

Lab – Android Development Environment

Setting up the ADT, Creating, Running and Debugging Your First Application

Objectives:

Familiarize yourself with the Android Development Environment

Important Note: This class has many students with a wide range of previous experience. Some students are fairly new to object-oriented (OO) programming. Some have OO programming experience, but are new to Android. Still others know some Android already, and want to just freshen up their knowledge.

Because of this, I'm not expecting that everyone can finish this entire lab. I suggest that you set a time limit for yourself, say 1 hour. Work through what you can in that time and then stop and take a break. If you later feel that you have some more time for this Lab, then repeat the process. Again – don't feel that you need to finish everything in this lab. That's not the goal here.

Specifically, if you are fairly new to programming, you should try to complete Parts 1 – 4 below. If you are familiar with programming and programming environments, you should try to complete parts 1 – 6 below. If you're an experienced programmer and reasonably comfortable with Java, try to do the whole thing.

This lab contains the following Parts.

1. Set up the Android SDK.
2. Create a new Android application.
3. Create an Android Virtual Device and start the Android Emulator.
4. Run the application you created in Part 2.
5. Import an application project.
6. Debug an Android application.

Additional helpful information can be found on the Android Developer website:

- <http://developer.android.com/sdk/installing/bundle.html>
- <http://developer.android.com/training/basics/firstapp/creating-project.html>
- <http://developer.android.com/tools/devices/managing-avds.html>
- <http://developer.android.com/training/basics/firstapp/running-app.html>

Part 1 – Setting Up The SDK.

In this part you will download and install the Android SDK and start the Eclipse Integrated Development Environment (IDE). Note that Android Studio is now the recommended IDE for Android. Therefore, we're including this section for backward compatibility only.

To use Eclipse you will need to have a copy of Eclipse. You will then install a custom Eclipse plugin called Android Developer Tools (ADT). See the following URL for installation instructions.

<https://developer.android.com/sdk/installing/installing-adt.html>

Advanced – Using Hardware Acceleration. Assuming it is compatible with your development machine, you may want to install the Intel Hardware Accelerated Execution Manager (HAXM). Installing HAXM will greatly speed up the emulator, making your development, testing and debugging much more productive.

See the following links for more information:

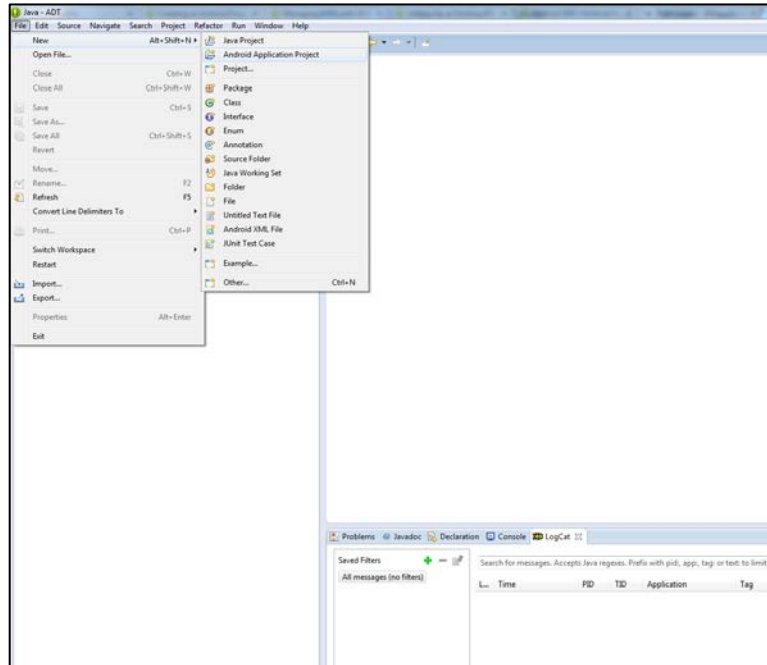
<http://developer.android.com/tools/devices/emulator.html#accel-vm>

for a complete description.

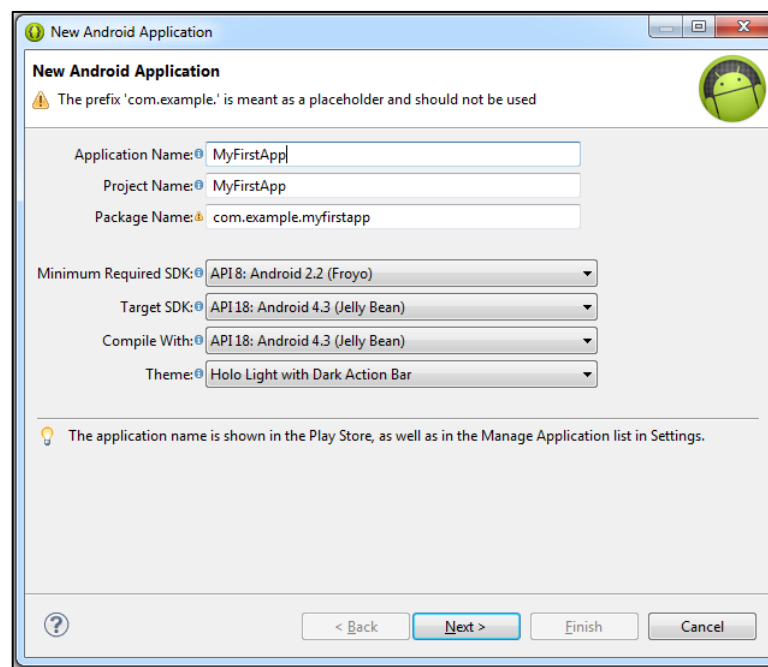
Part 2 – Creating A New Project

In this part you will create a simple Android application that displays the words, "Hello World!"

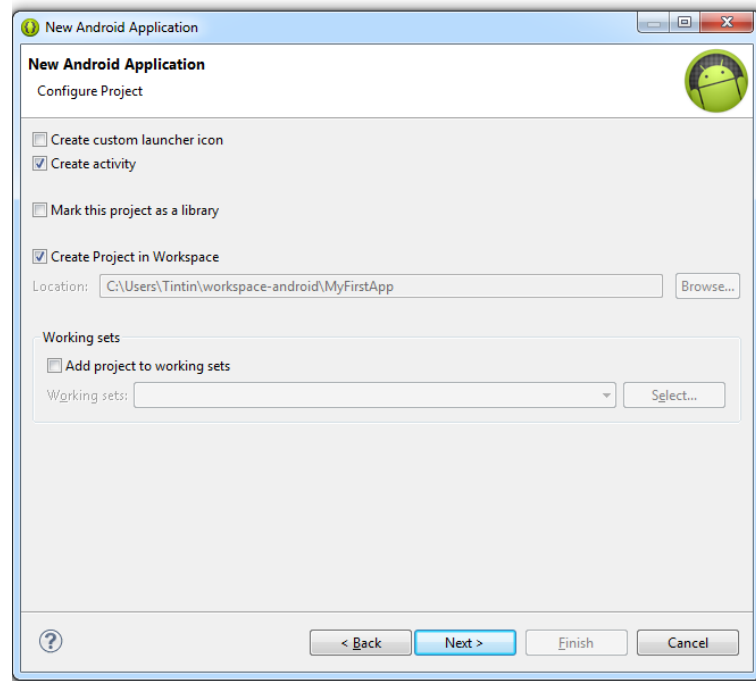
1. From the application menu bar, select File > New > Android Application Project



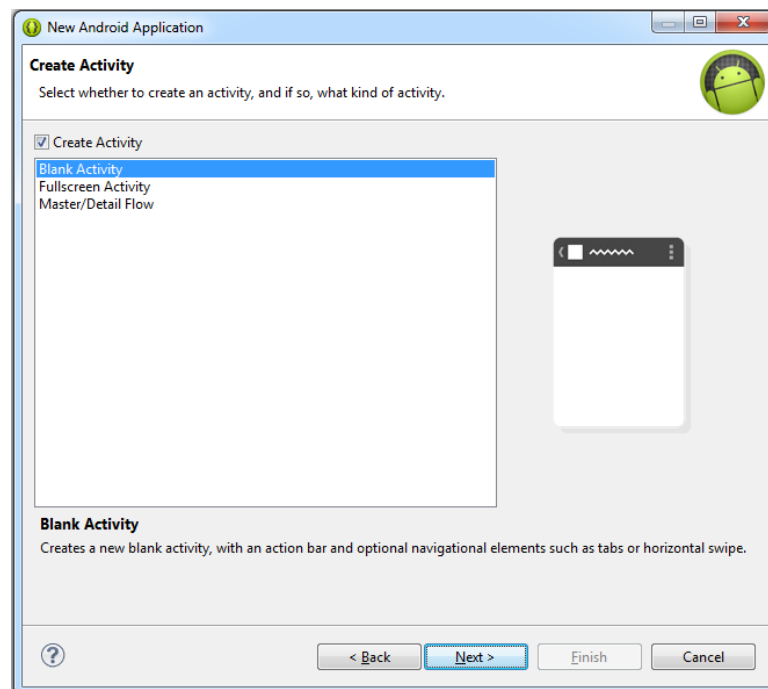
2. Next, a dialog box will appear, allowing you to enter an Application Name. Enter the string, "MyFirstApp." The Project Name and Package Name fields will fill in automatically. Leave the remaining settings alone and then click Next.



