape — 2d · Location
- Show All

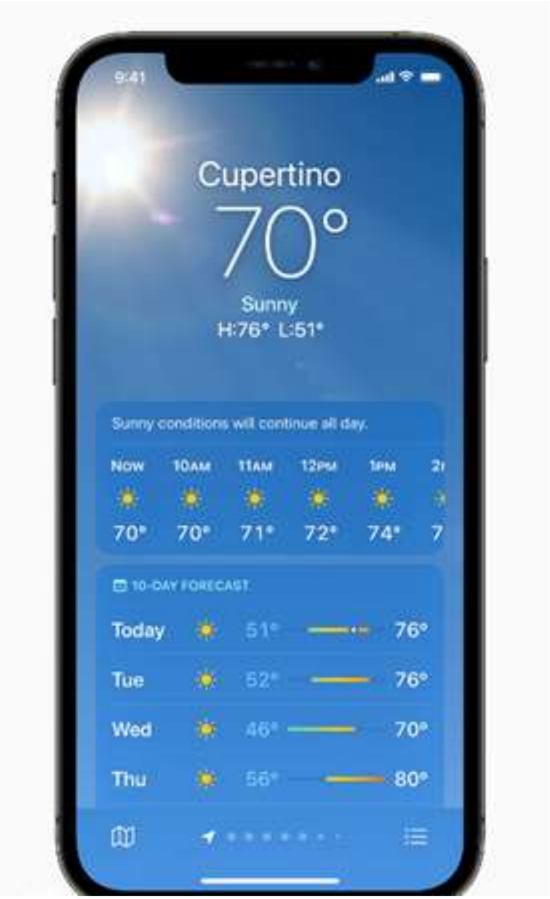
These apps accessed your data or sensors in the past 7 days.

APP NETWORK ACTIVITY

- New District Museum — 46
- Trio Ceramics — 30
- The Sketch Space — 25

New Privacy Features

iOS introduces even more privacy controls to help protect user information. With on-device speech recognition, audio of Siri requests is now processed entirely on iPhone by default, and performance improves significantly. Mail Privacy Protection stops senders from learning whether an email has been opened, and hides IP addresses so senders can’t learn a user’s location or use it to build a profile on them. App Privacy Report offers an overview of how apps use the access that has been granted to location, photos, camera, microphone, and contacts in the last seven days, and which other domains are contacted.



9:41

Cupertino

70°

Sunny

H:76° L:51°

Sunny conditions will continue all day.

Now	10AM	11AM	12PM	1PM	2
☀	☀	☀	☀	☀	☀
70°	70°	71°	72°	74°	7

10-DAY FORECAST

Today	☀	51°	76°
Tue	☀	52°	76°
Wed	☀	46°	70°
Thu	☀	56°	80°

Redesigned Weather and Notes Apps

Weather includes more graphical displays of weather data, full-screen maps, and dynamic layouts that change based on conditions. Beautifully redesigned animated backgrounds more accurately reflect the sun's position and precipitation, and notifications highlight when rain or snow starts and stops.

iCloud+ combines everything users love about iCloud with new premium features, including Hide My Email, expanded HomeKit Secure Video support, and an innovative new internet privacy service, iCloud Private Relay, at no additional cost.[9] Current iCloud subscribers will be upgraded to iCloud+ automatically this fall. All iCloud+ plans can be shared with people in the same Family Sharing group, so everyone can enjoy the new features, storage, and elevated experience that comes with the service.

The **Health** app gets a new sharing tab that lets users share their health data with family, caregivers, or a care team, Trends gives users a way to focus attention on meaningful changes in personal health metrics, and Walking Steadiness is a new metric that empowers people to proactively manage their fall risk.

Objective-C is a general-purpose, object-oriented programming language that adds Smalltalk-style messaging to the C programming language.

This is the main programming language used by Apple for the OS X and iOS operating systems and their respective APIs, Cocoa and Cocoa Touch.

This reference will take you through simple and practical approach while learning Objective-C Programming language.

https://www.tutorialspoint.com/objective_c/index.htm

It is primarily used in developing iOS and Mac OS X operating systems as well as its applications.

