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.
7. Further explore the IDE

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).
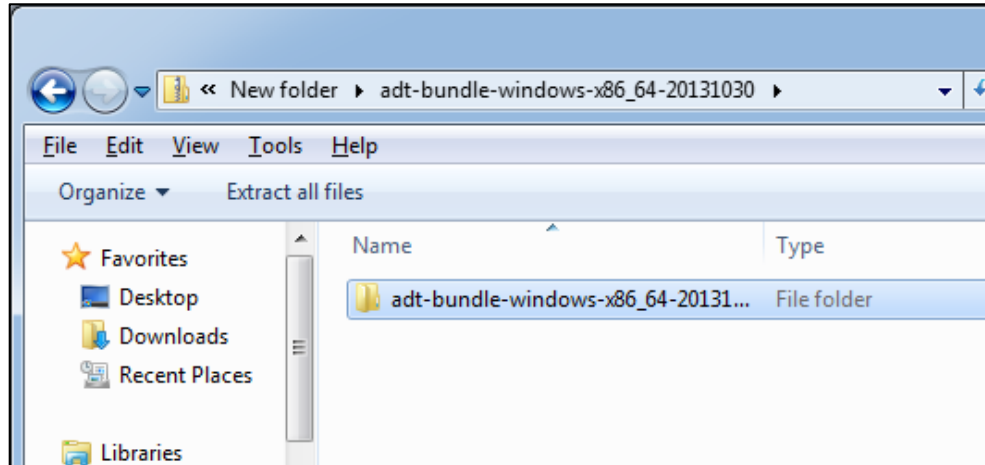
1. Download the SDK from http://developer.android.com/sdk/index.html. Remember where you save the downloaded file.
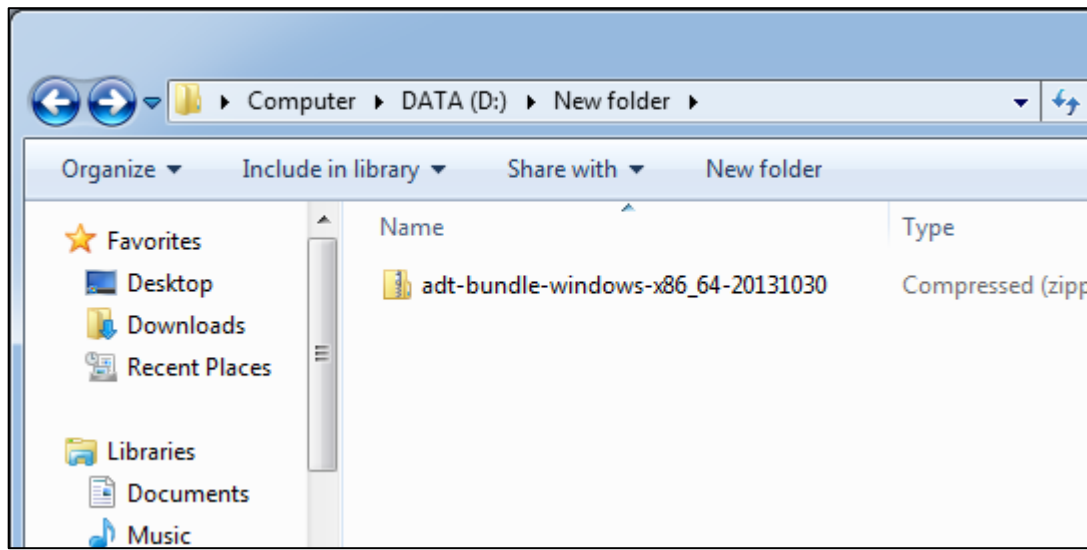
2. Double-click on the zip file you just downloaded to extract the zip file's contents.
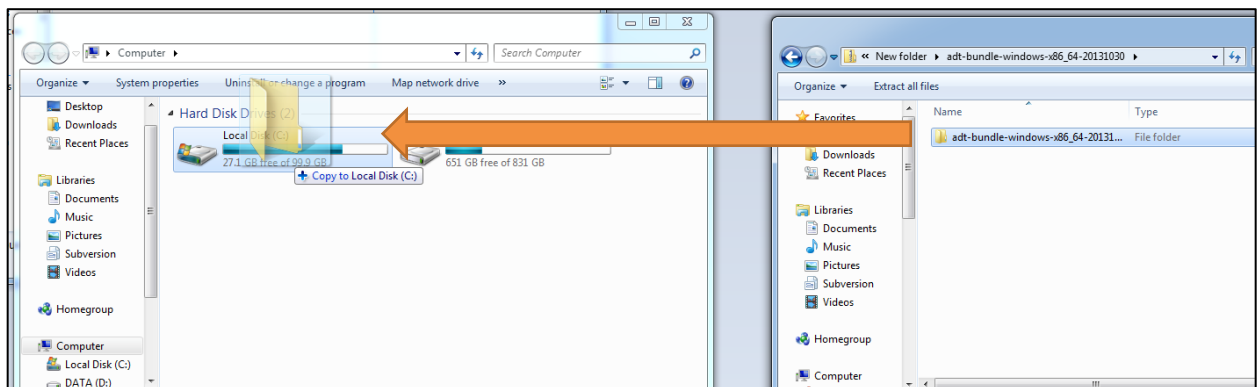
3. (Optional) Using Windows Explorer, drag and drop the ADT Bundle folder to any location you designate.


Assuming you have the Java Development Kit installed on your computer, you should now be able to use the Android SDK. If you do not have the Development Kit installed, see: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html for more information.
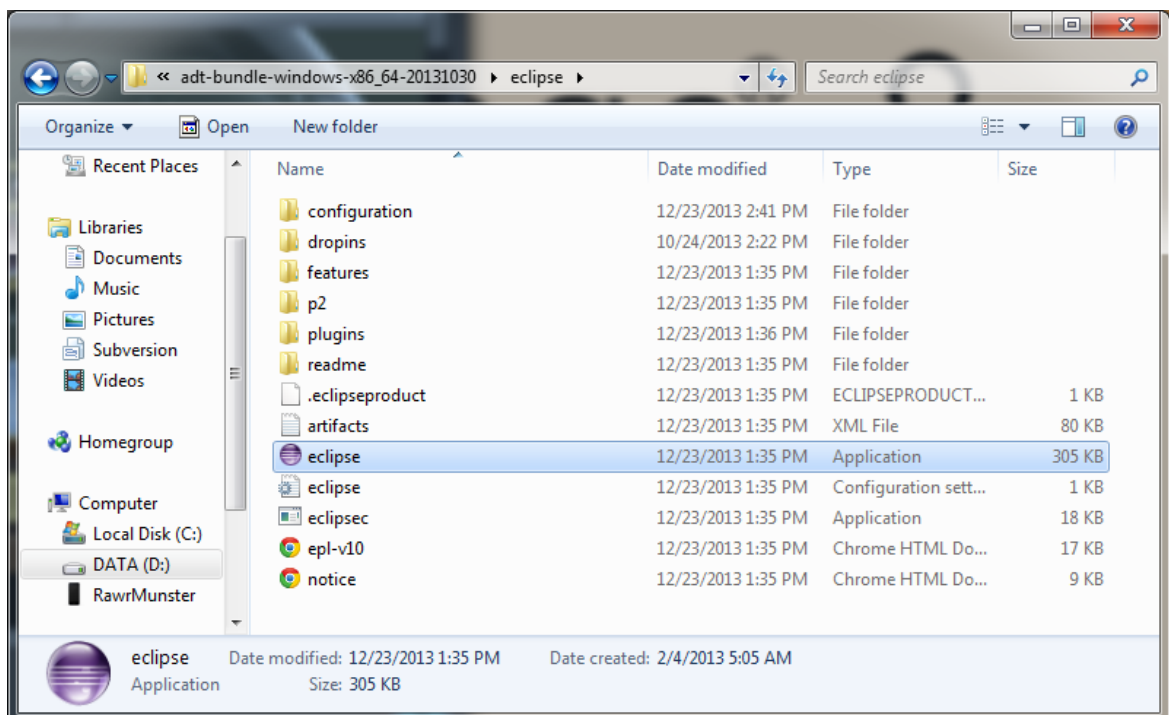
4. Start the Eclipse Integrated Development Environment (IDE). Begin navigating to the Eclipse application by double-clicking on the ADT Bundle folder.
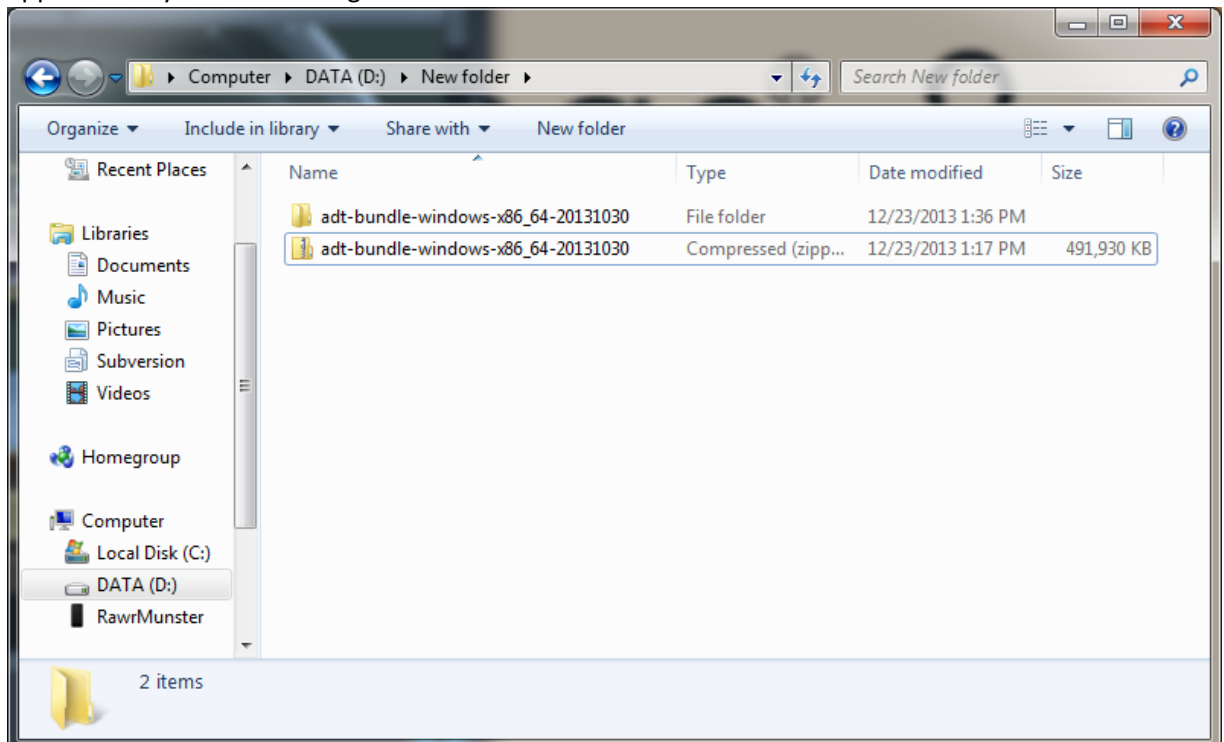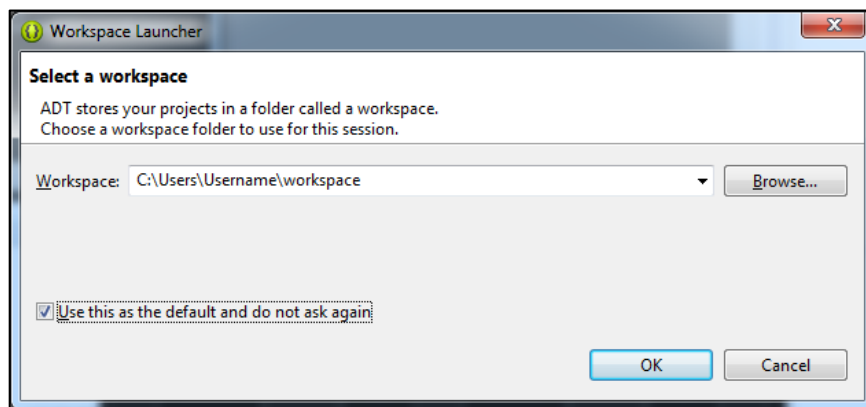
5. Next, double-click on the eclipse folder.
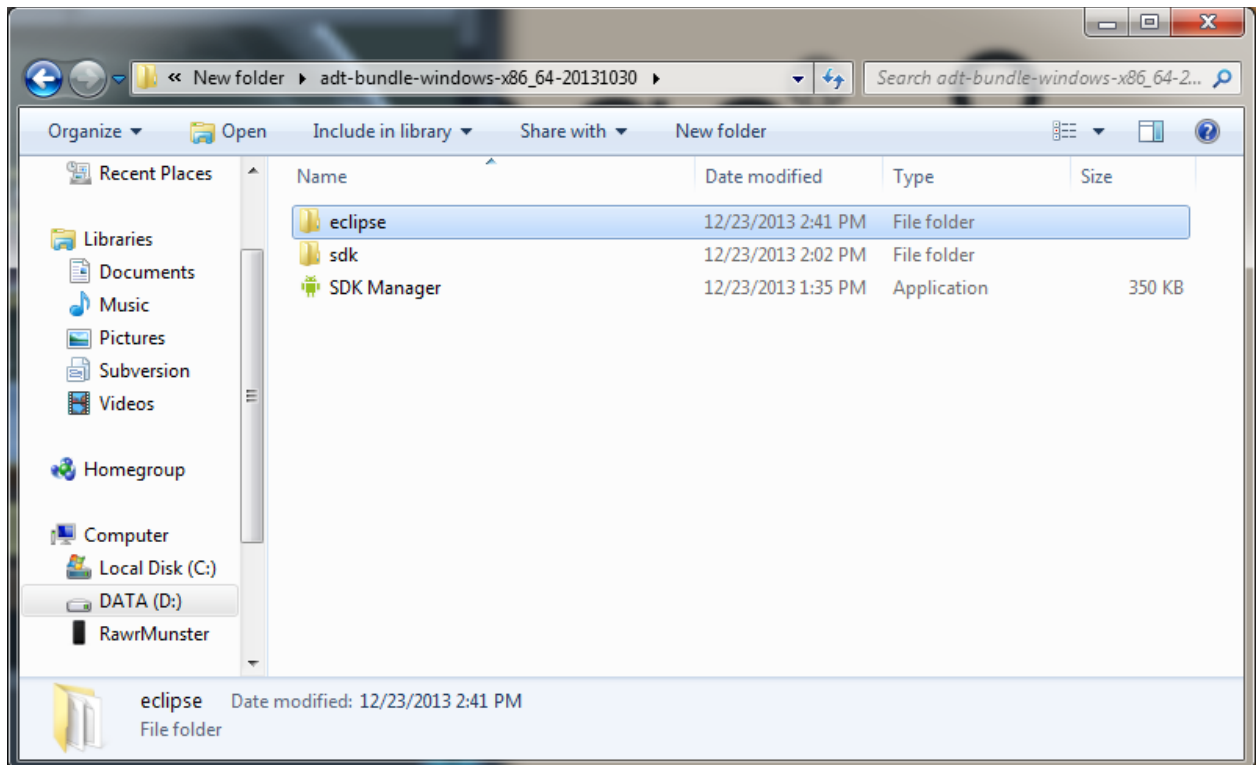
6. Next, double-click on the eclipse application icon.

