App Component

Activities

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen.

Broadcast Recievers

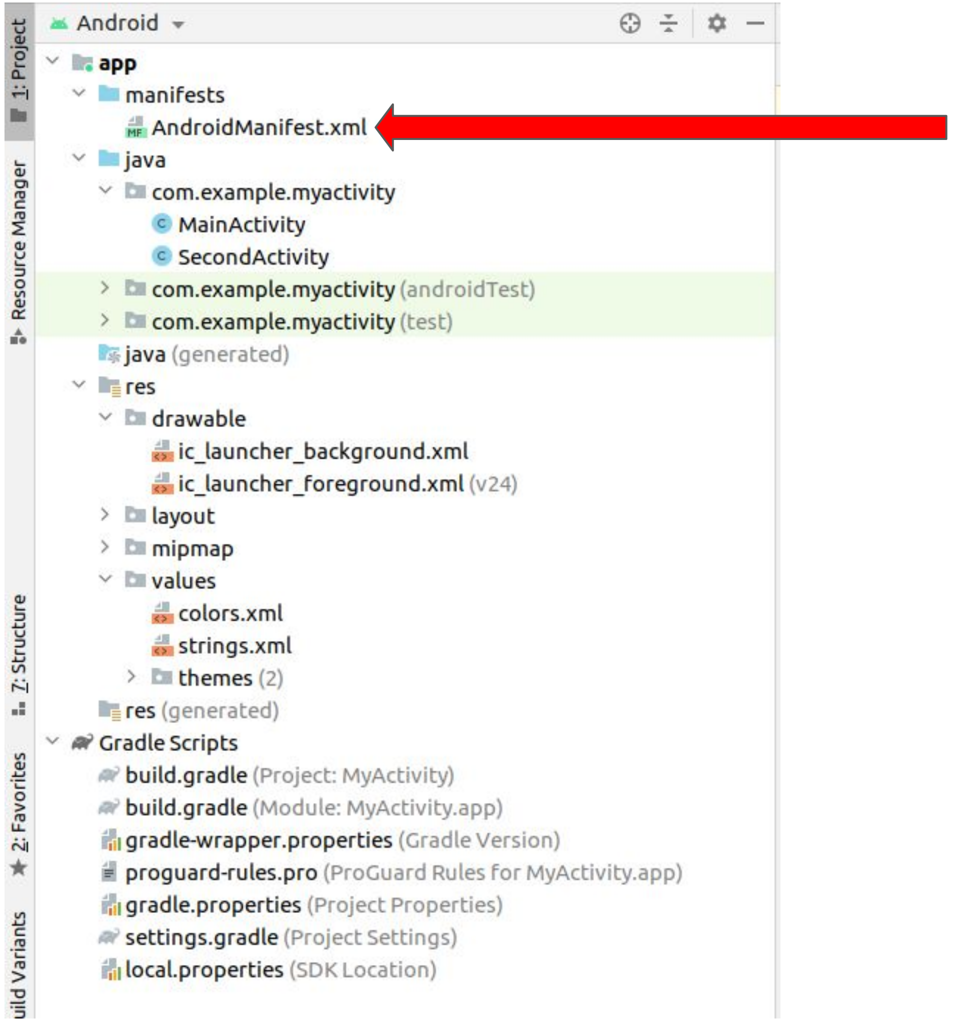
Broadcast Receivers simply respond to broadcast messages from other applications or from the system.

Services

A service is a component that runs in the background to perform long-running operations.

