Lecture 2

Activities, Fragments, and Intents

Course: **Mobile App Development**

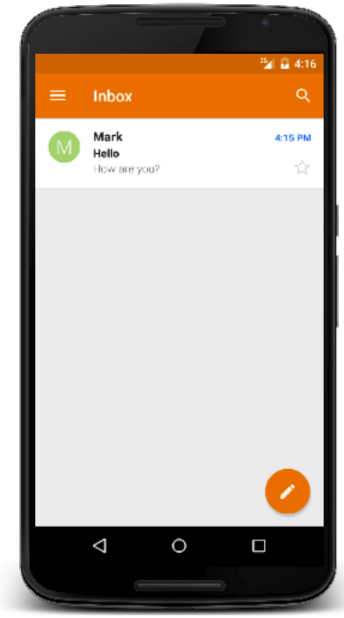
By: Tho C. Mai
Nha Trang University

Activities, Fragments, and Intents

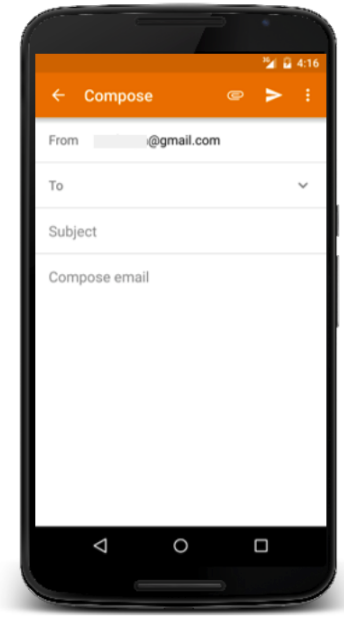
- The life cycles of an activity
- Using fragments to customize your UI
- Understanding the concept of intents

Activities

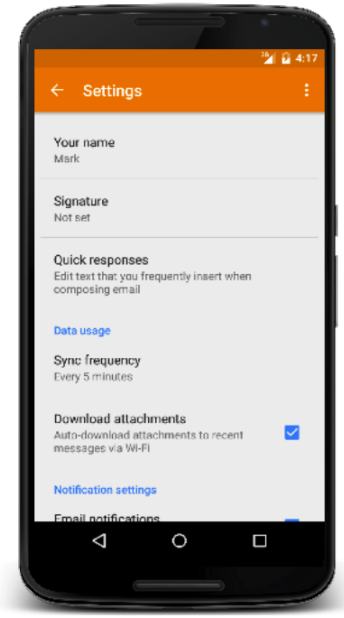
- Typically, applications have one or more activities
- The main purpose of an activity is to interact with the user
- An activity represents a single screen in your app with an interface the user can interact with.
  - An email app for example might contain several Activities:
    - Message Activity
    - Compose Activity
    - Settings Activity



