MOBILE COMPUTING

Lecture Three

Agenda

- Create ,navigate and run a project on virtual and physical devices.

- Create an emulator (virtual device) to run your app on your computer.

- Explore the AndroidManifest.xml file.

- Explore the project layout.

- Activity and Intent

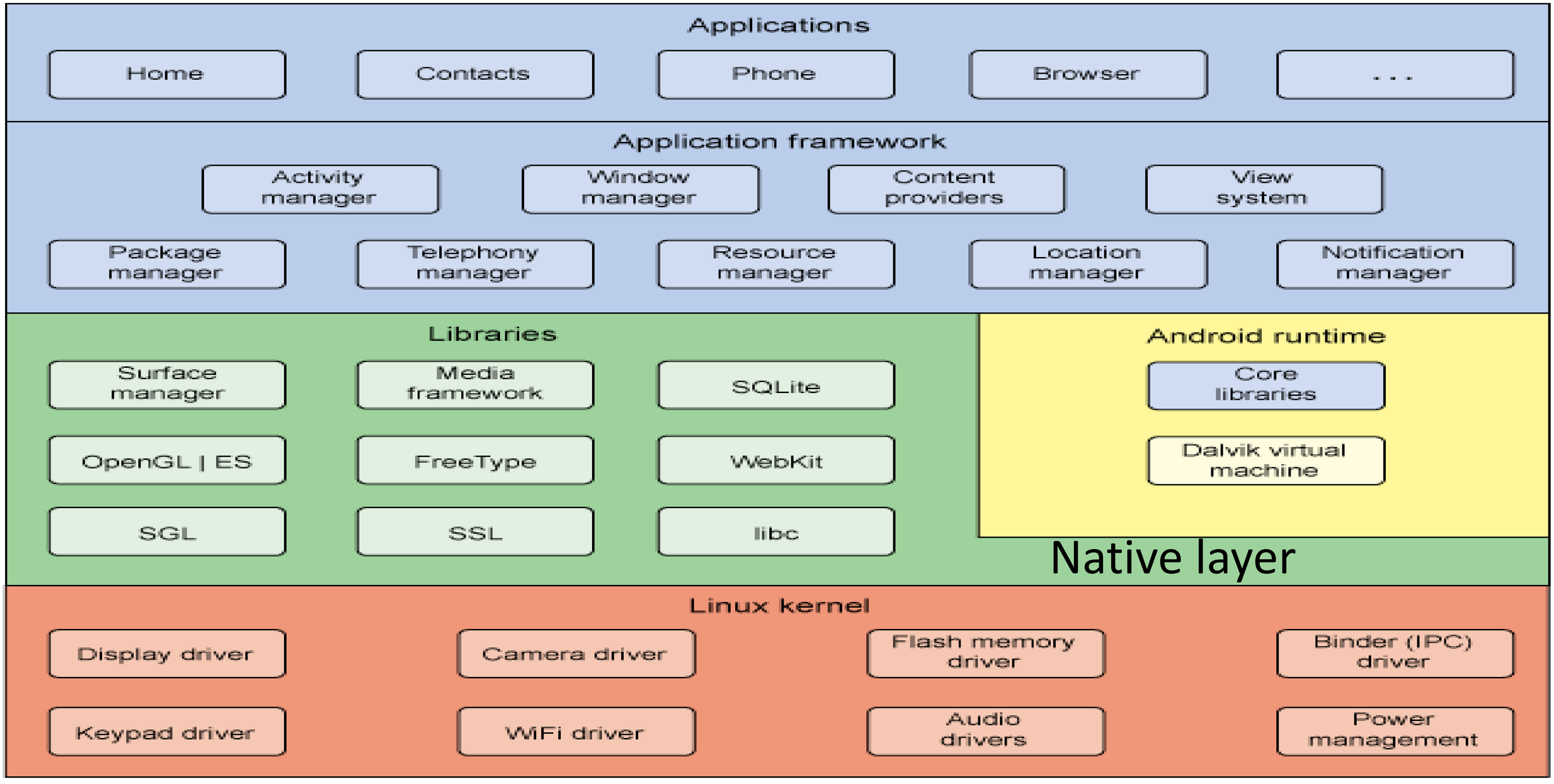
Native Applications

- These applications are developed using the platform's default solutions, developers have full and easier access to the device's capabilities; like all the device's sensors.

- Native applications has better performance since their code is closer to the 'hardware' and runs direct on OS.

- In addition to being faster, you will also have access to all of the native user interface (UI) controls and layouts.

Android Architecture



Structure of Android Studio project

app > manifests > AndroidManifest.xml

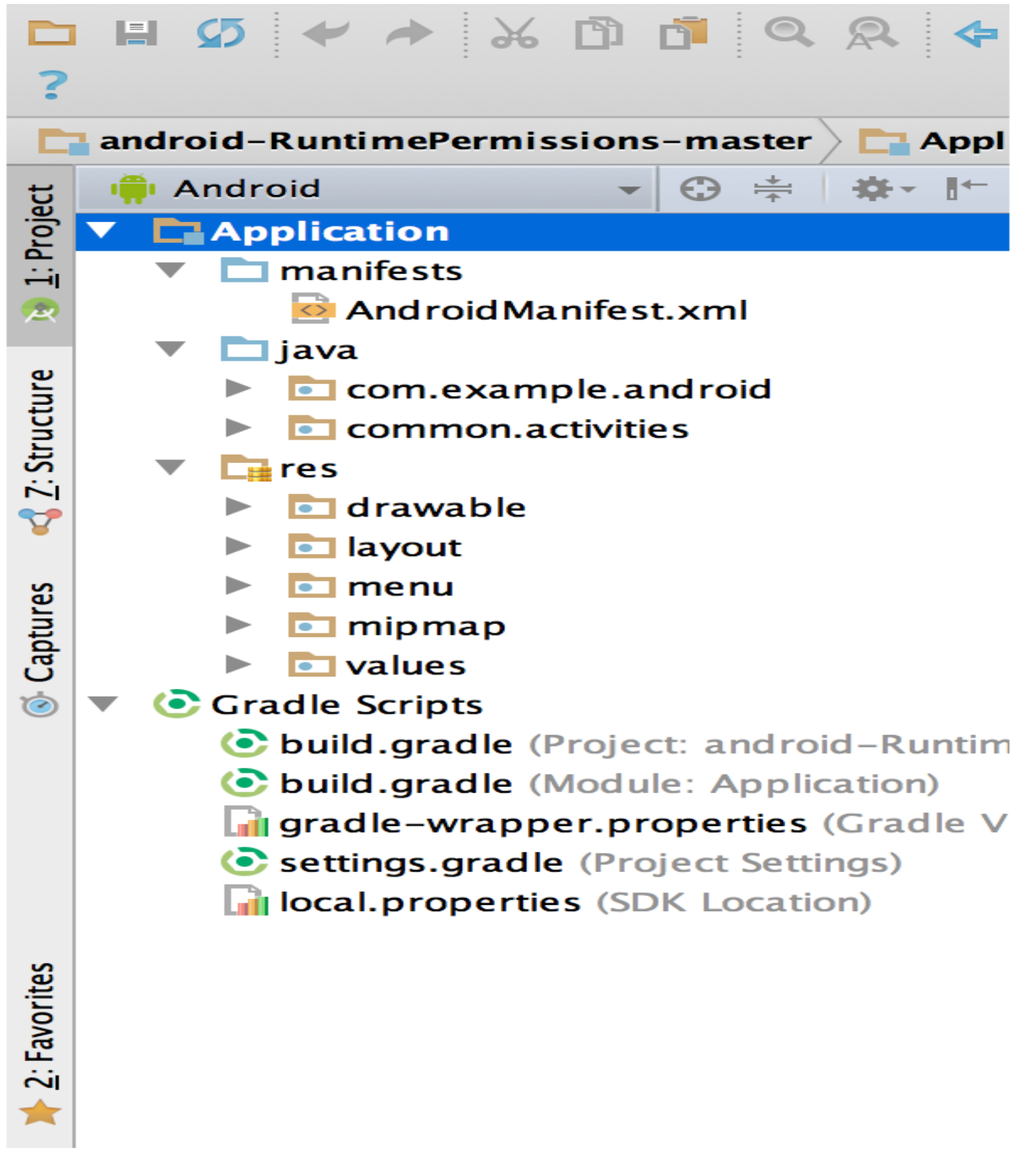
The manifest file defines each component included in the Android project (such as activities).

app > java > com.example.myfirstapp > MainActivity

This is the main activity. It's the entry point for your app. When you build and run your app, the system launches an instance of this Activity and loads its layout.

app > res > layout > activity_main.xml

This XML file defines the layout for the activity's user interface (UI).