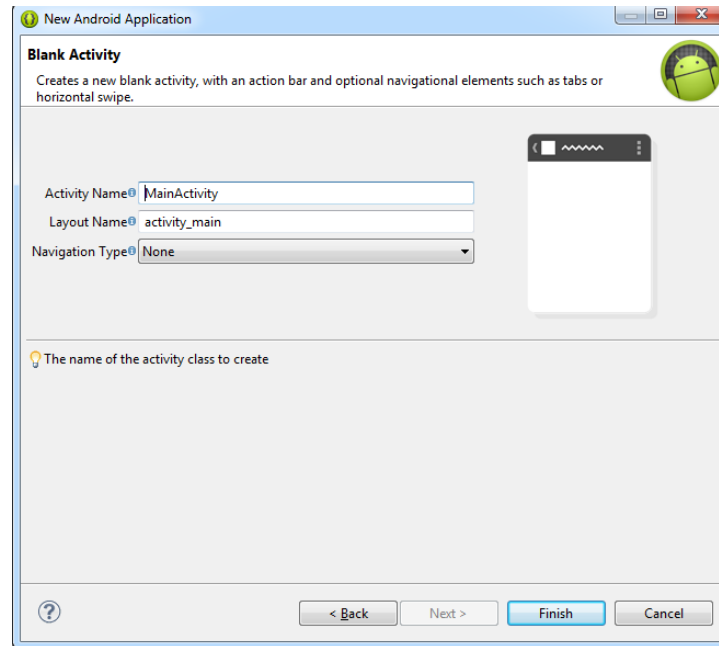
- Next, a new dialog box will appear. Deselect "Create custom launcher icon," as we will not be creating a custom icon for this app. Then click Next.



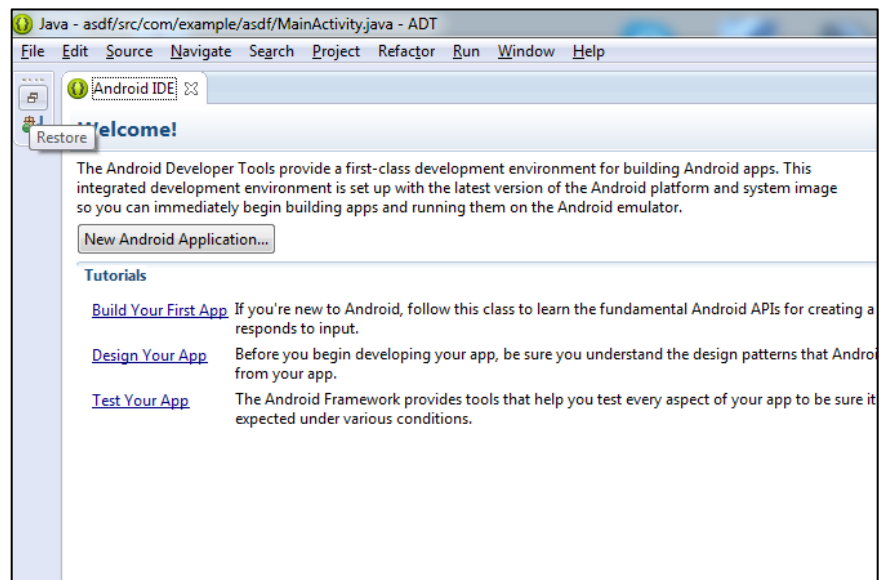
- In the next window, leave all the settings as default. Then click Next.



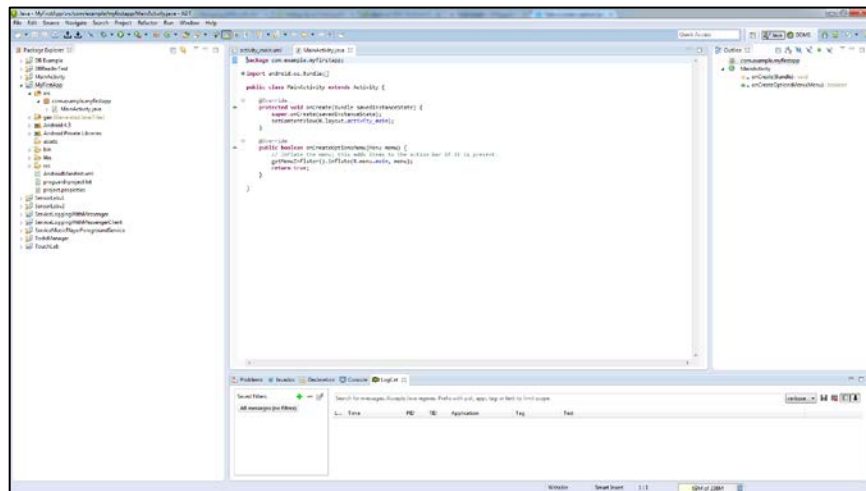
- For the next window, also leave all default settings in place and press Finish.



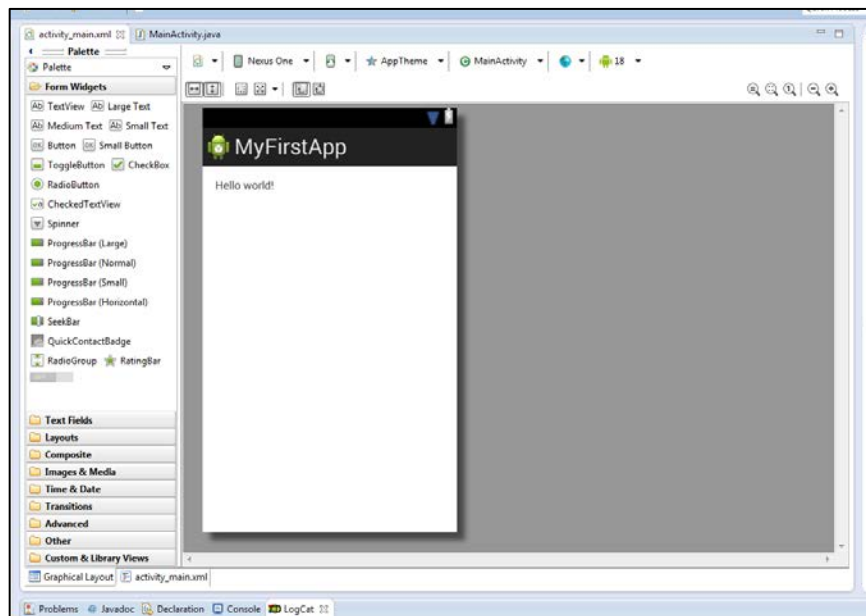
- Next, Find and press the restore icon on the left to bring the main editing window up.



- At this point, your screen should look similar to the one below. Open the MainActivity.java file to see the code that sets up the Main Activity of your new app.



- Back in the Package Explorer under res/layout/ there is the activity_main.xml file, which defines the layout of your Main Activity. If you click on the Graphical Layout tab, you can see words, "Hello world!" appearing on the Activity's user interface.

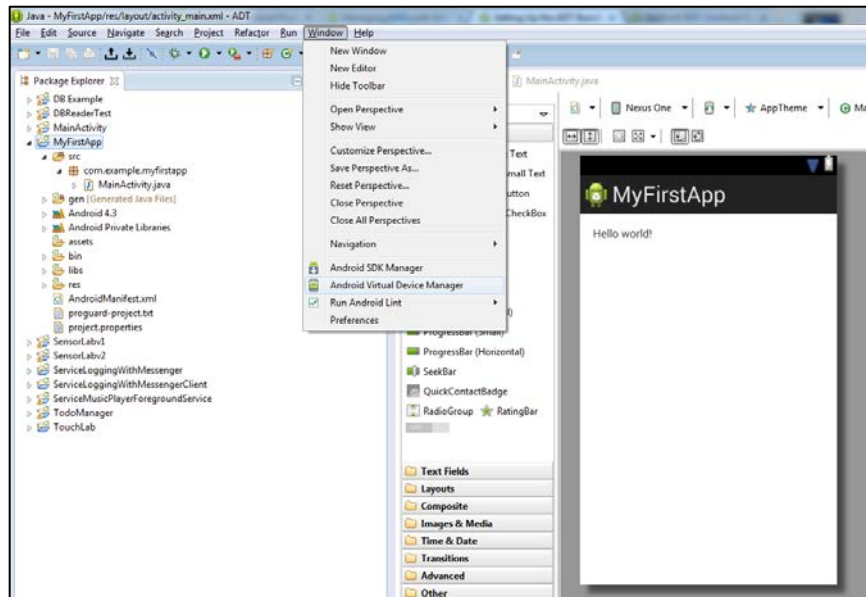


In Part 4 we will show you how to run this app in the Android Emulator.