Messages Activity



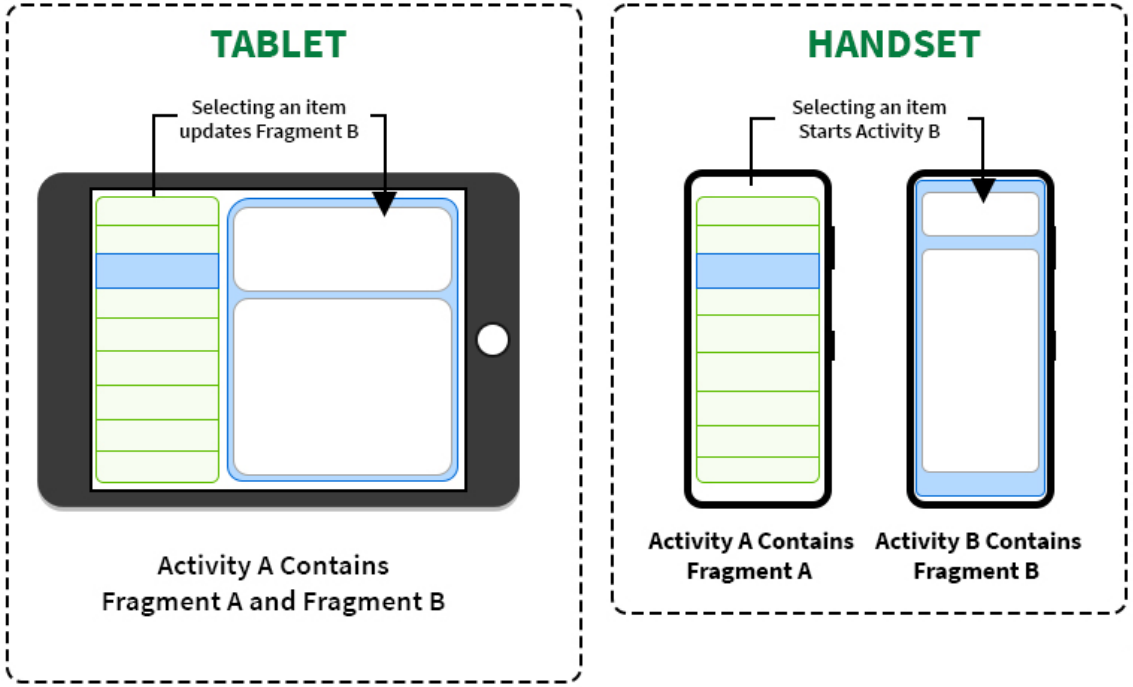
Compose Activity



Settings Activity

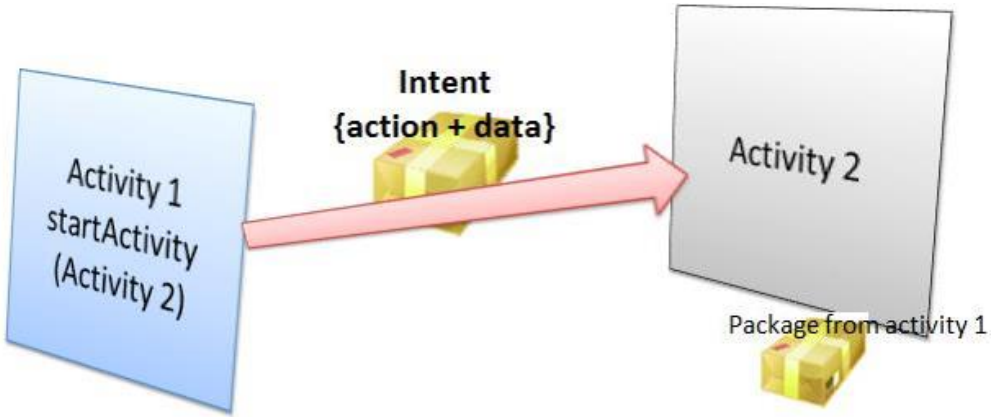
Fragments

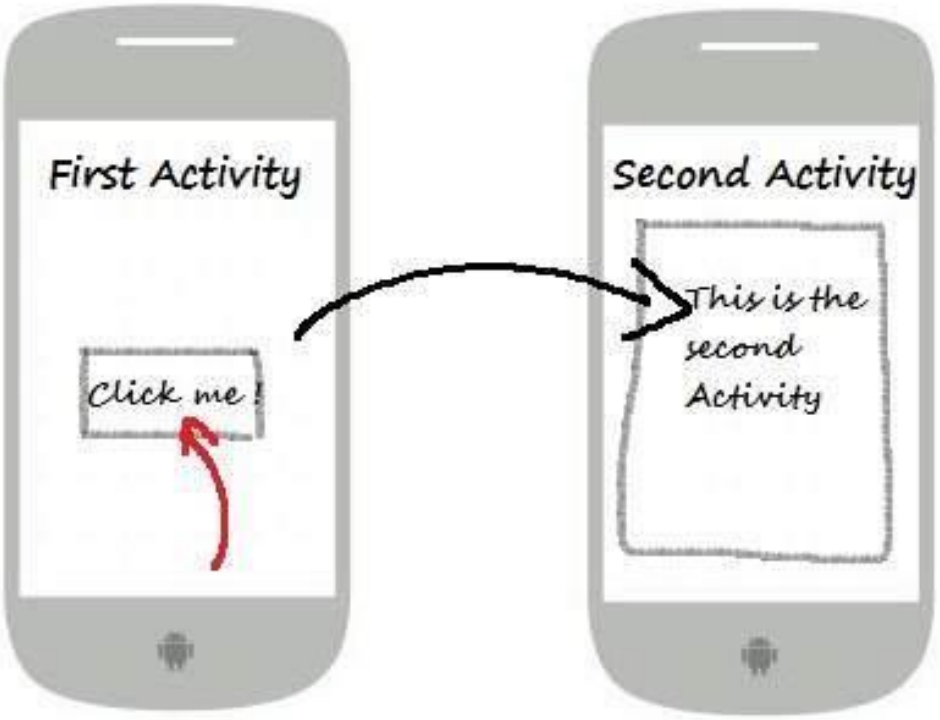
- Fragment is a feature that was introduced for tablets in
- Android 3.0 and for phones in Android 4.0
- Think of fragments as “miniature” activities that can be grouped to form an activity



Intent

- An intent is the “glue” that enables activities from different applications to work together seamlessly, ensuring that tasks can be performed as though they all belong to one single application