Android studio components

1. Activities

They dictate the UI and handle the user interaction to the smart phone screen.

2. Services

They handle background processing associated with an application.

3. Broadcast Receivers

They handle communication between Android OS and applications.

4. Layout

Represent what is seen on the scree

Create new project (1)

Go to main menu

File --→ New---→ New Project

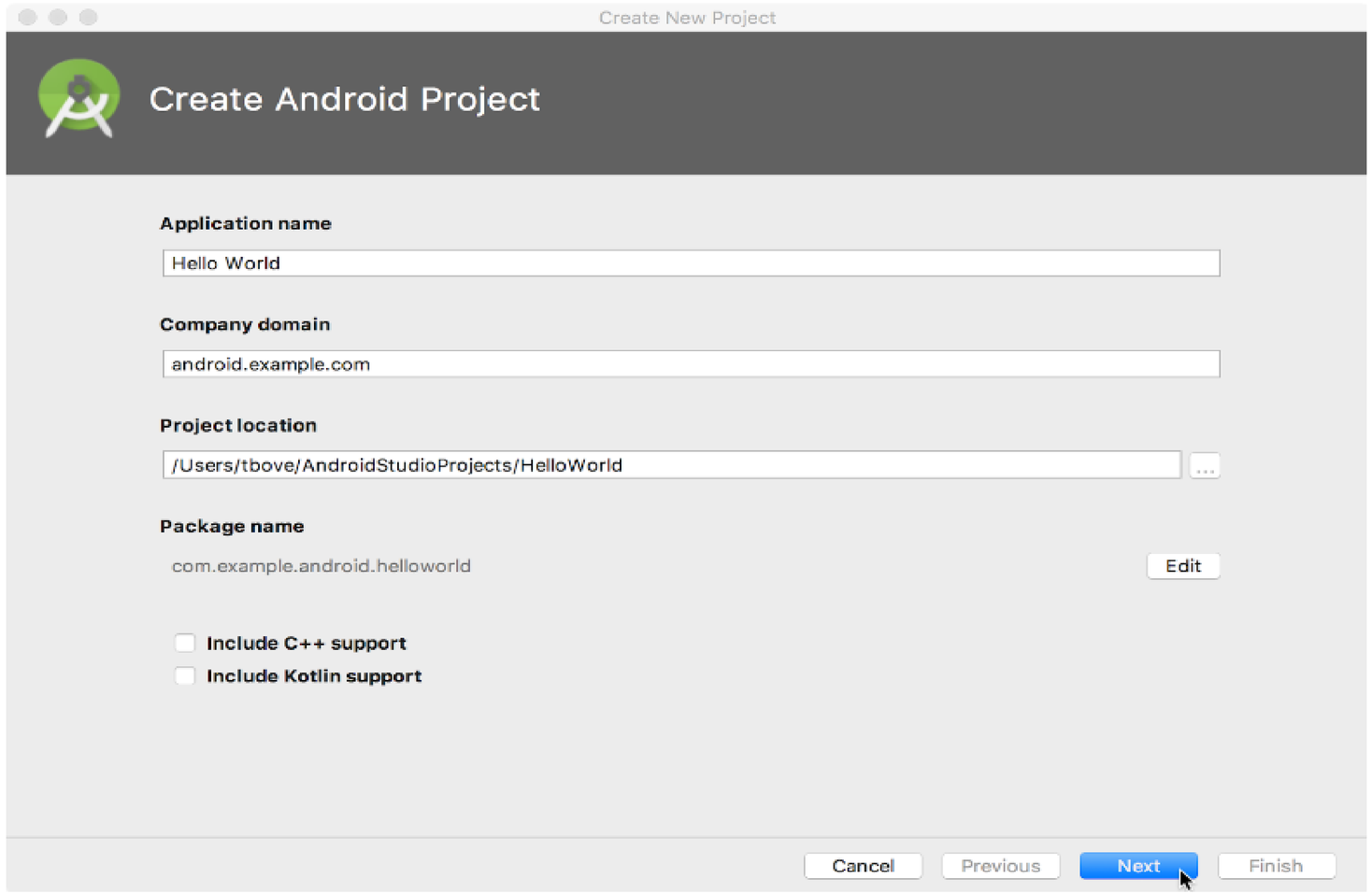
