COMP4336/9336 Lab 1

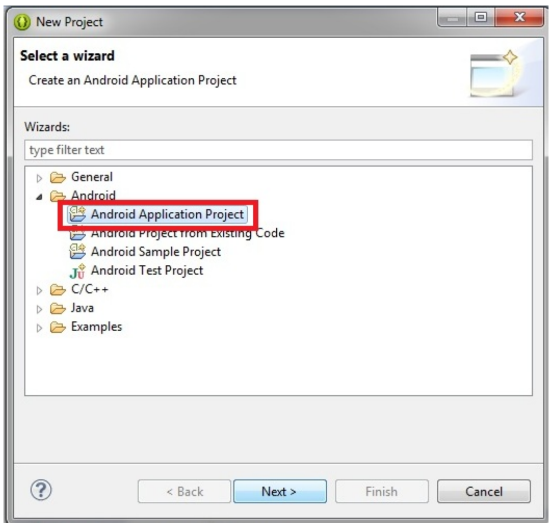
Lab Objectives

1. Learn how to run ADT and create a new project with eclipse.
2. Develop a simple Java program running on a Samsung android-based smart phone.
3. Learn how to deploy the program to both emulator and real device.
4. Learn how to copy your program to and run it on the smart phone.
5. Learn how to switch to and stop programs running on the smart phone.
6. Learn how to debug your program

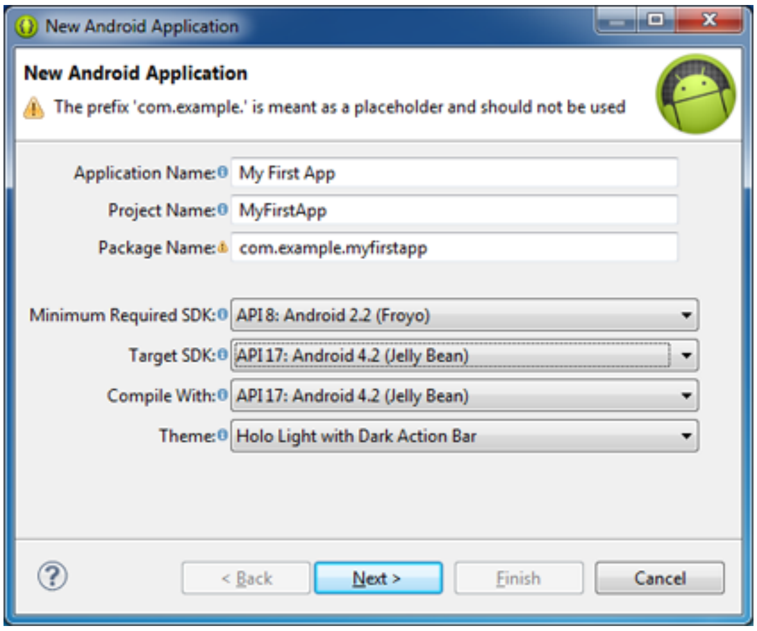
Lab Tasks

TASK1 (0.5 mark): "Hello World" Program with Eclipse

1. Connect the smart phone to your computer.
2. Run ADT by clicking on the eclipse icon.
3. Click **New** in the toolbar.



4. In the window that appears, open the **Android** folder, select **Android Application Project**, and click **Next**.


New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name:	My First App
Project Name:	MyFirstApp
Package Name:	com.example.myfirstapp
Minimum Required SDK:	API 8: Android 2.2 (Froyo)
Target SDK:	API 17: Android 4.2 (Jelly Bean)
Compile With:	API 17: Android 4.2 (Jelly Bean)
Theme:	Holo Light with Dark Action Bar

The New Android App Project wizard in Eclipse.

1. Fill in the form that appears:
   - **Application Name** is the app name that appears to users. For this project, use "My First App."
   - **Project Name** is the name of your project directory and the name visible in Eclipse.
   - **Package Name** is the package namespace for your app (following the same rules as packages in the Java programming language). Your package name must be unique across all packages installed on the Android system.

     Package name will be suggested automatically whn you select your project name.
   - **Leave other fields as their default values.**

2. On the next screen to configure the project, leave the default selections and click **Next**.
3. The next screen can help you create a launcher icon for your app.