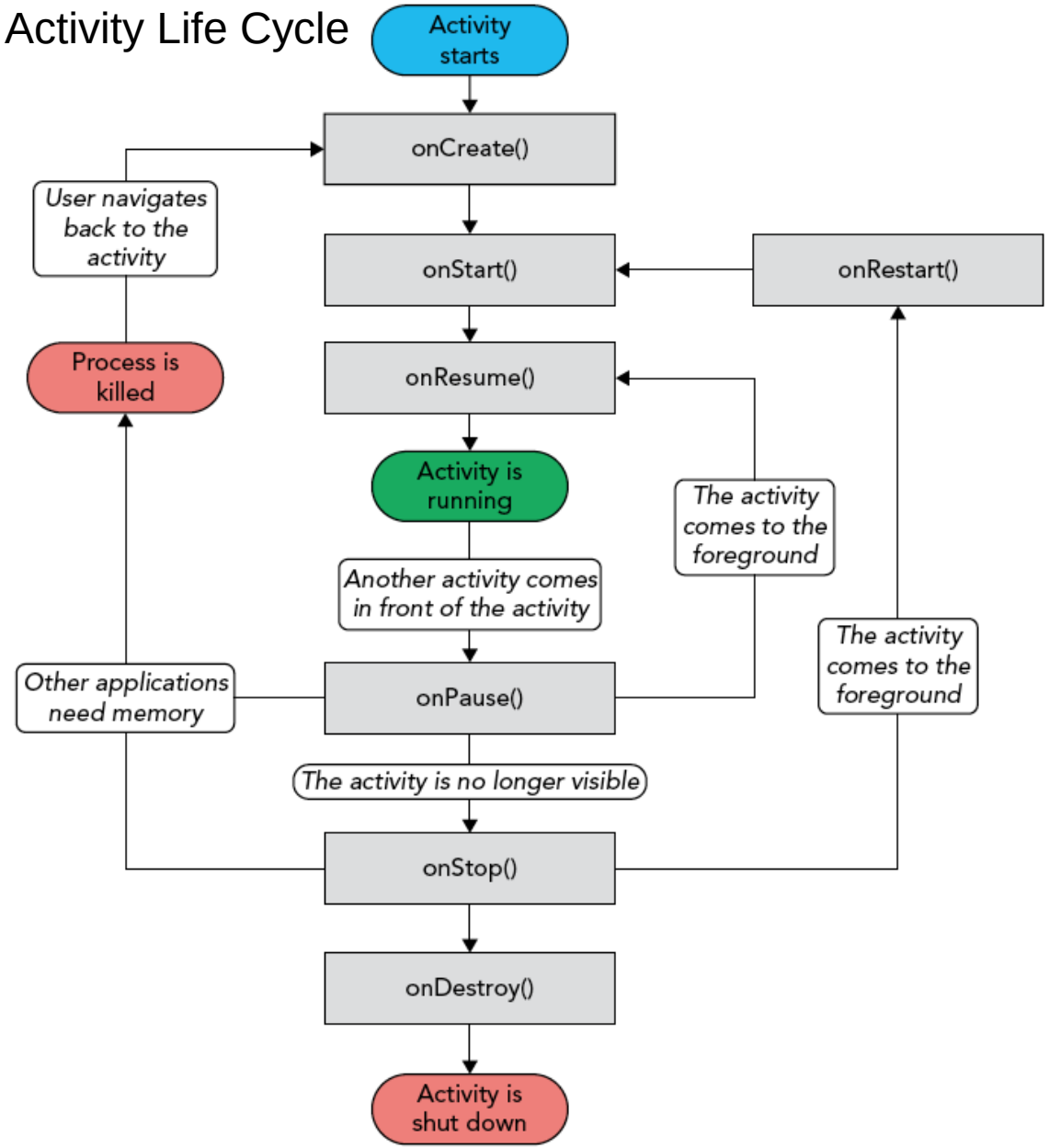


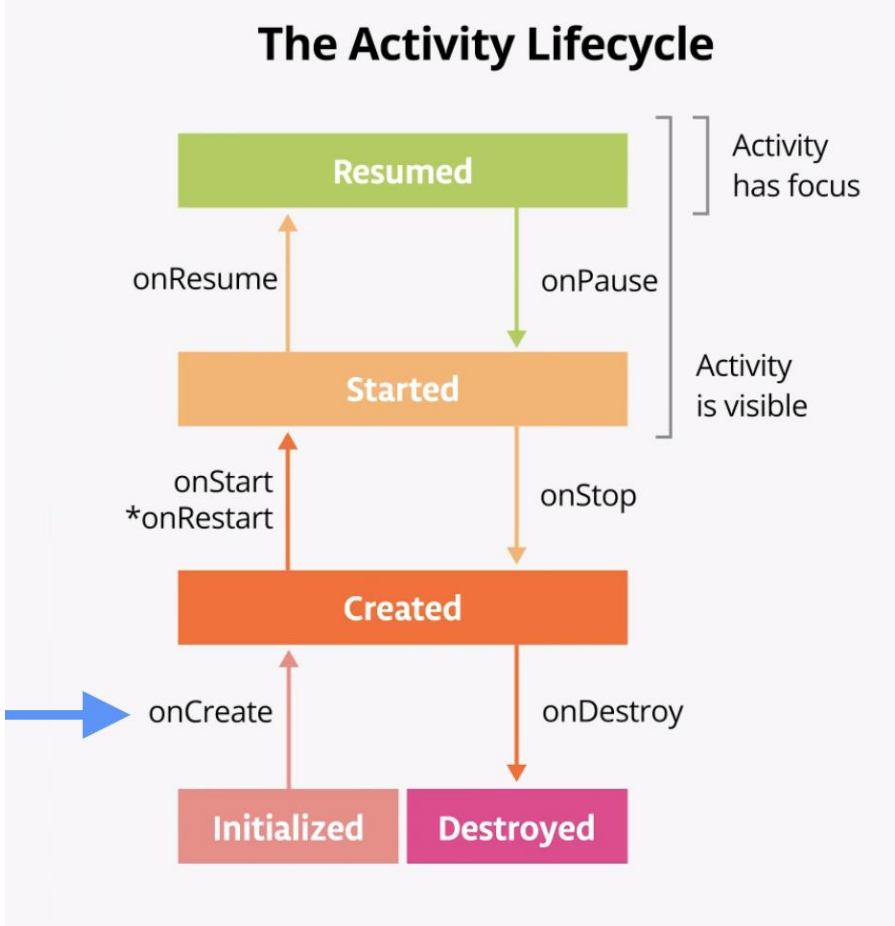


Activity Life Cycle

- *An activity's life cycle: from the moment an activity appears on the screen to the moment it is hidden, it goes through a number of stages*
- onCreate()—Called when the activity is first created
- onStart()—Called when the activity becomes visible to the user
- onResume()—Called when the activity starts interacting with the user
- onPause()—Called when the current activity is being paused and the previous activity is being resumed
- onStop()—Called when the activity is no longer visible to the user
- onDestroy()—Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- onRestart()—Called when the activity has been stopped and is restarting again

