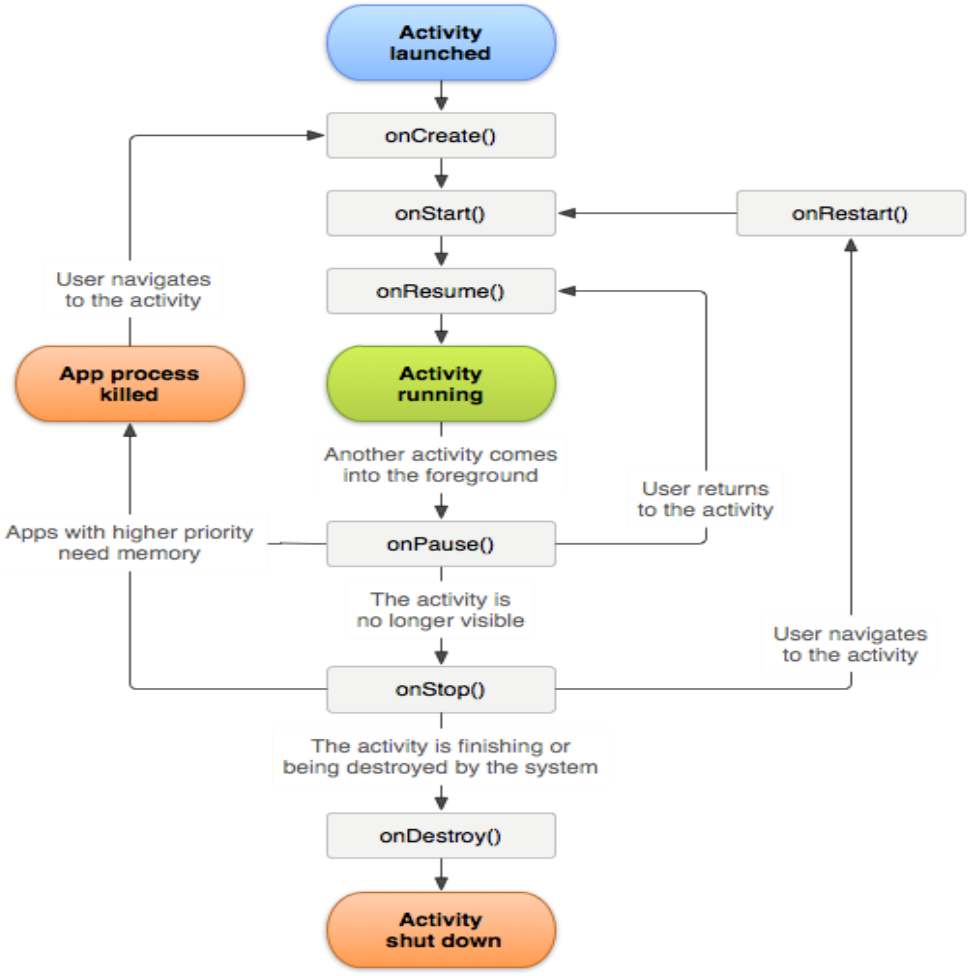
CMPS 312 Mobile Application Development

LAB 2: Activity LIFECYCLE

Objective

- Familiarize yourself with the Activity class and the Activity lifecycle,
- Create and monitor a simple application to observe multiple Activities as they move through their lifecycle.
- Learn how to handle screen orientation changes.

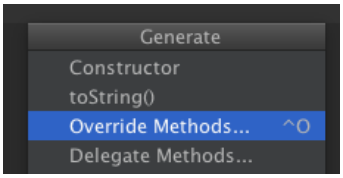
Once you've completed this lab you should understand: the Activity class, the Activity lifecycle, how to start Activity's programmatically, and the effect device orientation has on the Activity Lifecycle.

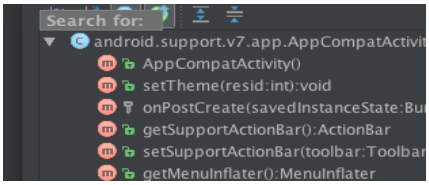


Part A
Activity Lifecycle Demo Application

1. Create an application and name it "ActivityLifecycleDemo"
2. Domain Name: cmps312.qu.edu.qa
3. Choose the (Phone Tablet) with Default Min SDK
4. Open the MainActivity under [app ➔java ➔ qa.edu.qu.cmps312.activitylifecycledemo package]
5. Override the following methods
   - protected void onStart();
   - protected void onRestart();
   - protected void onResume();
   - protected void onPause();
   - protected void onStop();
   - protected void onDestroy();

   ➔ You can use "Ctrl + N" on Windows or "Command + N" on Mac then select the "Override Methods…" and search for the above override methods.