7. As Eclipse starts, the Android Developer Tools splash screen will appear.

8. Eclipse will ask you to choose the location of your workspace. If you would like that workspace to be the default from now on, select the appropriate checkbox.

9. At this point you should see Eclipse open. The first time it opens you may see a Welcome screen, such as the one shown below.
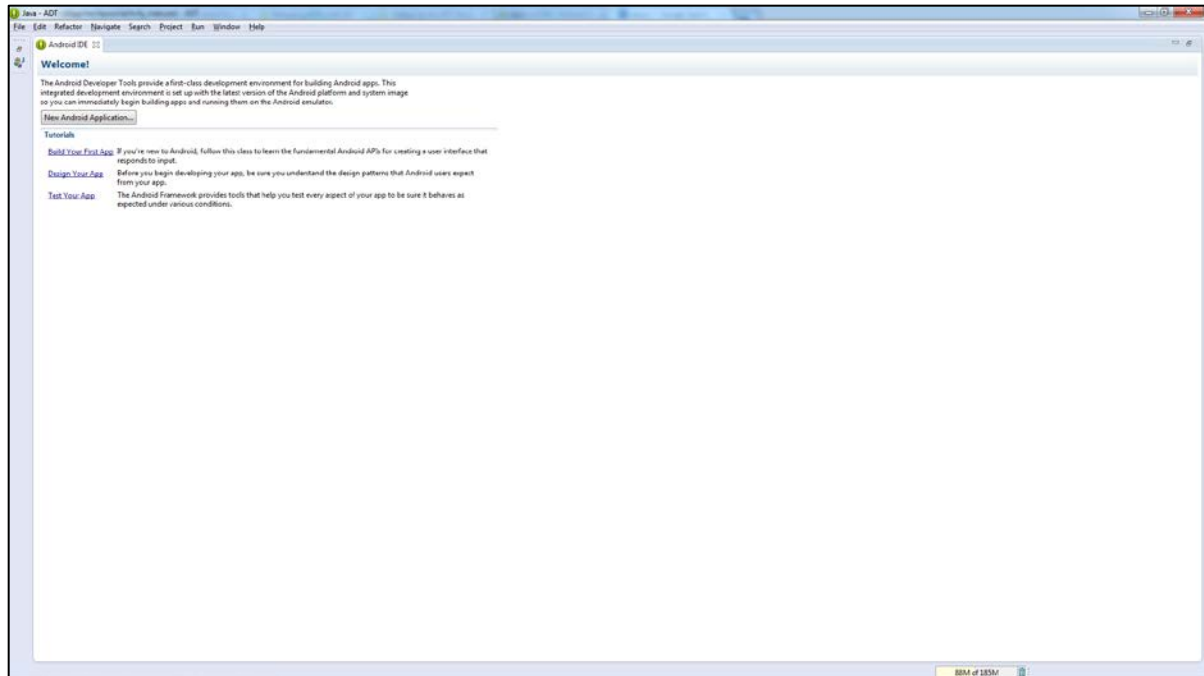
Advanced – Using Hardware Acceleration. Student Ed B. reminded me to mention the Intel Hardware Accelerated Execution Manager (HAXM). If compatible with your development machine, installing HAXM will greatly speed up the emulator, making your development, testing and debugging much more productive. HAXM is bundled with the ADT, and also available for download separately.

See the following links for more information:

http://developer.android.com/tools/devices/emulator.html#acceleration for a complete description.

About: http://software.intel.com/en-us/articles/speeding-up-the-android-emulator-on-intel-architecture

Downloads: http://software.intel.com/en-us/articles/intel-hardware-accelerated-execution-manager

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
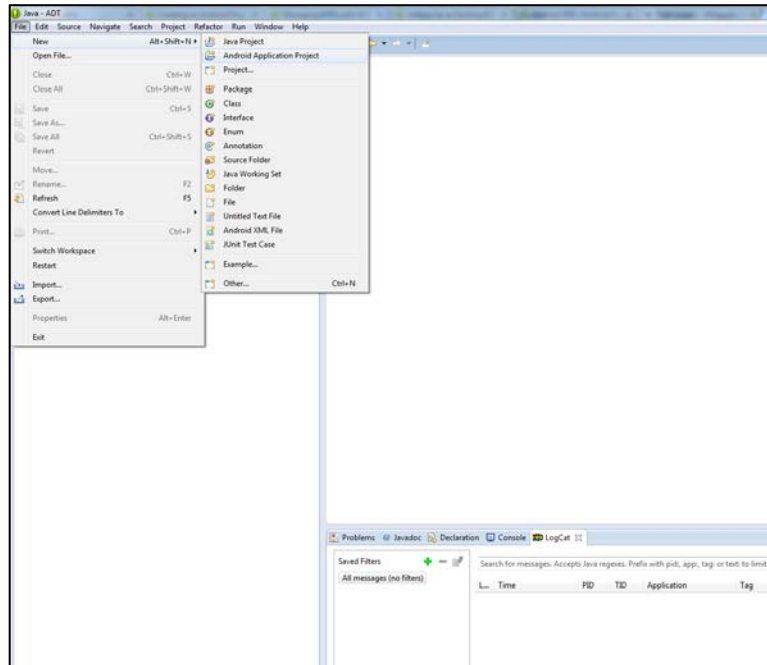

2. Next, a dialog box will appear, allowing you to enter an Application Name. Enter the string, "MyFirstApp." The Project Name and Package Name fields will fill in automatically. Leave the remaining settings alone and then click Next.
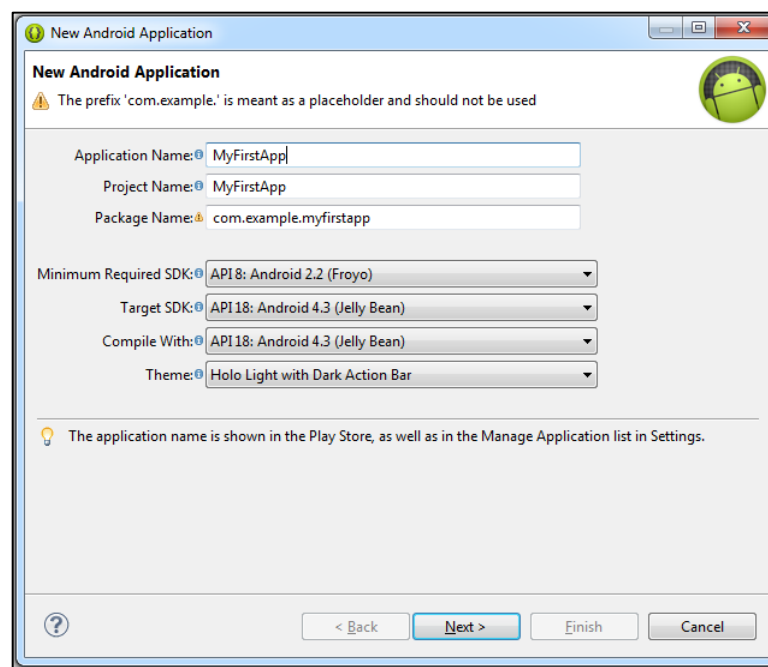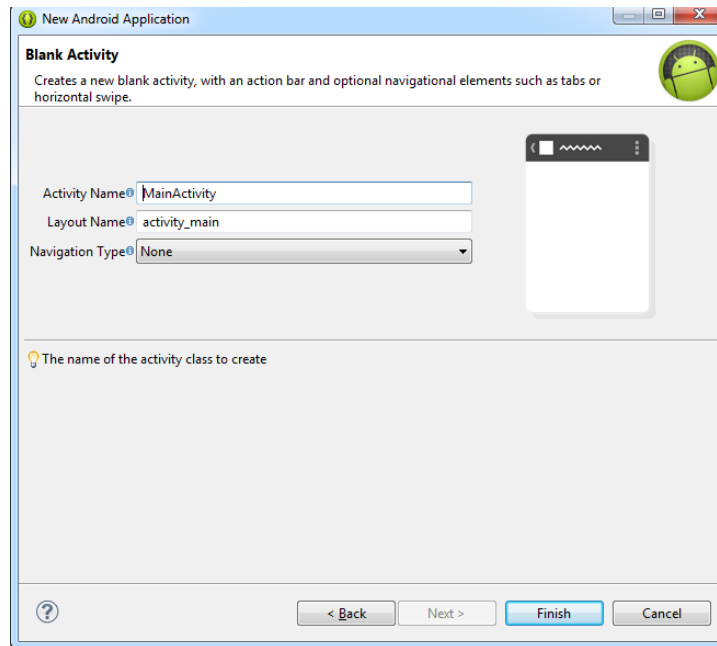
3. Next, a new dialog box will appear. Deselect "Create custom launcher icon," as we will not be creating a custom icon for this app. Then click Next.
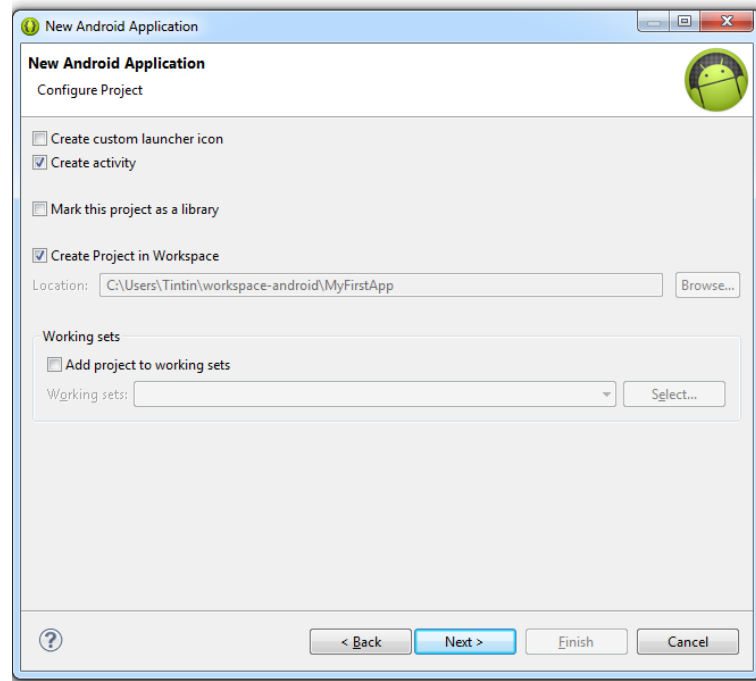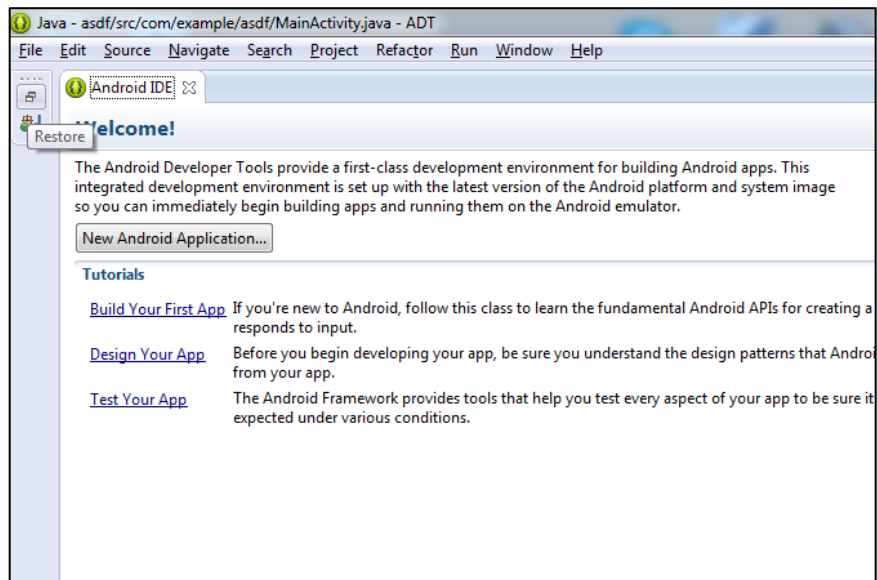

