Splash Screen

Splash Screen

- Starting in Android 12, the SplashScreen API enables a new app launch animation for all apps when running on a device with Android 12 or higher.
- This includes an into-app motion at launch, a splash screen showing your app icon, and a transition to your app itself.
- A SplashScreen is a Window and therefore occludes an Activity.
- This experience brings standard design elements to every app launch, but it’s also customizable so your app can maintain its unique branding.
- To understand why splash screens are essential, you need to understand the various app startup states.

App Startup States

- Users launch Apps in many ways:
  - first-time open from a fresh install,
  - quickly switching between **Recents** or
  - from the main launcher for the first time in days.
- Your app needs to handle all the above scenarios.
- Your app will launch from one of the following states:
  - Cold start
  - Warm start
  - Hot start

App Startup States
Cold Start

- Any time your app is freshly launched, including the first launch after installation or device restart, it’s considered a cold start.
- The system also kills apps in the background to free memory.
- If your app hasn’t been used for a while or the device is low on memory, then even if your app has previously been open, the system might still treat it as a cold start.
- Launching from that state takes the longest because the system has to create the app process and load in the app.
- This is a perfect use case for showing a splash screen that gives the impression of a shorter delay while giving you the opportunity to get creative with your branding.

App Startup States
Warm Start

- Your app can be in various states of retainment as the system starts to release memory. That creates a warm start.
- The system does not need to initialize everything but has lost some information and needs to reload it.
- For example, the app process might still be running, but the activity requires re-creation to display.
- That doesn’t take as long as a cold start, but it’s often long enough to warrant a splash screen.

App Startup States
Hot Start

- If the system has the app process and layout information in memory, it doesn't need to re-initialize anything and can bring the app back to the foreground.
- This is a hot start and it's quick, leaving no time for a splash screen.

How the Splash Screen Works

- When a user launches an app while the app's process is not running (a cold start) or the Activity has not been created (a warm start), the following events occur. (The splash screen is never shown during a hot start.)

1. The system shows the splash screen using themes and any animations that you've defined.
2. When the app is ready, the splash screen is dismissed and the app is displayed.

- The splash screen can be customized, allowing you to supply your own logo animation and branding.