Activity Life Cycle





Observe Activity Life Cycle

- Using Android Studio, create a new Android project and name it *ActivityLifeCycle*, package name like this: com.usename.
- In the MainActivity.java file, add the following highlighted statements

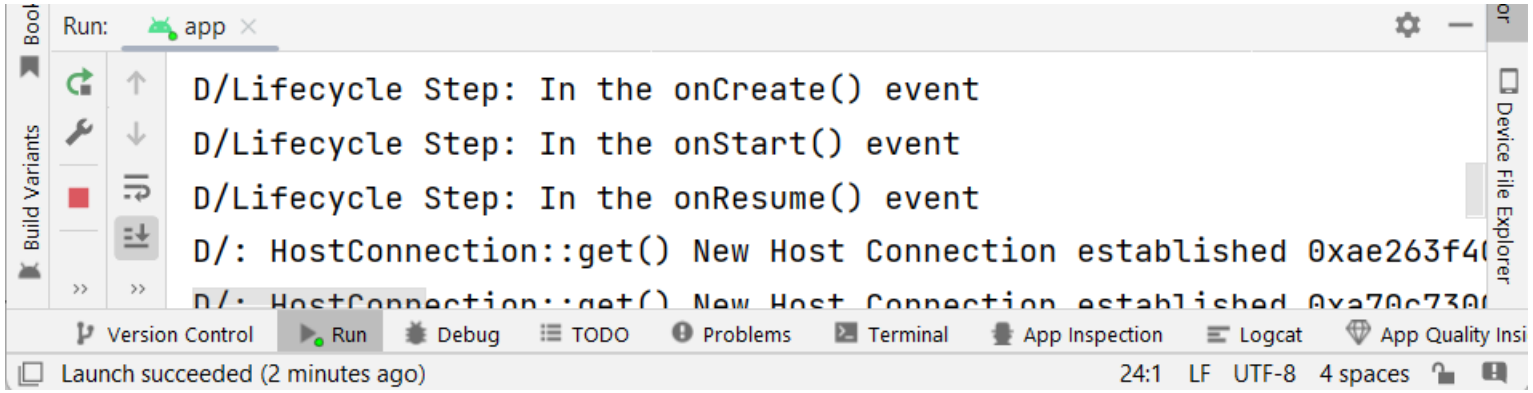
```
package com.maicuongtho.activitylifecycle;
..
import android.util.Log;
public class MainActivity extends AppCompatActivity {
   String tag = "Lifecycle Step";
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(tag, "In the onCreate() event");
   }
```

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

setContentView(R.layout.activity_main);
   Log.d(tag, "In the onCreate() event");
}
public void onStart()
{
   super.onStart();
   Log.d(tag, "In the onStart() event");
}
public void onRestart()
{
   super.onRestart();
   Log.d(tag, "In the onRestart() event");
}
```

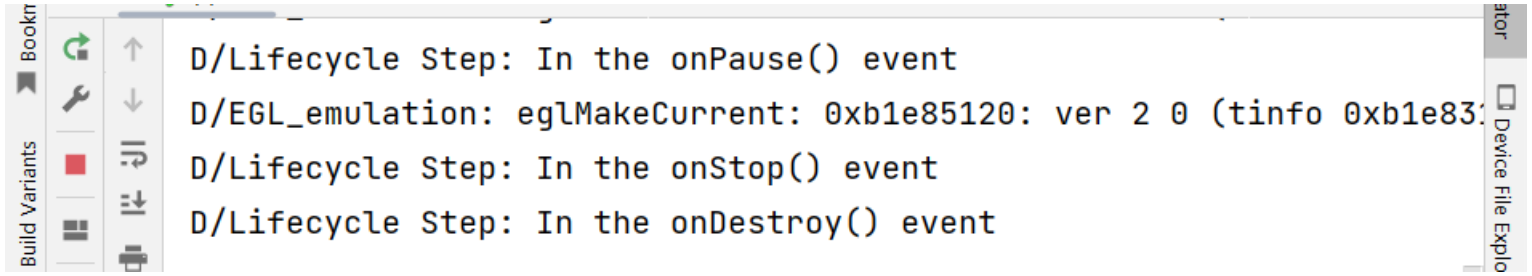
```
public void onResume()
  {
     super.onResume();
     Log.d(tag, "In the onResume()
event");
  }
  public void onPause()
  {
     super.onPause();
     Log.d(tag, "In the onPause() event");
  }
  public void onStop()
  {
     super.onStop();
     Log.d(tag, "In the onStop() event");
  }
  public void onDestroy()
  {
     super.onDestroy();
     Log.d(tag, "In the onDestroy()
event");
  }
}
```

Observe Activity Life Cycle

- When the activity is first loaded, you should see something very similar to the following in the logcat console



- If you click the Back button on the Android emulator, you will see:



Observe Activity Life Cycle

- Click the Home button, click the Overview icon, select the Activity101 app, you will see:

```
11-16 06:31:08.905: D/Lifecycle Step(559): In the onCreate() event
11-16 06:31:08.905: D/Lifecycle Step(559): In the onStart() event
11-16 06:31:08.925: D/Lifecycle Step(559): In the onResume() event
```

- Click the Home button and then click the Phone button on the Android emulator so that the activity is pushed to the background

```
11-16 06:32:00.585: D/Lifecycle Step(559): In the onPause() event
11-16 06:32:05.015: D/Lifecycle Step(559): In the onStop() event
```

- Exit the phone dialer by clicking the Back button, the activity is now visible again:

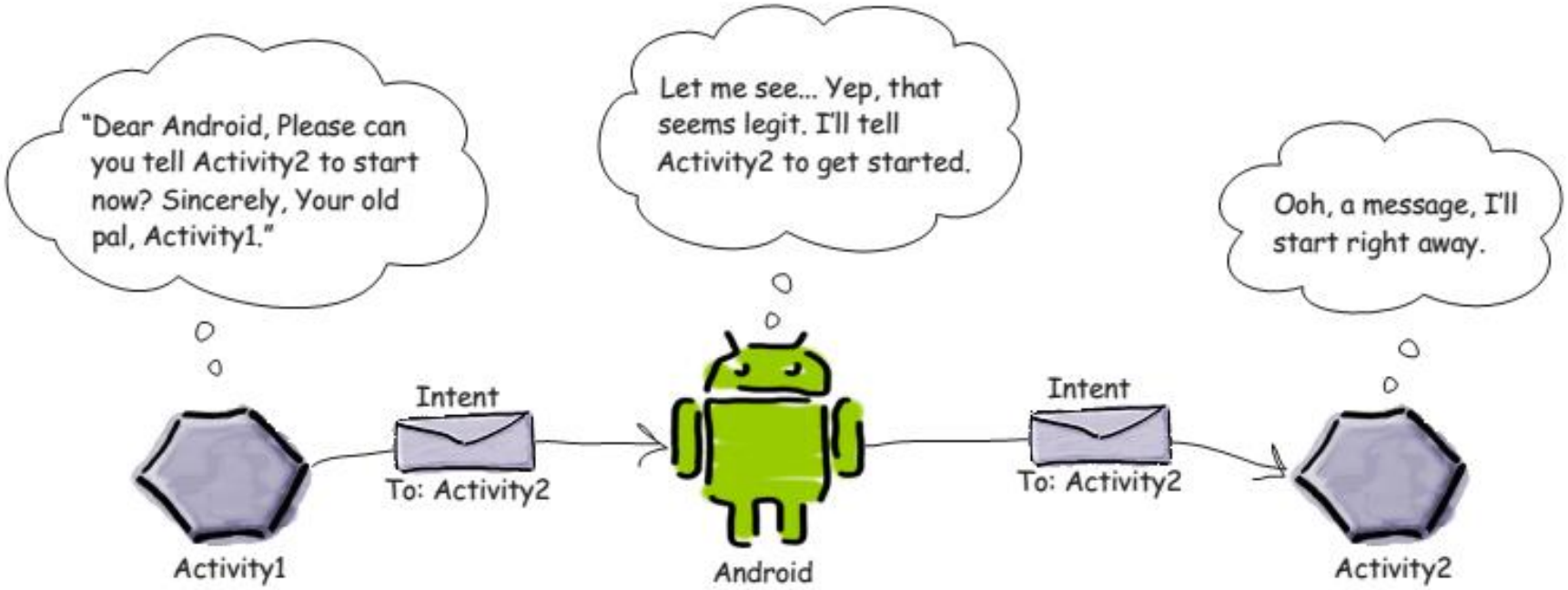
```
11-16 06:32:50.515: D/Lifecycle(559): In the onRestart() event
11-16 06:32:50.515: D/Lifecycle(559): In the onStart() event
11-16 06:32:50.515: D/Lifecycle(559): In the onResume() event
```