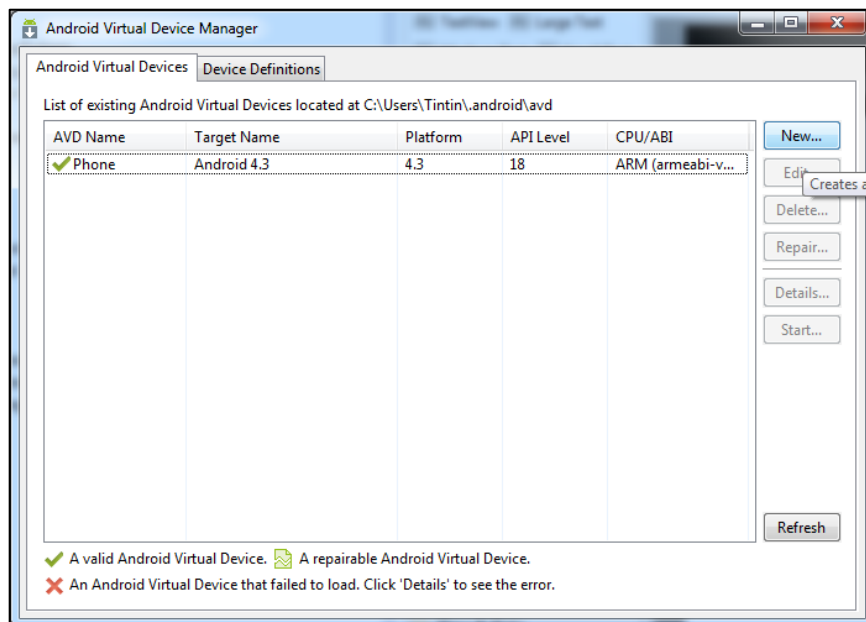
Part 3 – Using the Emulator

In this part you will learn how to set up and use the Android Emulator.

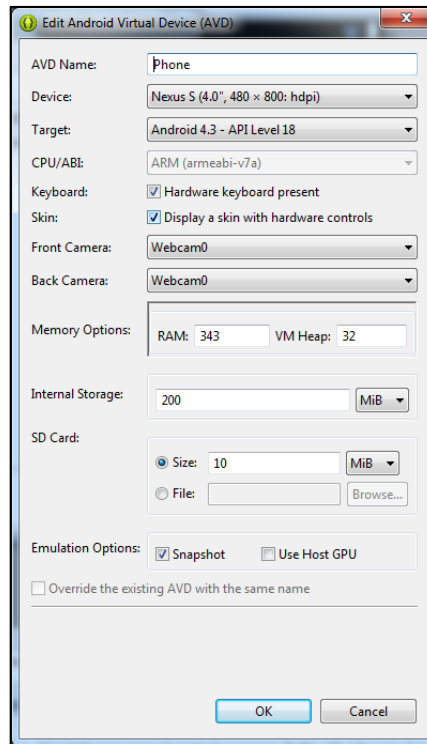
1. First start up the Android Virtual Device Manager. You can do that by selecting Window > Android Virtual Device Manager from the Eclipse menu bar.



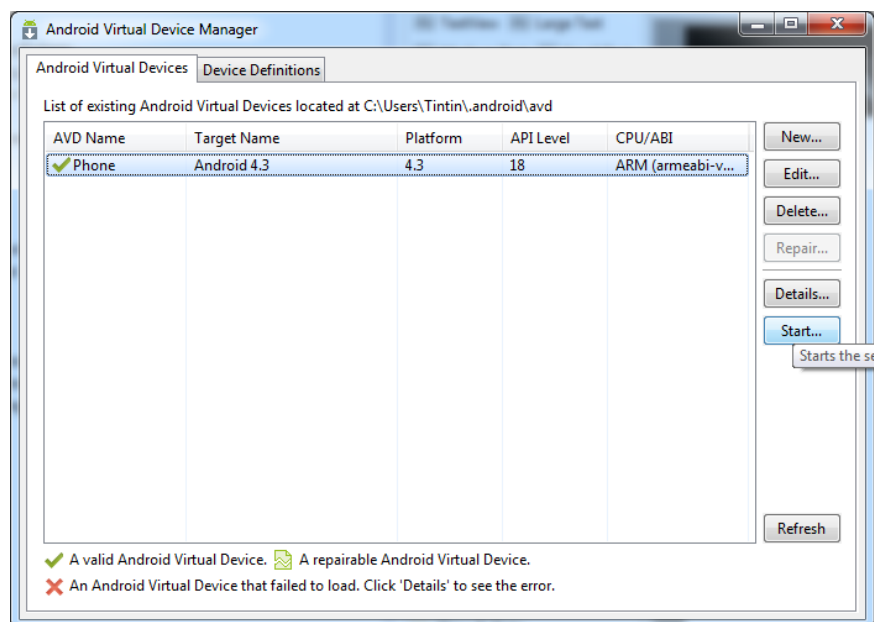
2. A new dialog box will pop up. Click "New" to create a new Android Virtual Device (AVD).



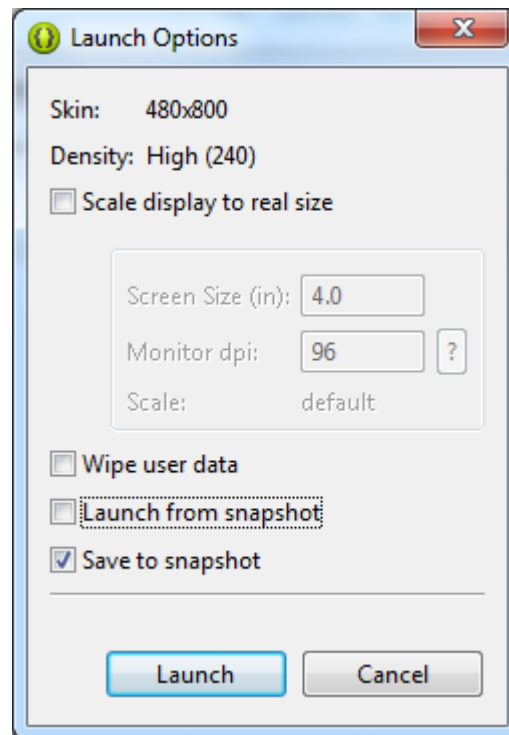
- Another dialog box will pop up displaying various AVD parameters. You can play with these parameters to create different virtual devices. Press "Ok" to finish.



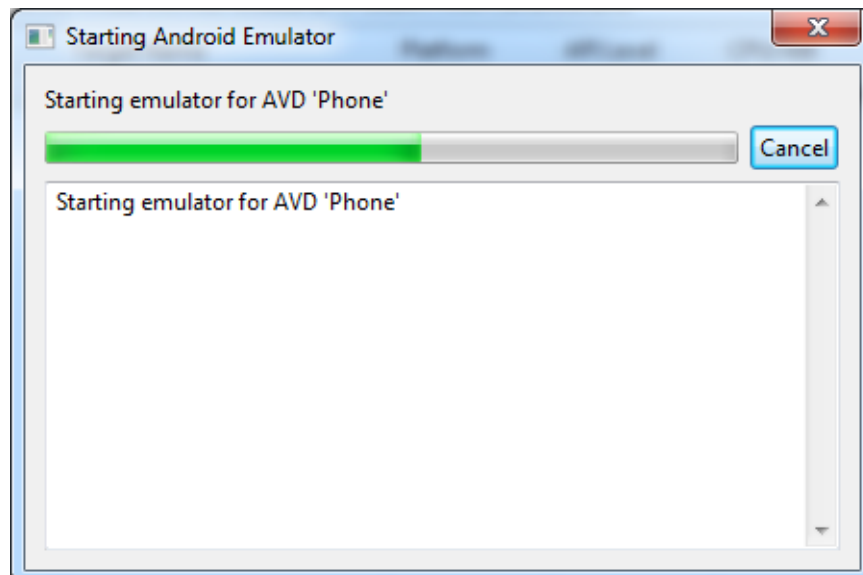
- A new AVD will now be listed in the AVD Manager window. Select it and then press the Start button.



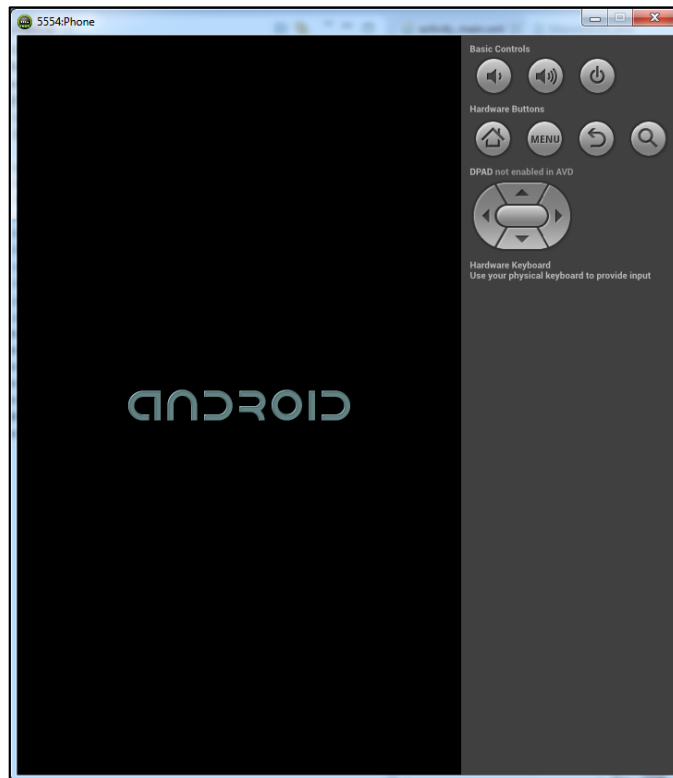
5. A new dialog box will pop up. Press the Launch button to start the emulator.