4. In the next window, leave all the settings as default. Then click Next.
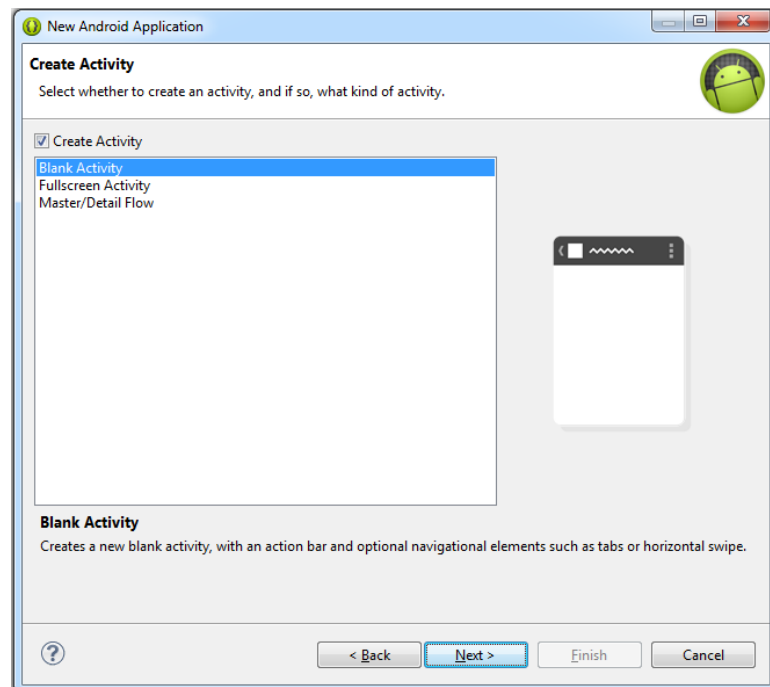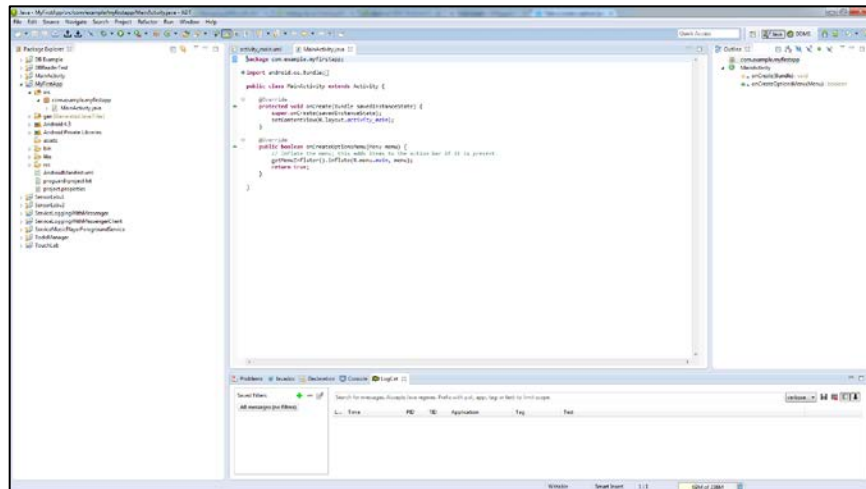
5. For the next window, also leave all default settings in place and press Finish.

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

7. 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.

8. Under activity_main.xml, is the xml file associated with 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.
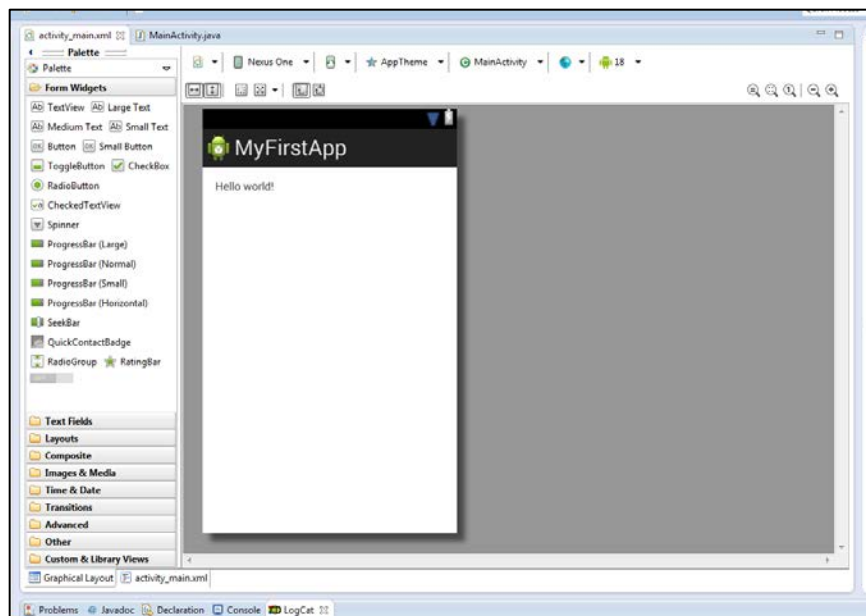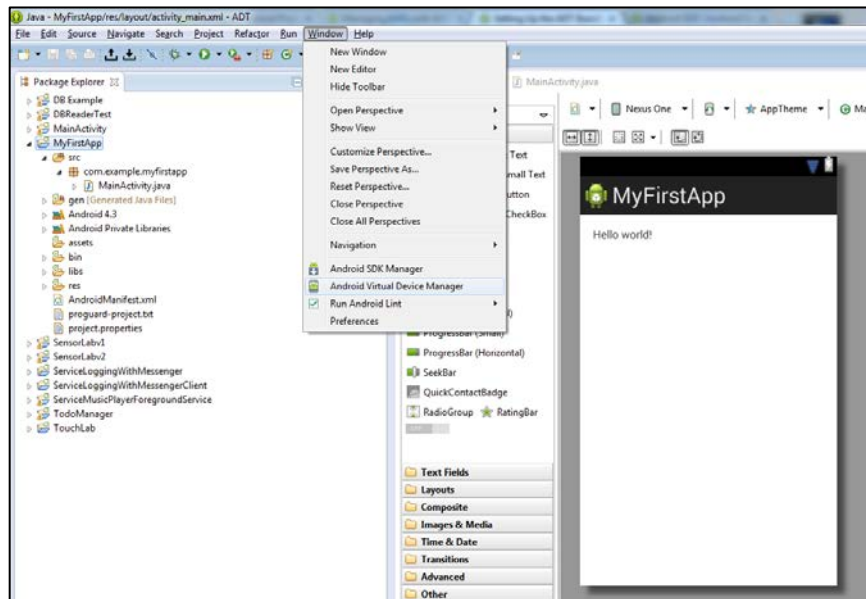

In part 4 we will show you how to run this app in the Android Emulator.

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.
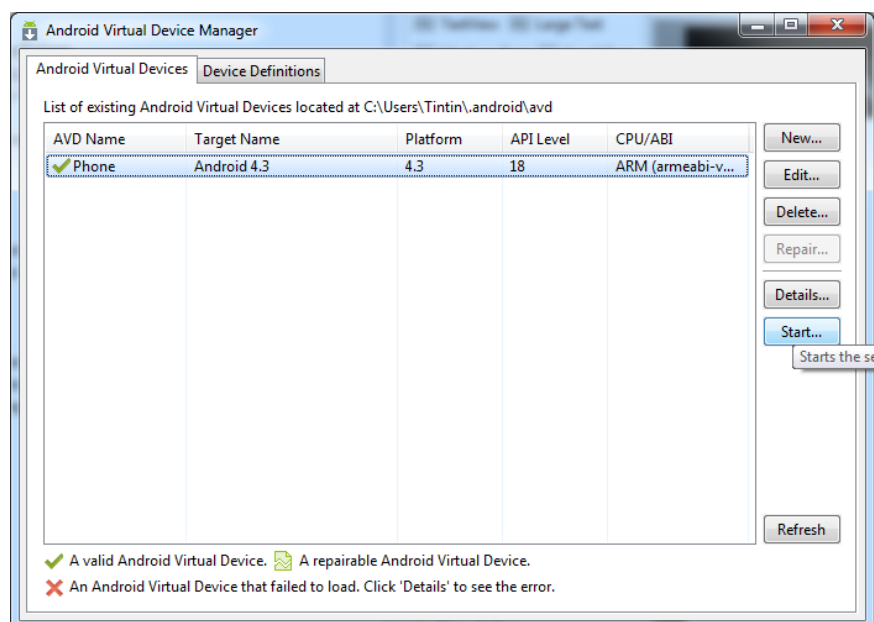

2. A new dialog box will pop up. Click "New" to create a new Android Virtual Device (AVD).
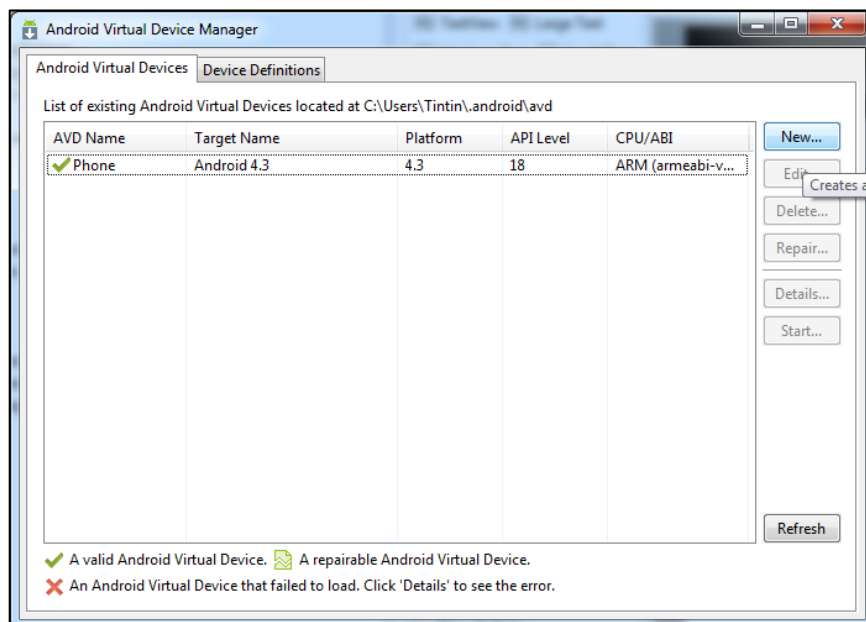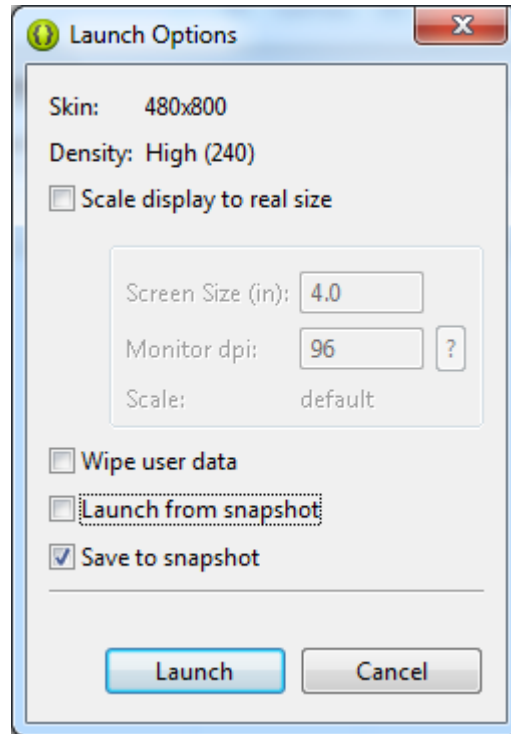
3. Another dialog box will pop up displaying various AVD parameters. You can play with these parameters to create different virtual devices. Press "Ok" to finish.
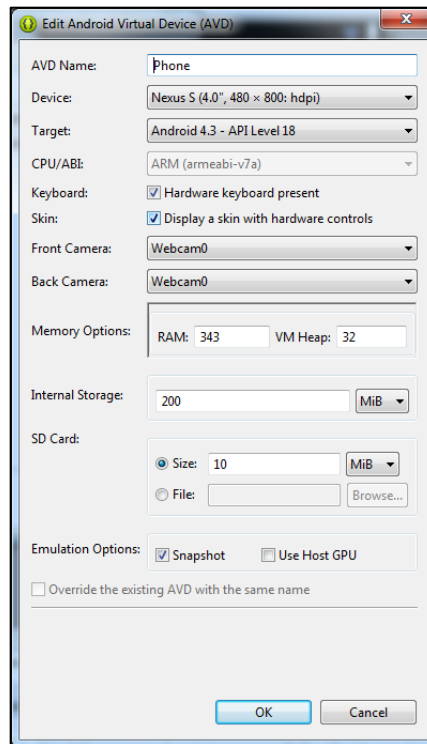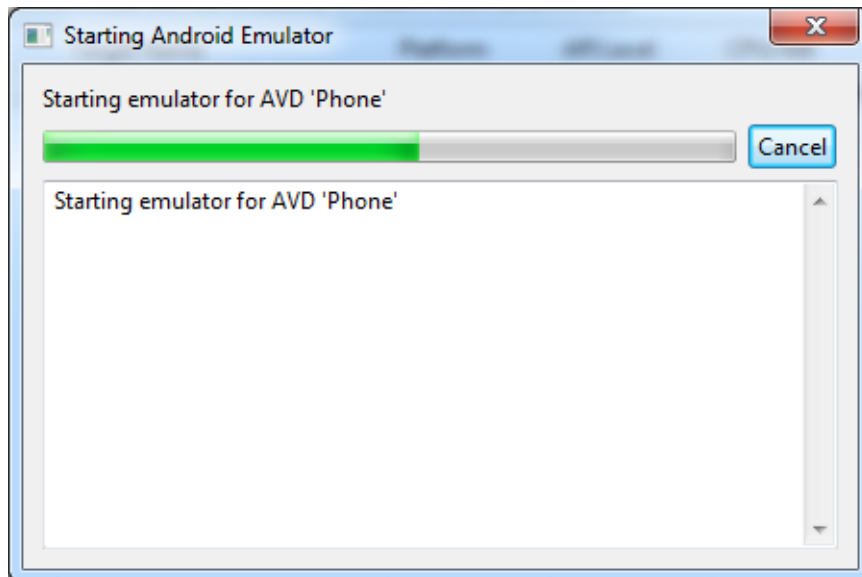