Write the application name

Leave unchecked the options to

include C++ support and **Include**

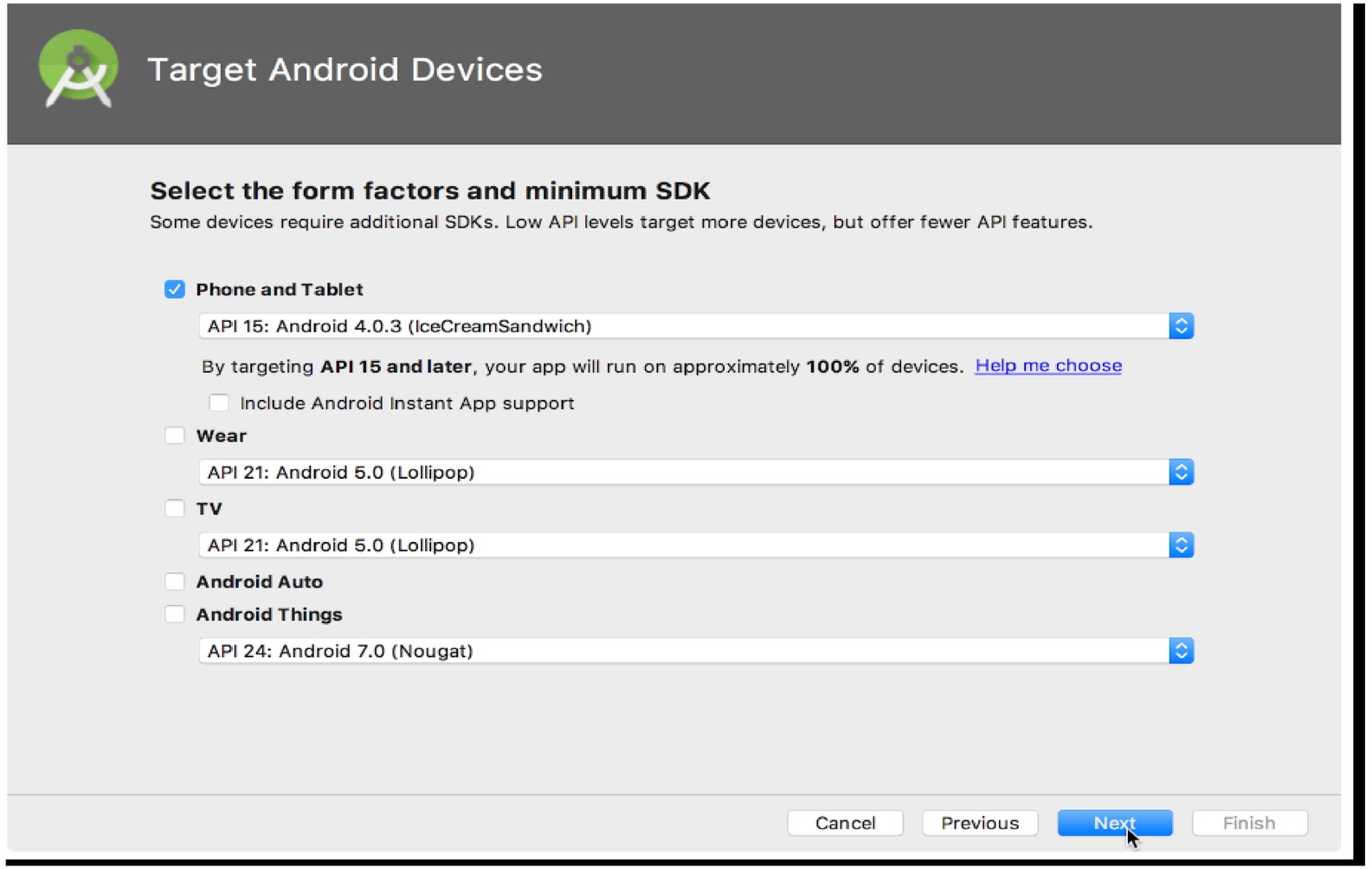
Kotlin support

Then press Next



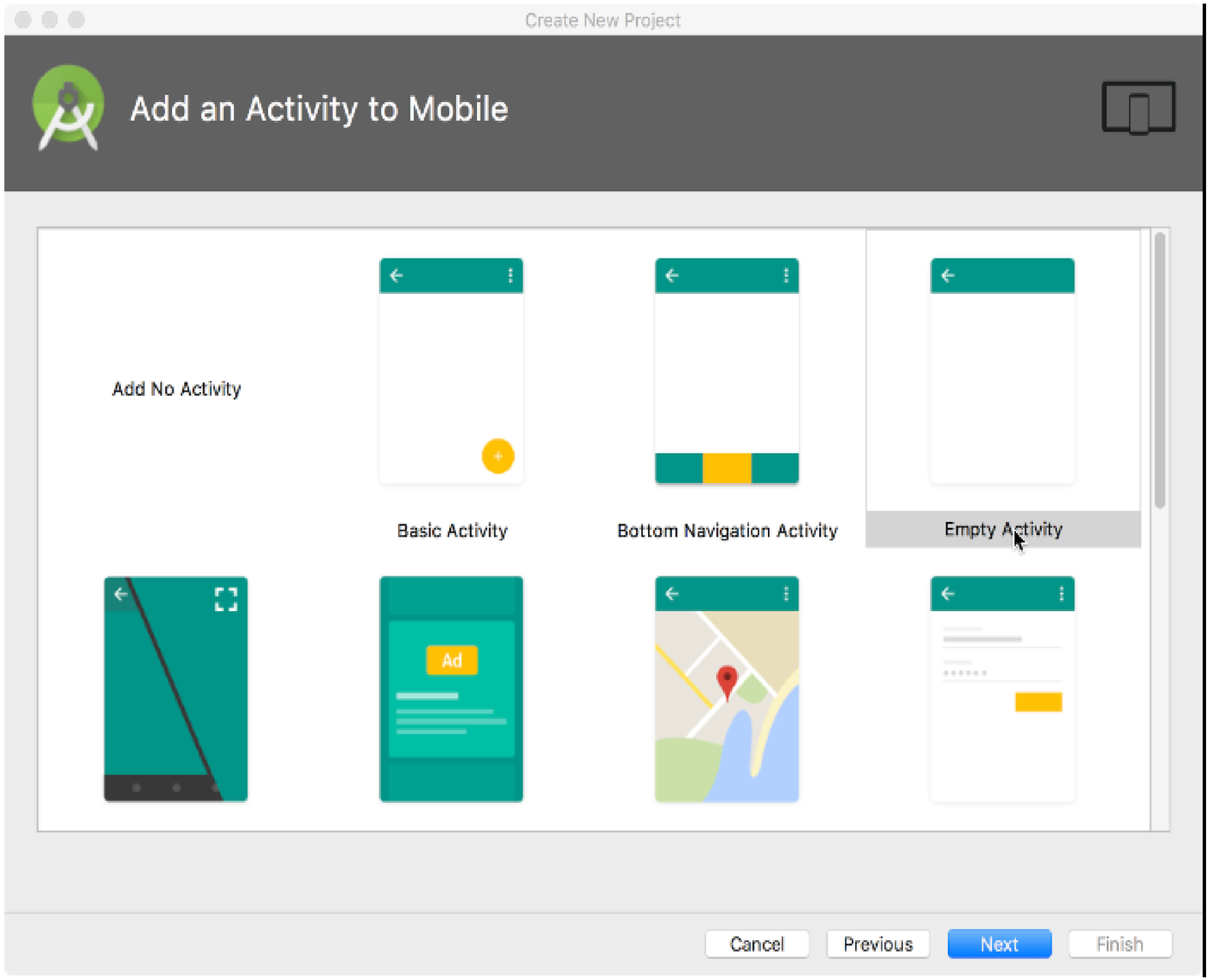
Select target device and version (2)

- On the Target Android Devices screen, Phone and Tablet should be selected.
- Set the Minimum SDK; Android 5.0.0 (for example)
- Then click next



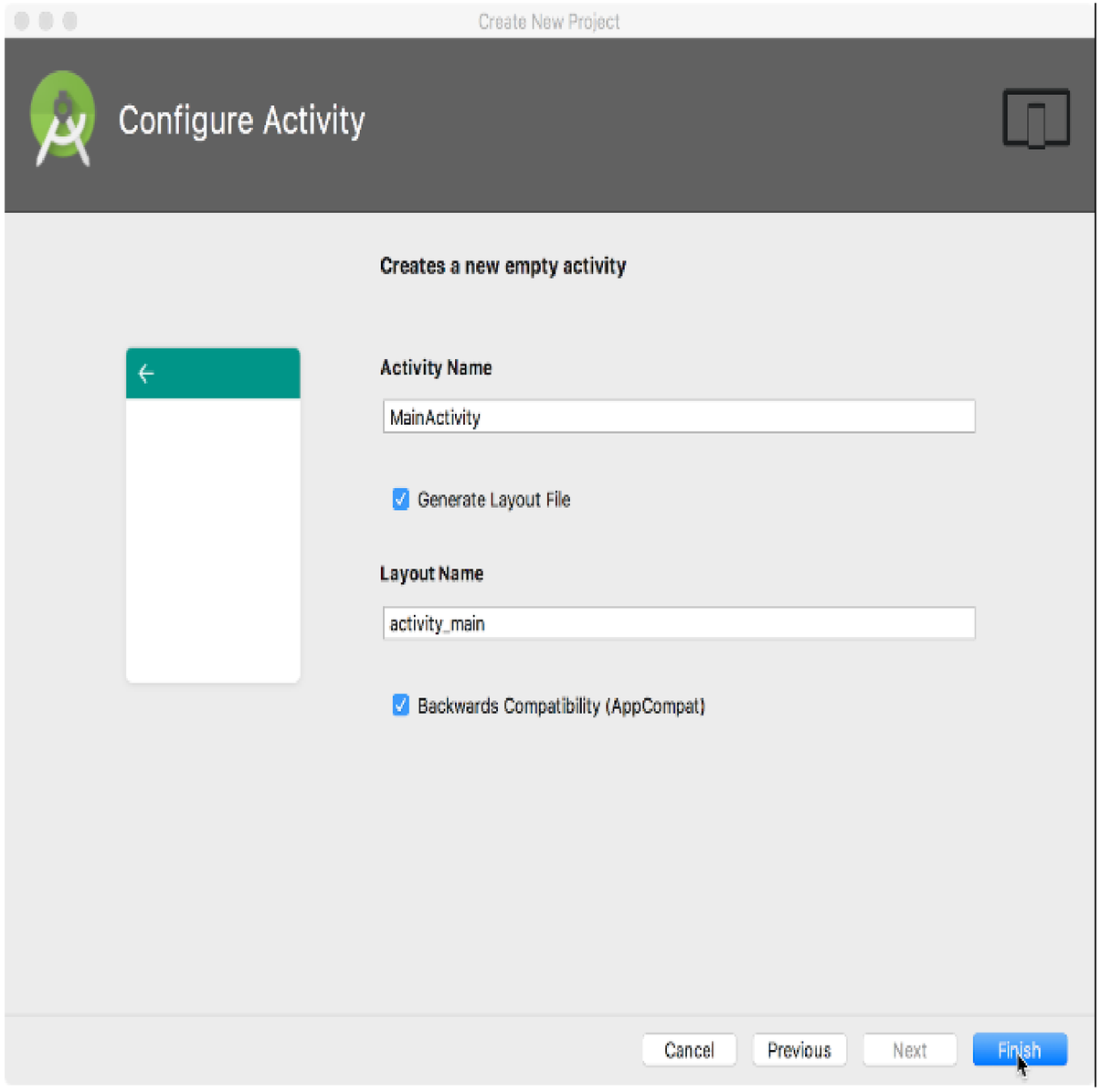
Add an activity (3)