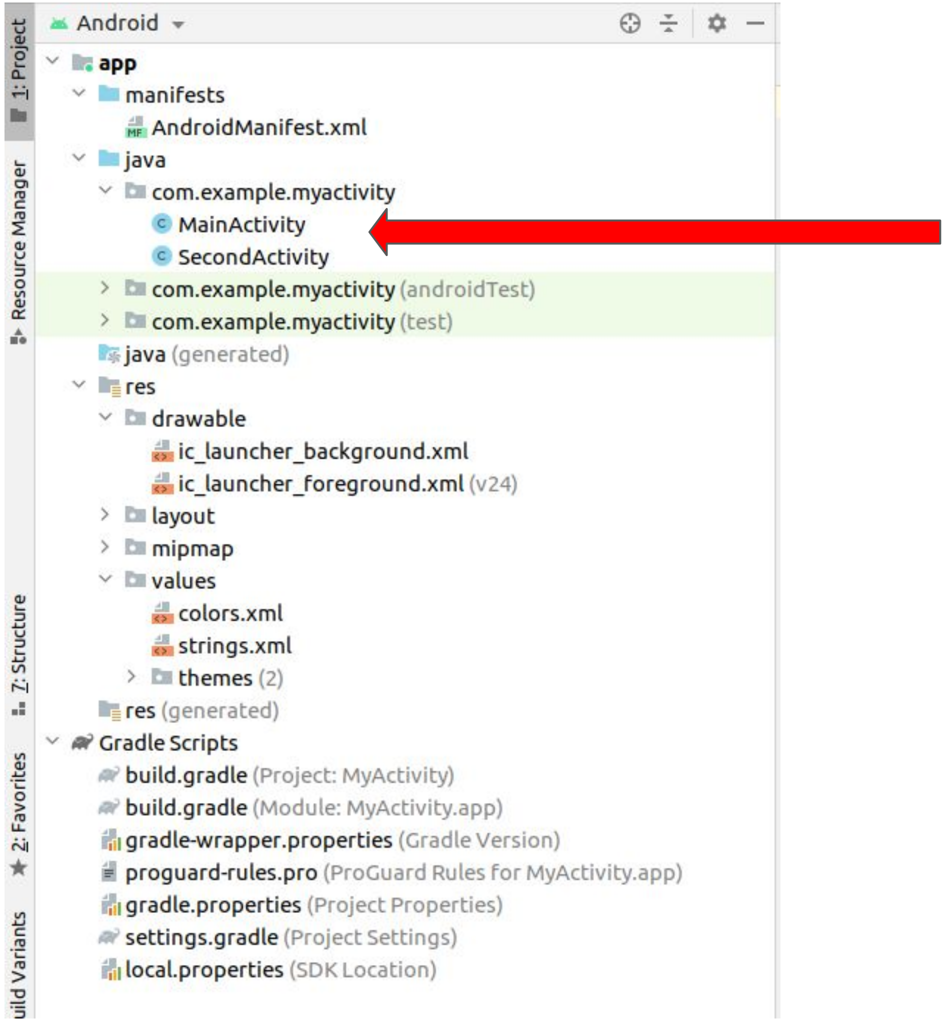
Content Providers

A content provider component supplies data from one application to others on request.



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.myactivity
      MainActivity
      SecondActivity
    com.example.myactivity (androidTest)
    com.example.myactivity (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
    mipmap
    values
      colors.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
  build.gradle (Project: MyActivity)
  build.gradle (Module: MyActivity.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for MyActivity.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Every project in Android includes a Manifest XML file, which is AndroidManifest.xml, located in the root directory of its project hierarchy. The manifest file is an important part of our app because it defines the structure and metadata of our application, its components, and its requirements. This file includes nodes for each of the Activities, Services, Content Providers, and Broadcast Receivers that make the application, and using Intent Filters and Permissions determines how they coordinate with each other and other applications.