4. A new AVD will now be listed in the AVD Manager window. Select it and then press the Start button.
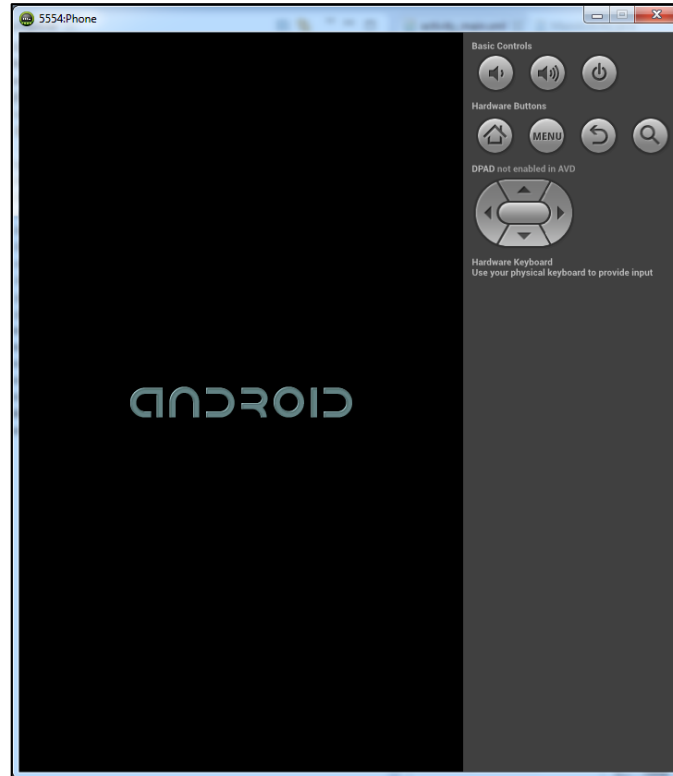
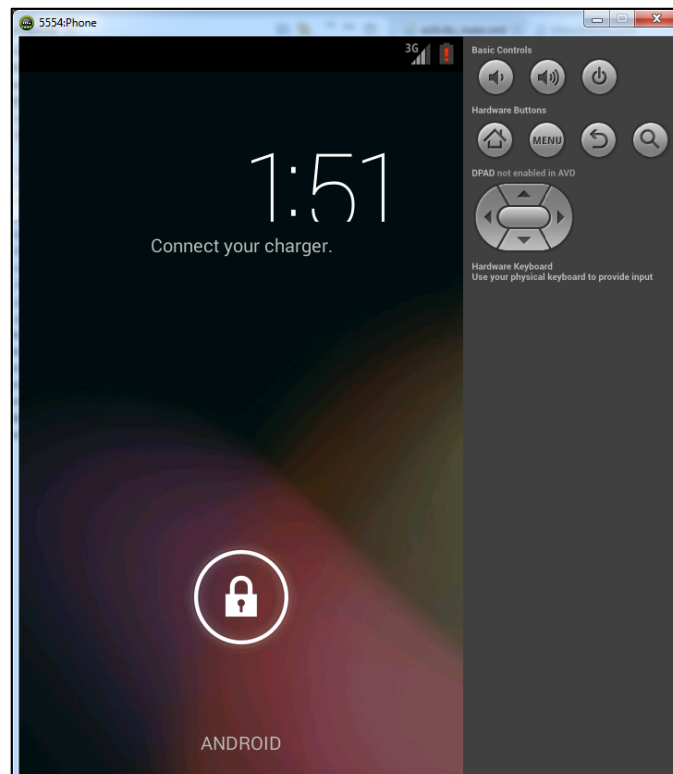5. A new dialog box will pop up. Press the Launch button to start the emulator.

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

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

8. 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.
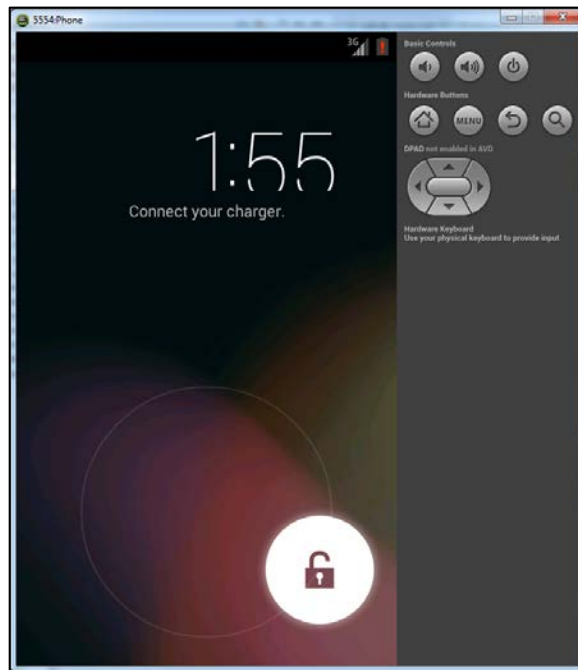
1. Return to Eclipse and right-click your app's project name. From there select Run As > Android Application.
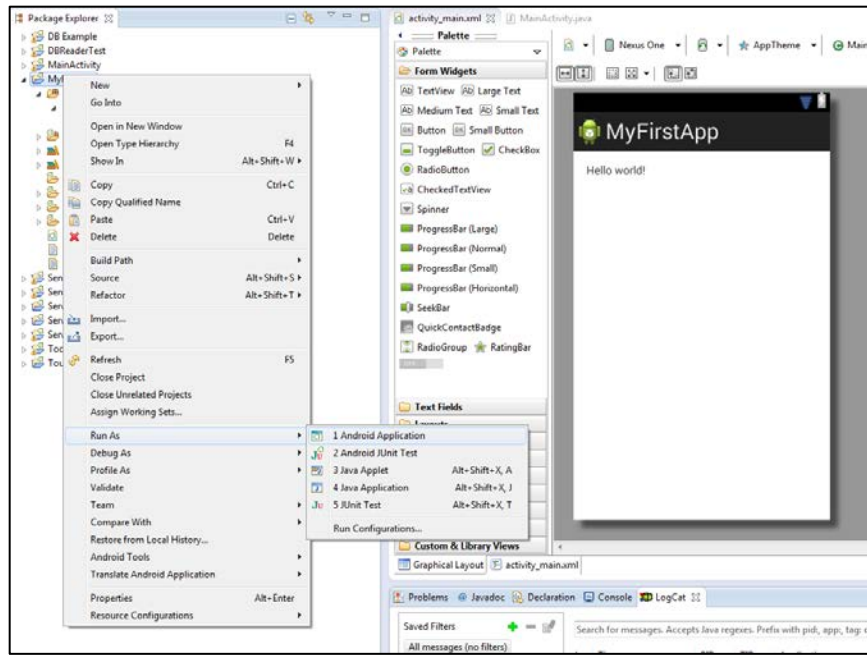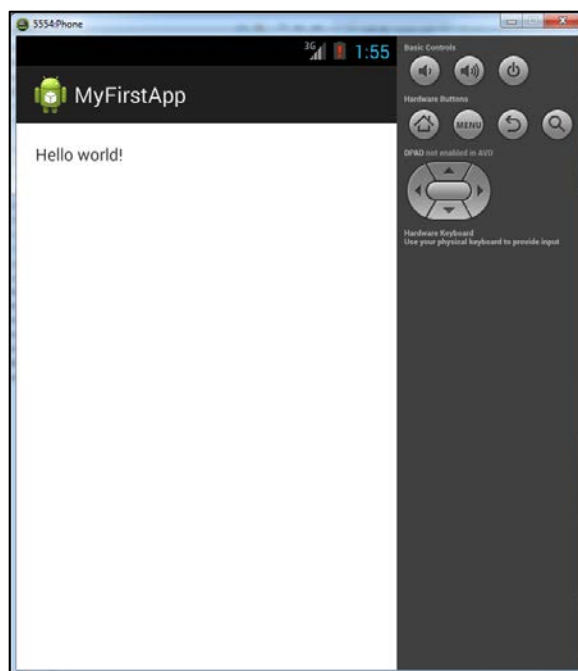

2. In the Console panel, below the editing window, you will see output indicating that the application is being loaded onto the Android Emulator.
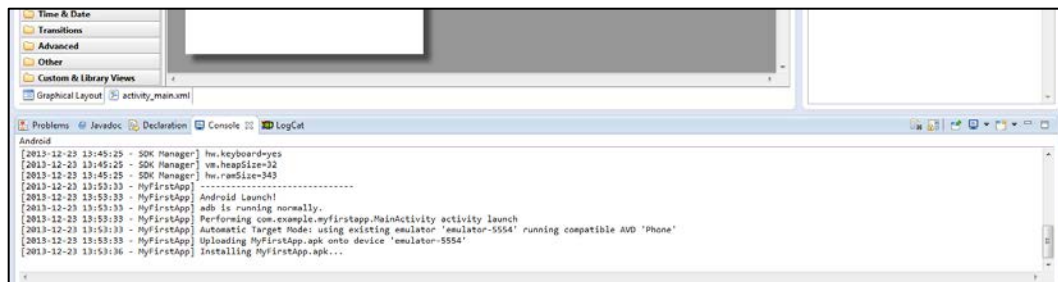
3. Return to your Emulator instance. If neessary, drag the lock icon to unlock your device.