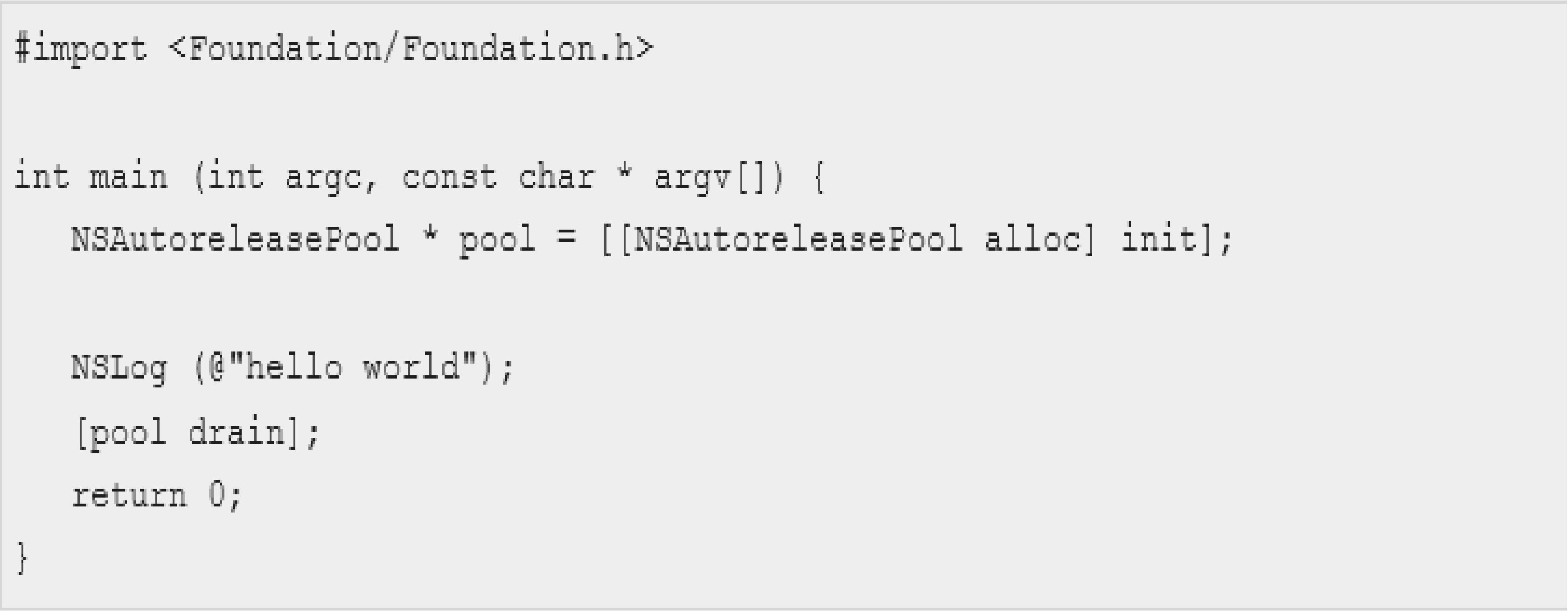
Initially, Objective-C was developed by NeXT for its NeXTSTEP OS from whom it was taken over by Apple for its iOS and Mac OS X.

Object-Oriented Programming

Objective-C fully supports object-oriented programming, including the four pillars of object-oriented development −

