Thompson Rivers University

# COMP 2160

# Mobile App Development I

## MODULE 3:
## Activities, Intents, and Fragments

## Module 3

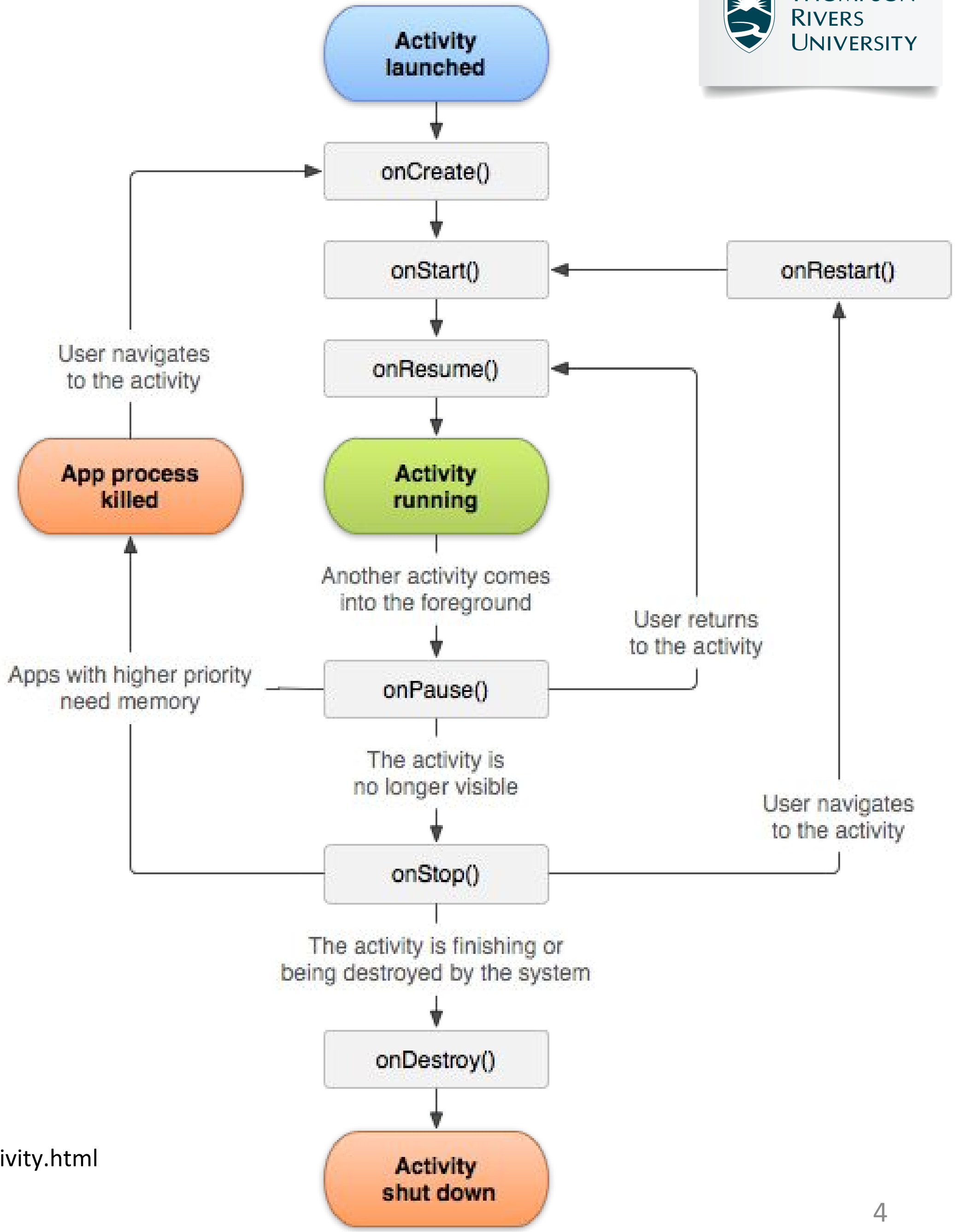
1. The life cycles of an activity
2. Intent and Intent Filters
3. Common Intents
4. Sharing data using Intents
5. Using the Log class to track the order of execution
6. Managing Multiple Activities
7. The life cycles of a Fragment
8. Managing Fragments
9. Communicating with Activities