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.
2. Extract the contents of the Example App to any folder. The contents should contain a sample project contained within a folder.
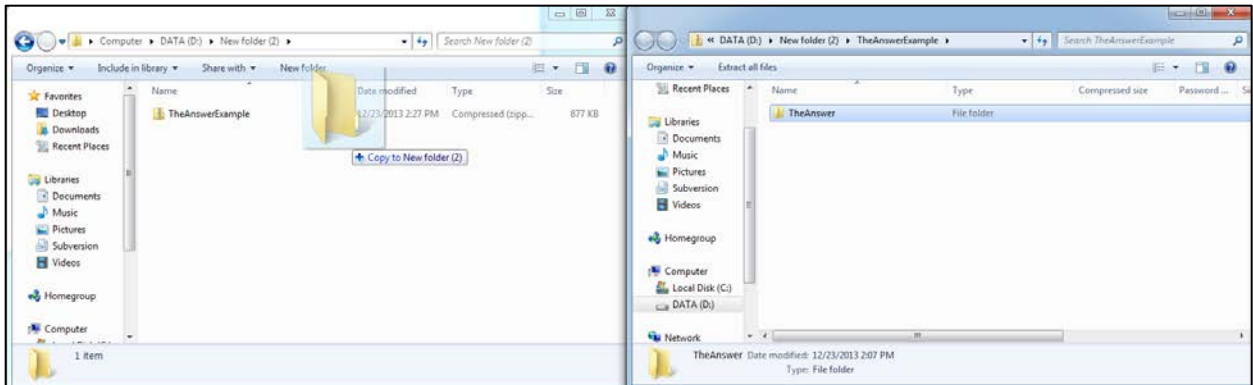
3. Return to Eclipse. Select File > Import from the menu bar to import the application project.
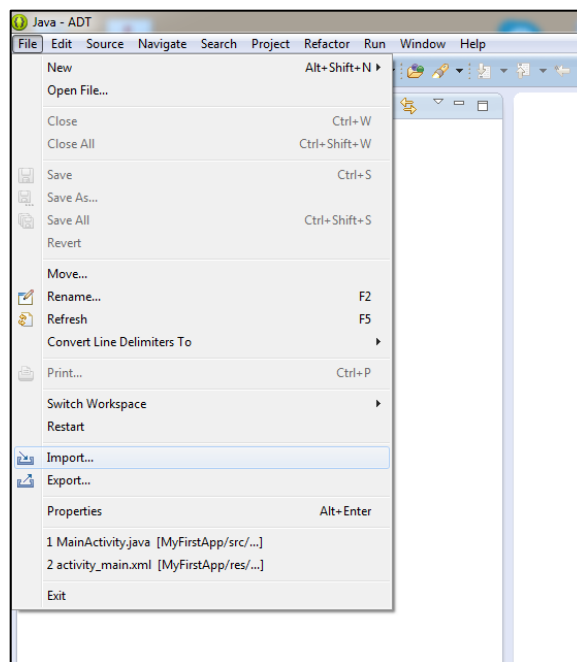
4. Next, in the dialog box that appears, select Android > Existing Android Code Into Workspace, and then press the Next Button.
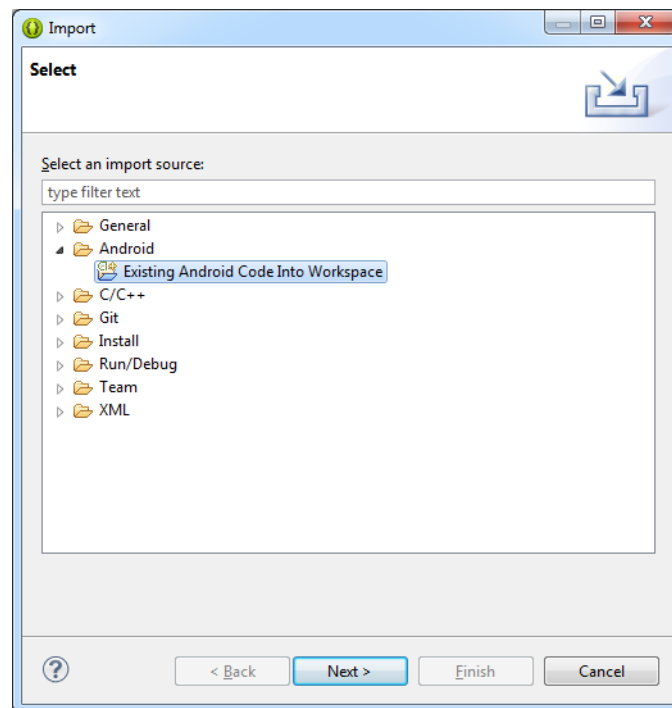
5. Next, you'll see a dialog box, allowing you to select the project to import. Press the Browse... Button.
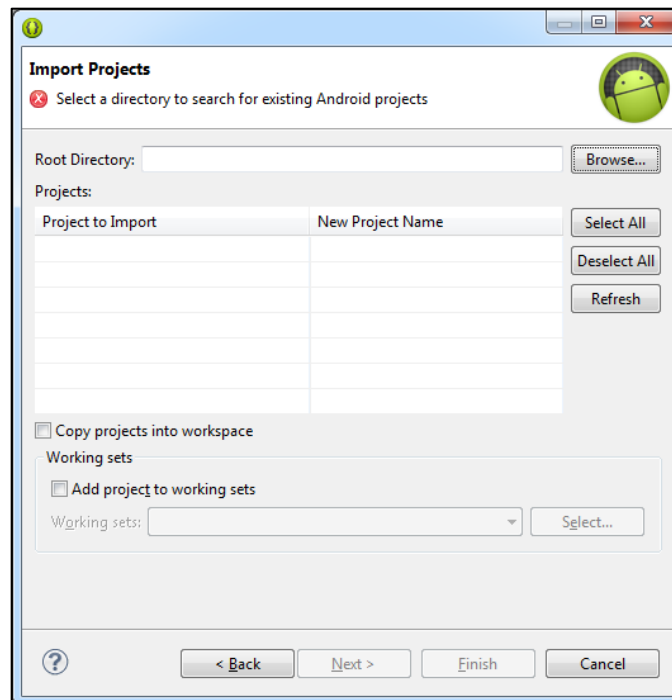
6. Select the location of the Example Application contents, and press OK.
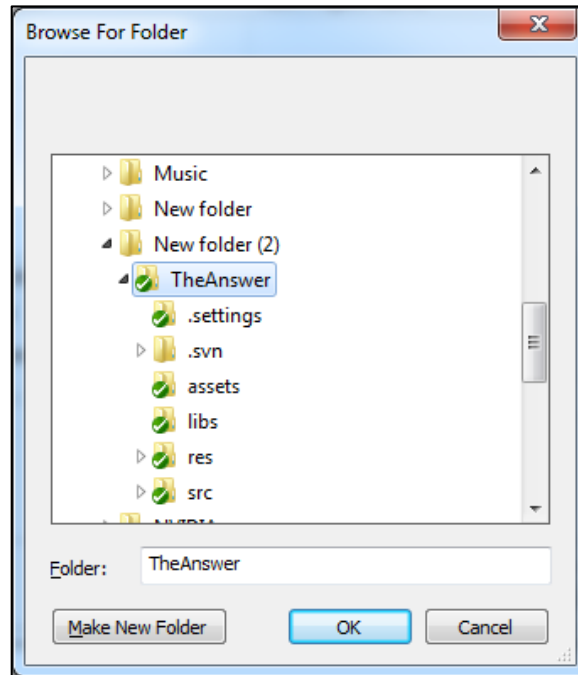
7. In the next dialog box, make sure that you select "Copy projects into workspace" and then click the Finish Button.
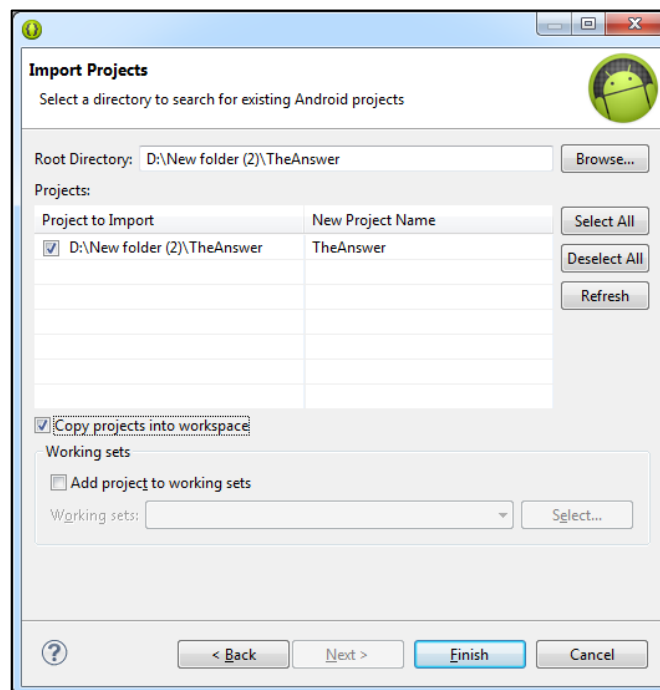
8. At this point the application should appear in the Project Explorer on the left side of the IDE.
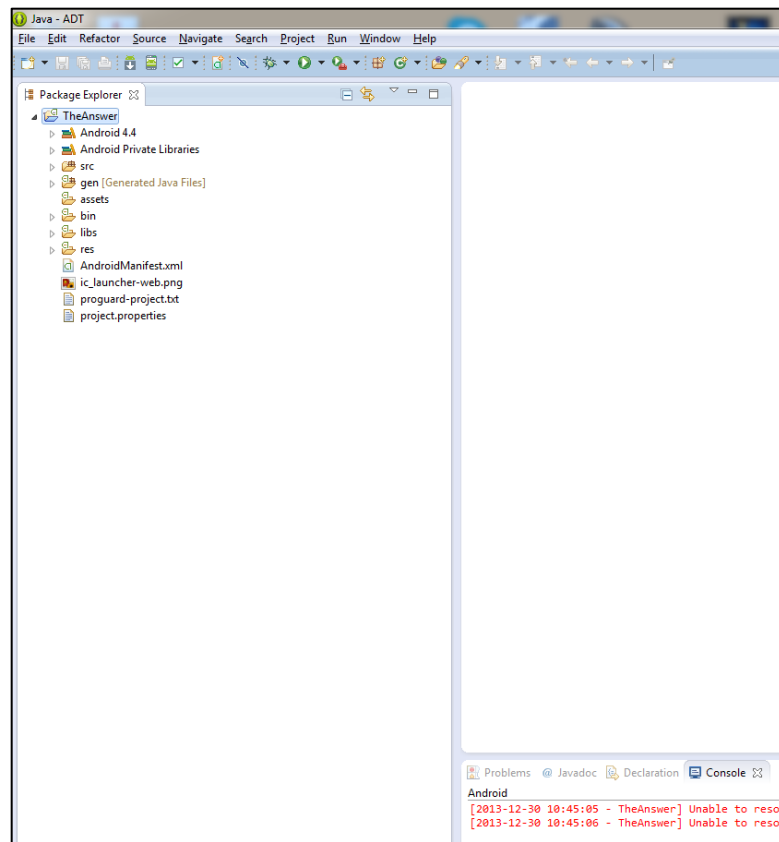
9. Right-click the folder go to Run As > Android Application
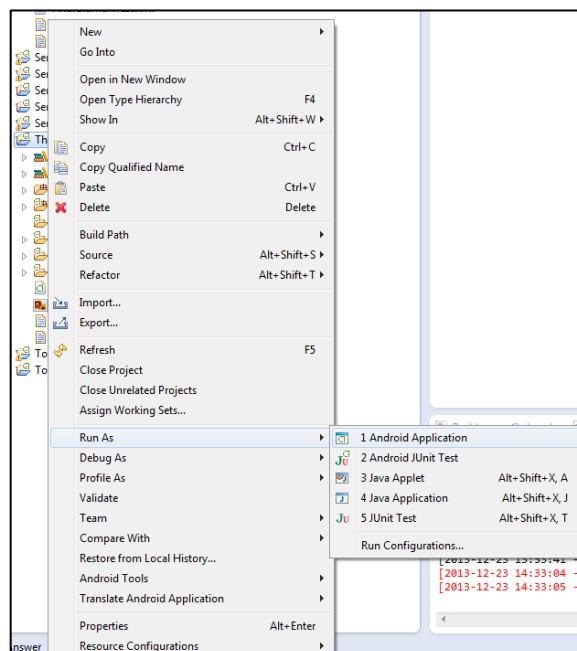
10. The Android Emulator will now open up and run the example application.
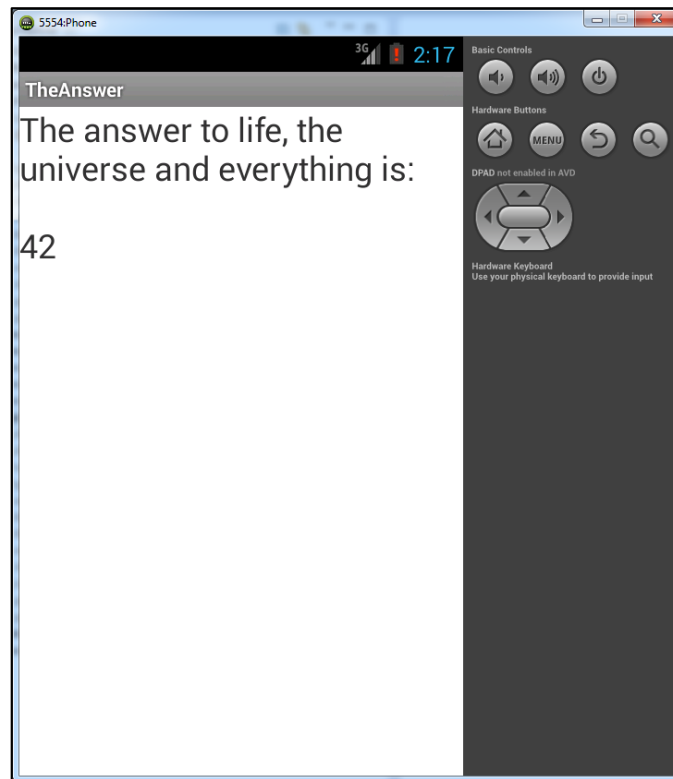
Part 6 – Debugging

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

1. Double-click the TheAnswer.java file under src > course.examples.theanswer
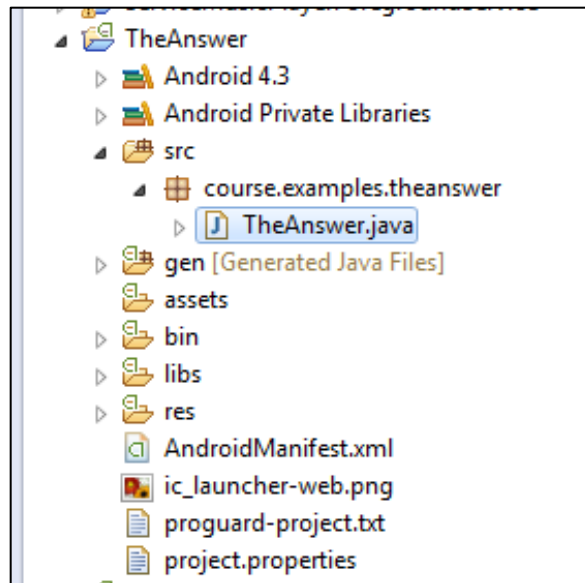
2. Next, open the Debugging Perspective by selecting Window > Open Perspective > Debug from the menu bar.
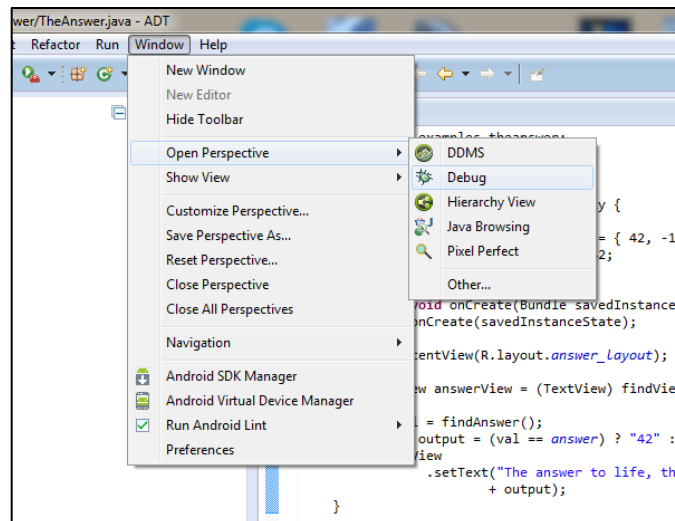
3. A new perspective will appear, similar to that shown below.
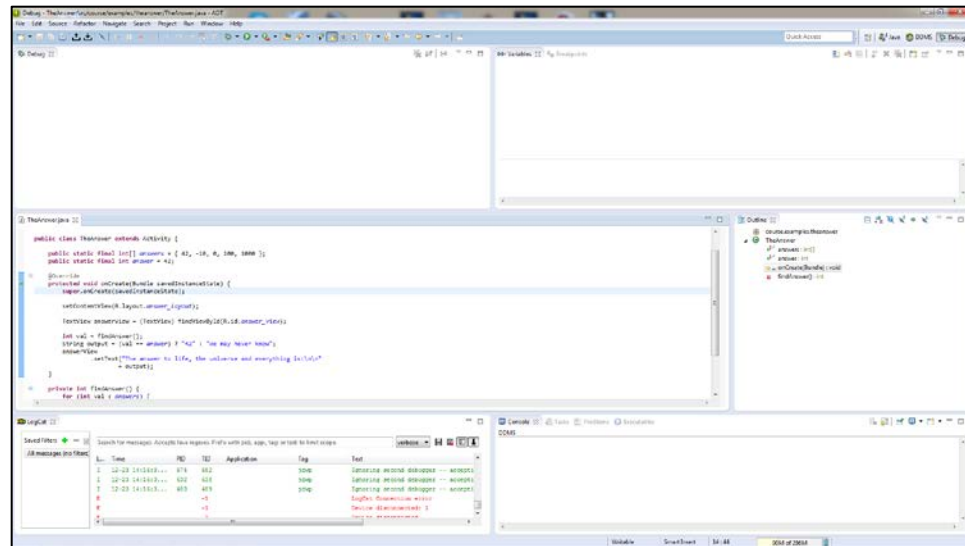
4. On this screen, double click the highlighted blue area next to the line:
   "int val = findAnswer();"


```
        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.

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

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

Line breakpoint:TheAnswer [line: 20] - onCreate(Bundle)
        String output = (val == answer) ? "42" : "We may never know";
        answerView
                .setText("The answer to life, the universe and everything is:\n\n"
                        + output);
```
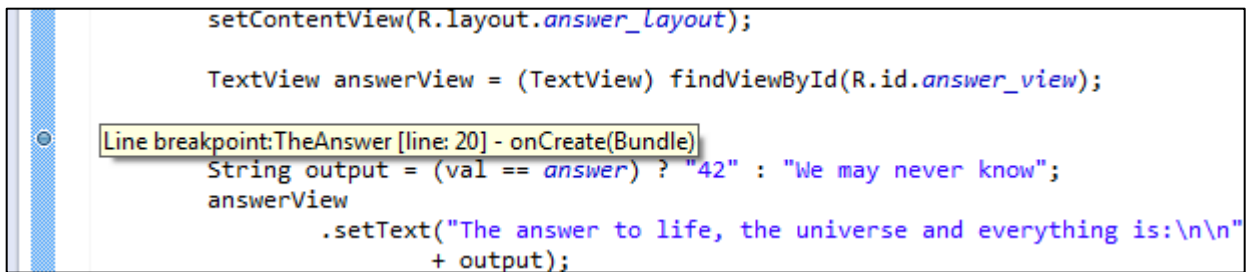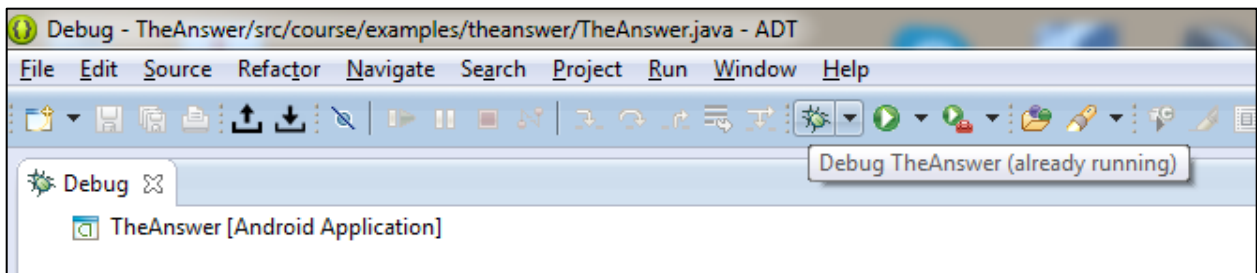
6. Next, press the Debug icon at the top of the window in the menu bar to start debugging the application.

```
Debug - TheAnswer/src/course/examples/theanswer/TheAnswer.java - ADT
File   Edit   Source   Refactor   Navigate   Search   Project   Run   Window   Help

                                                    Debug TheAnswer (already running)
Debug ⊠
    TheAnswer [Android Application]
```