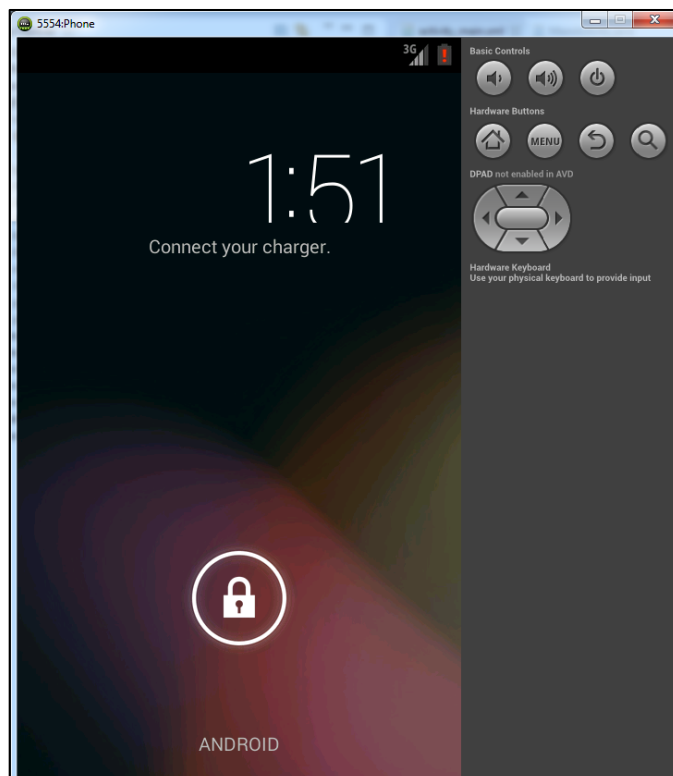
6. As the emulator starts up, you will see another dialog box. Launch errors will be displayed here.



- Next, The emulator will appear and start its boot sequence.



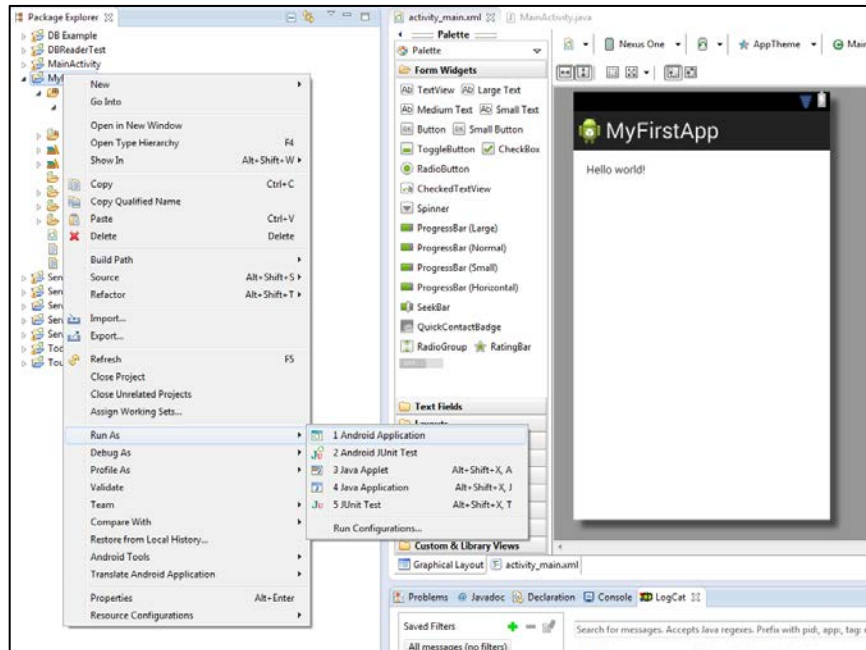
- After the device has booted, the emulator will be ready for user interaction.



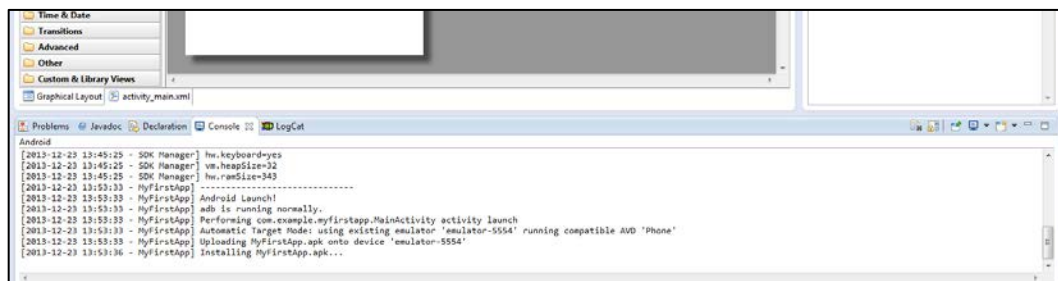
Part 4 – Running Your First App

In this part you will learn how to run the application you created in Part 2 in the Android Emulator.

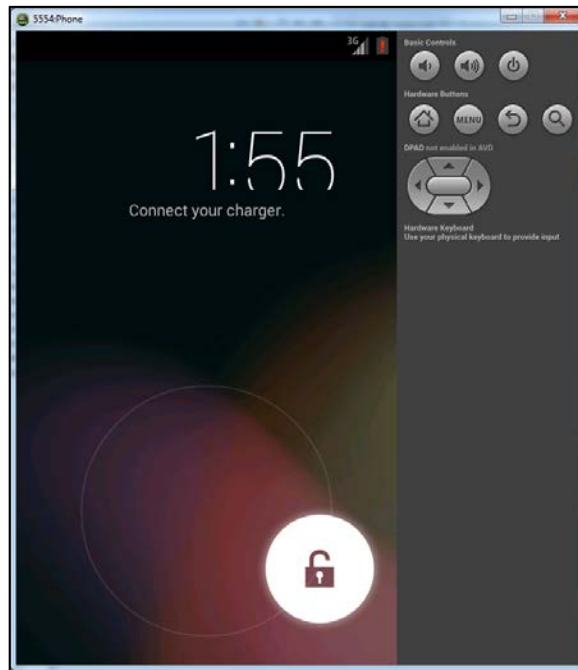
1. Return to Eclipse Package Explorer and right-click your app's project name. From there select Run As > Android Application.



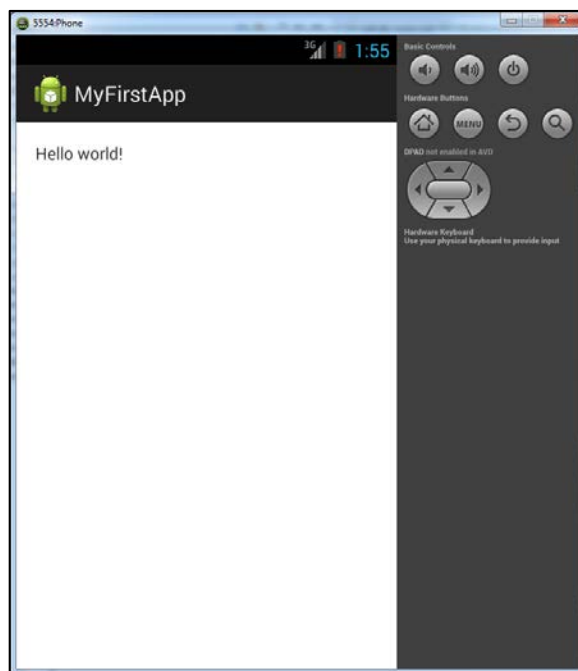
2. In the Console panel, below the editing window, you will see output indicating that the application is being loaded onto the Android Emulator.



3. Return to your Emulator instance. If necessary, drag the lock icon to unlock your device.



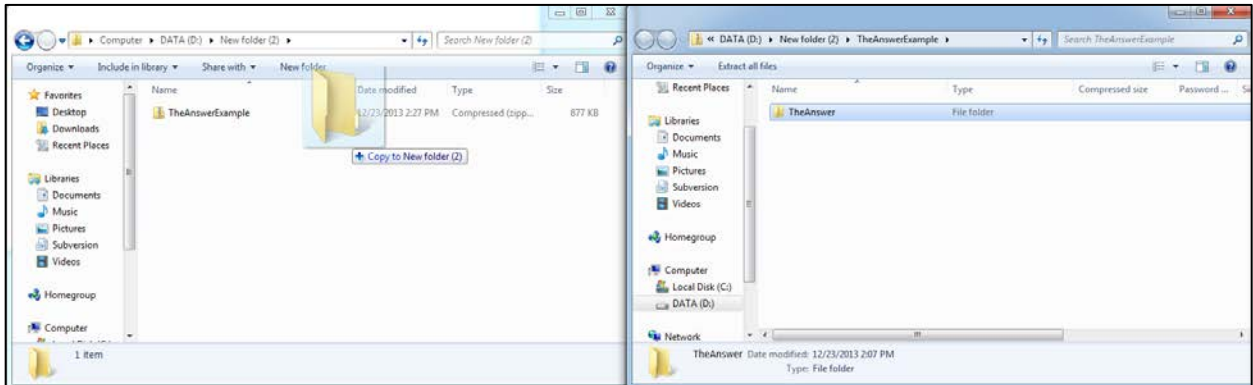
4. You should now see your application, running in the Android Emulator.