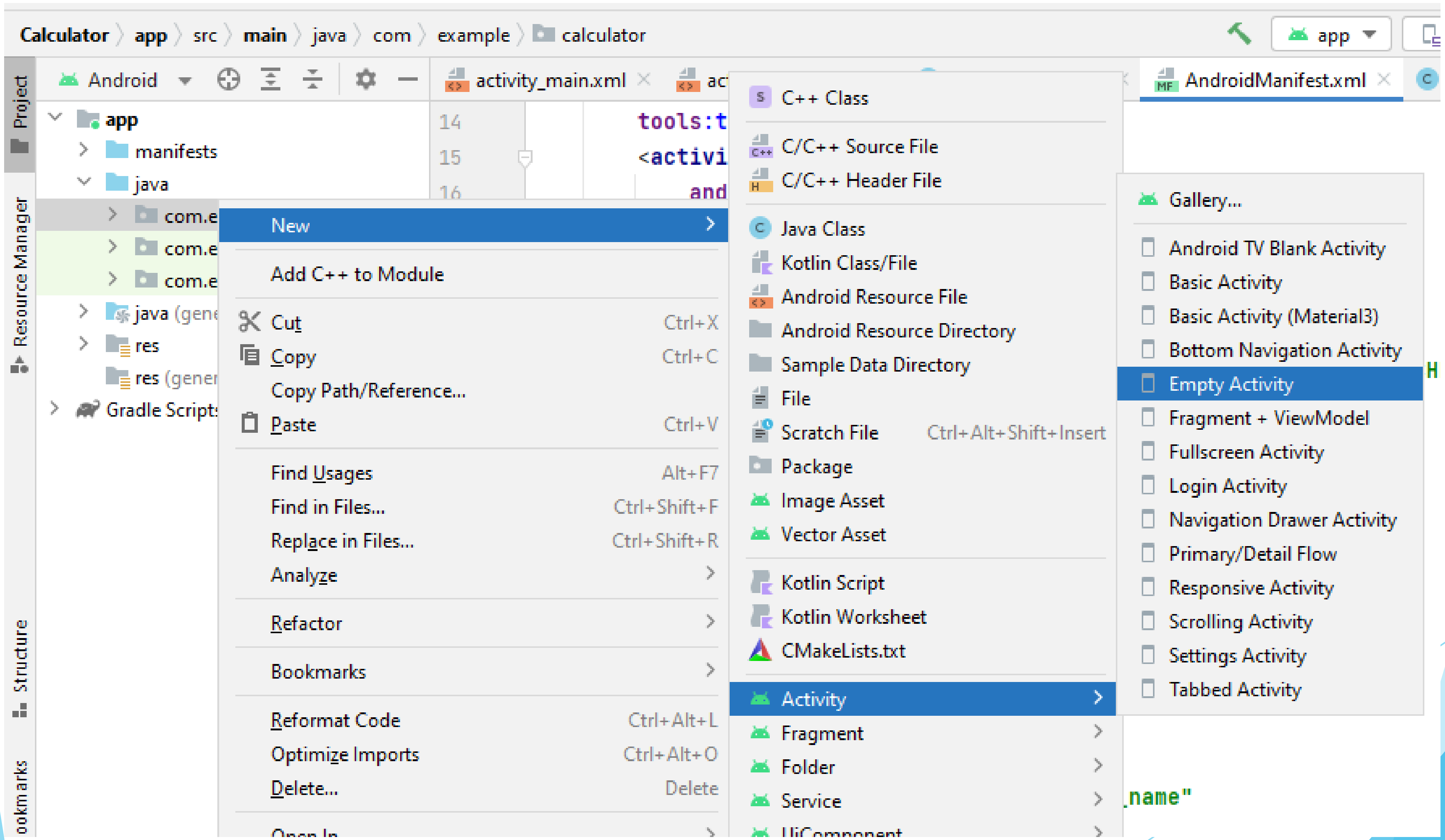
To insert and use ImageView

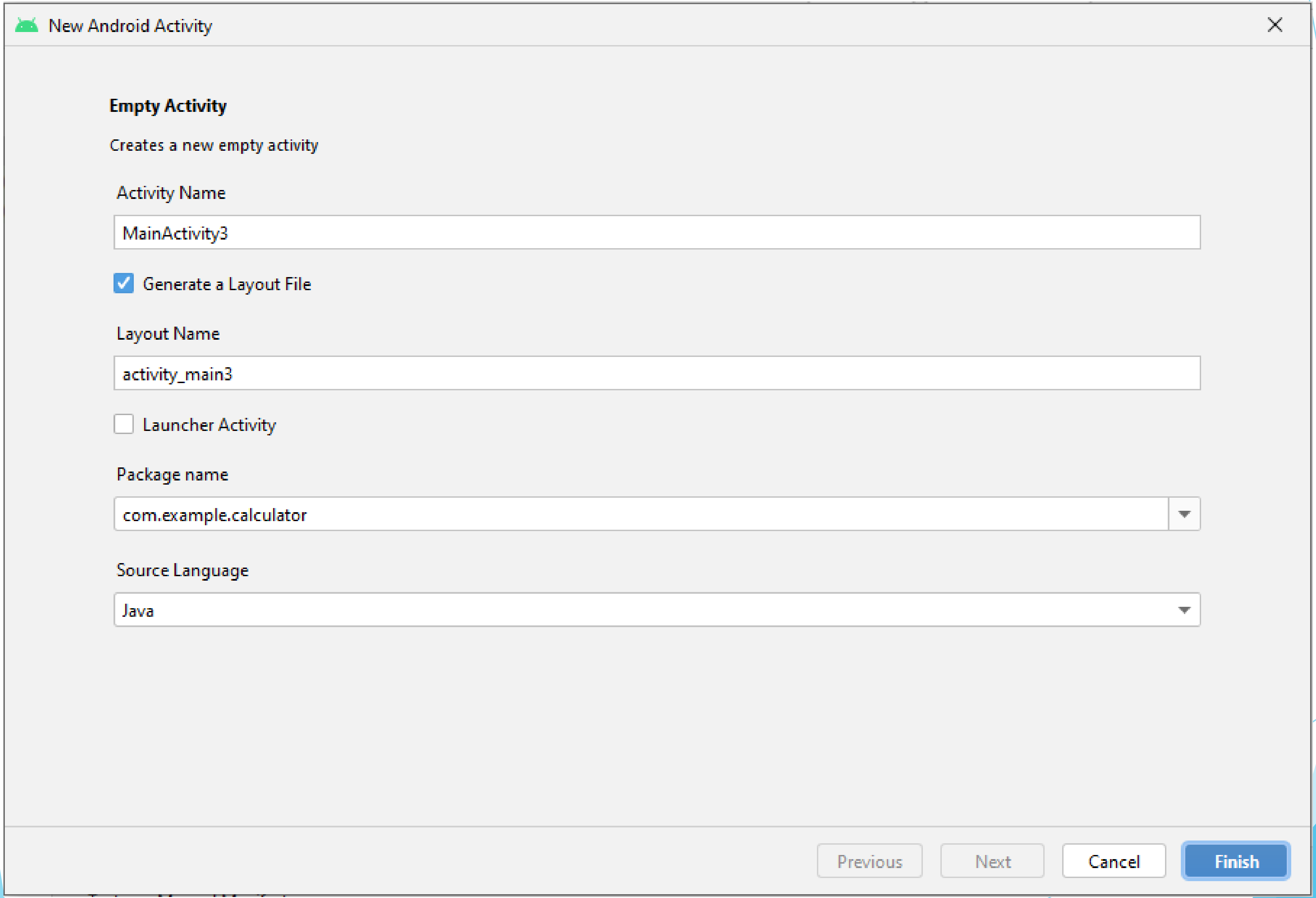
- Download a background image and calculator png image from google.
- Open the image and copy it using ctrl + c (or right click the opened image and click on copy)
- Paste the image in android studio.
- To paste:
    1. On the left most pane of your android studio project, select the first tab i.e. **Project**
    2. Expand the folder **res,** then expand the folder **Drawable**. The folder already contains two image files.
    3. Paste the copied image here, on pasting a dialog box will appear write the name of the image (must not contain spaces in name, must not contain any capital letter in the name, only SMALL letters)
    4. To use that image, go to palette, click on **Widgets**, drop **ImageView** to the design view.
    5. A dialog box appear, choose the name of the image you want to choose and press ok.

To add Splash Screen (XML)

- Create a new empty activity. To create:
    1. On the left most pane open the tab **Project**
    2. Expand the folder **Java**
    3. Right click on the package name, select **New**, the select **Activity**, then select **Empty Activity**
    4. Give the name of the activity e.g. SplashScreen
    5. A new java and xml file appears with the given name.
- Set the background of your splash screen
- Drop an **ImageView** and choose the logo you wish to appear on your splash screen.
- Set the constraints of your image dropped.