## The life cycles of an activity

# The life cycles of an activity

## States

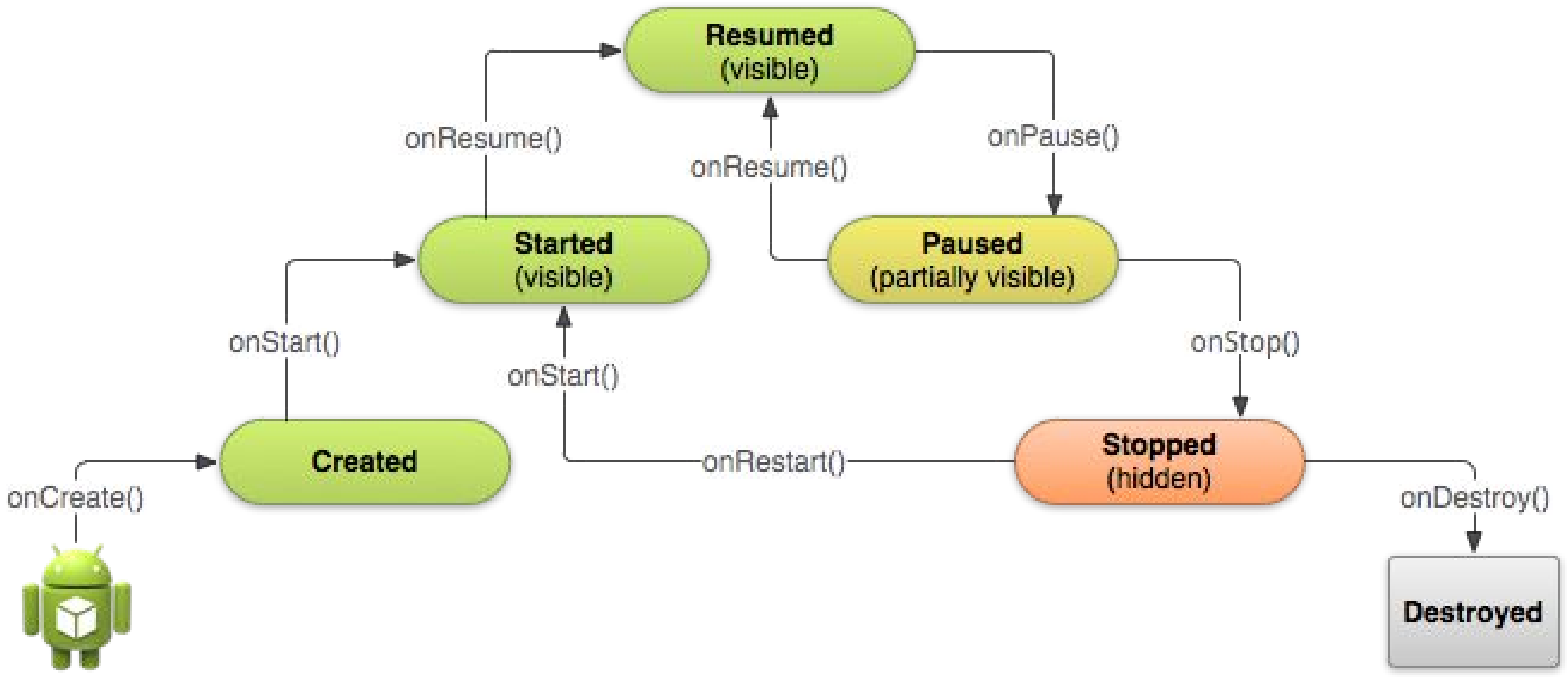
- Activity running
- Activity paused
- Activity stopped
- Activity destroyed


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
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