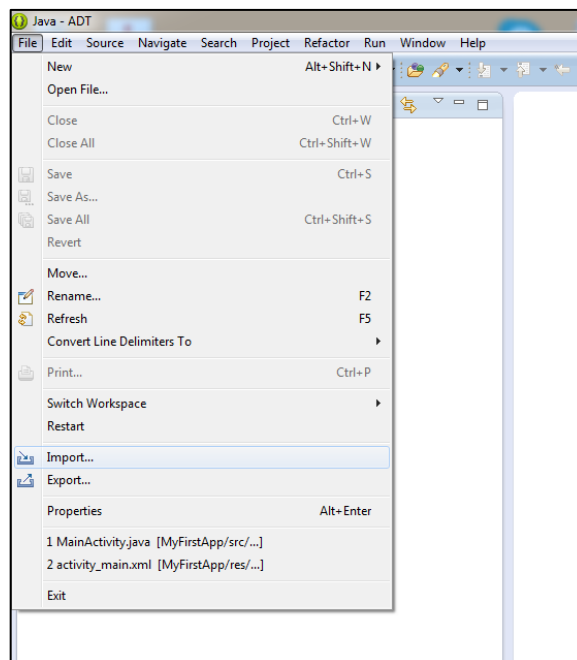
Part 5 – Importing and Running an Existing Application

In this part you'll learn to import a pre-existing application into Eclipse and then run it.

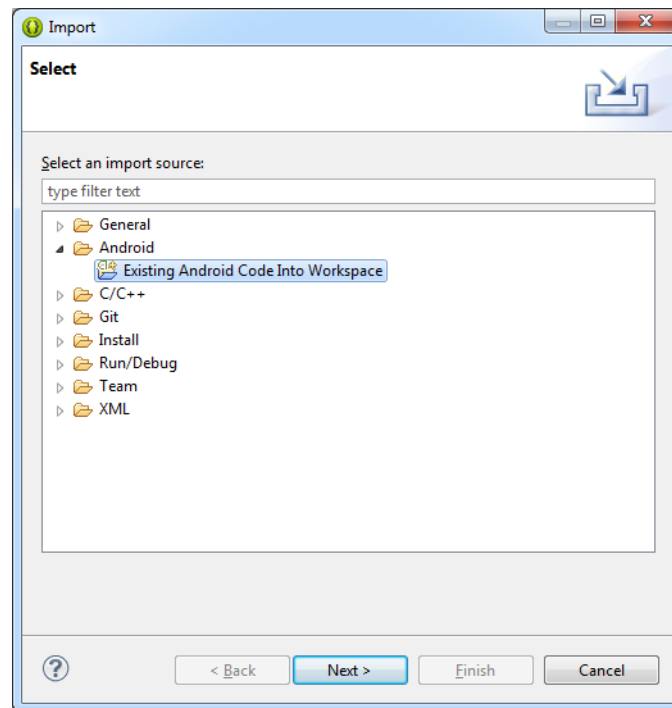
1. Download the TheAnswer application from the course source code repository.
2. Extract the contents of the App to any folder. The contents should contain a sample project contained within a folder called TheAnswer.



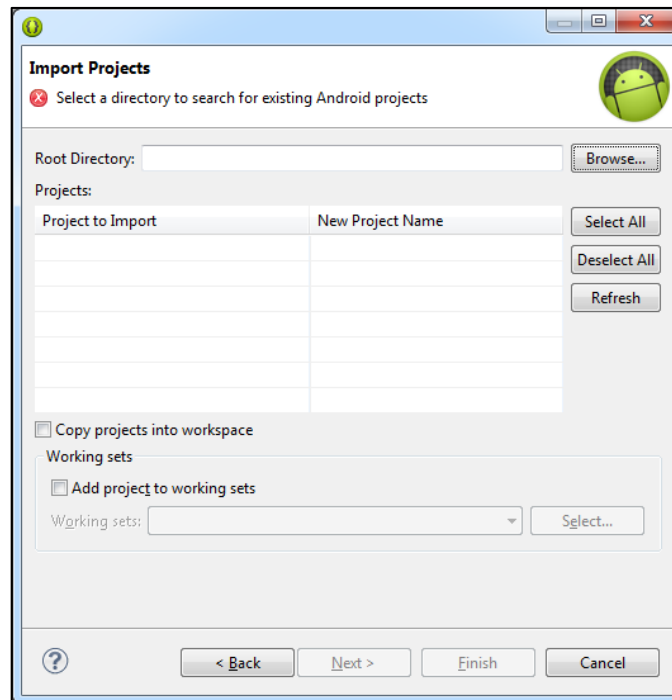
3. Return to Eclipse. Select File > Import from the menu bar to import the application project.



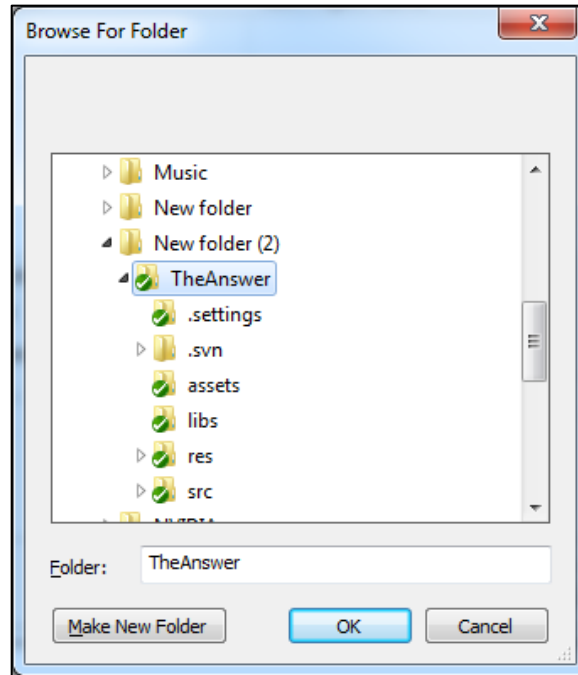
- Next, in the dialog box that appears, select Android > Existing Android Code Into Workspace, and then press the Next Button.



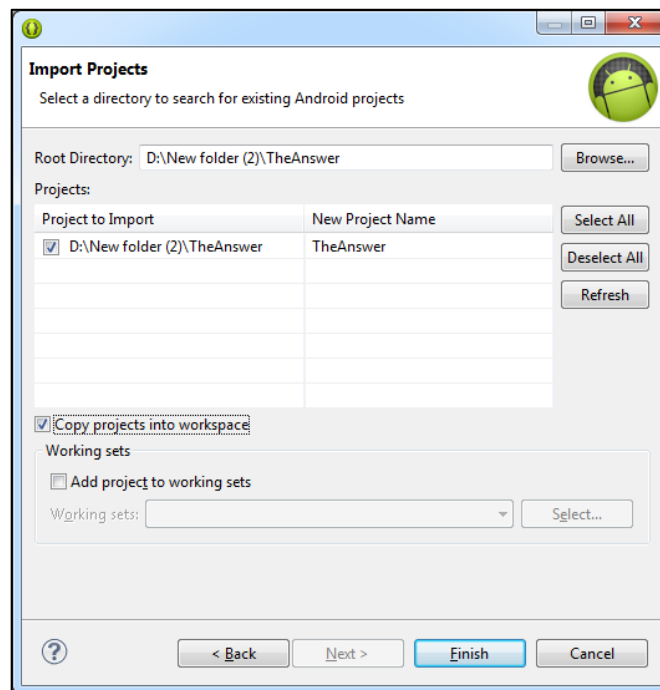
- Next, you'll see a dialog box, allowing you to select the project to import. Press the Browse... Button.