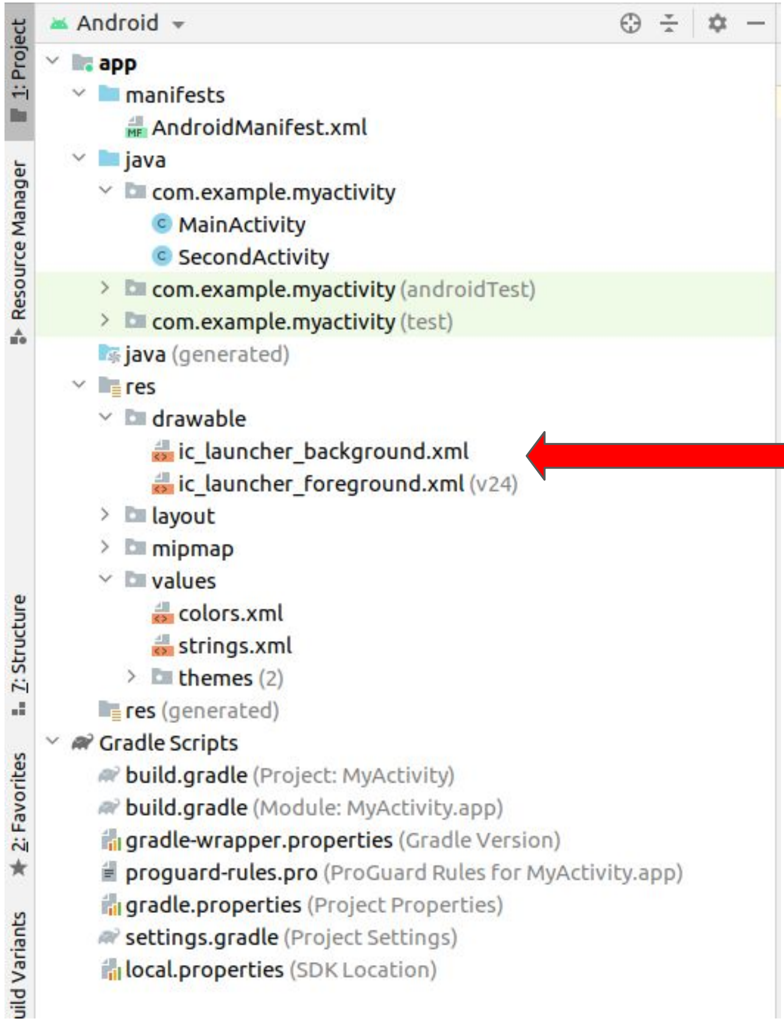


```
app
  manifests
    AndroidManifest.xml
  java
    com.example.myactivity
      MainActivity
      SecondActivity
    com.example.myactivity (androidTest)
    com.example.myactivity (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
    mipmap
    values
      colors.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
  build.gradle (Project: MyActivity)
  build.gradle (Module: MyActivity.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for MyActivity.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Here, "com.example.myactivity" is the package name of the application. Each android application is identified by its package name. The package name should be unique.

For example, package name of instagram is "com.instagram.android"
https://play.google.com/store/apps/details?id=com.instagram.android&hl=en&gl=US&pli=1

java/com.example.myactivity will contain our java code. Here, MainActivity and SecondActivity are java class files for our two activity components.

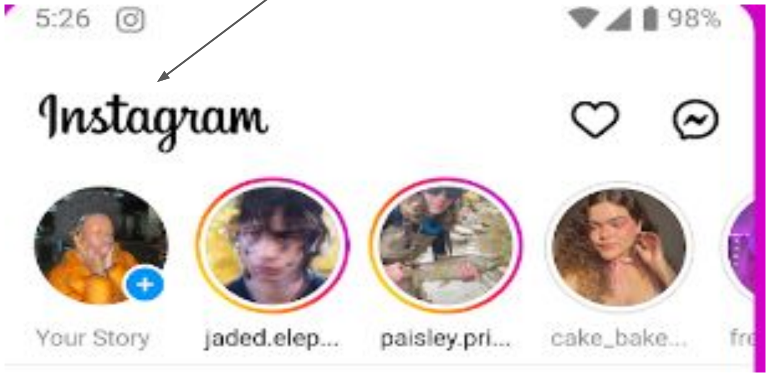


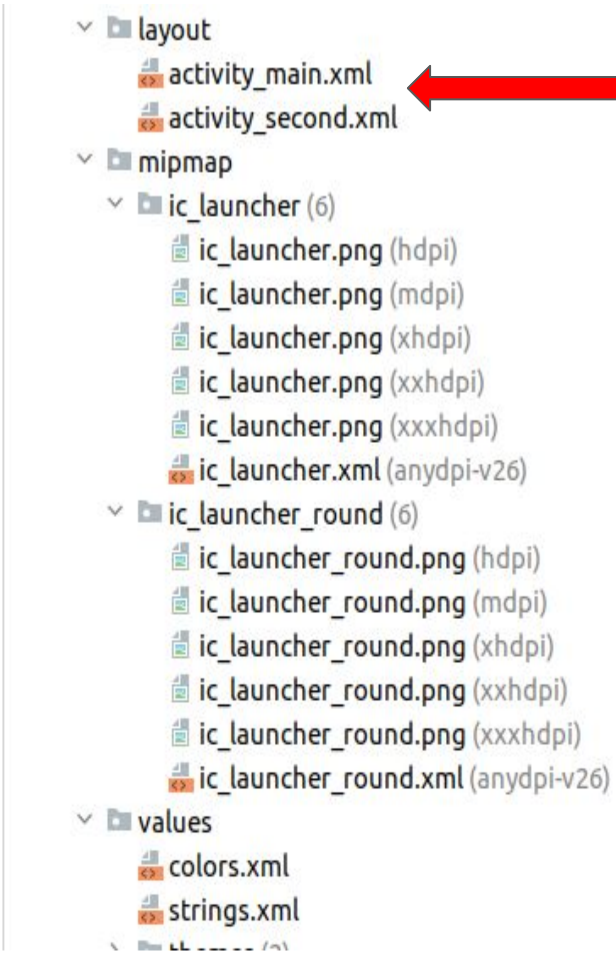
```
app
  manifests
    AndroidManifest.xml
  java
    com.example.myactivity
      MainActivity
      SecondActivity
    com.example.myactivity (androidTest)
    com.example.myactivity (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
    mipmap
    values
      colors.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
  build.gradle (Project: MyActivity)
  build.gradle (Module: MyActivity.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for MyActivity.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

The resources folder will contain the application's resources like icons, images, strings, layouts.

The res/drawable folder will contain images used by the application

Here, the "Instagram" image will be stored inside drawable folder





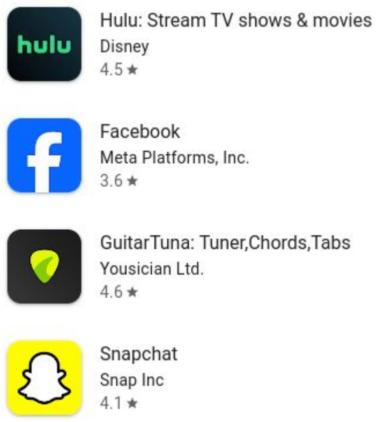
```
layout
    activity_main.xml
    activity_second.xml
mipmap
    ic_launcher (6)
        ic_launcher.png (hdpi)
        ic_launcher.png (mdpi)
        ic_launcher.png (xhdpi)
        ic_launcher.png (xxhdpi)
        ic_launcher.png (xxxhdpi)
        ic_launcher.xml (anydpi-v26)
    ic_launcher_round (6)
        ic_launcher_round.png (hdpi)
        ic_launcher_round.png (mdpi)
        ic_launcher_round.png (xhdpi)
        ic_launcher_round.png (xxhdpi)
        ic_launcher_round.png (xxxhdpi)
        ic_launcher_round.xml (anydpi-v26)
values
    colors.xml
    strings.xml
```

res/layout folder contains the different views that are visible to uses.