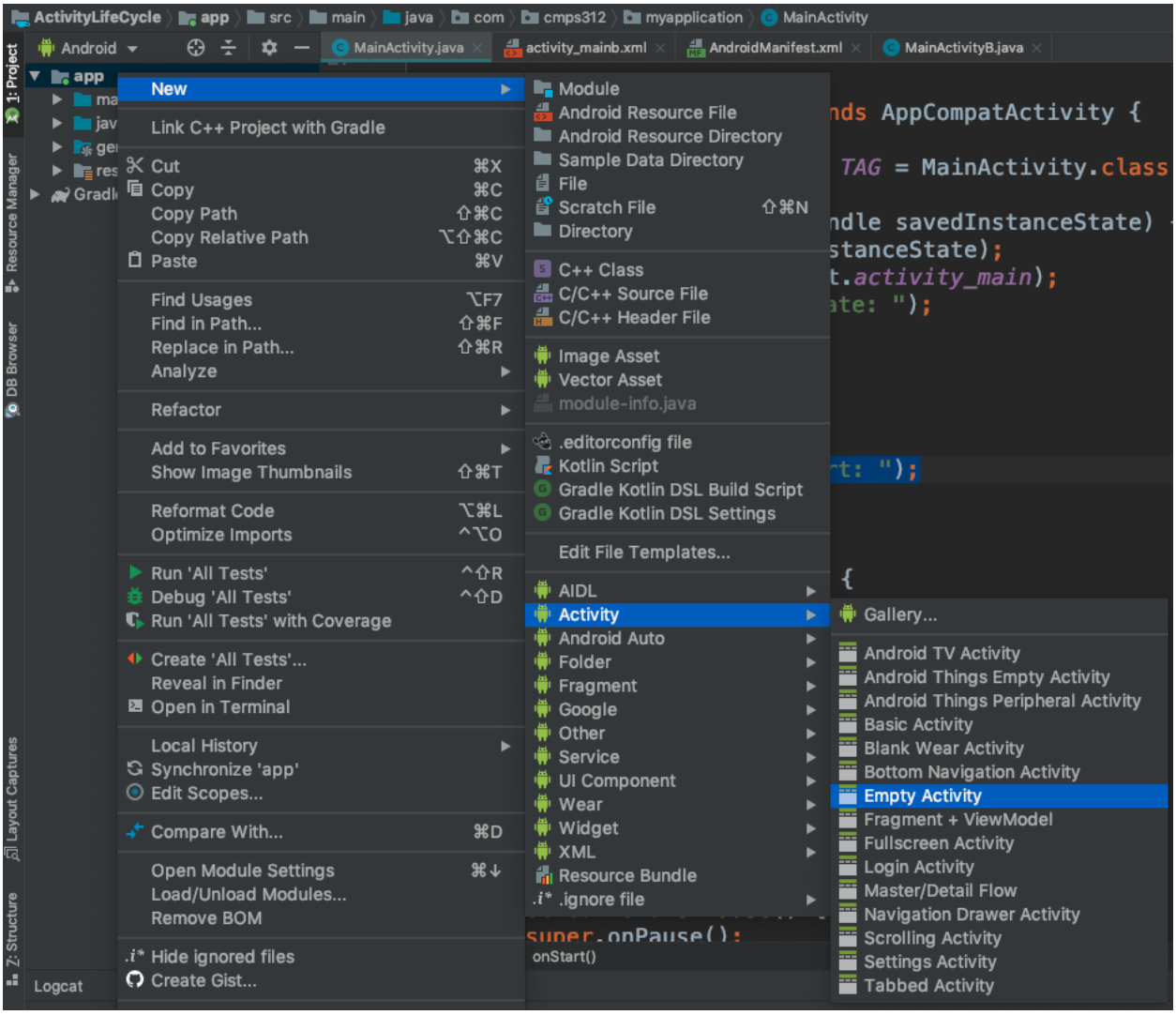


6. Inside each of the Override method add the following LOG message

   ➔ Log.d(TAG, "<INSERT NAME OF THE METHOD>") ;

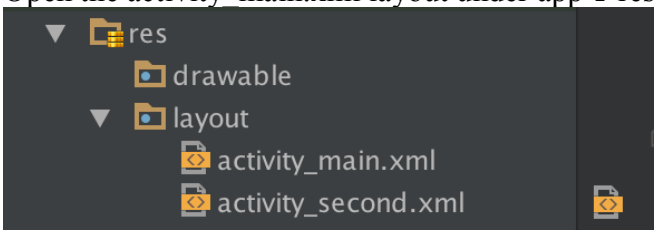
   Example: *Log.d(TAG, "onStart()");*
   Note**: Make sure to declare your* **TAG** *as Static final String initialize it using the* ***getSimpleName() *method .*

7. Now Create a second Activity and Name it "SecondActivity"
   a. Right Click on your Package and select New -> Activity-> Empty Activity as shown in the image below.