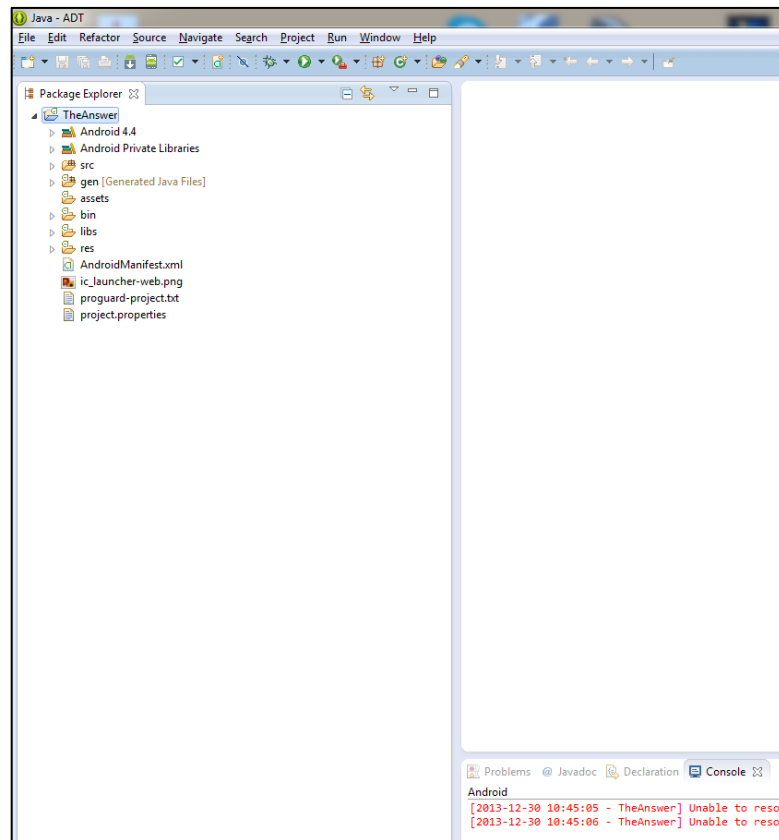
6. Select the location of the Example Application contents, and press OK.



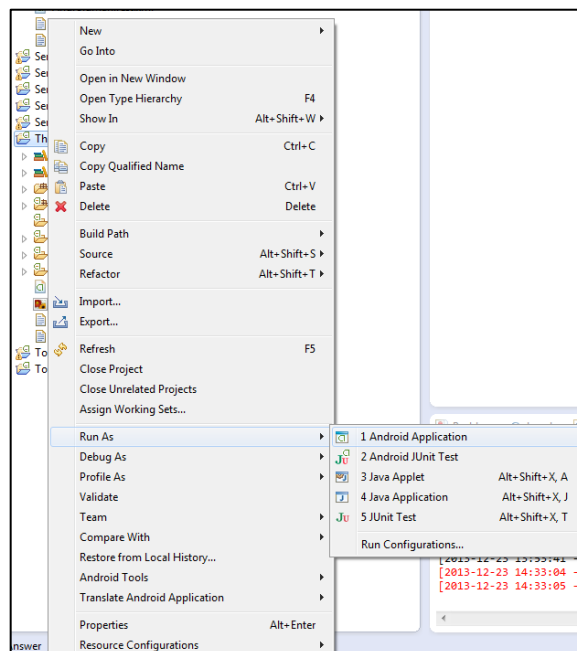
7. In the next dialog box, make sure that you select "Copy projects into workspace" and then click the Finish Button.



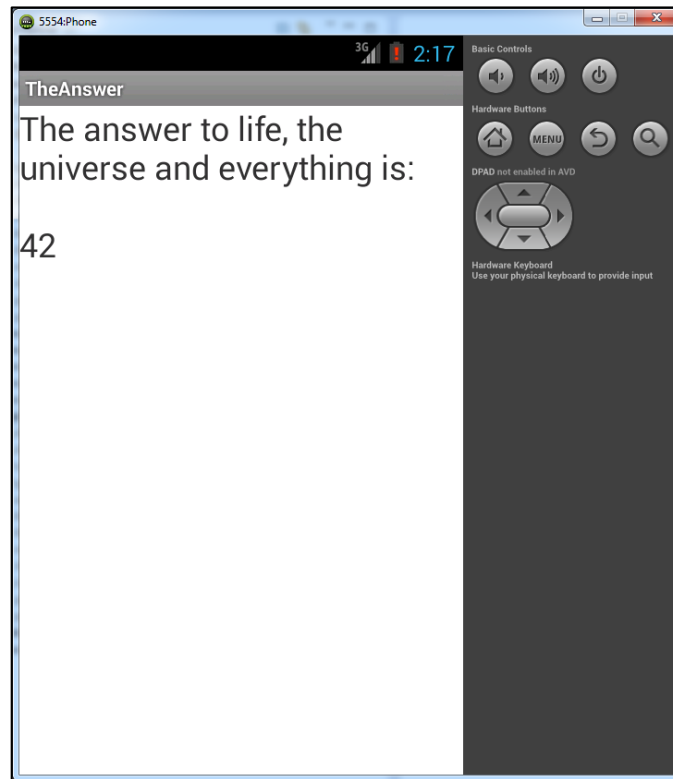
8. At this point the application should appear in the Project Explorer on the left side of the IDE.



9. Right-click the folder go to Run As > Android Application



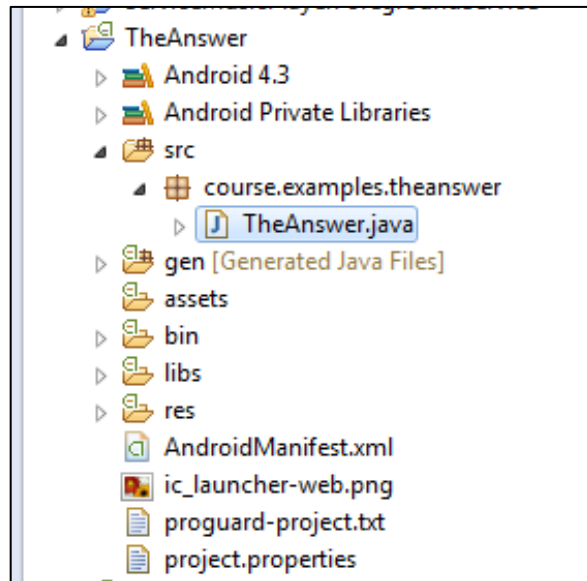
10. The Android Emulator will now open up and run the example application.



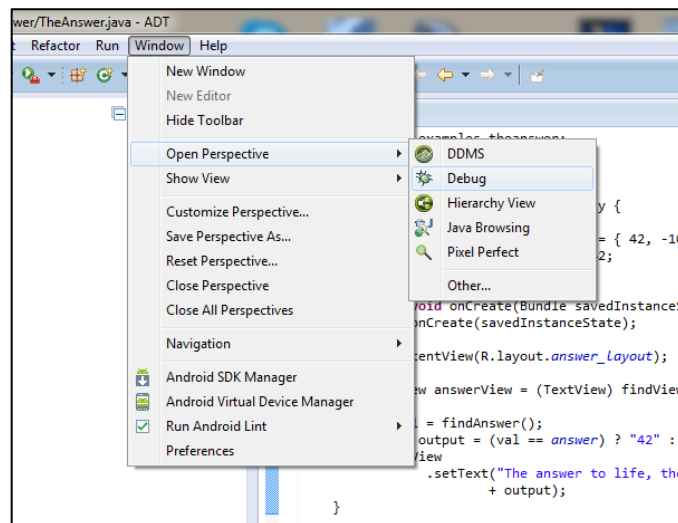
Part 6 – Debugging

In this part of the lab you will learn how to use the Eclipse debugger to debug the TheAnswer application you imported in Part 5.

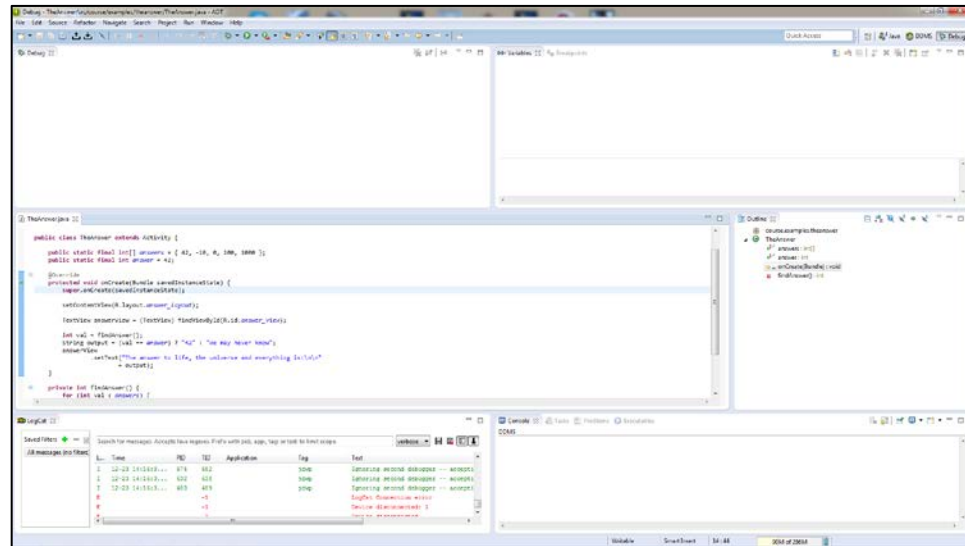
1. Double-click the TheAnswer.java file under src > course.examples.theanswer