- **An Activity is a single, focused thing that the user can do.**
- All activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with.
- An Activity typically has a layout associated with it that defines how UI elements appear on a screen.
- Generally, one activity implements one screen in an app.
- Most apps contain multiple screens, which means they comprise multiple activities.



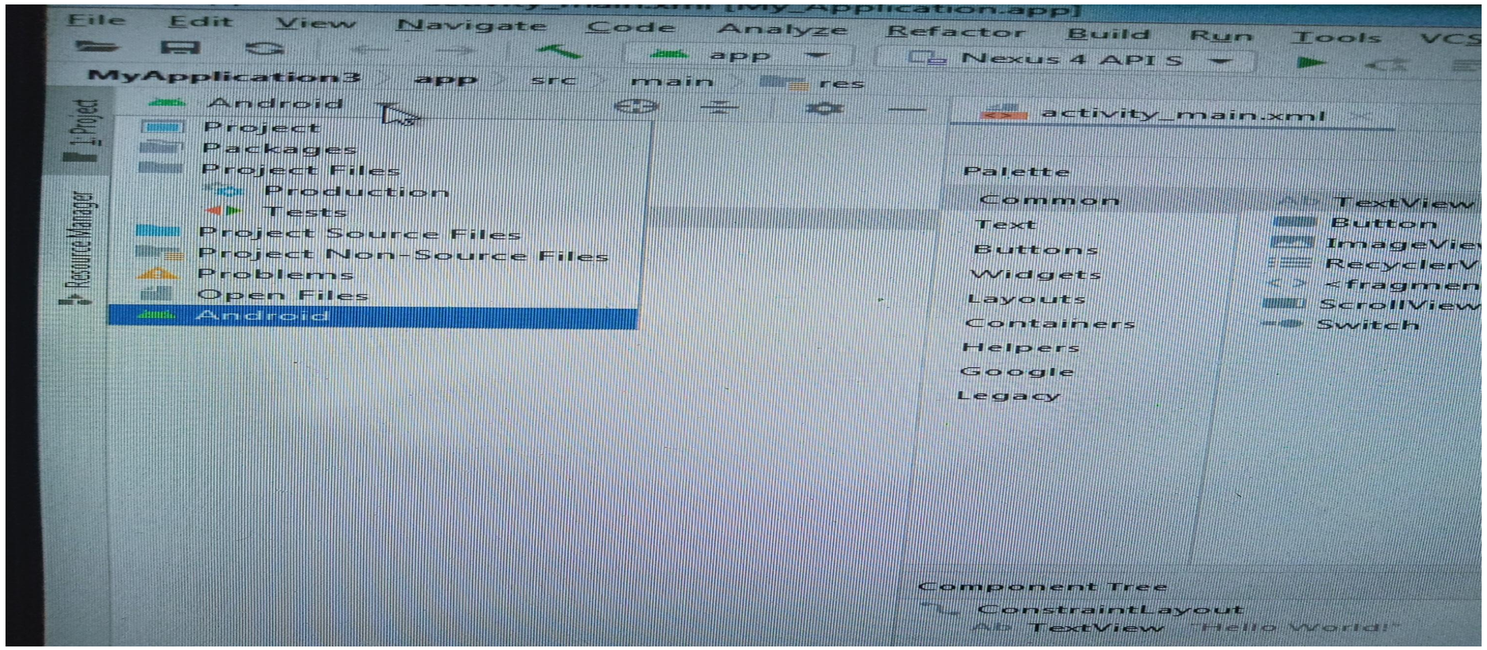
Main Activity (4)

- One activity in an app is specified as the (***MainActivity)***, which is the first screen to appear when the user launches the app.
- It is like the main method in Java which the OS starts the program by calling it.
- It is possible to invoke (go to) another activities from the Main Activity
- If another application invoked your application it will be also directed to the main activity



Project Navigation(5)