7. Your Emulator should have loaded the App and stopped before the words ,"The answer to life……" , is displayed on the screen.
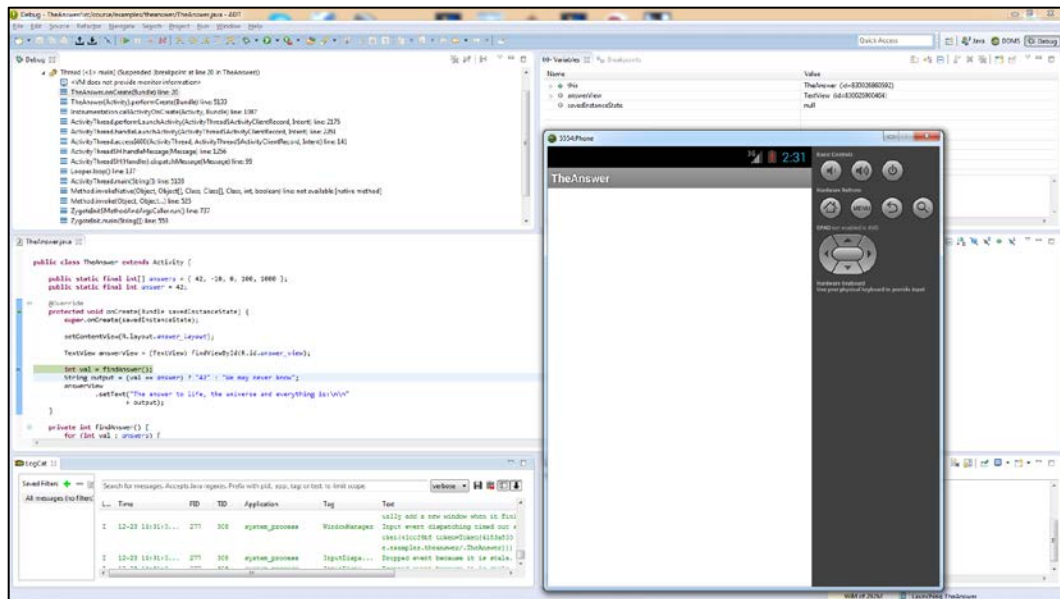
8. Now that the app is stopped, you can examine the app's state and step through the app's execution using the following buttons appearing in the menu bar.
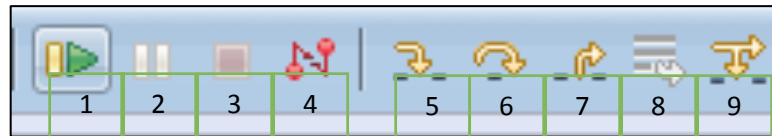
Buttons from left to right on the navigation bar:
1 - Resume
2 – Suspend
3 – Terminate
4 – Disconnect debugger from emulator
5 – Step into
6 – Step over
7 – Step return
8 – Drop to frame
9 – Use step filters

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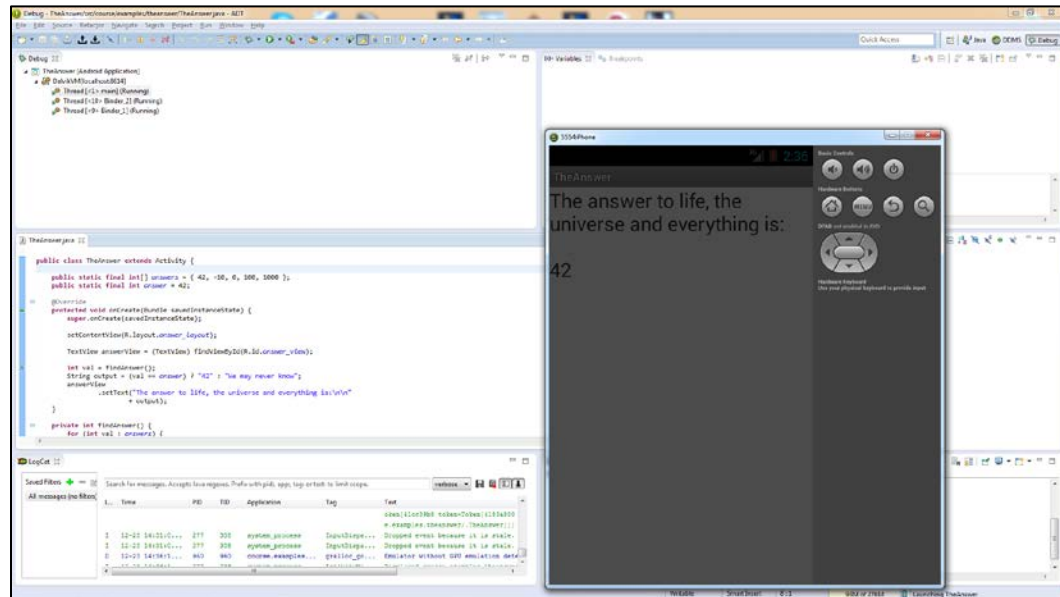

10. The next debugging task will have you create and display informational messages to the LogCat panel, to help you better understand the application's runtime behavior. To generate these messages, you will use methods in the android.util.Log class. You will also need to import this class into your 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.
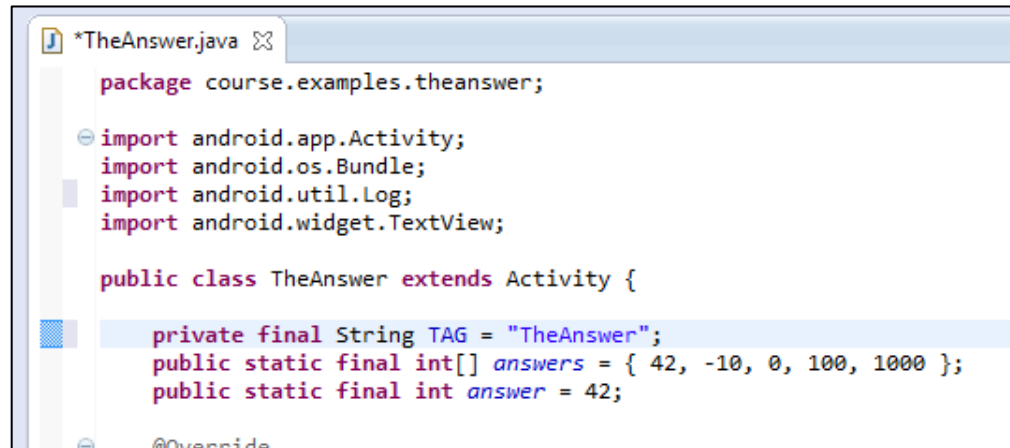
11. Return to the Java Editing Perspective by clicking the Java Icon in the top-right corner.
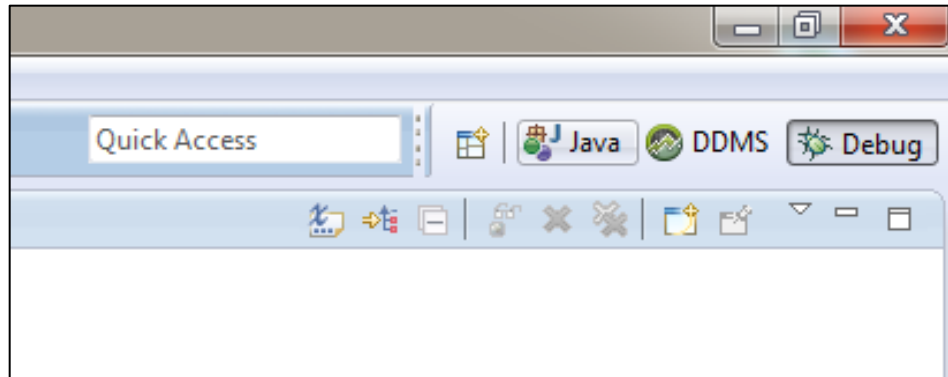
12. Import the android.util.Log library by typing, "import android.util.Log;" near the beginning of the code for TheAnswer.java. You can also use the Organize Imports function (Ctrl + Shift + 0 is the shortcut).
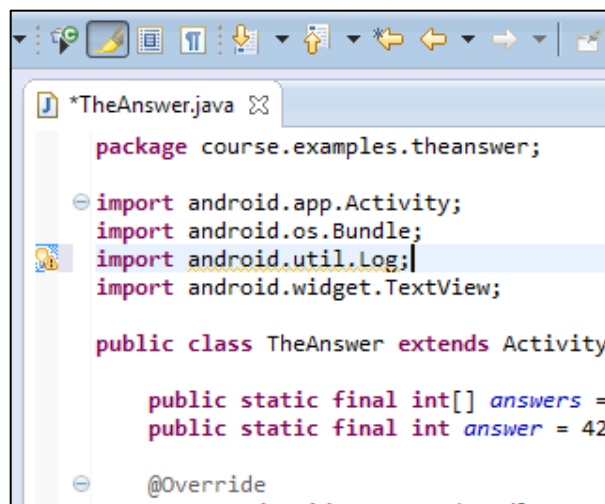
13. The Log class' methods require a string called a Tag, which identifies the creator of the message and can be used to sort and filter the messages when they are displayed. Create a constant called TAG within the TheAnswer class , by typing, "private final String TAG = "TheAnswer";"

```java
🇯 *TheAnswer.java ⊠
    package course.examples.theanswer;

    import android.app.Activity;
    import android.os.Bundle;
    import android.util.Log;
    import android.widget.TextView;

    public class TheAnswer extends Activity {

        private final String TAG = "TheAnswer";
        public static final int[] answers = { 42, -10, 0, 100, 1000 };
        public static final int answer = 42;

        @Override
```
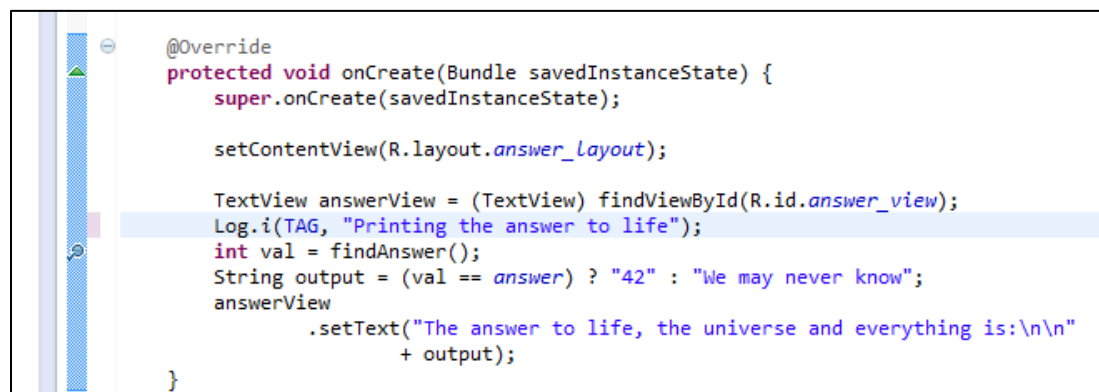
14. Use the Log.i() function to create and output a log message. Just before the line that starts, "int val = …" type in a new line: "Log.i(TAG, "Printing the answer to life");"
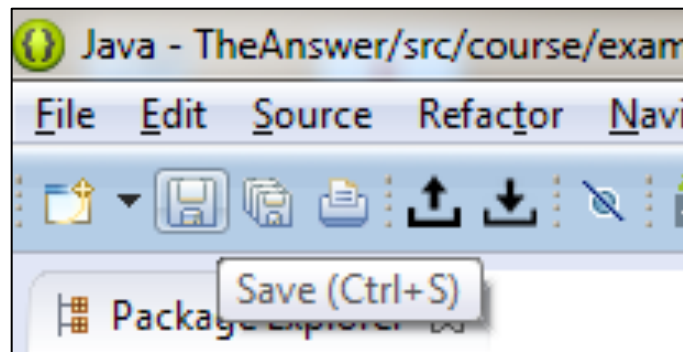
```java
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            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);
        }
```
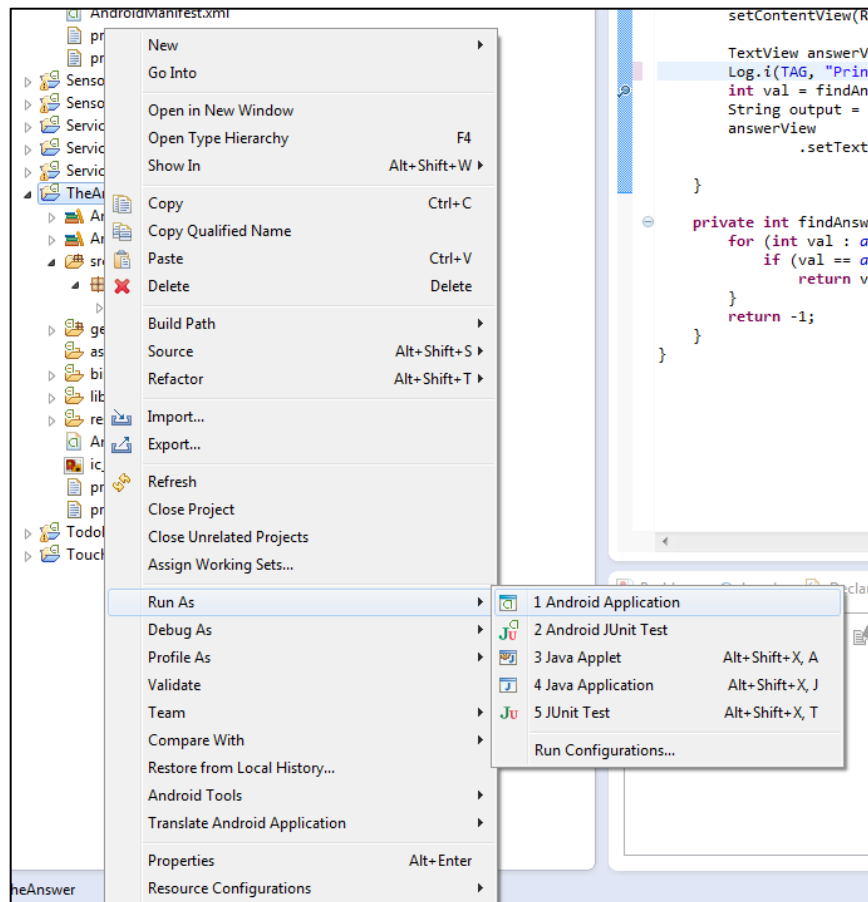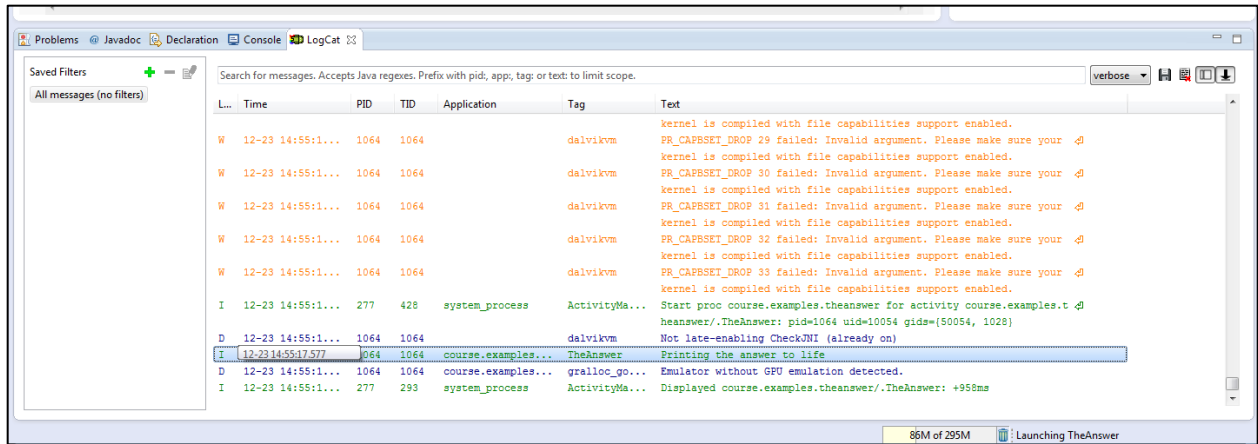
15. Save your changes.