Observe Activity Life Cycle : Summary

- Use the onCreate() method to create and instantiate the objects that you will be using in your application
- Use the onResume() method to start any services or code that needs to run while your activity is in the foreground
- Use the onPause() method to stop any services or code that does not need to run when your activity is not in the foreground
- Use the onDestroy() method to free up resources before your activity is destroyed

Intents

- When your application has more than one activity, you often need to navigate from one to another. In Android, you navigate between activities through what is known as an intent

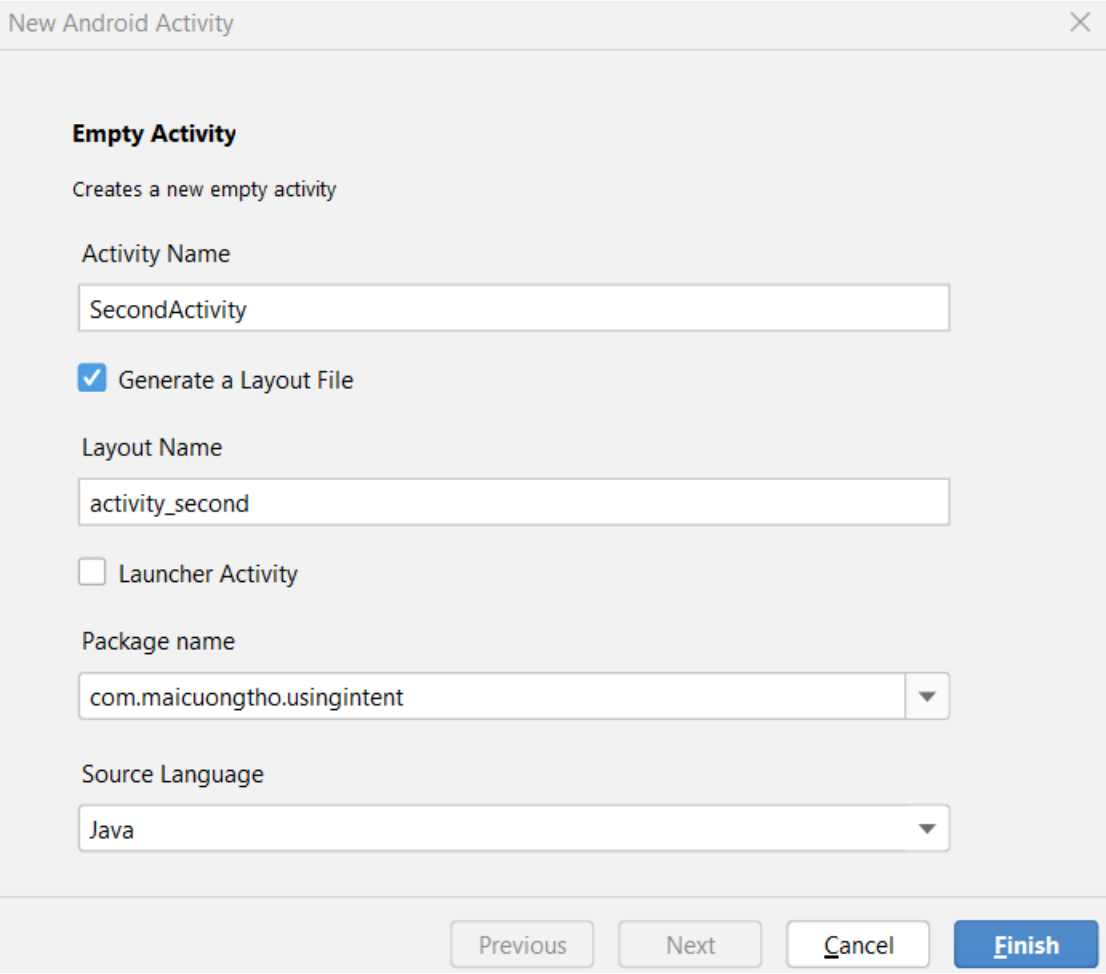


Linking Activities with Intents: Example

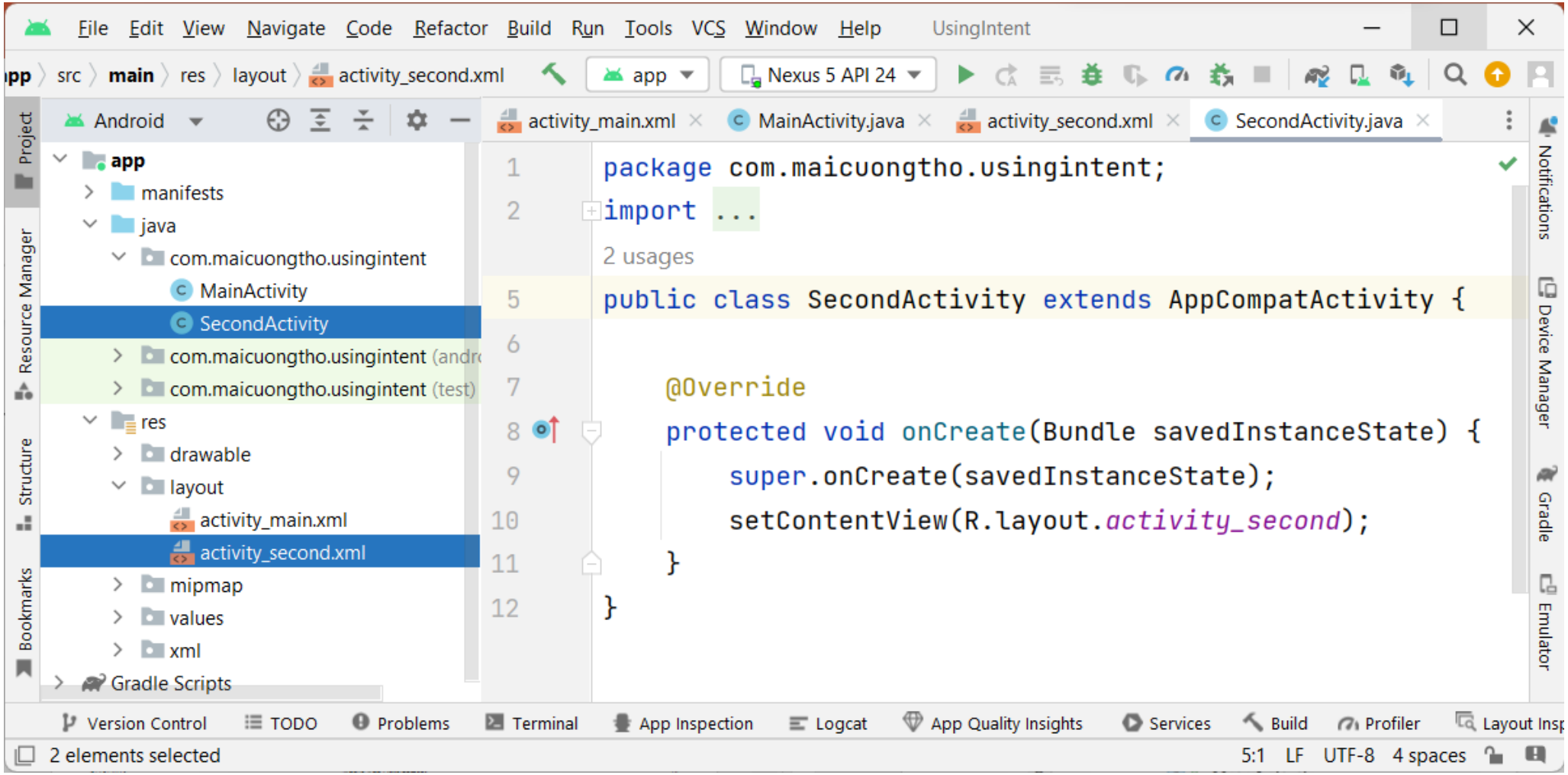