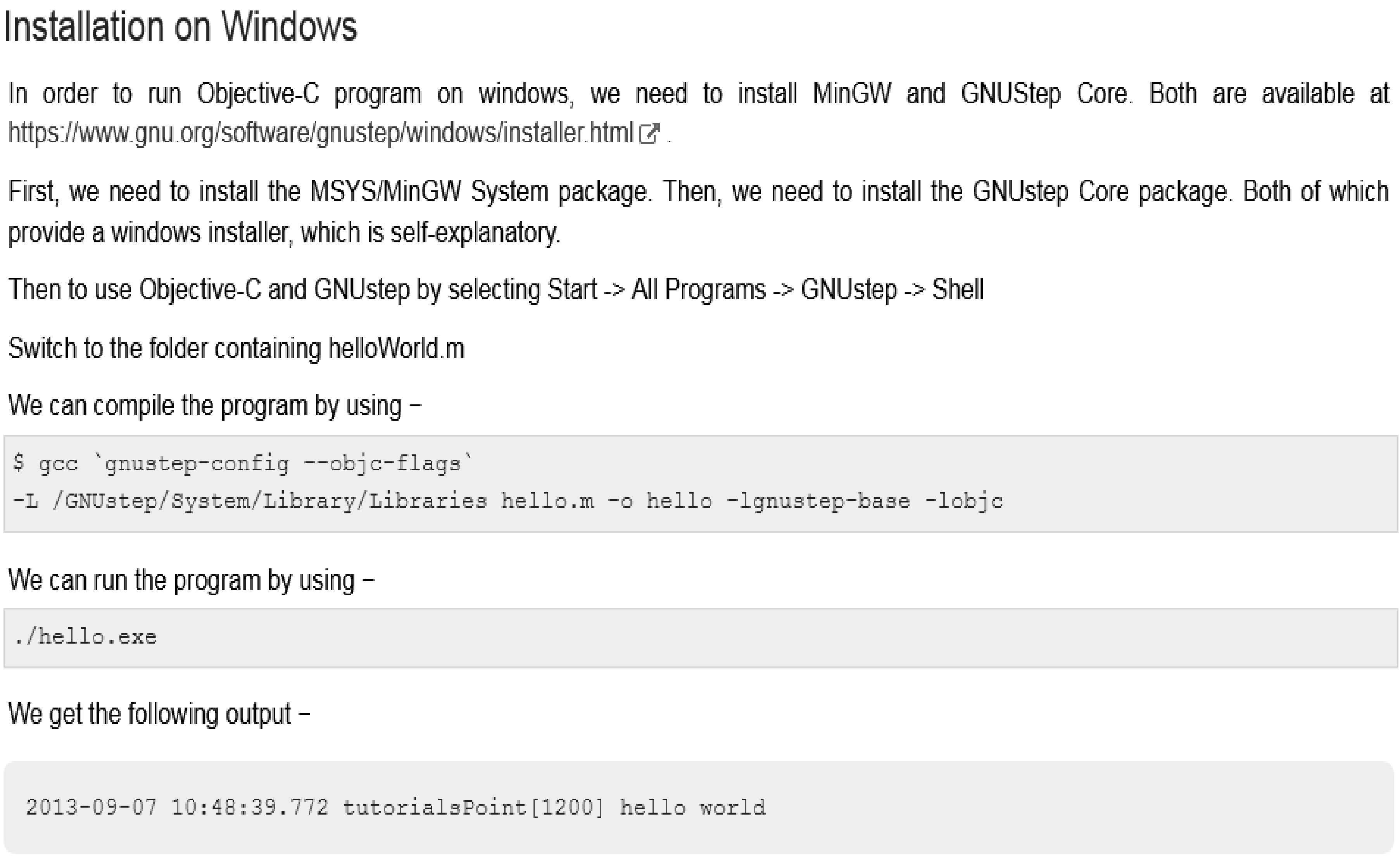
Encapsulation
Data hiding
Inheritance
Polymorphism

Example Code



```
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[]) {
   NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

   NSLog (@"hello world");
   [pool drain];
   return 0;
}
```

https://www.tutorialspoint.com/objective_c/index.htm



Installation on Windows

In order to run Objective-C program on windows, we need to install MinGW and GNUStep Core. Both are available at https://www.gnu.org/software/gnustep/windows/installer.html .

First, we need to install the MSYS/MinGW System package. Then, we need to install the GNUstep Core package. Both of which provide a windows installer, which is self-explanatory.

Then to use Objective-C and GNUstep by selecting Start -> All Programs -> GNUstep -> Shell

Switch to the folder containing helloWorld.m

We can compile the program by using –

```
$ gcc `gnustep-config --objc-flags`
-L /GNUstep/System/Library/Libraries hello.m -o hello -lgnustep-base -lobjc
```

We can run the program by using –

```
./hello.exe
```

We get the following output –

```
2013-09-07 10:48:39.772 tutorialsPoint[1200] hello world
```