Ref: https://developer.android.com/reference/android/app/Activity.html

# The life cycles of an activity

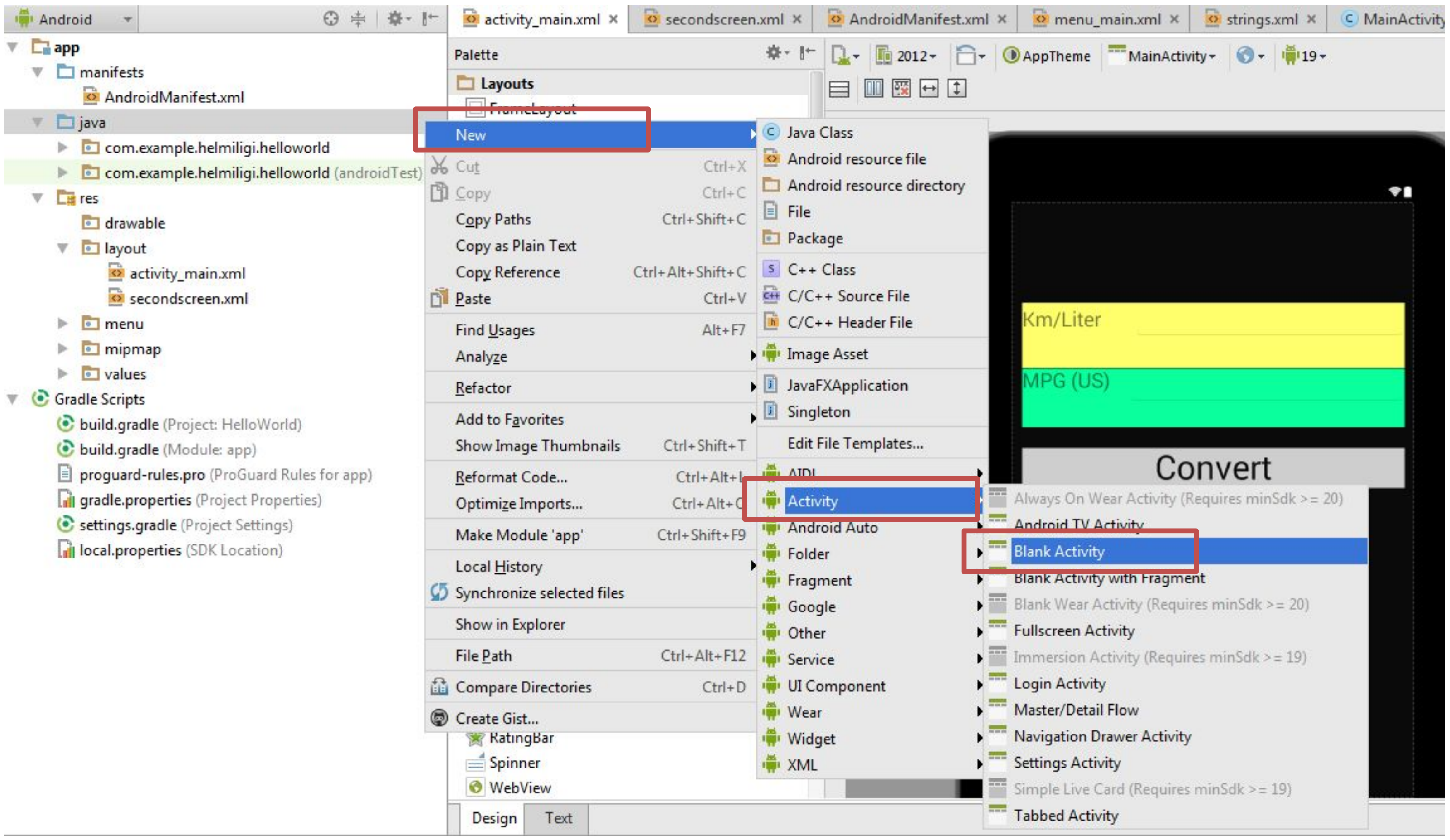
## Methods and Key Loops



Source: http://developer.android.com/training/basics/activity-lifecycle/starting.html