- Using Android Studio, create a new Android project with an empty Activity named MainActivity; name the project **UsingIntent**
- Right-click your package name under the java folder in the Project Files windows and select

New ➪ Activity ➪ Empty Activity

- Name the new activity SecondActivity and click OK

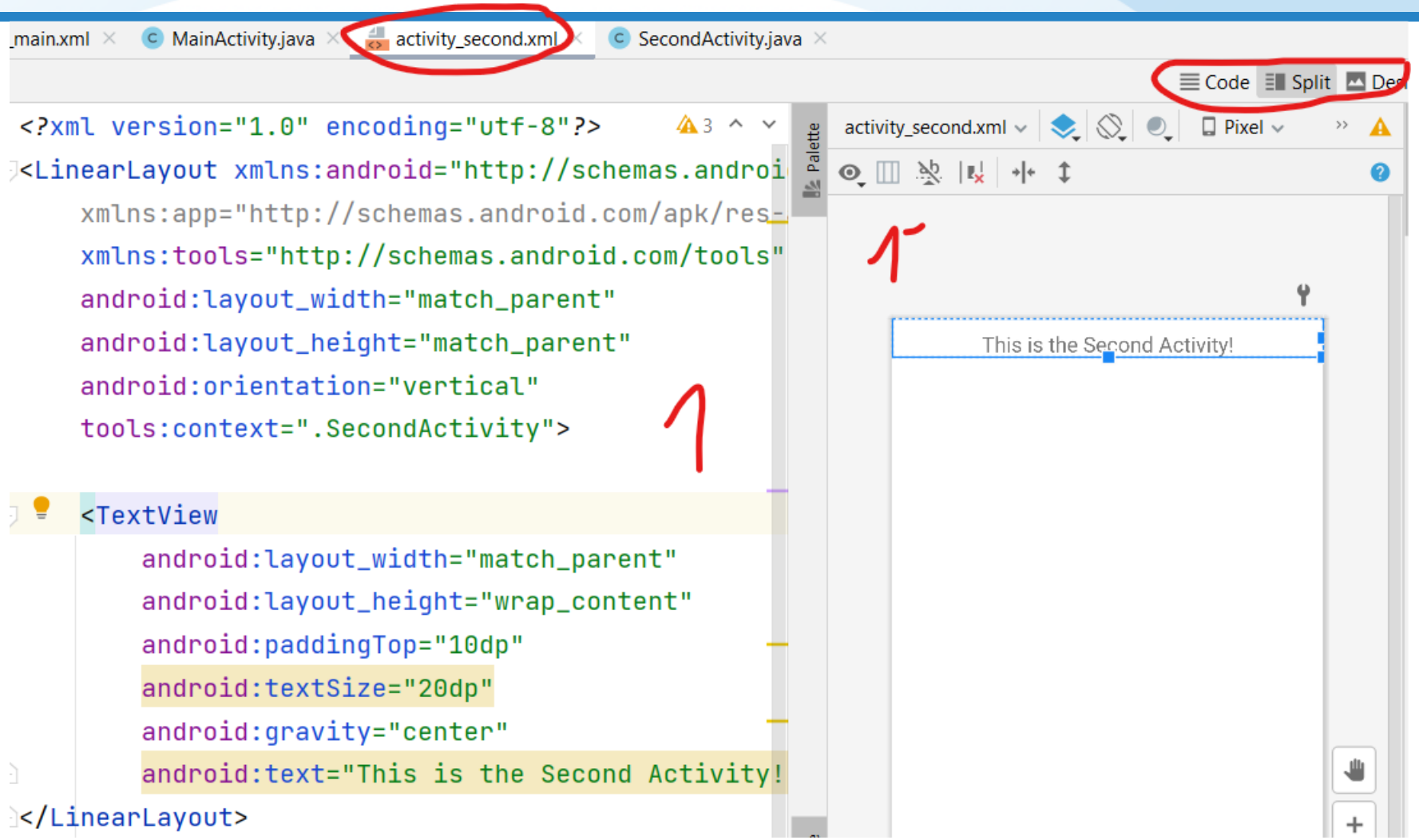


Linking Activities with Intents: Example



Linking Activities with Intents: Example

- Modify the activity_second.xml file as follows:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.androi
    xmlns:app="http://schemas.android.com/apk/res-
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".SecondActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="10dp"
        android:textSize="20dp"
        android:gravity="center"
        android:text="This is the Second Activity!
</LinearLayout>
```