Create New Activity in the project





Empty Activity

Creates a new empty activity

Activity Name

MainActivity3

☑ Generate a Layout File

Layout Name

activity_main3

☐ Launcher Activity

Package name

com.example.calculator

Source Language

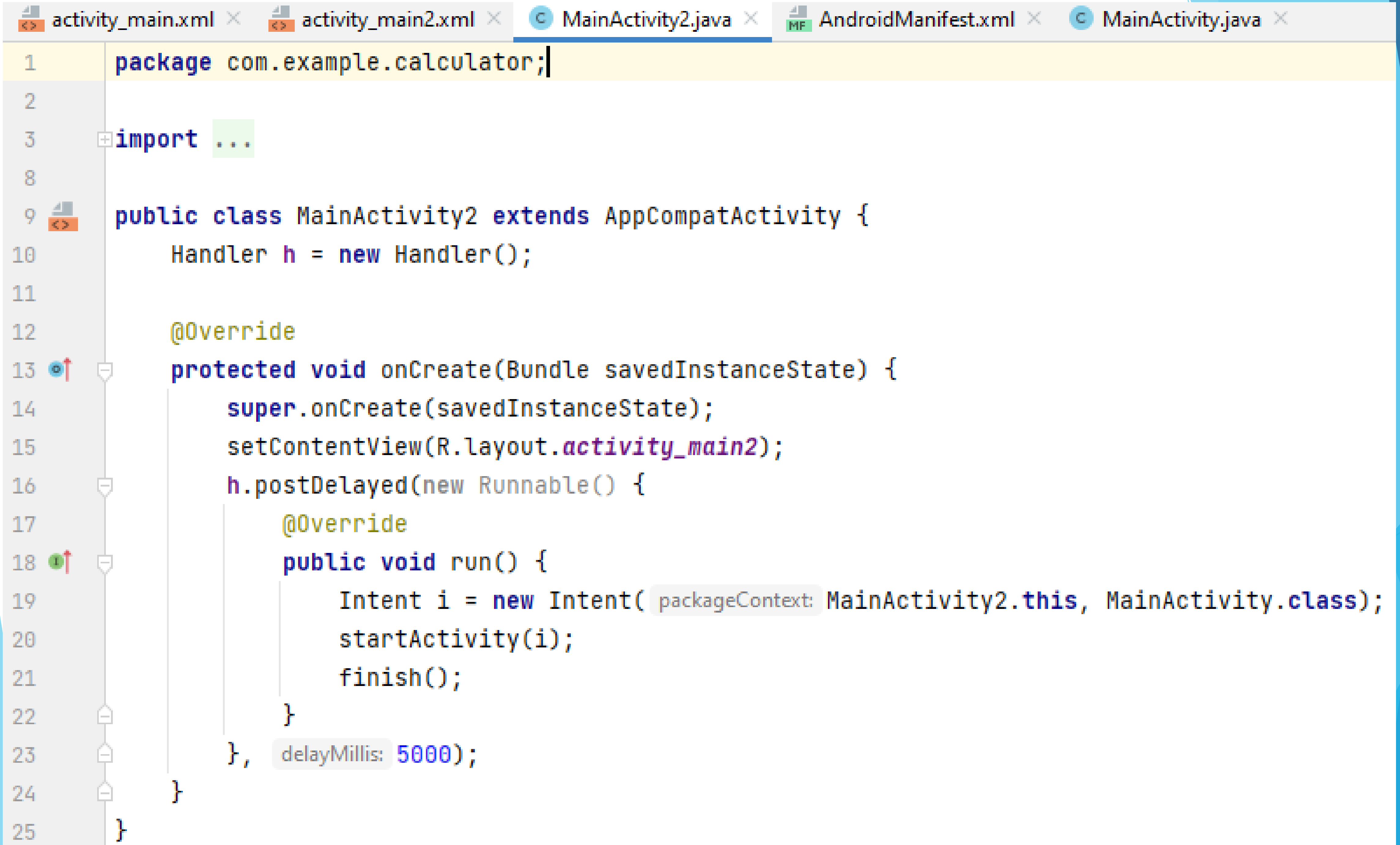
Java

To add Splash Screen (Java)

1. Open the Java file of your splash screen activity.
2. Under the main class activity class (i.e. “public class MainActivity2 extends AppCompatActivity {“) make an object of Hnadler class.
    1. Handler h = new Handler(); // here the object name is h and Handler is the class name.
    2. Syntax to create object of class in java is as follows:
    3. ClassName objectName = new ClassName();
3. Now go to onCreate Method, under the R.layout method
    1. Use postDelayed method, this method is used to delay the execution time of your code. Write new Runnable and hit enter it will do rest of code on its own.
    2. Right after where the method runnable ends },5000)
    3. It means you want to delay the code for 5 seconds (5000 miliseconds)
    4. Now create an object of Intent class under the **public void run()**
    5. **Intent I = New Intent(SplashScreen.this , MainActivity.class)**
    6. Here we pass two parameter one for this current activity and one for the activity where we want to go