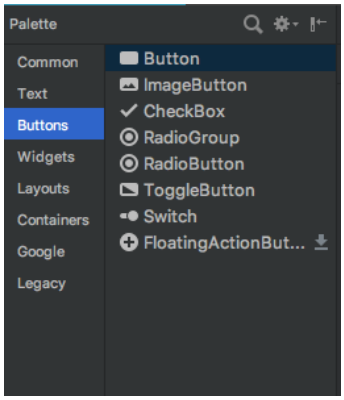


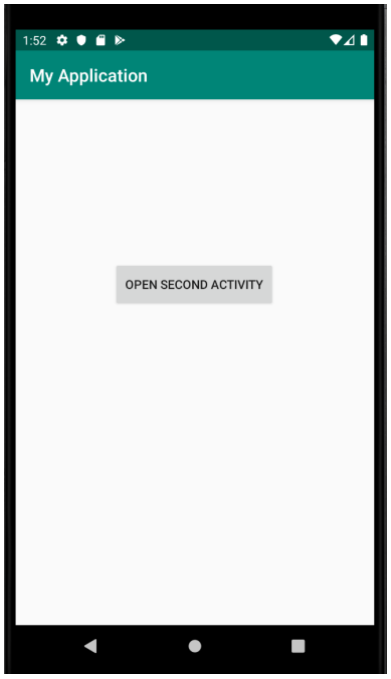
8. Open the "SecondActivity" and repeat **Steps 5 and 6** above.

9. Open the activity_main.xml layout under app➔res➔layout folder

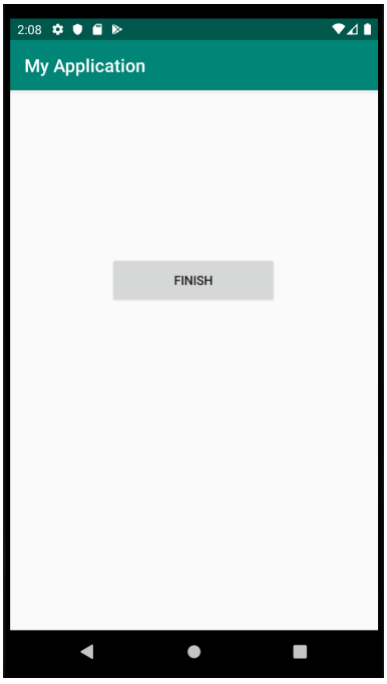


10. Add a single button to the layout by selecting the Button Widget from the Left Palette and clicking on the blank phone screen on the right window as shown in below image. Name the Button "**Go To Second Activity**".





11. Open "**activity_second**" layout and add another Button. Name the button "**FINISH**"



12. Open the "**MainActivity**" class and add the highlighted code. Make sure to import Button, Intent and View classes.

```
public class MainActivity extends Activity {

    private static final String MY_TAG = MainActivity.class.getSimpleName();

    @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(MainActivity.this, SecondActivity.class);
                startActivity(intent);

            }
        });
```

13. Open the "**SecondActivity**" class and add the highlighted code. Make sure to import Button class.

```
public class SecondActivity extends Activity {

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

        Button finishButton =(Button) findViewById(R.id.finishButton);
        finishButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

14. Save your code and run it on the Emulator
15. Now perform the following Actions and see how the Log Messages Change.
    a. Click on the "**Go To Second Activity**" Button
    b. Press the Back Finish and view the log messages
    c. Minimize the Application
    d. Close the Application
    e. Change the Orientation of the Phone by making it landscape/portrait

➔In Each of the above Actions View the log messages displayed in the IDE and see which methods are being called and when?
➔Also view the sequence in which the methods are being executed. This will help you understand the Activity Class Lifecycle.

Part B-Bundles

Bundle allow you to have a mapping from String values to various Parcelable types. Bundle is generally used for passing data between various activities of android. We usally attach them to Intents as an extra.

Bundles are generally used for two main reasons .

1. Passing data between Android Activities.

From MainActivity

```
Intent intent = new Intent(this,SecondActivity.class);
Bundle bundle = new Bundle();
bundle.putString("myValue", myValue);
intent.putExtras(bundle);
startActivity(intent);
```

In ***SecondActivity***