Consider this to be the equivalent of html in web apps.

res/mipmap contains the application icons in different format and sizes.

These are app icons

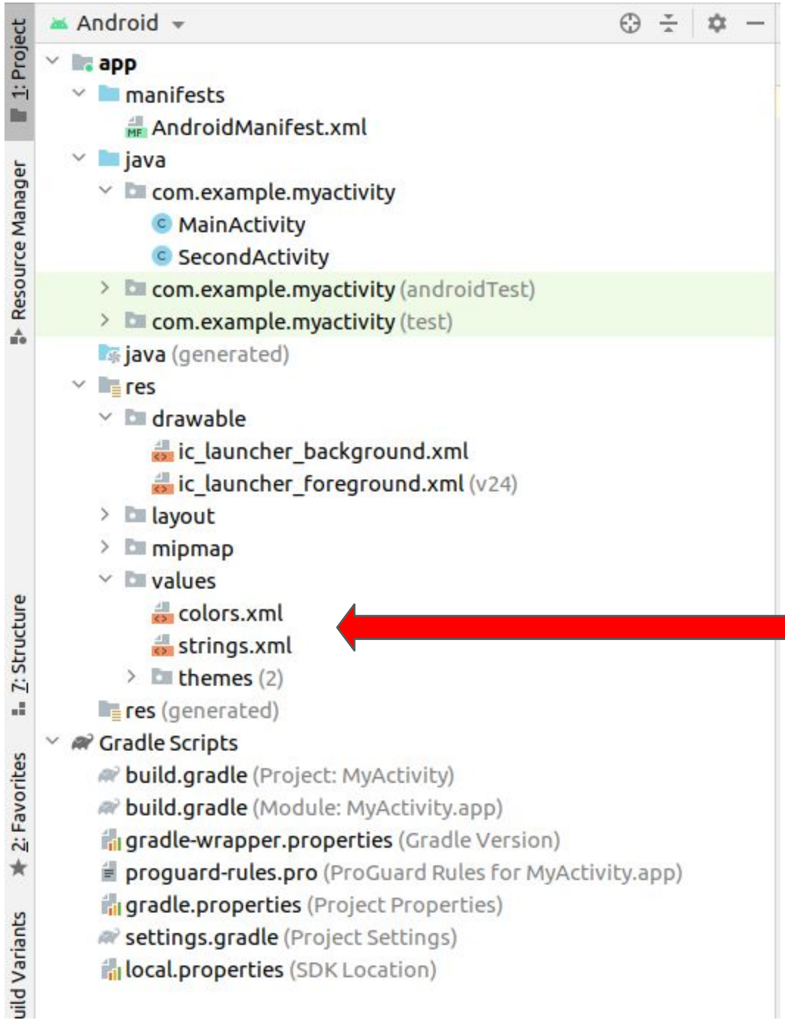


Hulu: Stream TV shows & movies
Disney
4.5 ★

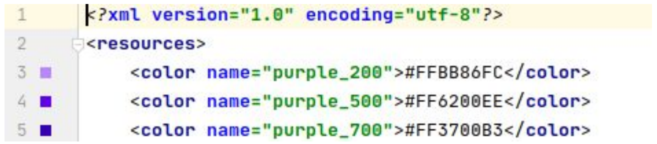
Facebook
Meta Platforms, Inc.
3.6 ★

GuitarTuna: Tuner,Chords,Tabs
Yousician Ltd.
4.6 ★

Snapchat
Snap Inc
4.1 ★



```
Android
app
  manifests
    AndroidManifest.xml
  java
    com.example.myactivity
      MainActivity
      SecondActivity
    com.example.myactivity (androidTest)
    com.example.myactivity (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
    mipmap
    values
      colors.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
  build.gradle (Project: MyActivity)
  build.gradle (Module: MyActivity.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for MyActivity.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

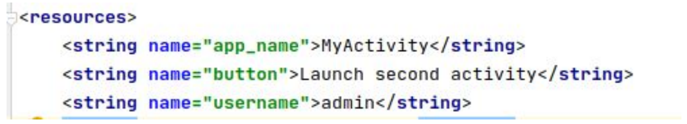