On the left side expand the menu to chose Android

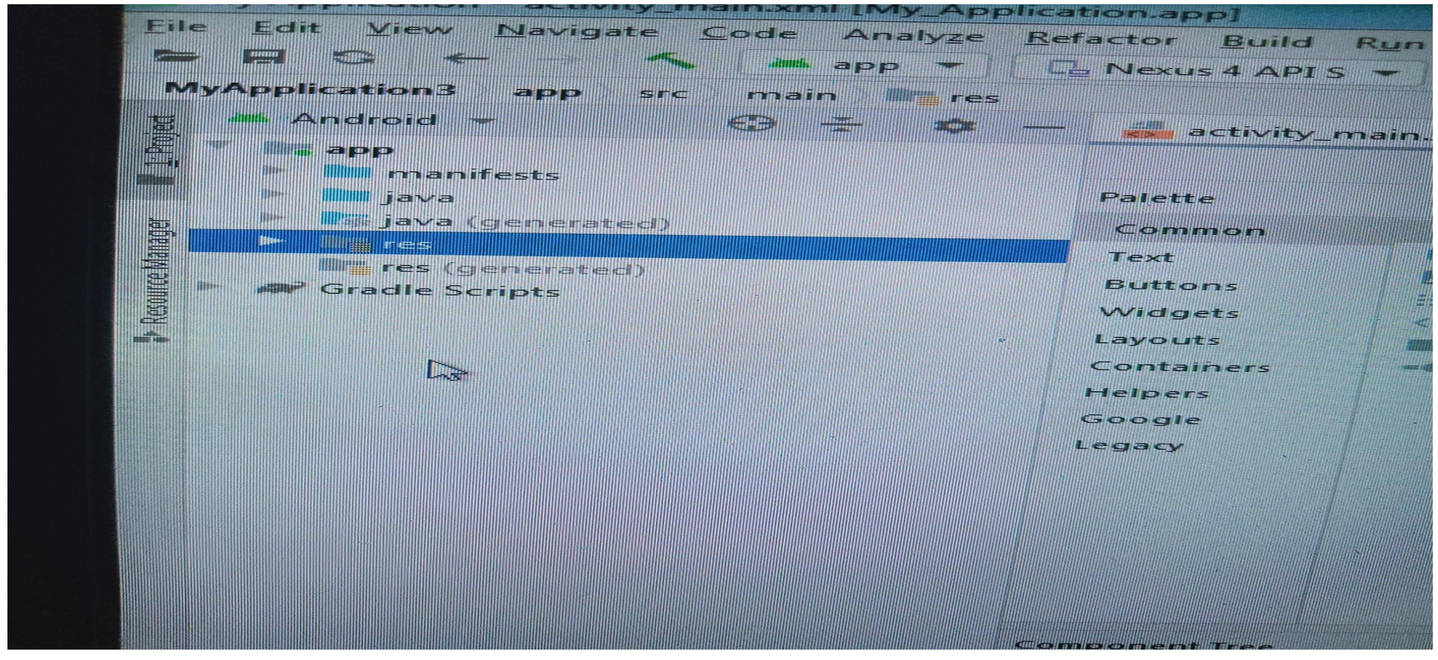


Three main parts of your project appear under app directory

manifests > AndroidManifest.xml

app > java > com.example.myfirstapp > MainActivity

app > res > layout > activity_main.xml



Structure of Android Studio project

app > manifests > AndroidManifest.xml

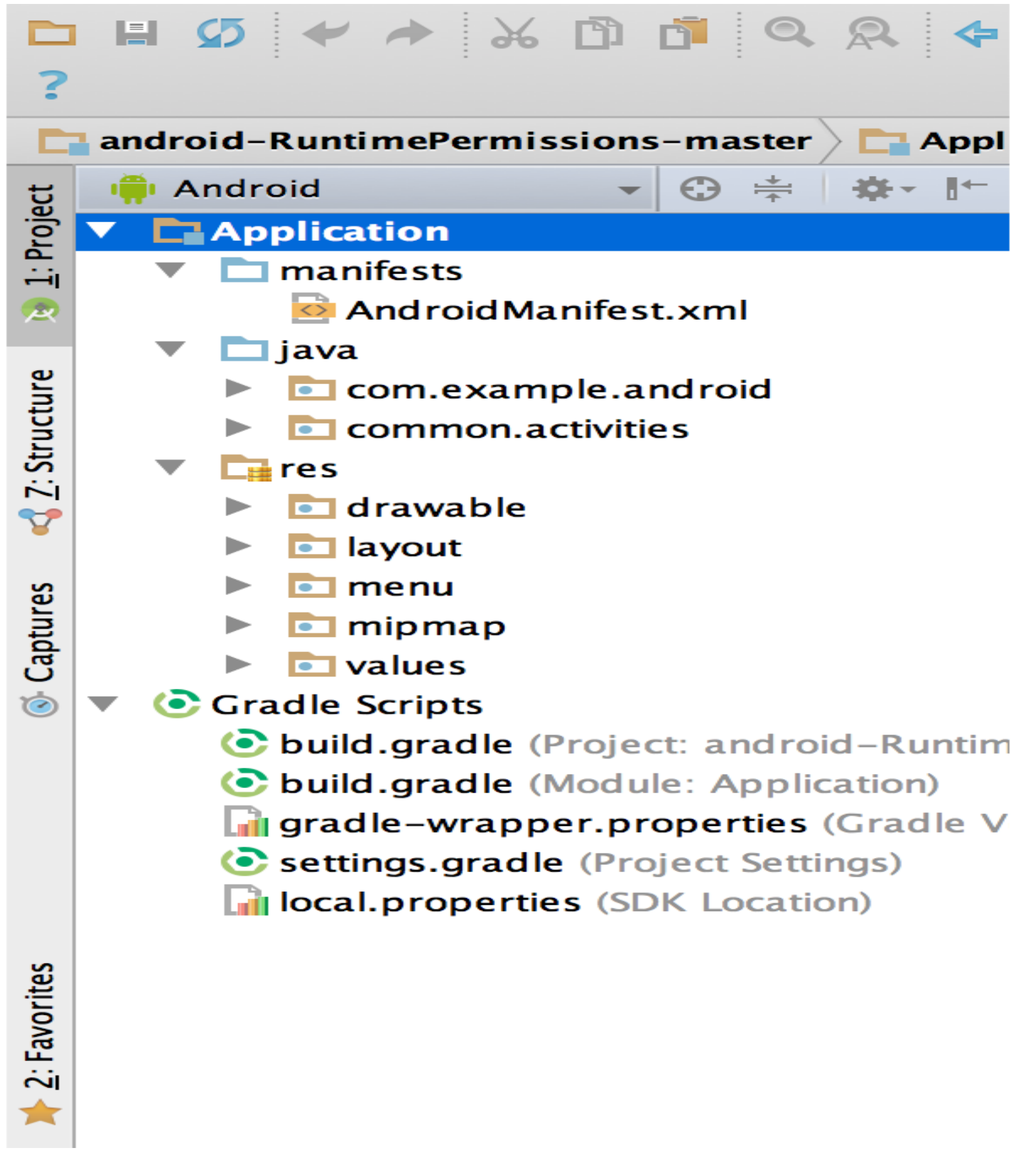
The manifest file describes the fundamental characteristics of the app and defines each of its components.

app > java > com.example.myfirstapp > MainActivity

This is the main activity. It's the entry point for your app. When you build and run your app, the system launches an instance of this Activity and loads its layout.

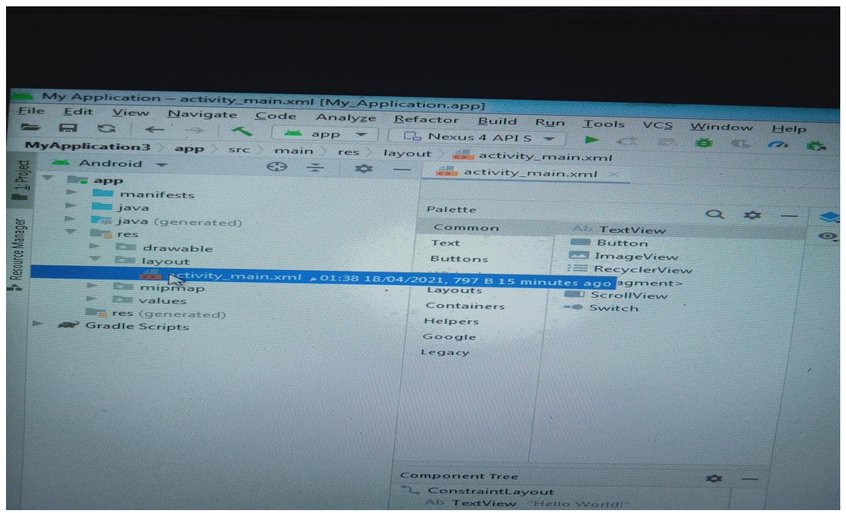
app > res > layout > activity_main.xml

This XML file defines the layout for the activity's user interface (UI). It contains a TextView element with the text "Hello, World!"