Linking Activities with Intents: Example

- Add the bolded lines in the following code to the activity_main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout …
 tools:context=".MainActivity">
  <TextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="15dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="15dp"
    android:text="Main Activity!"
    android:textSize="20dp"
    android:gravity="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
  <Button
    android:id="@+id/button"
    android:onClick="onClick"
    android:text="Display second activity"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginStart="20dp"
    android:layout_marginEnd="20dp"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
   />
</androidx.constraintlayout.widget.ConstraintLayout>
```

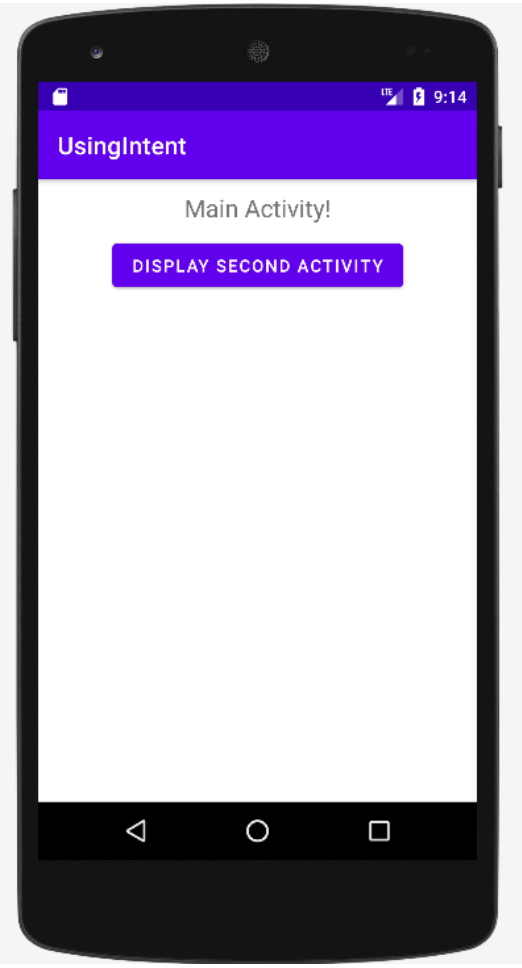
Linking Activities with Intents: Example

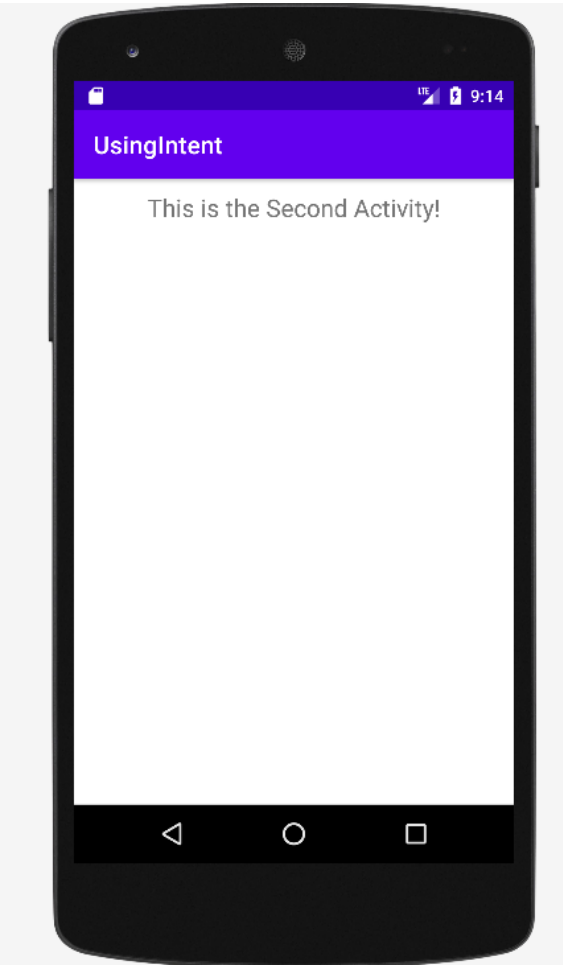
- Modify the MainActivity.java file as shown in the bolded lines in the following code:

```
package com.maicuongtho.usingintent;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void onClick(View view) {
      Intent intent_for_SecondActiviy = new Intent(this,SecondActivity.class);
      startActivity(intent_for_SecondActiviy);
   }
}
```

Linking Activities with Intents: Example

- Press Shift+F9 to debug (Shift+F10 to run) the application on the Android emulator
- When the first activity is loaded, click the button and the second activity also loads





Your turn

I fear not the man who has practiced 10,000 kicks once,
but I fear the man who has practiced one kick 10,000 times.
Bruce Lee

- Practice 2/ Excercie 2
  1) Repeat the example by yourself
  2) Push it to your github repository
     - ✓ With a report with screenshots of the final app in action, data structures used/class design, and the implementation logic.
- Practice 3/ Excercie 3 (Homework #1)

Homework #1

- Add a UI control on the screen of the second activity so that you can go back to the first activity (i.e., the main activity). In addition, on the main activity, display an iteration count on the number of times the main activity is displayed.

What to submit:

- Push it to your github repository,
  - a report with screenshots of the final app in action, data structures used/class design, and the implementation logic.