# The life cycles of an activity

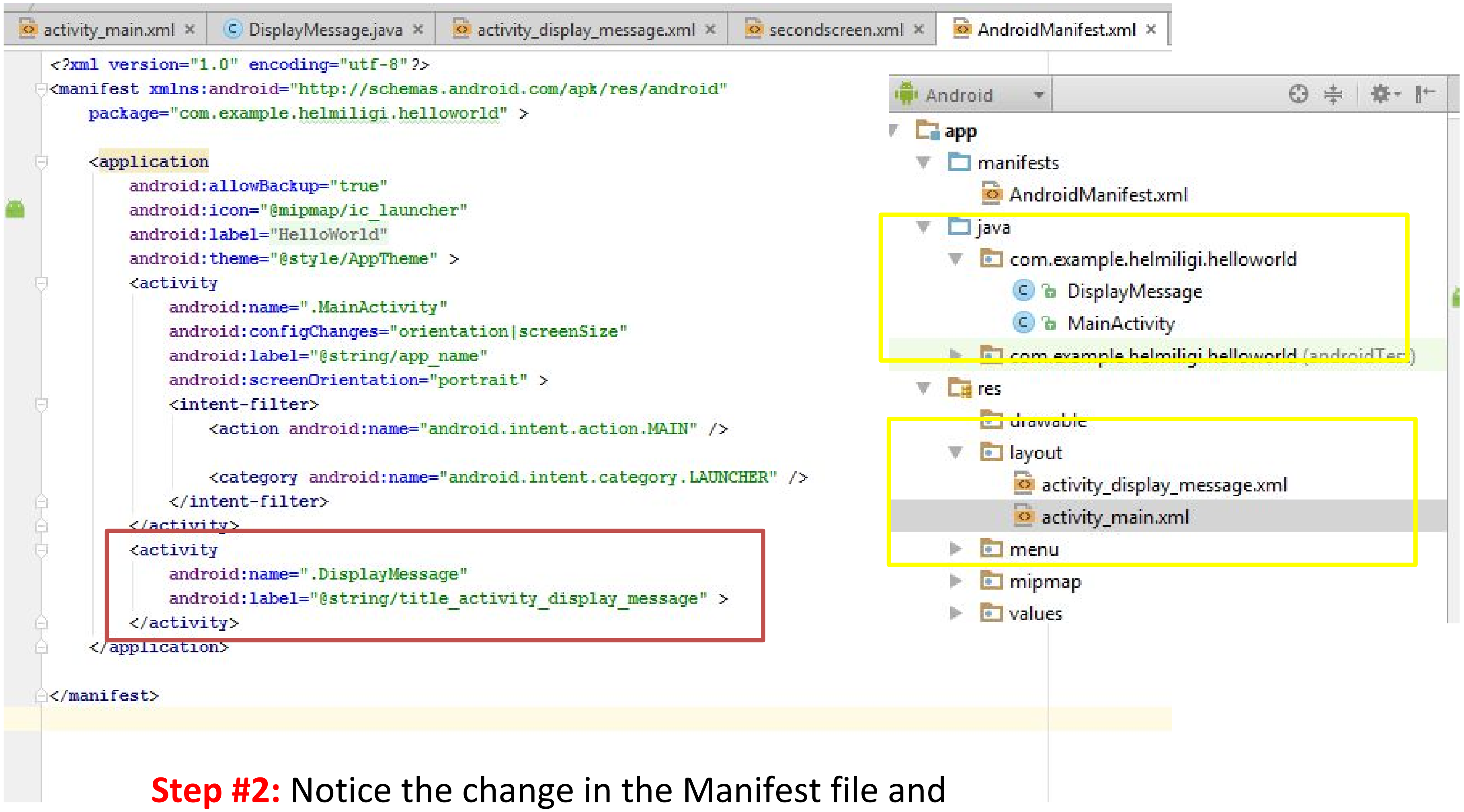
## Creating a new activity - 1



**Step #1:** Create a new Blank Activity File

# The life cycles of an activity

## Creating a new activity - 2



activity_main.xml × | DisplayMessage.java × | activity_display_message.xml × | secondscreen.xml × | AndroidManifest.xml ×