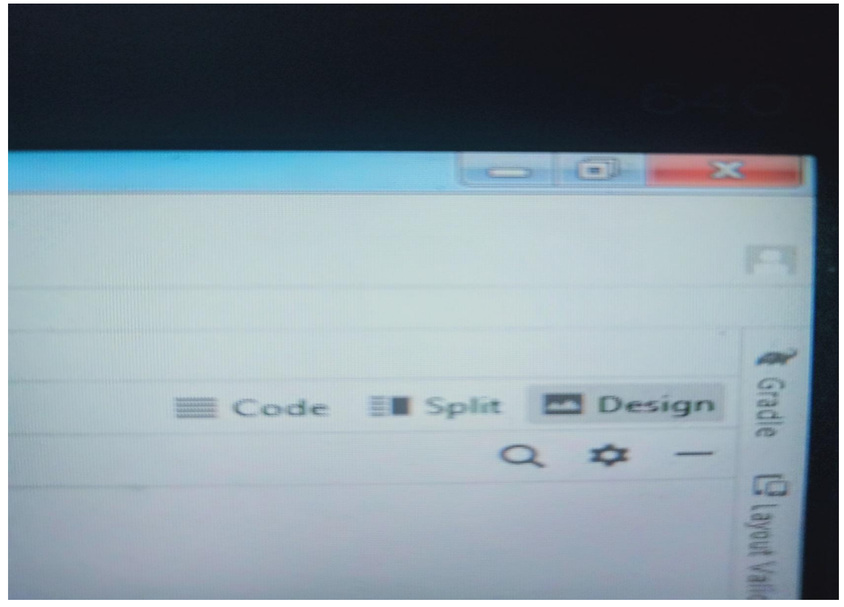


Activity layout and user interface(6)

Expand (res) then layout then click on **activity_main.xml**



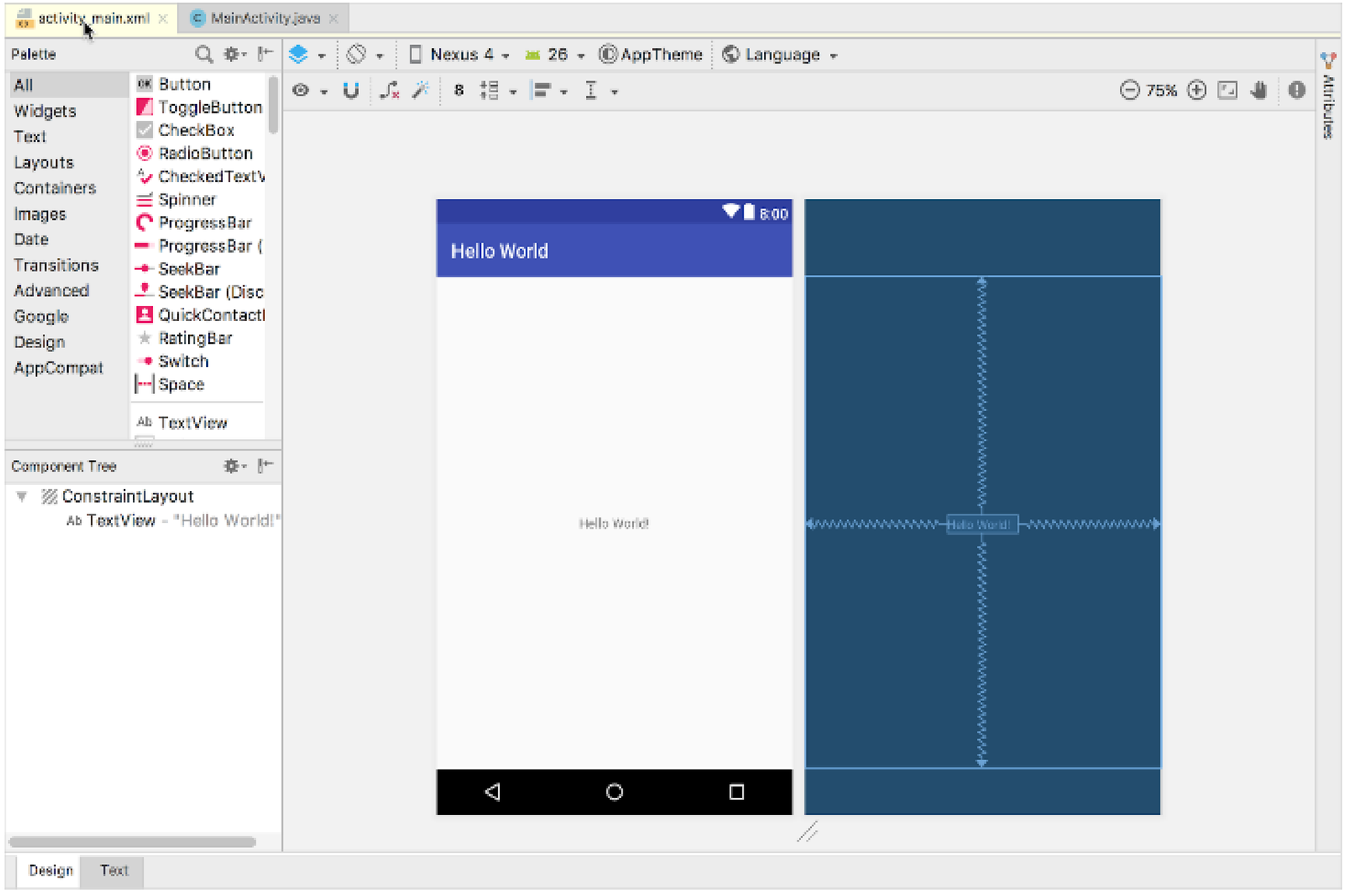
You can see the layout as XML code or as a graphical view,
3 options
Code: see xml code only
Design: see graphical interface only
Split: see both



Main Activity XML file(6)

layout editor.

- On the right the scree and on the left set of tools that allows you to drag and drop “textbox” ,”image”,….



View class

The View class represents the basic building block for all UI components

A View has a location, expressed as a pair of left and top coordinates, and two dimensions, expressed as a width and a height.