```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
```

Colors.xml

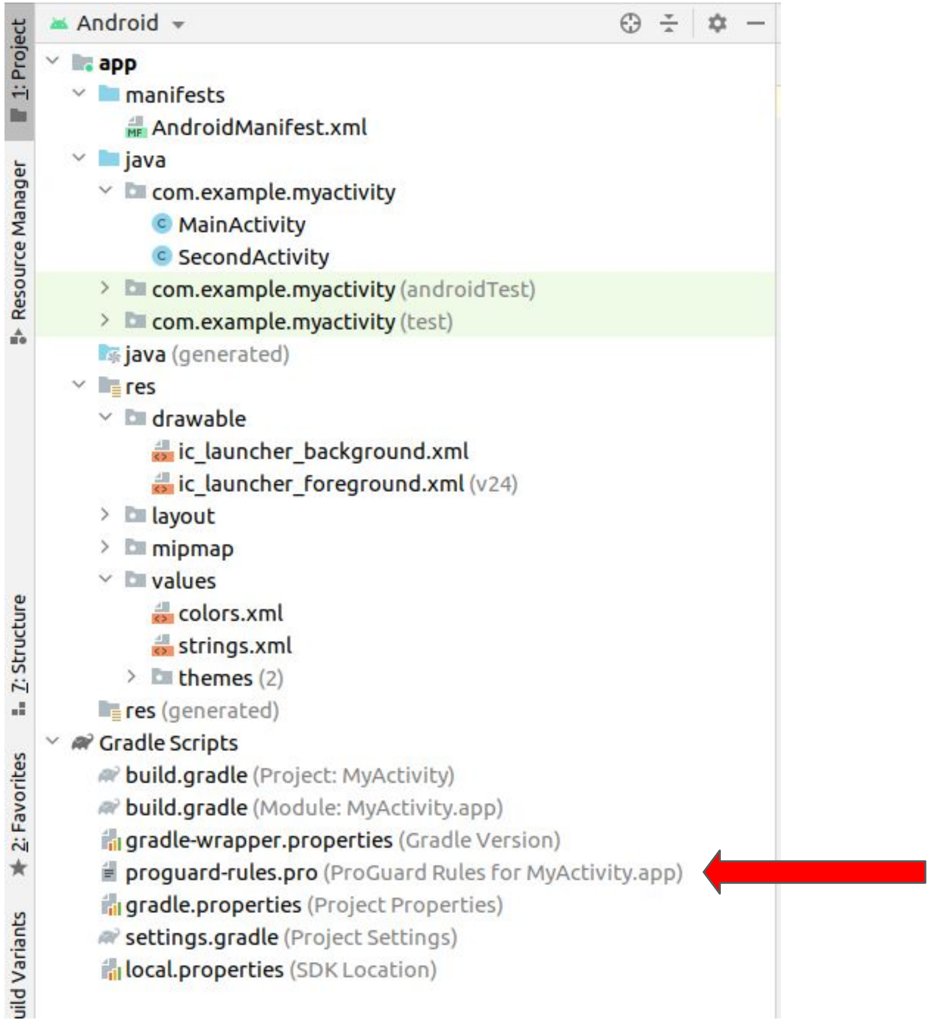
All the colors (red, grey, white) used by the apps are defined inside res/values/colors.xml.

All strings values used by application are defined in res/values/strings.xml



```xml
<resources>
    <string name="app_name">MyActivity</string>
    <string name="button">Launch second activity</string>
    <string name="username">admin</string>
```

Strings.xml



```
Android
app
  manifests
    AndroidManifest.xml
  java
    com.example.myactivity
      MainActivity
      SecondActivity
    com.example.myactivity (androidTest)
    com.example.myactivity (test)
  java (generated)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
    mipmap
    values
      colors.xml
      strings.xml
      themes (2)
  res (generated)
Gradle Scripts
  build.gradle (Project: MyActivity)
  build.gradle (Module: MyActivity.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for MyActivity.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