   You can customize an icon in several ways and the tool generates an icon for all screen densities. Before you publish your app, you should be sure your icon meets the specifications defined in the Iconography design guide.

   Click **Next**.
4. Now you can select an activity template from which to begin building your app.

   For this project, select **BlankActivity** and click **Next**.
5. Leave all the details for the activity in their default state and click **Finish**.

Your Android project is now set up with some default files and you're ready to begin building the app. Continue to the next part.

If you followed the previous steps to create an Android project, it includes a default set of "Hello World" source files that allow you to immediately run the app.

How you run your app depends on two things: whether you have a real Android-powered device and whether you're using Eclipse. This tutorial shows you how to install and run your app on a real device and on the Android emulator, and in both cases with either Eclipse or the command line tools.

Before you run your app, you should be aware of a few directories and files in the Android project:

Run on a Real Device

If you have a real Android-powered device, here's how you can install and run your app:

1. Plug in your device to your development machine with a USB cable. If you're developing on Windows, you might need to install the appropriate USB driver for your device. For help installing drivers, see the OEM USB Drivers document.
2. Enable **USB debugging** on your device.
   - On most devices running Android 3.2 or older, you can find the option under **Settings > Applications > Development**.
   - On Android 4.0 and newer, it's in **Settings > Developer options**.

     Note: On Android 4.2 and newer, **Developer options** is hidden by default. To make it available, go to **Settings > About phone** and tap **Build number** seven times. Return to the previous screen to find **Developer options**.

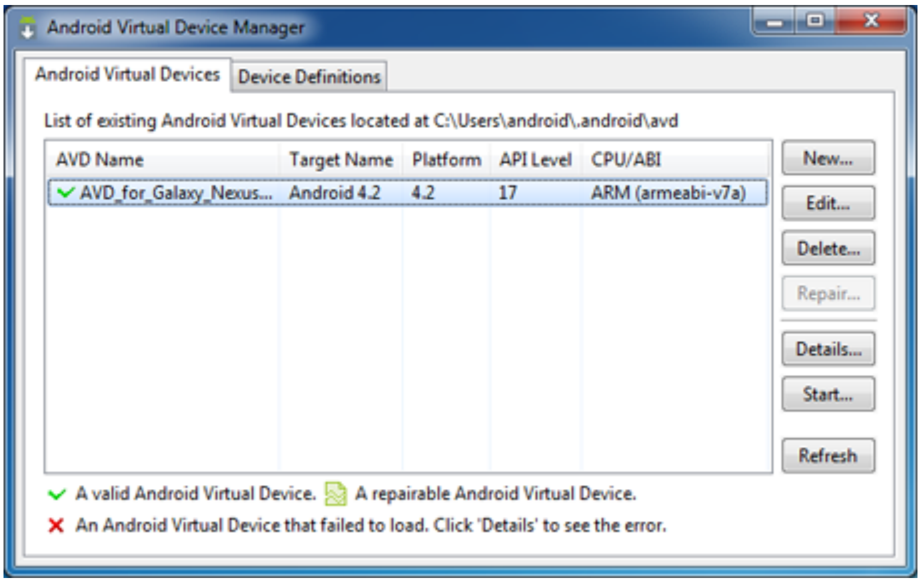
To run the app from Eclipse:

1. Open one of your project's files and click **Run** ▶ from the toolbar.
2. In the **Run as** window that appears, select **Android Application** and click **OK**.

Eclipse installs the app on your connected device and starts it.

Run on the Emulator

You need to first create an Android Virtual Device (AVD). An AVD is a device configuration for the Android emulator that allows you to model different devices.



List of existing Android Virtual Devices located at C:\Users\android\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ AVD_for_Galaxy_Nexus...	Android 4.2	4.2	17	ARM (armeabi-v7a)

✓ A valid Android Virtual Device. ▨ A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

The AVD Manager showing a few virtual devices.

To create an AVD:

1. Launch the Android Virtual Device Manager:
    a. In Eclipse, click Android Virtual Device Manager from the toolbar.
2. In the *Android Virtual Device Manager* panel, click **New**.
3. Fill in the details for the AVD. Give it a name, a platform target, an SD card size, and a skin (HVGA is default).
4. Click **Create AVD**.
5. Select the new AVD from the *Android Virtual Device Manager* and click **Start**.
6. After the emulator boots up, unlock the emulator screen.