TextView: for displaying text

EditText: to enable the user to enter and edit text

Button, RadioButton, CheckBox: to provide interactive behavior

ScrollView: to display scrollable items

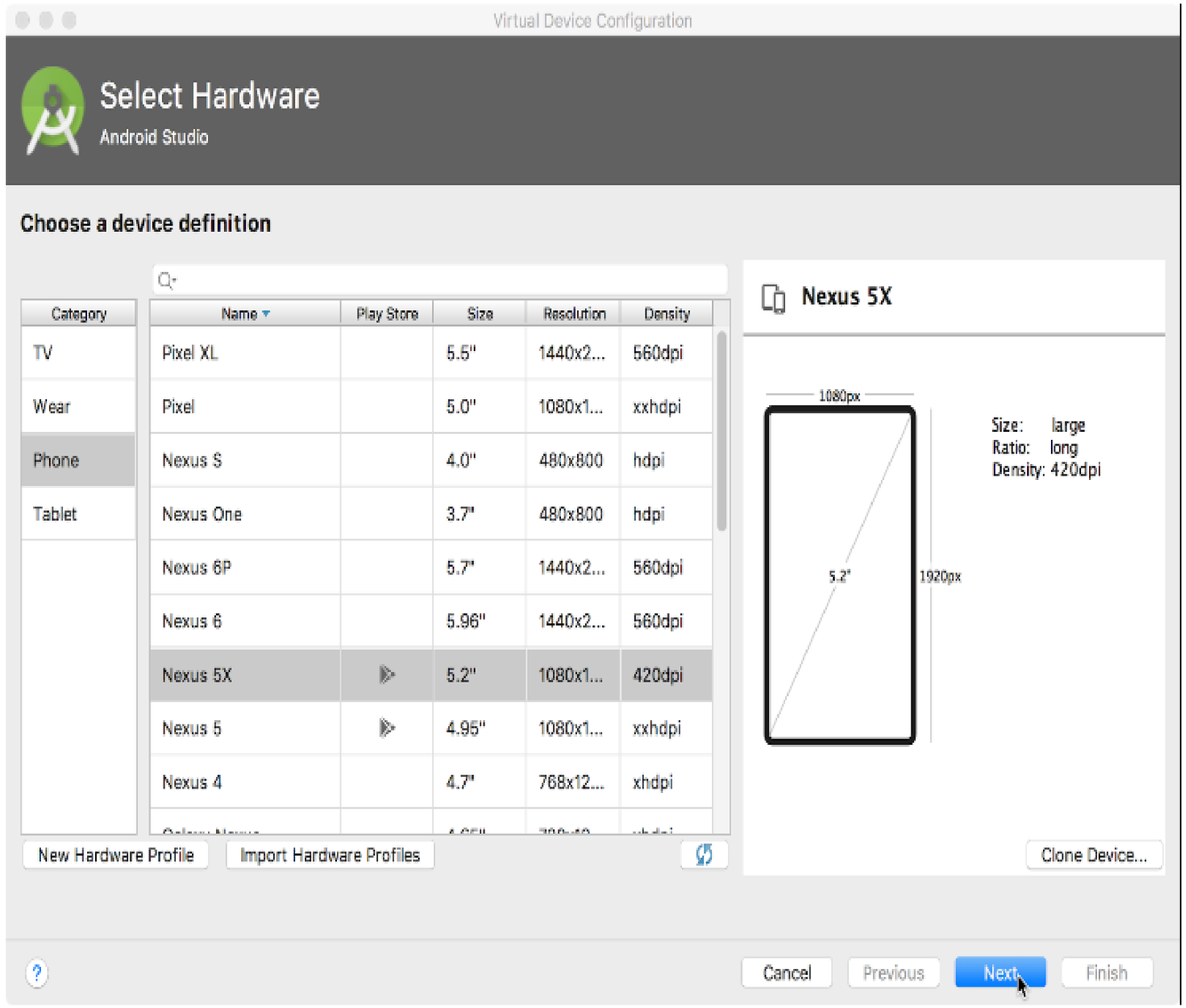
ImageView: for displaying images

Example

```
<TextView
    android:layout_width="244dp"
    android:layout_height="91dp"
    android:text="Hello SIM"
    android:textSize="30sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.758"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />


<ImageView
   android:id="@+id/imageView"
   android:layout_width="274dp"
   android:layout_height="142dp"
   tools:layout_editor_absoluteX="68dp"
   tools:layout_editor_absoluteY="488dp"
   tools:srcCompat="@tools:sample/avatars" />
```

Create an emulator (8)

- Select **Tools > Android > AVD Manager**, or click the AVD Manager icon in the toolbar.
- The **Your Virtual Devices** screen appears.
- If you've already created virtual devices, the screen shows them



Running app on virtual device

- The virtual devise will simulate the behavior of actual device.

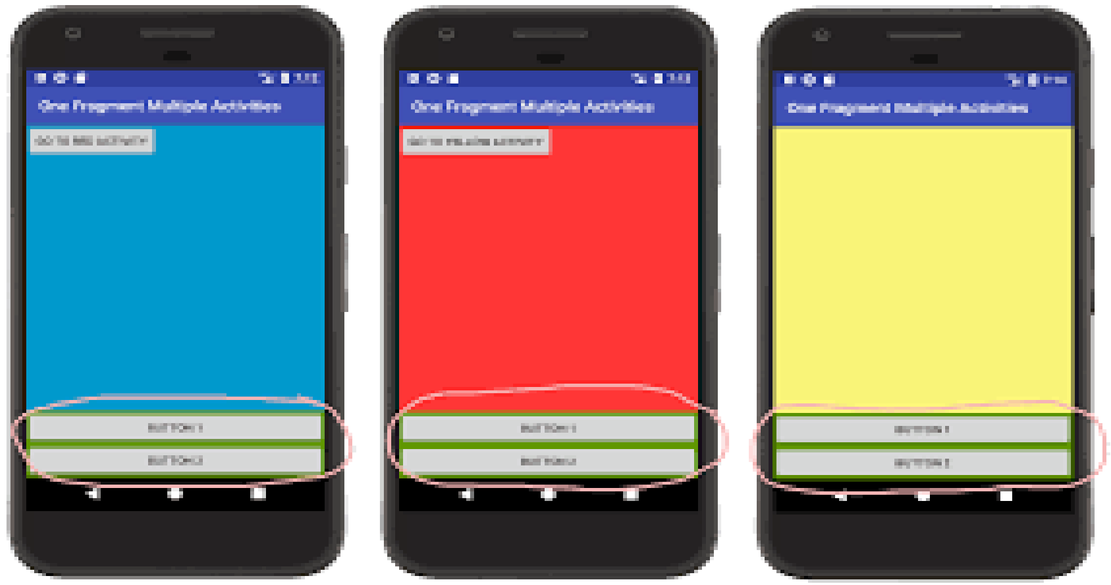
- In Android Studio, choose **Run > Run app** or click the **Run** icon in the toolbar.

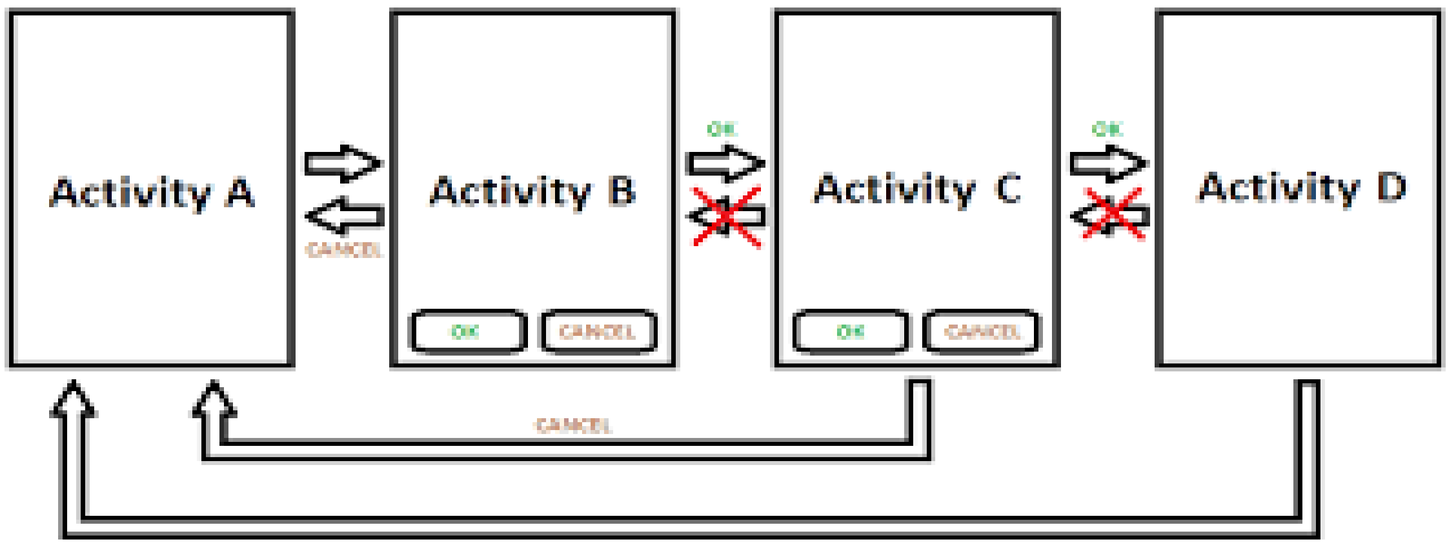
- The **Select Deployment Target** window, under **Available Virtual Devices**, select the virtual device, which you just created, and click **OK**.

Running app on physical drive

- Turn on **USB Debugging** on your Android device.
- This is enabled in the **Developer options** settings of your device.
- The **Developer options** screen is hidden by default.
- Open device **Settings** then **About phone**, and tap **Build number** seven times.
- Return to the previous screen (**Settings / System**),**developer options will** appears in the list.
- Tap **Developer options**.
- Choose **USB Debugging**.