7. Now in the next line simply call the **startActivity(i)** and pass the Intent object.
8. Now go to the next line and just add the **finish();** function
9. Here is the code:

```
public class MainActivity2 extends AppCompatActivity {
    Handler h = new Handler();                //create object of Handler class
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        h.postDelayed(new Runnable() {     //use the postDelayed method with Handler class object
            @Override
            public void run() {
                Intent i = new Intent(MainActivity2.this, MainActivity.class); //MainActivity2 is splash screen activity and
                                                                               //MainActivity is the calculator screen
                startActivity(i);        //startActivity function is called and intent class object is passed
                finish();          //finish function is used
            }
        }, 5000); //the delay time is given in miliseconds
    }
}
//NOTE: already written code in red color, comments in blue color. Screenshots on next slides.
```

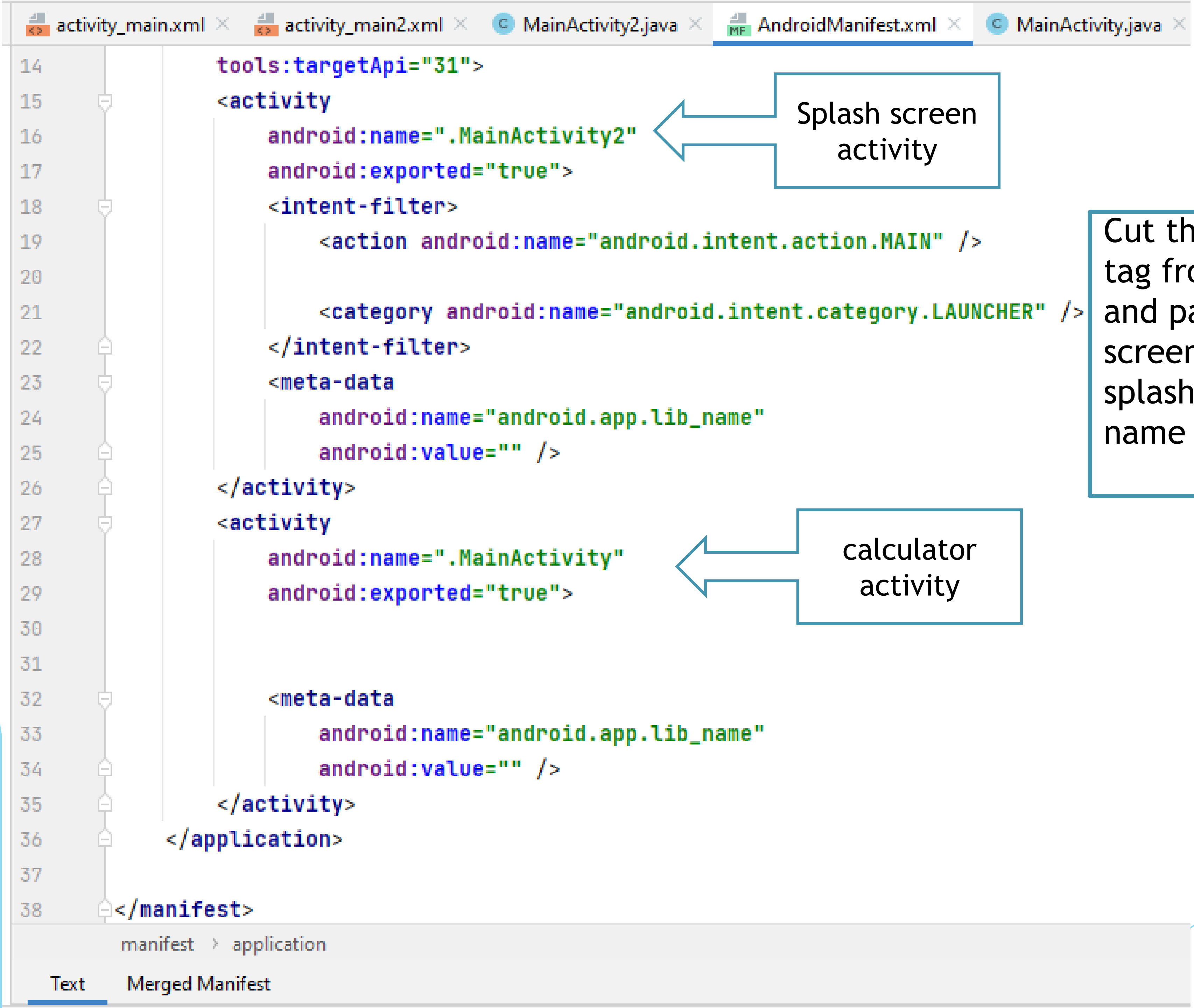

activity_main.xml | activity_main2.xml | MainActivity2.java | AndroidManifest.xml | MainActivity.java

```java
package com.example.calculator;

import ...

public class MainActivity2 extends AppCompatActivity {
    Handler h = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        h.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent i = new Intent( packageContext: MainActivity2.this, MainActivity.class);
                startActivity(i);
                finish();
            }
        }, delayMillis: 5000);
    }
}
```

Android Manifest XML file

- As we created the calculator on MainActivity class and splash screen later on.
- We need to tell the android studio which file needs to be run first.
- To do so:

1. Go to **Project,** expand the folder **app,** expand the folder **manifests.**
2. Open **AndroidManifest.xml** file.
3. Cut the intent filter tag(the complete tag from start till end) from **MainActivity** tag and paste the complete tag in SplashScreen Acitivity tag.
4. Set the **android:exported** value as “**true**”



```
        tools:targetApi="31">
        <activity
            android:name=".MainActivity2"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
        <activity
            android:name=".MainActivity"
            android:exported="true">


            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>

</manifest>
```

Splash screen activity

calculator activity