To run the app from Eclipse:

1. Open one of your project's files and click **Run** from the toolbar.
2. In the **Run as** window that appears, select **Android Application** and click **OK**.

Eclipse installs the app on your AVD and starts it.

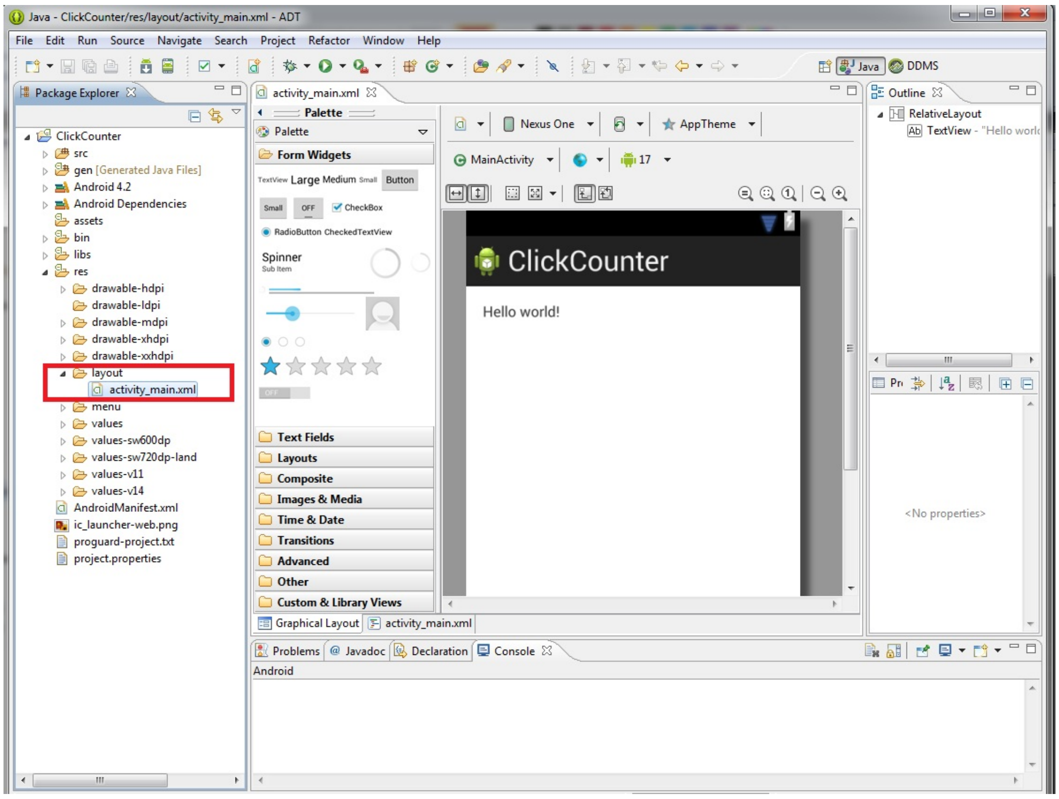
TASK2 (0.5 mark): "Click Counter" Program

AIMS:

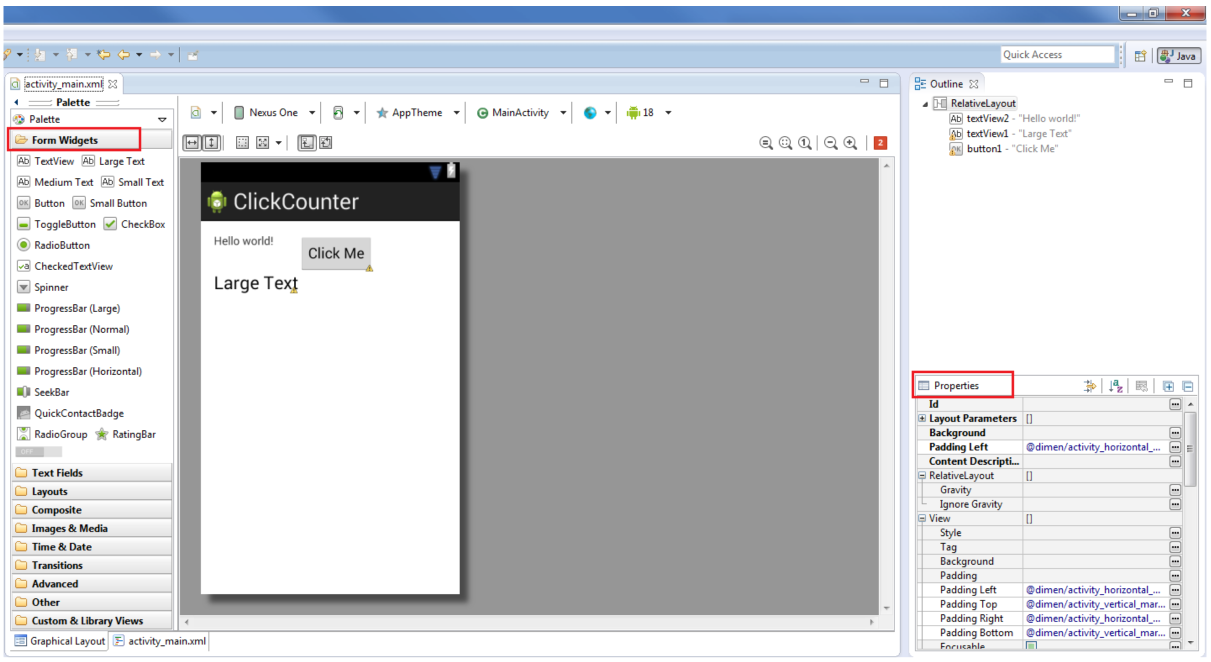
1. Study the code of your *HellowWorld* program.
2. Develop the *Click Counter* program so that it records and shows the number of times *Click Me* button has been clicked

Instruction:

1. Create a new Android Application Project named *ClickCounter*. Leave all options as their default values except the project name.
2. Go to the layout designer by double click on res\layout \activity_main.xml such as below picture:



3. Now, you can design the graphical layout of your project by drag-and-drop the component from "**Form Widgets**". By adding a Button and a Large Text, create the following layout.
   - You can change the Text of your button using Properties window in the right side of your IDE such as you can see in the picture.



Id	
Layout Parameters	[]
Background	
Padding Left	@dimen/activity_horizontal_...
Content Descripti...	
RelativeLayout	[]
Gravity	
Ignore Gravity	
View	[]
Style	
Tag	
Background	
Padding	
Padding Left	@dimen/activity_horizontal_...
Padding Top	@dimen/activity_vertical_mar...
Padding Right	@dimen/activity_horizontal_...
Padding Bottom	@dimen/activity_vertical_mar...
Focusable	

- By clicking on each object within the graphical layout (such as the button), you can change the details of the object in Properties window. These properties are stored in an XML file which is named "activity_main.xml" in this project.
- You don't need to edit the xml file directly.
- For each graphical object, you need to know the id name which is necessary for manipulating its properties from your Java classes.

4. In order to manage click activity for the button in Android, you can do as below:
   - Open MainActivity class from the src folder in Package Explorer window (at the left side of your IDE)
   - Then, declare your button in **onCreate** method of the class like

```
final Button btn = (Button) findViewById(R.id.button1);
```

   - Then use the button variable as below:

```
btn.setOnClickListener(new View.OnClickListener() {

  public void onClick(View v) {
    myClick(v); /* my method to call new intent or activity */
  }
});
```

   - As an example, you can handle the click event as :

```
public void myClick(View v) {
  TextView txCounter = (TextView) findViewById(R.id.textView1);
  txCounter.setText("yourtext");
}
```

- Now, the following is the source code of class MainActivity for implementing our simple *ClickCounter* program. You should complete the **myClick** method by your own code.
  Note: As this is the first lab, the below code has been provided for you but for next lab exercises you should be able to create the project and manage all methods and codes by yourself.

```
package com.example.clickcounter;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // set default counter for the text box

      Button clickMeBtn = (Button) findViewById(R.id.button1);
        clickMeBtn.setOnClickListener(new
View.OnClickListener() {
            public void onClick(View v) {
                myClick(v); /* my method to call new intent or
activity */
            }
        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar
if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    public void myClick(View v) {
```