Android application is a sets of activates





Activity in android studio

- **Activity** is a java class that creates and default window on the screen where we can place different components such as Button, EditText, TextView, Spinner etc.
- Unlike programming paradigms in which apps are launched with a main( ) method, the Android system initiates code in an Activity instance.
- The entry point of lunching (starting) application by OS is the Main Activity class, then the main activity can call another activity and so on.
-

Activity in android studio

- Each activity is represented by a screen and has its layout XML file for example if you have **two activities** in the application you will have

MainActivity.java

SecodActivity.java

activity_main.XML

activity_second.XML

The two activities should be declared in the Manifest file

MainActivity

- **public class MainActivity extends AppCompatActivity** {

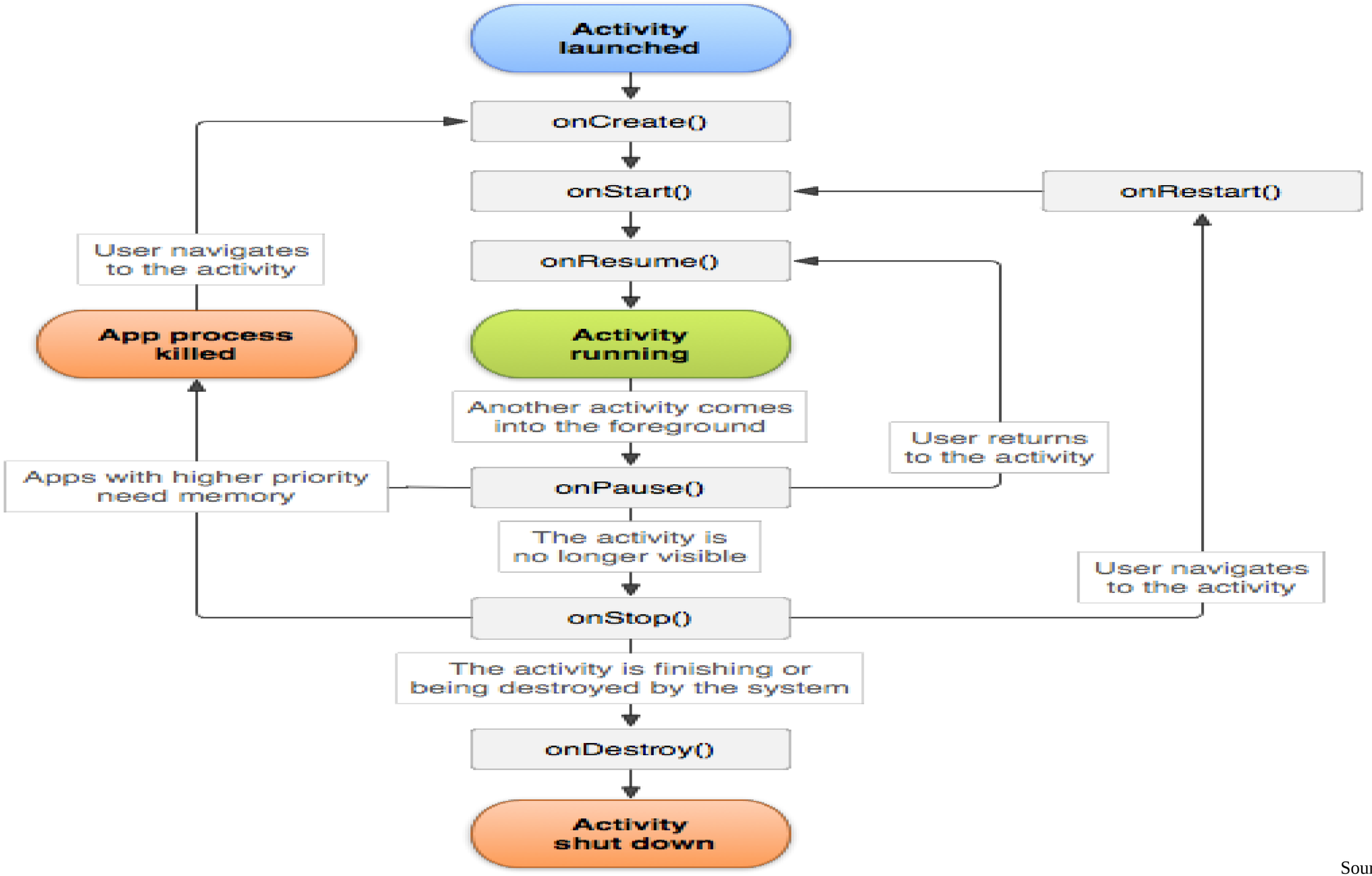
```
    @Override
    protected void  onCreate  (Bundle savedInstanceState) {
        super. onCreate (savedInstanceState);

        setContentView(R.layout.activity_main);
    }
}
```

Store information about activity layout to be loaded on recreating the activity

res layout XML file associated with UI

Activity Lifecycle

- Dalvik VM, AVR can stop any `Activity` without warning, so saving state is important!
- Activities need to be “responsive”, otherwise Android shows user “App Not Responsive”



Source: [12]

Essentially states of an activity:

1.Active and visible : If an activity is in the foreground of the screen, this is the activity that the user is currently interacting with.

2. Visible but not active : If an activity has lost focus but is still *visible*. It is possible if a new non-full-sized activity has focus. Such activity is completely alive (it maintains all state and information when user returns to it **(onResume())** is called.

3. Invisible and not active (resident in memory): If an activity is completely obscured by another activity, (navigation between application activates) when user returns to it **(onstart())** is called.

4. Invisible and not active (not resident in memory): If activity is killed by the system when memory is needed elsewhere. When user return to it **oncreate()** is called by saved instance.

5. Distorted or shut down by user (should be lunched by operating system

Activity methods onCreate() onStart()

```
public class MainActivity extends Activity {
  String msg = "Android : ";

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d (msg, "The onCreate() event");
}

/** Called when the activity is about to become visible. */
@Override
protected void onStart() {
  super.onStart();
  Log.d (msg, "The onStart() event");
}
```

Define a String variable with name msg

This code will be executed when the activity is created

This code will be executed when the activity starts

Activity methods

- **onCreate()** Called when the activity is first created. This method also provides you with a Bundle containing the activity's previously frozen state

- **onStart()** Called when the activity is becoming visible to the user.

- **onResume()** Called when the activity will start interacting with the user.

- **onPause()** Called when the activity loses foreground state, is no longer focusable

- **onStop()** Called when the activity is no longer visible to the user. This may happen either because a new activity is being started on top.

Bundle class

Bundle class is used to stored the data of activity whenever above condition occur in app.

Bundles can stores all types of values and pass them to the new activity.

Manifest file

1. **The <activity> tag** is used to specify an activity and *android:name* attribute specifies the fully class name of the *Activity* subclass

2. You can specify **multiple activities** using <activity> tags.

Each activity should be declared in the XML manifest file.

Declaration of the activity includes declaration of its intent.

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

Manifest file

3. The **action** for the intent filter is named ***android.intent.action.MAIN*** to indicate that this activity serves as the entry point for the application.

4. The **category** for the intent-filter is named *android.intent.category.LAUNCHER* to indicate that the application can be launched from the device's launcher icon

Each activity should be declared in the XML manifest file.

Declaration of the activity includes declaration of its intent.

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

List of tags that describe components in manifest XML file
Mian components of Android studio

- **<activity>** elements for activities
- **<service>** elements for services
- **<receiver>** elements for broadcast receivers
- **<provider>** elements for content providers

Main Activity

- Bundle class is used to stored the data of activity whenever above condition occur in app.

Bundle class

is used to store data about activities. The values that are to be passed are mapped to String keys which are later used in the next activity to retrieve the values.

```
public class MainActivity extends

AppCompatActivity {



    private static String TAG = "ActivityName";

public void onCreate(Bundle savedInstanceState)

{

    super.onCreate(savedInstanceState);

    setContentView(R.layout.content_main);

}
```