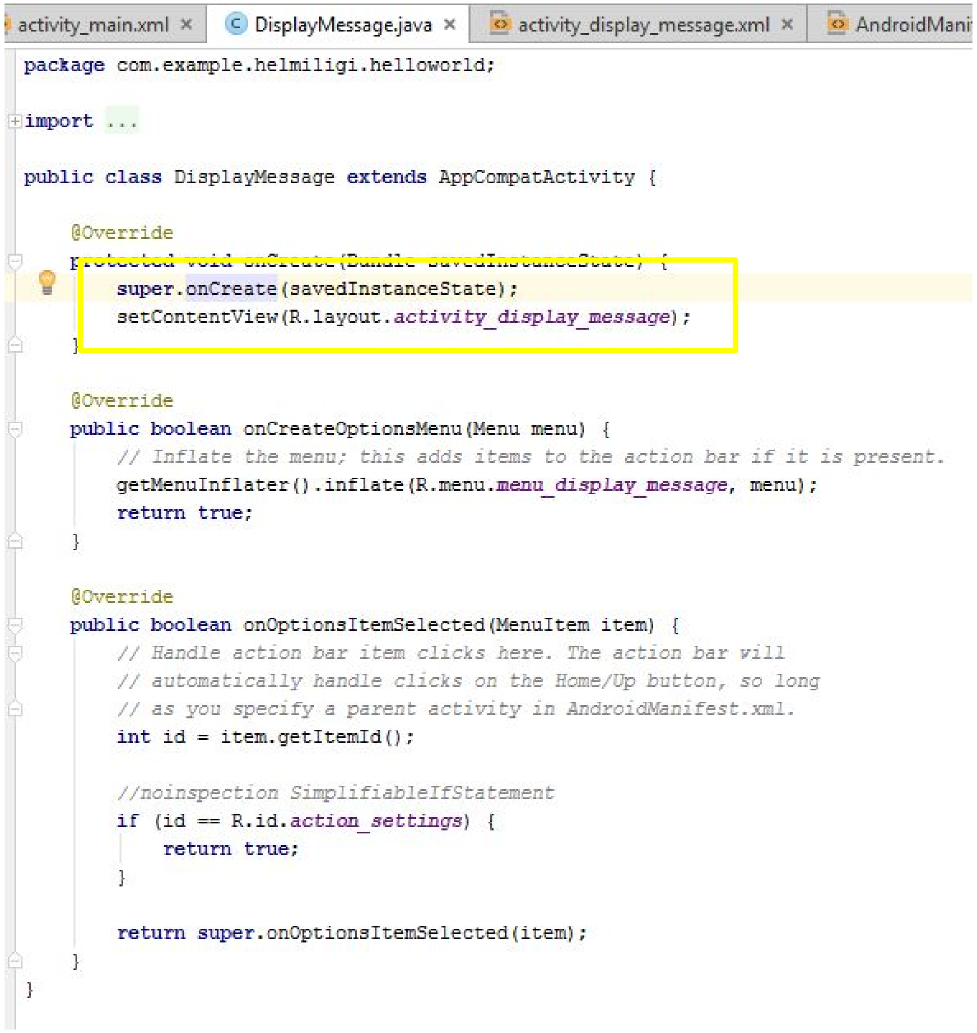
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helmiligi.helloworld" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:configChanges="orientation|screenSize"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".DisplayMessage"
            android:label="@string/title_activity_display_message" >
        </activity>
    </application>

</manifest>
```

```
Android
app
  manifests
    AndroidManifest.xml
  java
    com.example.helmiligi.helloworld
      DisplayMessage
      MainActivity
    com.example.helmiligi.helloworld (androidTest)
  res
    drawable
    layout
      activity_display_message.xml
      activity_main.xml
    menu
    mipmap
    values
```

**Step #2:** Notice the change in the Manifest file and layout directory

# The life cycles of an activity

## Creating a new activity - 3



```
package com.example.helmiligi.helloworld;

import ...

public class DisplayMessage extends AppCompatActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_display_message, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

**Step #3:** Open the new activity Java file.

## Handling Runtime Changes

## Handling Runtime Changes

### Screen Orientation (FAQ)

1. What happens when the orientation changes at runtime ?
2. Can you lock the screen orientation?
3. How can you handle changes during runtime?
4. Can you change the layout design at runtime?

https://developer.android.com/guide/topics/resources/runtime-changes.html#RetainingAnObject

## Handling Runtime Changes

### Screen Orientation (FAQ)

1. **What happens when the orientation changes at runtime ?**

- When such a change occurs, Android restarts the running Activity (onDestroy() is called, followed by onCreate()). The restart behavior is designed to help your application adapt to new configurations by automatically reloading your application with alternative resources that match the new device configuration.

https://developer.android.com/guide/topics/resources/runtime-changes.html#RetainingAnObject

## Handling Runtime Changes

### Screen Orientation (FAQ)

**2. Can you lock the screen orientation?**

- Yes.

```
<activity android:name="MyActivity"
          android:screenOrientation="landscape"
          android:configChanges="keyboardHidden|orientation|screenSize">
          ...
</activity>
```