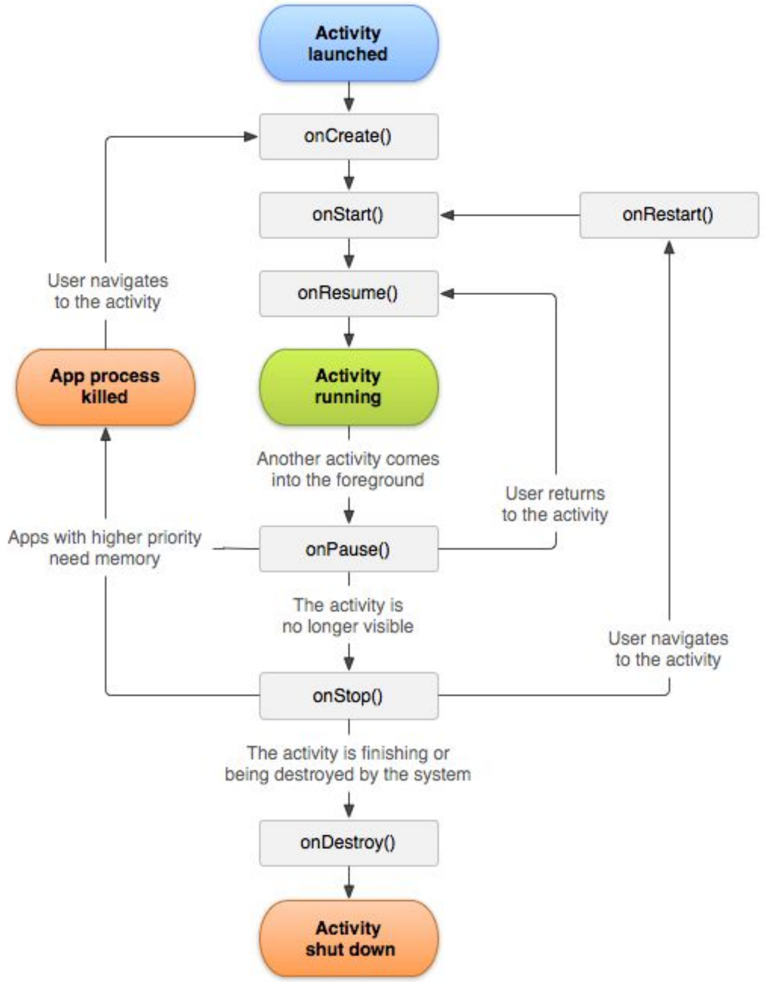
proguard-rules.pro is a config file for proguard.

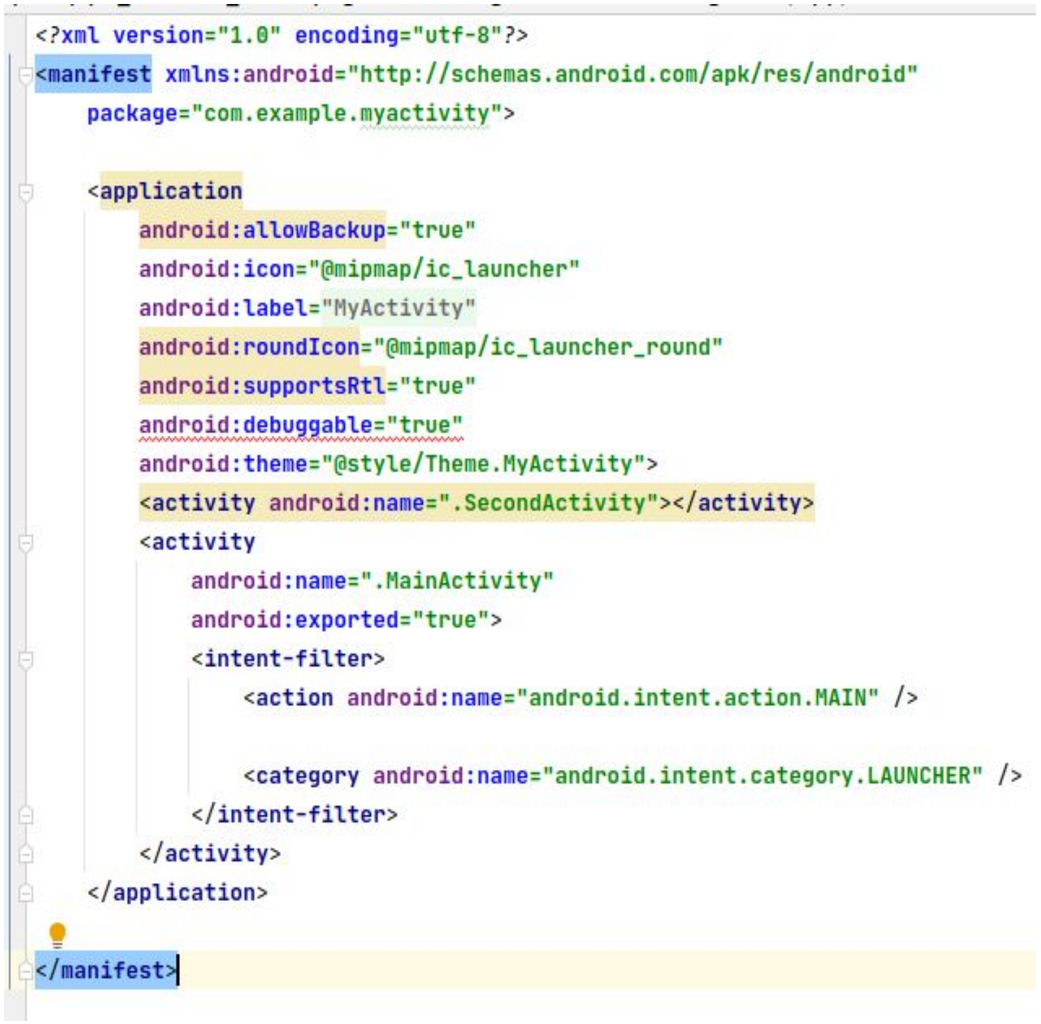
Proguard is a great tool for creating a production-ready application in Android.
It obfuscates the code, which means that it changes the names to something smaller, such as A for MainViewModel. After obfuscating the app, reverse engineering becomes difficult.
It shrinks the resources, ignoring resources that are not called by our Class files, are not used in our Android app, such as images from drawable, and so on. This will significantly reduce the app's size. To keep your app light and fast, you should always shrink it.