```
Bundle bundle = getIntent().getExtras();
act2MyValue= bundle.getString("myValue");
```

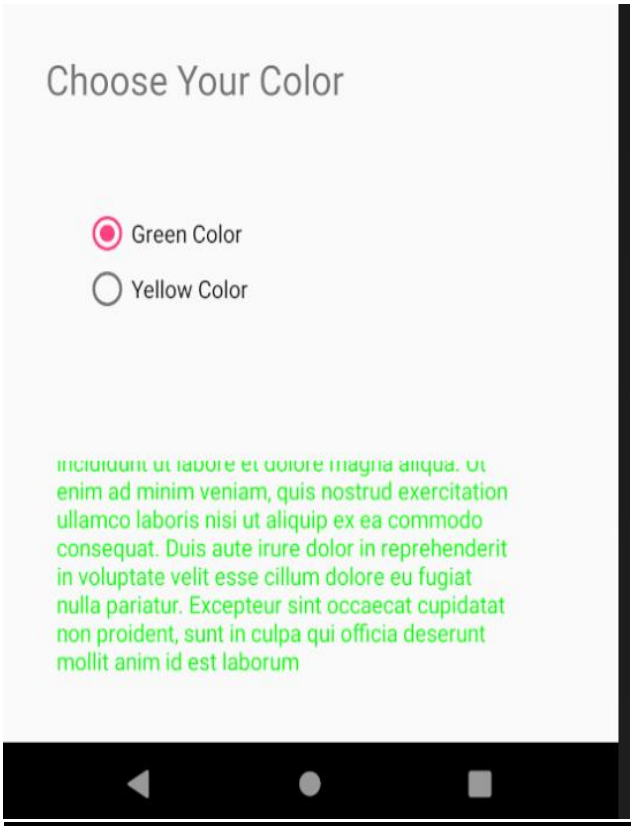
2. Restoring data from same activity.

```
public static final String USER_SCORE = "UserScore";

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

    //You can read the saved values from the Bundle
    if (savedInstanceState != null) {
        int score = savedInstanceState.getInt(USER_SCORE);
        Log.d(USER_SCORE, score + "");
    }

}

public void onSaveInstanceState(Bundle savedInstanceState) {

    //In here you can save the values inside the bundle
    savedInstanceState.putInt(USER_SCORE, 32);

}
```

Part D

Practice Exercise # 1

Create the following simple color picker application that changes the color or the text depending on the user's selection.



Practice Exercise # 2

1. Activity Lifecycle APP

The **ActivityLifeCycle** app stores the number of times each lifecycle stage of an Android Activity is triggered and displays the counts in the current Activity as in Fig 1. As your app gets in the background of another activity, or destroyed, these numbers should reflect the times the lifecycle stages are triggered.

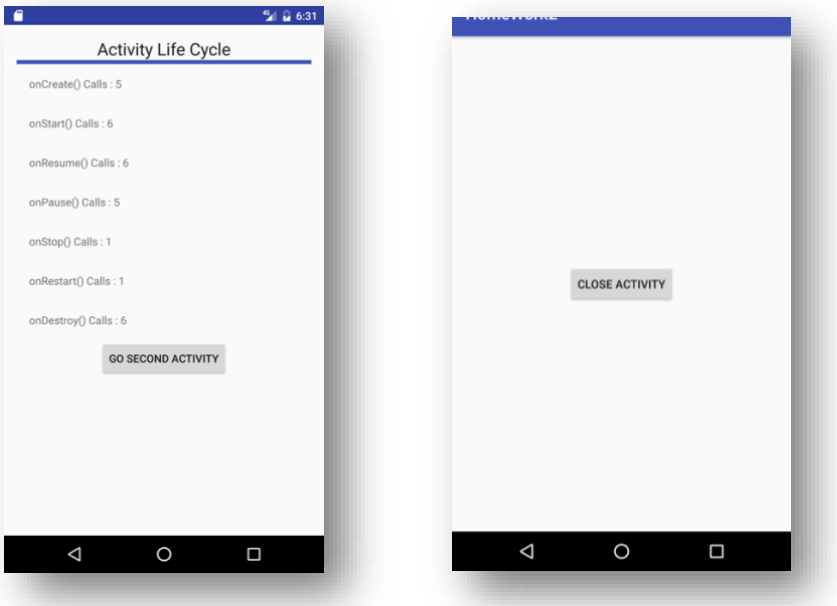


Figure 1 : Main Activity Screen

Important note: When you rotate your device (CTRL + F12 in windows or CTRL + FN + F12 in Mac in the emulator), android destroys and recreates the activity by default in Android. It is your job to make sure that these counters do not get reset when a screen rotation happens and instead, the counts should increment as if they were never reset!

You will need to use the **onSaveInstanceState()** and **onRestoreInstanceState()** (or use the bundle parameter in **onCreate())** to save and restore the counter values when a screen rotation happens, or when Android decides to destroy your activity when it is in the background due to memory constraints.

Note: You will have your first In-assessment next week. So, make sure you practice before coming to the lab by solving both the homework and the practice exercise in the Lab sheet.