2. Next, open the Debugging Perspective by selecting Window > Open Perspective > Debug from the menu bar.



3. A new perspective will appear, similar to that shown below.



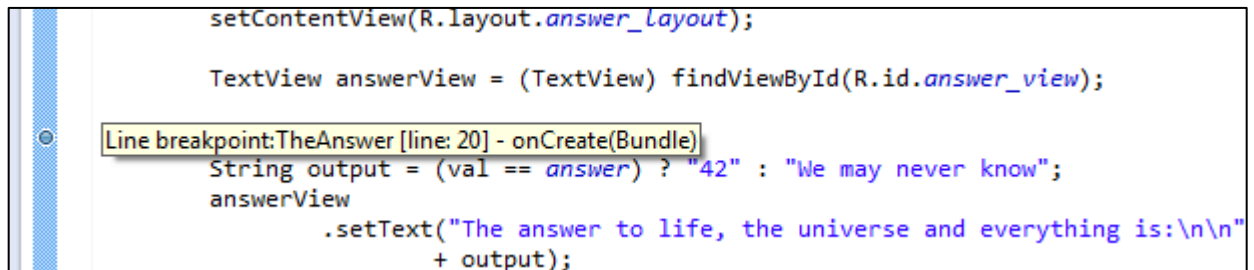
4. On this screen, double click the highlighted blue area next to the line:
"int val = findAnswer();" 

```
setContentView(R.layout.answer_layout);

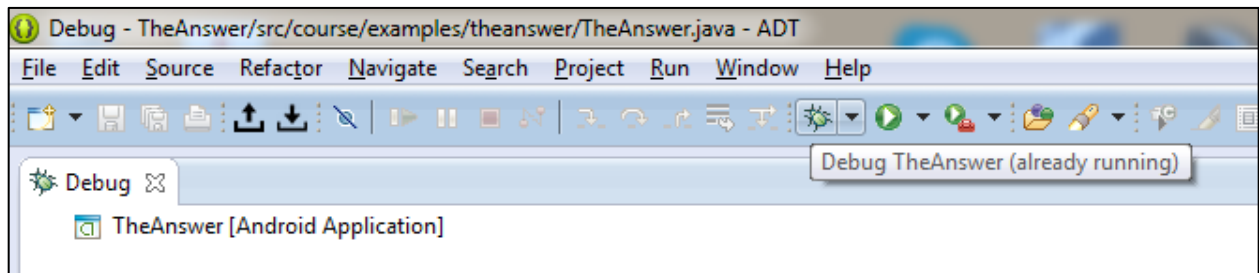
TextView answerView = (TextView) findViewById(R.id.answer_view);

int val = findAnswer();
String output = (val == answer) ? "42" : "We may never know";
answerView
    .setText("The answer to life, the universe and everything is:\n\n"
        + output);
```

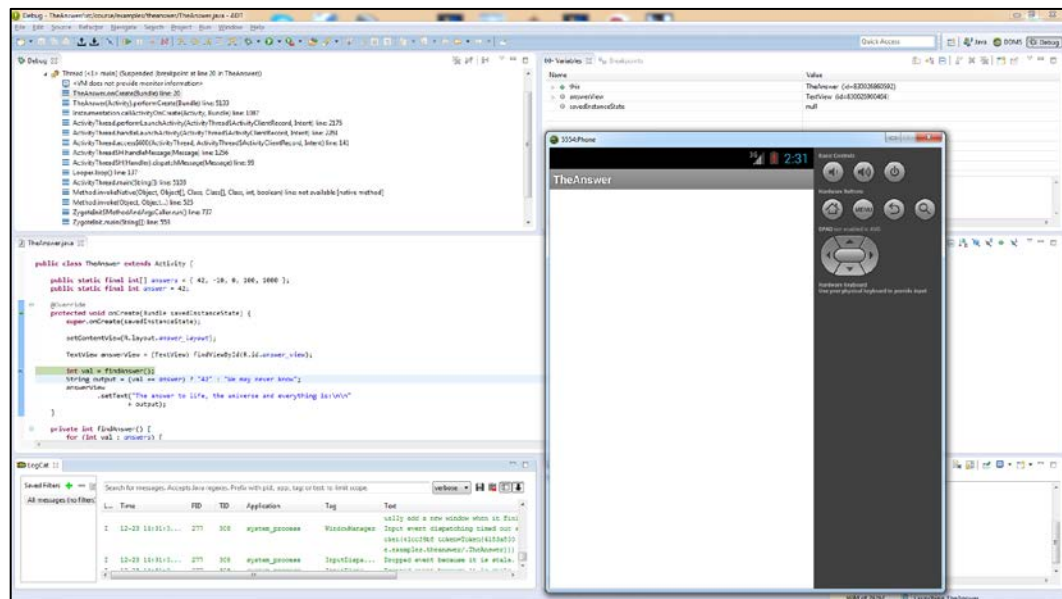
5. A new breakpoint will be placed at that line, indicated by the small circle that now appears in the highlighted blue area to the left of the text.



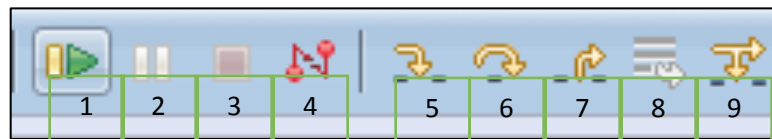
6. Next, press the Debug icon at the top of the window in the menu bar to start debugging the application.



7. Your Emulator should have loaded the App and stopped before the words, "The answer to life....." , is displayed on the screen.



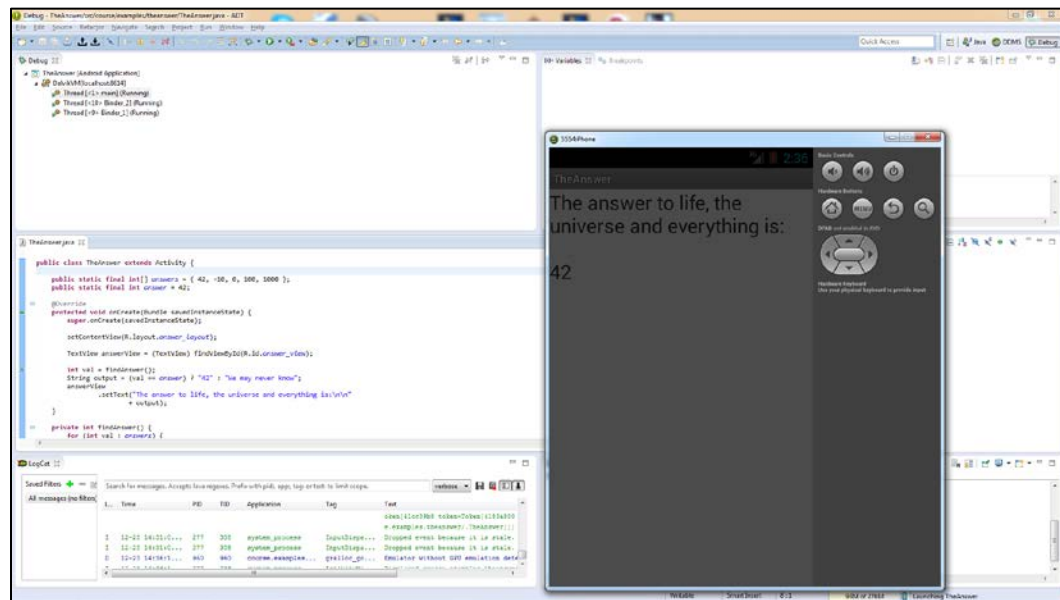
8. Now that the app is stopped, you can examine the app's state and step through the app's execution using the following buttons appearing in the menu bar.



Buttons from left to right on the navigation bar:

- 1 - Resume
- 2 – Suspend
- 3 – Terminate
- 4 – Disconnect debugger from emulator
- 5 – Step into
- 6 – Step over
- 7 – Step return
- 8 – Drop to frame
- 9 – Use step filters

9. Next, press the Resume icon to continue executing the app. The app will finish loading and will display the text.

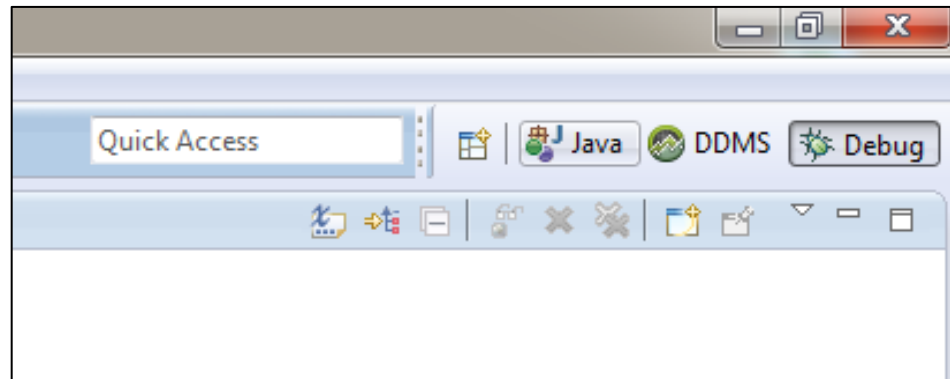


10. The next debugging task will have you create and display informational messages to the LogCat panel, to help you better understand the application's runtime behavior. To generate these messages, you will use methods in the android.util.Log class. You will also need to import this class into your application. Some LogCat functions include:

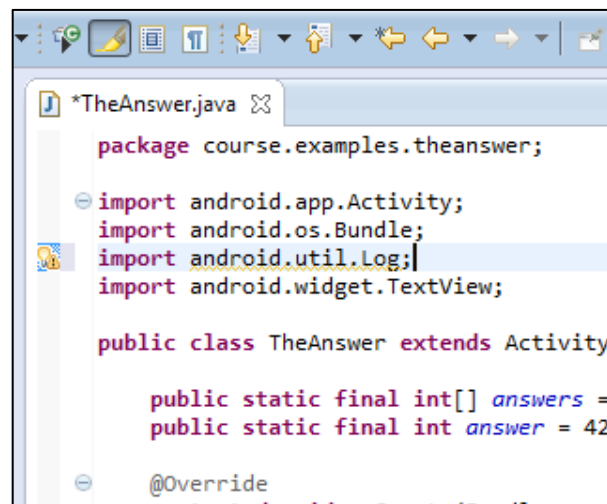
- 1 – Log.i(..., ...) – Sends an INFO LogCat message
- 2 – Log.d(..., ...) – Sends a DEBUG LogCat message
- 3 – Log.e(..., ...) – Sends an ERROR LogCat message
- 4 – Log.v(..., ...) – Sends a VERBOSE LogCat message

See <http://developer.android.com/reference/android/util/Log.html> for more information.