Cut the intent-filter tag from MainActivity and paste it in splash screen activity, here splash screen activity name is MainActivity2

Explanation

- Handler Class:
  - In android Handler is mainly used to update the main thread from background thread or other than main thread.
  - A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue.
  - Each Handler instance is associated with a single thread and that thread's message queue. When you create a new Handler it is bound to a Looper.
  - It will deliver messages and runnables to that Looper's message queue and execute them on that Looper's thread.
  - Scheduling message is accomplished with post(Runnable)
- Runnable:
  - The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.
  - The class must define a method of no arguments called run .

- Looper:
  - Class used to run a message loop for a thread.
- Thread:
  - A thread is a lightweight sub-process, it going to do background operations without interrupt to UI.
  - When the user launches your app, Android creates a new Linux process along with an execution thread.
  - This main thread, also known as the UI thread, is responsible for everything that happens onscreen. Understanding how it works can help you design your app to use the main thread for the best possible performance.
  - To keep your application responsive, it is essential to avoid using the main thread to perform any operation that may end up keeping it blocked.
- postDelayed:
  - postDelayed is a method that causes the Runnable r to be added to the message queue, to be run after the specified amount of time elapses.

Intent

- Intent is to perform an action.
- It is mostly used to start activity, send broadcast receiver, start services and send message between two activities.
- There are two intents available in android as Implicit Intents and Explicit Intents.
- **Explicit Intent**
  - It connects the internal world of an application such as start activity or send data between two activities. T
  - o start new activity we have to create Intent object and pass source activity and destination activity.

```
Intent i = new Intent(MainActivity.this, SecondActivity.class);

startActivity(i);
```

Intent

- **Implicit Intents**
  - It connects our app with any other application on our device such as call, mail, phone, visit any website ..etc.
  - In implicit intent we have to pass an action using setAction() as shown below example.

```
Intent i = new Intent();
i.setAction(Intent.ACTION_VIEW);
i.setData(Uri.parse("www.uaf.edu.pk"));
//uri is uniform resource identifier
startActivity(i);
```