Activities

Activities are said to be the presentation layer of our applications. The UI of our application is built around one or more extensions of the Activity class. By using Fragments and Views, activities set the layout and display the output and also respond to the user’s actions. An activity is implemented as a subclass of class Activity.

Example from: https://www.geeksforgeeks.org/introduction-to-activities-in-android/


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down



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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MyActivity"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:debuggable="true"
        android:theme="@style/Theme.MyActivity">
        <activity android:name=".SecondActivity"></activity>
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

<application> is the root level tag. Attributes of <application> tag define what the behaviour of application will be.

Here,
android:icon and **android:roundIcon** define the application's icon.

android:allowBackup - When an app is being backed up, should the data stored inside application directory be backed up or not. Arguable if this is a security risk or not. In an dystopian case, consider that sensitive data is stored in application directory and that gets uploaded on some untrusted servers. (Eg. whatsapp chats on one's device being uploaded to untrusted server during backup)

android:debuggable - defines if the application can be debugged using a debugger or not. Again needs to be turned of for production apps.

Other components (activities/services/receivers/providers) needed by the application also needs to be defined in manifest file. Here we have defined two activity components we are using.

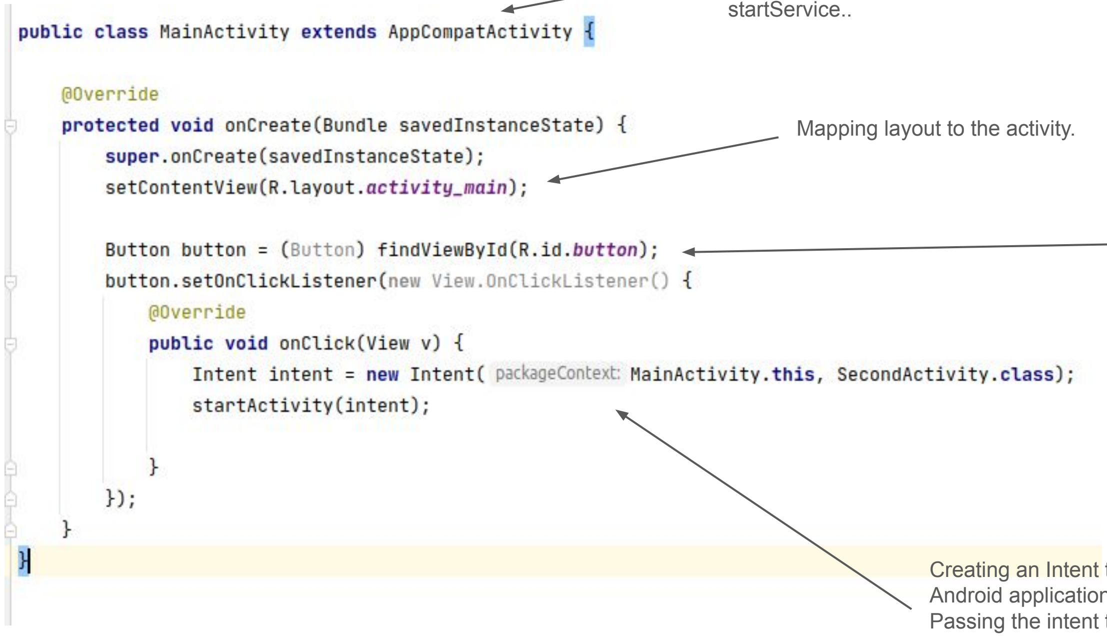
android:exported - The android:exported attribute sets whether a component (activity, service, broadcast receiver, etc.) can be launched by components of other applications:

- If true, any app can access the activity and launch it by its exact class name.
- If false, only components of the same application, applications with the same user ID, or privileged system components can launch the activity.
-

This setting needs to be properly considered, if set inappropriately Other apps will have access to internal components to modify your app's internal functionality. Leaking of sensitive data. Code execution in the context of the vulnerable application.

Intents

It is a powerful inter-application message-passing framework. They are extensively used throughout Android. Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.



```java
public class MainActivity extends AppCompatActivity {

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

        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent( packageContext: MainActivity.this, SecondActivity.class);
                startActivity(intent);

            }
        });
    }
}
```

Contains definitions for activity lifecycle functions like onCreate, onStart etc. and other utility functions like startActivity, startService..