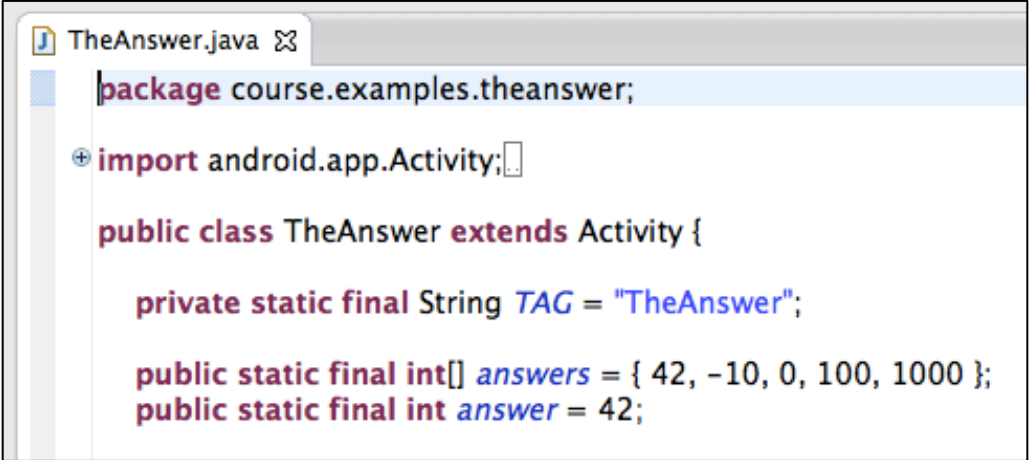
11. Return to the Java Editing Perspective by clicking the Java Icon in the top-right corner.



12. Import the android.util.Log library by typing, "import android.util.Log;" near the beginning of the code for TheAnswer.java. You can also use the Organize Imports function (Ctrl + Shift + O is the shortcut).



13. The Log class' methods require a string called a Tag, which identifies the creator of the message and can be used to sort and filter the messages when they are displayed. Create a constant called TAG within the TheAnswer class, by typing, for example, "private static final String TAG = "TheAnswer";"



```
TheAnswer.java
package course.examples.theanswer;

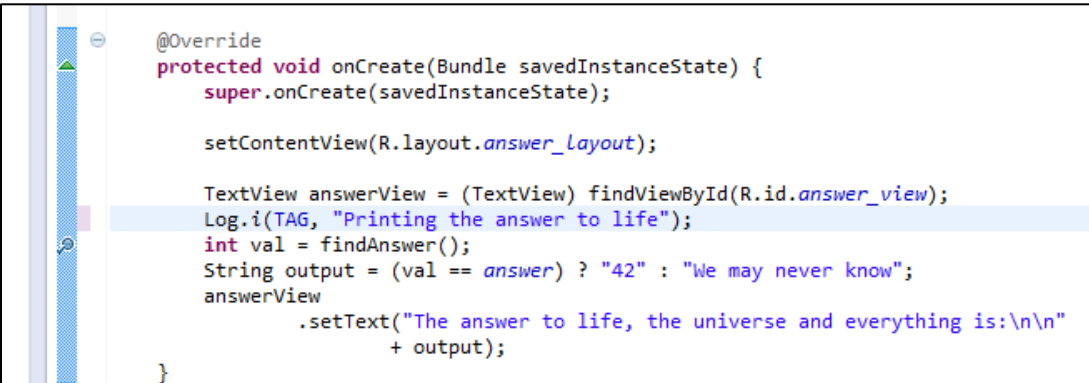
import android.app.Activity;

public class TheAnswer extends Activity {

    private static final String TAG = "TheAnswer";

    public static final int[] answers = { 42, -10, 0, 100, 1000 };
    public static final int answer = 42;
```

14. Use the Log.i() function to create and output a log message. Just before the line that starts, "int val = ..." type in a new line: "Log.i(TAG, "Printing the answer to life");"

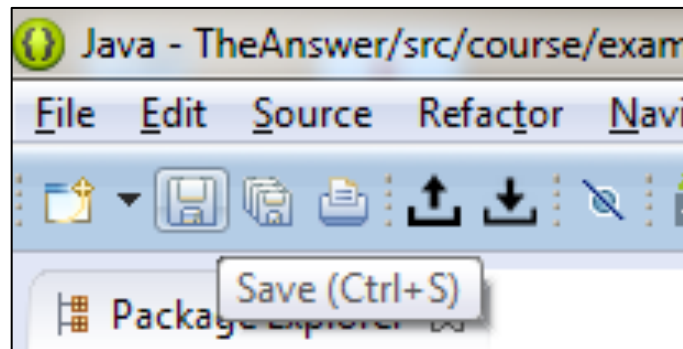


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

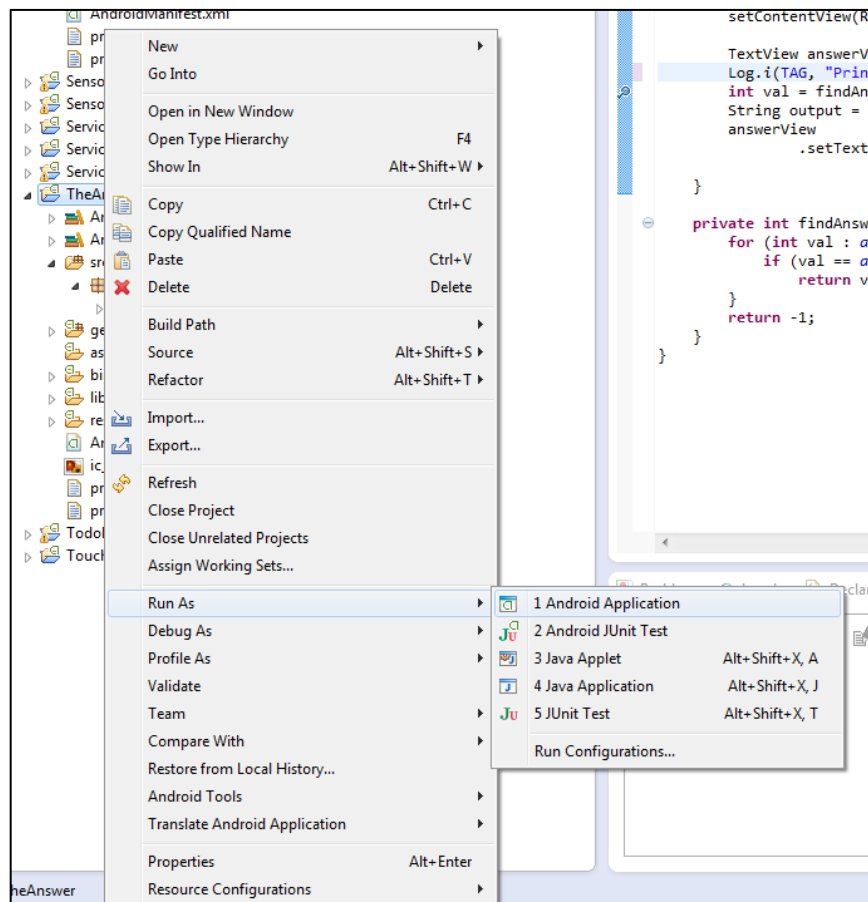
    setContentView(R.layout.answer_layout);

    TextView answerView = (TextView) findViewById(R.id.answer_view);
    Log.i(TAG, "Printing the answer to life");
    int val = findAnswer();
    String output = (val == answer) ? "42" : "We may never know";
    answerView
        .setText("The answer to life, the universe and everything is:\n\n"
            + output);
}
```


15. Save your changes.



16. Run the application.



17. Once the app is running, open the LogCat panel at the bottom of the Java Perspective. Look for the search box, enter, "tag:TheAnswer" and hit Return. You will now see the log message from the TheAnswer application in the LogCat panel.