16. Run the application.

17. Once the app is run, 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.

Part 7 – Exploring the IDE

7.1 SCREENSHOTS - In this part of the lab you'll experiment with different emulated devices and take a screenshot. If necessary, return to Part 3, following the directions until you reach this screen.
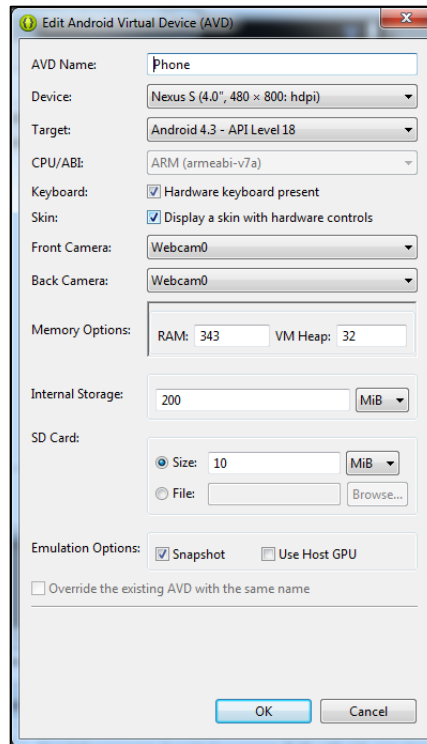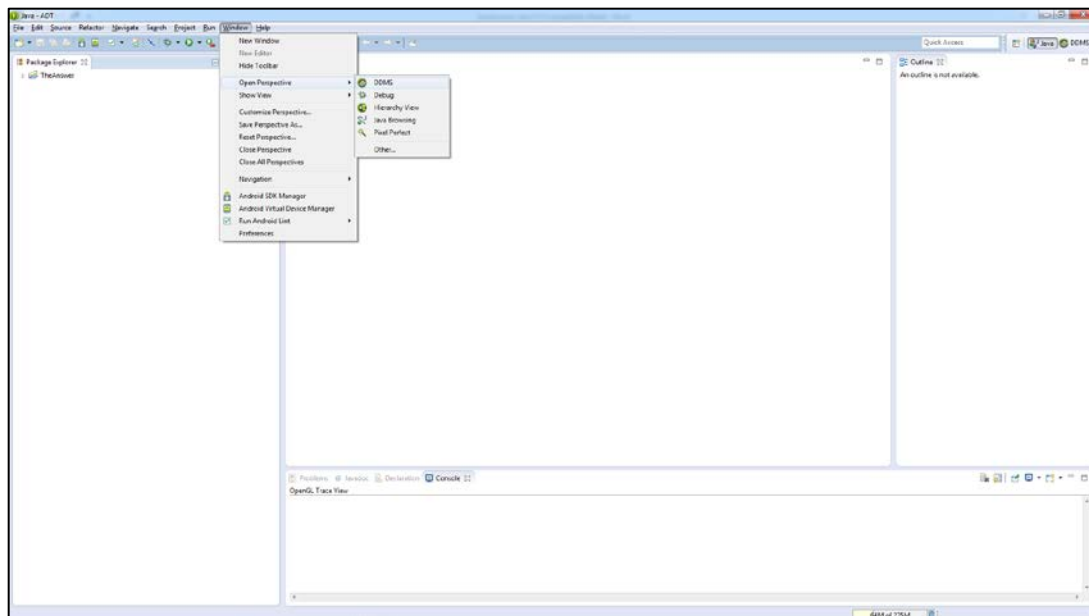

1. This time around change the Device from the Nexus S to any other device, such as a tablet or the Nexus 4.
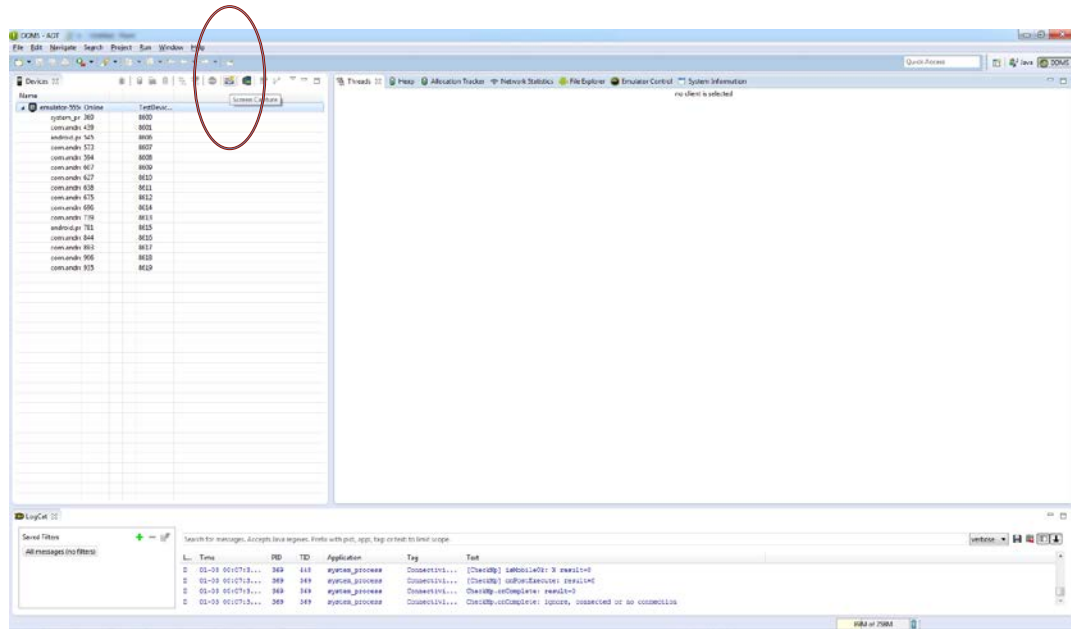
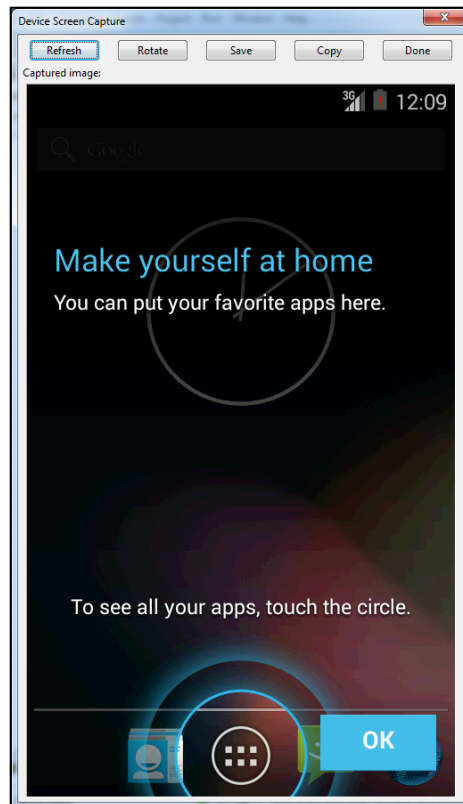2. Once you've created the AVD, start the emulator.
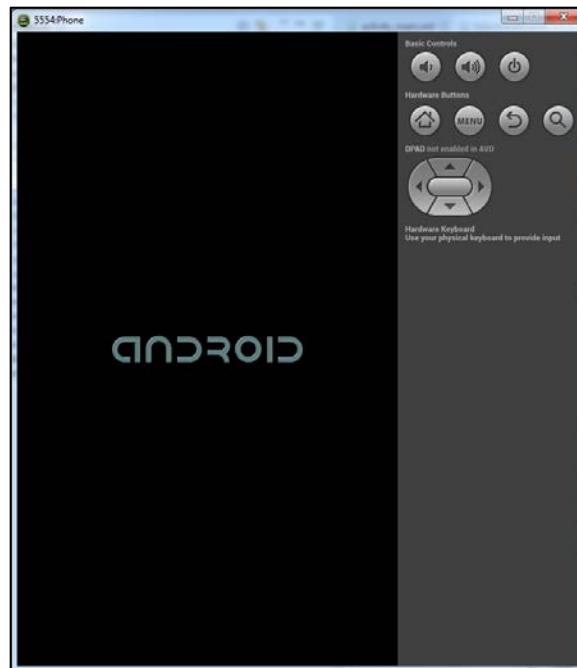
3. Once your emulator has loaded, take a screenshot of the emulated device. Navigate to the DDMS Perspective of the development kit.
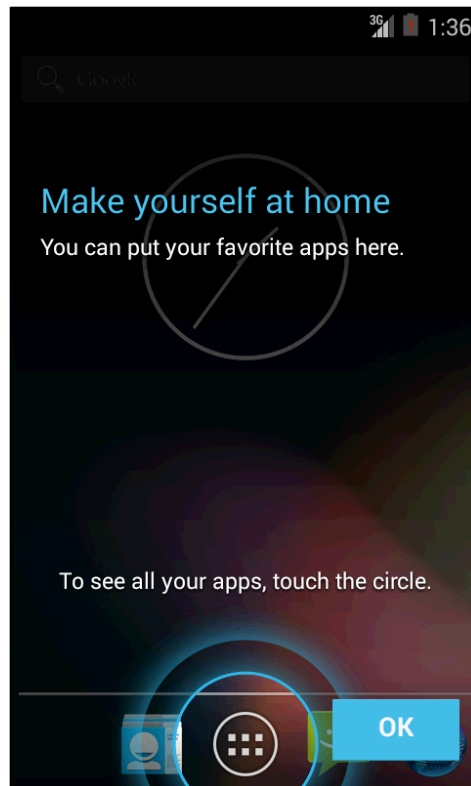
4. Click on the Screen Capture button.