Mapping layout to the activity.

Create a button variable. Mapping the variable to the button element on layout. And then writing a onClick listener for it. The onClick listener defines what to do when user clicks the button.

Similar to what we do in HTML..

<button id=”test1”>MyButton</button>

let va1 = Document.getElementById(“test1”).value;

Creating an Intent to navigate from the MainActivity to the SecondActivity in an Android application.
Passing the intent to startActivity function which is a method in Android used to begin the execution of an activity specified by the provided Intent.

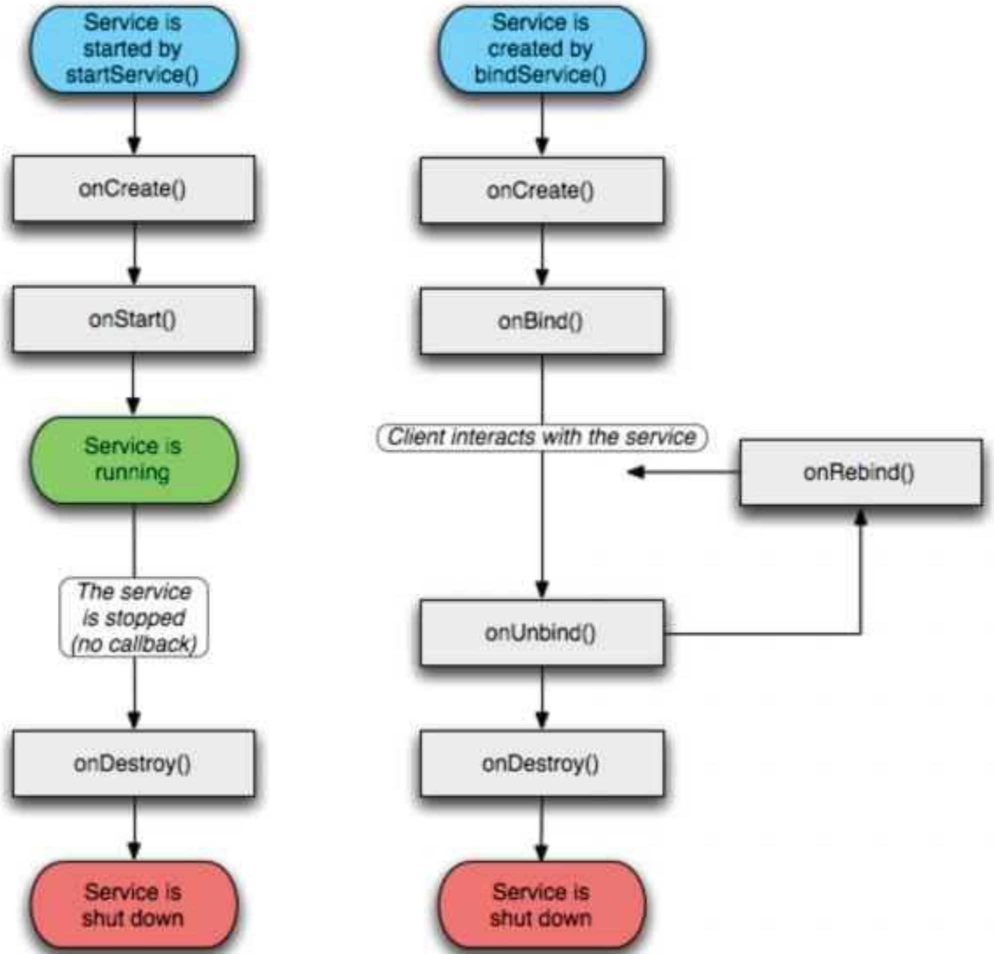
Services

Services are like invisible workers of our app. These components run at the backend, updating your data sources and Activities, triggering Notification, and also broadcast Intents. They also perform some tasks when applications are not active. A service can be used as a subclass of class Service.

List running services:

```
ps -A | grep -i myservice
```

`Example from : https://www.geeksforgeeks.org/services-in-android-with-example/`


Service is started by startService()
onCreate()
onStart()
Service is running
The service is stopped (no callback)
onDestroy()
Service is shut down
Service is created by bindService()
onCreate()
onBind()
Client interacts with the service
onRebind()
onUnbind()
onDestroy()
Service is shut down

Content Providers

Content Providers are responsible for sharing the data beyond the application boundaries. The Content Providers of a particular application can be configured to allow access from other applications, and the Content Providers exposed by other applications can also be configured.

A content provider should be a sub-class of the class ContentProvider

Example from: https://www.geeksforgeeks.org/content-providers-in-android-with-example/

Broadcast Receivers

They are known to be intent listeners as they enable your application to listen to the Intents that satisfy the matching criteria specified by us. Broadcast Receivers make our application react to any received Intent thereby making them perfect for creating event-driven applications.

Example from: https://www.geeksforgeeks.org/broadcast-receiver-in-android-with-example/