## Handling Runtime Changes

### Screen Orientation (FAQ)

**Discussion**

**When should you utilize the lock screen orientation feature in your app?**

Give a few examples of good and bad use of lock screen orientation

## Handling Runtime Changes

### Screen Orientation (FAQ)

**3. How can you handle changes during runtime?**

**Retain an Object During a Configuration Change**

- Option 1: Bundle
- Option 2: Fragment

**Handle the Configuration Changes Yourself**

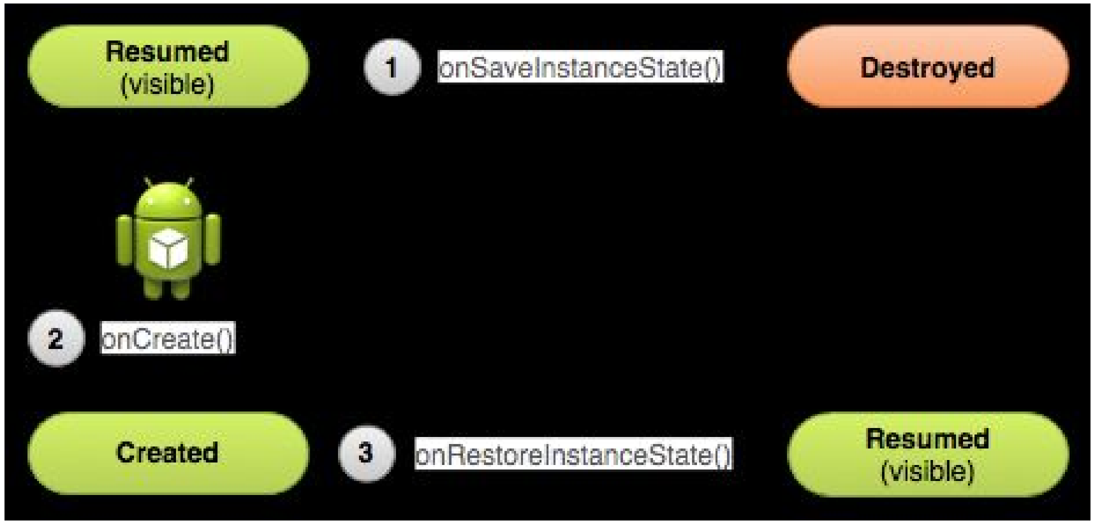
* You should always retain your activity state during normal activity life cycle

https://developer.android.com/guide/topics/resources/runtime-changes.html#RetainingAnObject

## Handling Runtime Changes

### Retain an Object During a Configuration Change Using Bundle

**Override the onSaveInstanceState() callback method**



*As the system begins to stop your activity, it calls onSaveInstanceState() (1) so you can specify additional state data you'd like to save in case the Activity instance must be recreated. If the activity is destroyed and the same instance must be recreated, the system passes the state data defined at (1) to both the onCreate() method (2) and the onRestoreInstanceState() method (3)*

https://developer.android.com/training/basics/activity-lifecycle/recreating.html

## Handling Runtime Changes

### Retain an Object During a Configuration Change Using Bundle

#### Save Your Activity State

```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
...

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
   // Save the user's current game state
   savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
   savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

   // Always call the superclass so it can save the view hierarchy
state
   super.onSaveInstanceState(savedInstanceState);
}
```

#### Restore Your Activity State

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState); // Always call the superclass first

   // Check whether we're recreating a previously destroyed instance
   if (savedInstanceState != null) {
      // Restore value of members from saved state
      mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
      mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
   } else {
      // Probably initialize members with default values for a new instance
   }
   ...
}
```

**OR**

```
public void onRestoreInstanceState(Bundle savedInstanceState) {
   // Always call the superclass so it can restore the view hierarchy
   super.onRestoreInstanceState(savedInstanceState);

   // Restore state members from saved instance
   mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
   mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
}
```

https://developer.android.com/training/basics/activity-lifecycle/recreating.h

## Handling Runtime Changes

### Screen Orientation (FAQ)

**Handle the Configuration Changes Yourself**

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|keyboardHidden"
          android:label="@string/app_name">

@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // Checks the orientation of the screen
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation ==
Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
    }
}
```

https://developer.android.com/guide/topics/resources/runtime-changes.html#RetainingAnObject

## Handling Runtime Changes

### Screen Orientation (FAQ)

**4. Can you change the layout design at runtime?**

**Yes.**

Option 1: Instantiate UI elements at runtime.

Option 2: Change UI properties at runtime.

Option 3: Create separate layouts using qualifiers.

Option 4: Use Fragments.

https://developer.android.com/guide/topics/resources/runtime-changes.html#RetainingAnObject

## Android Activities

### Activity Lifecycle Management

#### Activity

In the activity lifecycle, which method should be used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc.?

*onPause().*