5. The snapshot will appear in a separate window.

6. Press Save to save the image to anywhere on your computer. You have now created a screenshot of your android device.
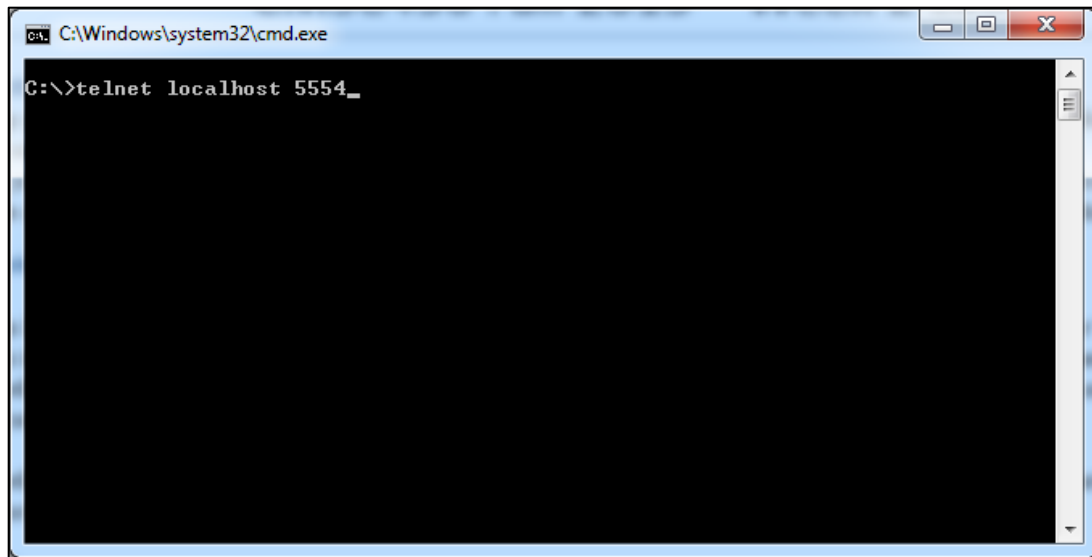
7. **7.2 TELNET** - For this next part you'll use telnet to change the network speed configuration in the emulator. You will need to use Windows' command prompt, Mac OS X's Terminal, or the Linux shell. (Pictures in these instructions will use Wndows 7). On Windows you may need to enable the Telnet client.
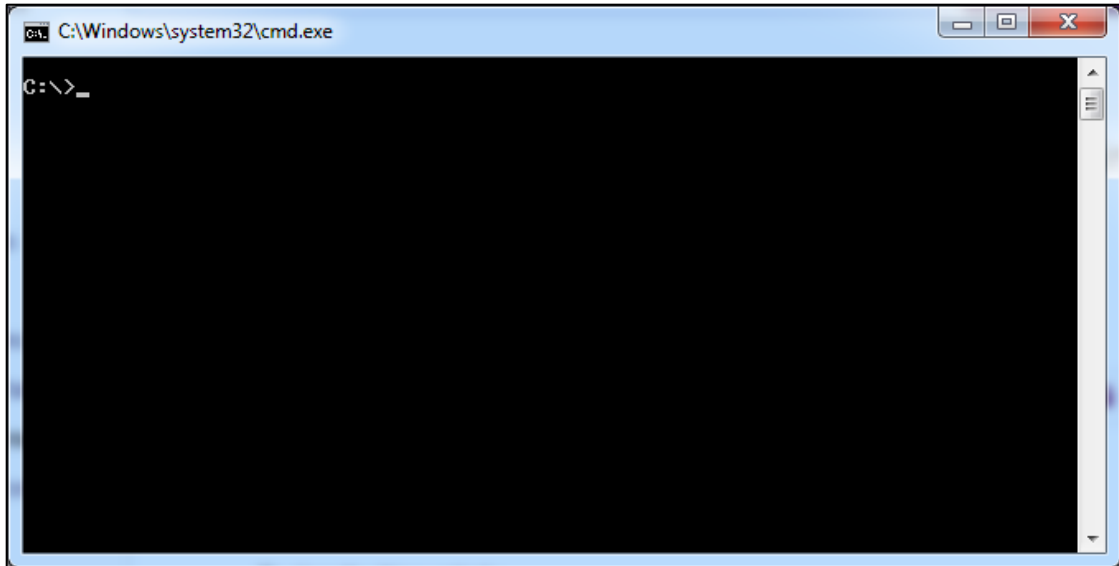
8. Return to the Android DDMS Perspective in Eclipse to find the port the android emulator is running on (It's also in the title bar of the emulator).
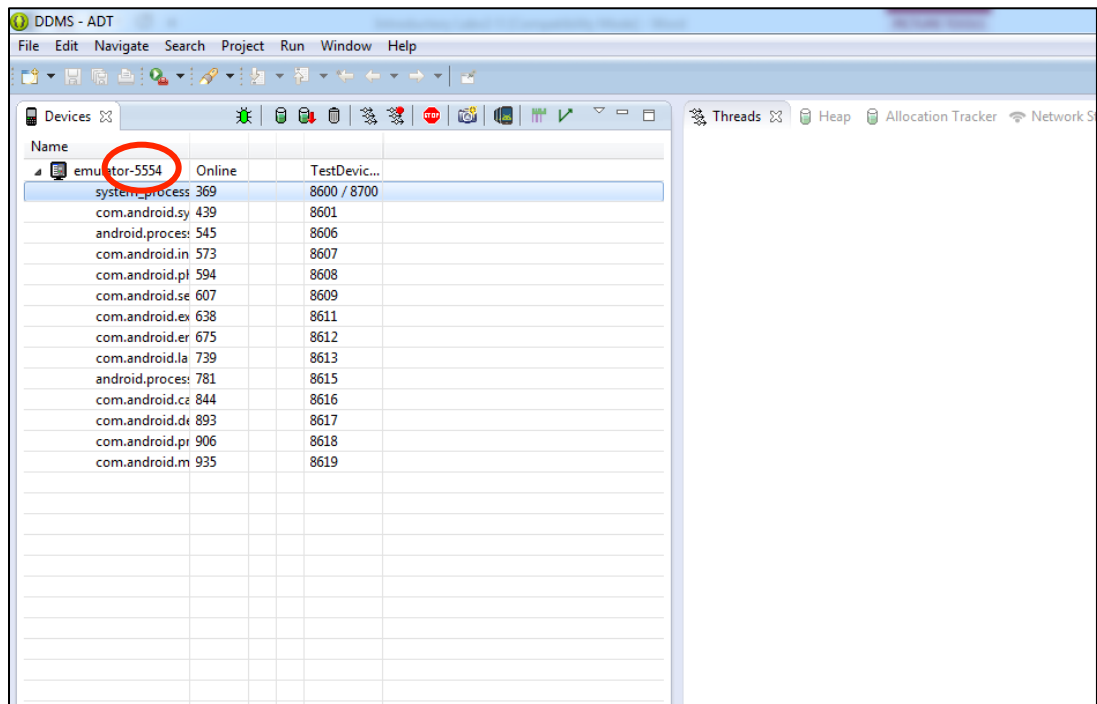
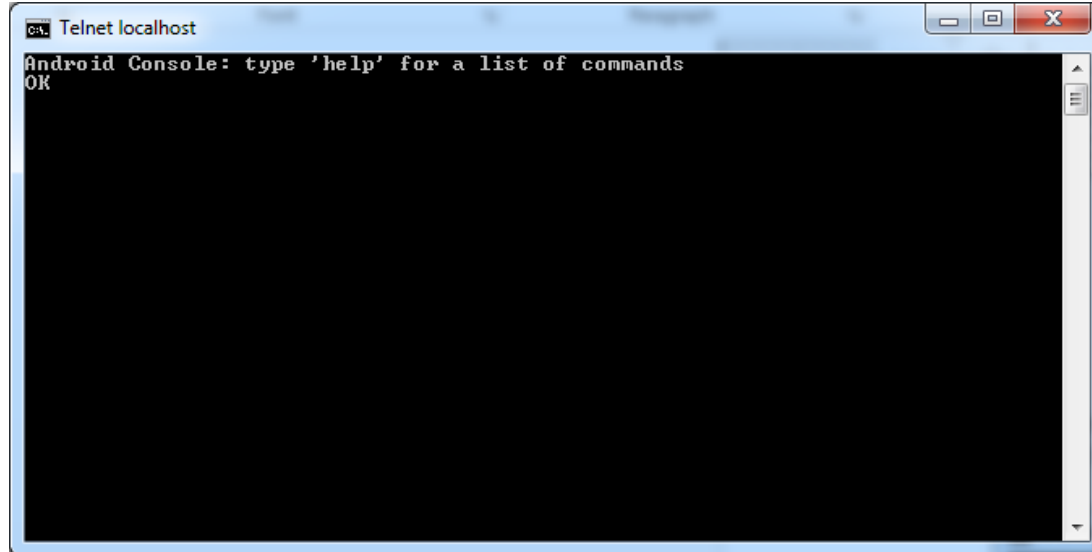9. Return to the command line interface and enter "telnet localhost <your port>". In my case I entered "telnet localhost 5554". If your computer doesn't recognize localhost, try 127.0.0.1 instead.
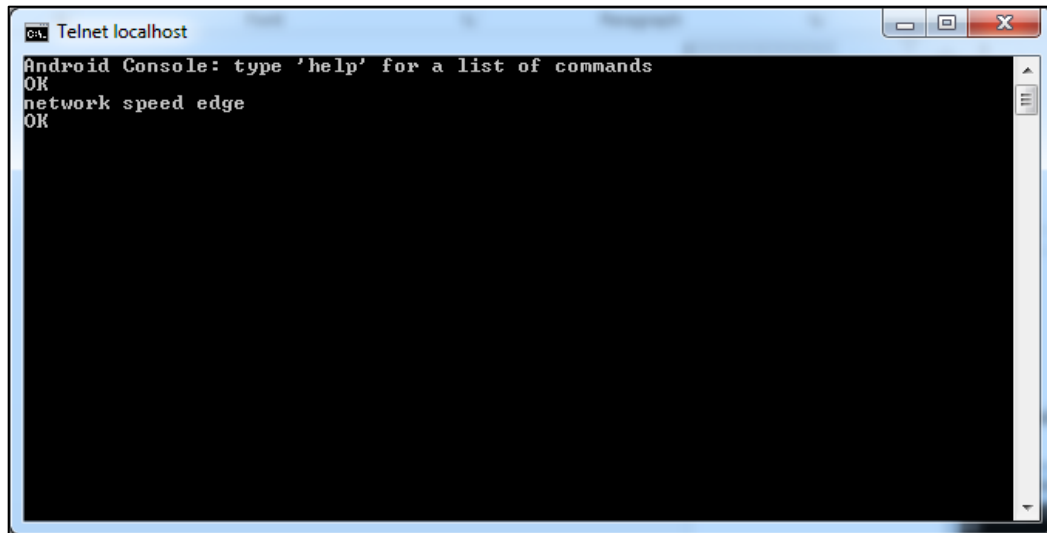
```
C:\Windows\system32\cmd.exe

C:\>telnet localhost 5554_
```

10. The console will display text indicating that you're connected to the emulator.
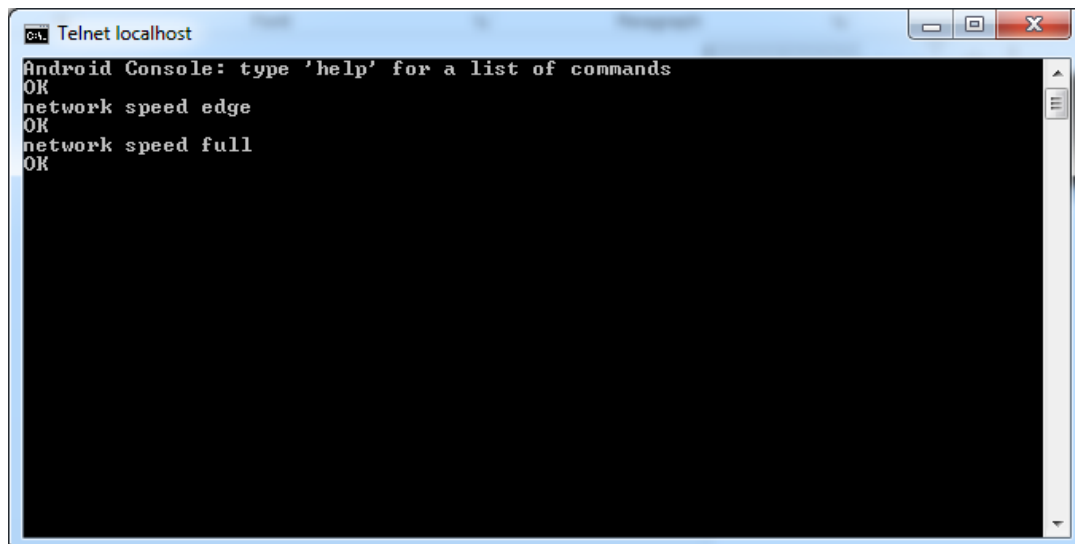
```
Telnet localhost

Android Console: type 'help' for a list of commands
OK
```

11. To set the network speed to EDGE, enter "network speed edge". Afterwards, you will see the response "OK".
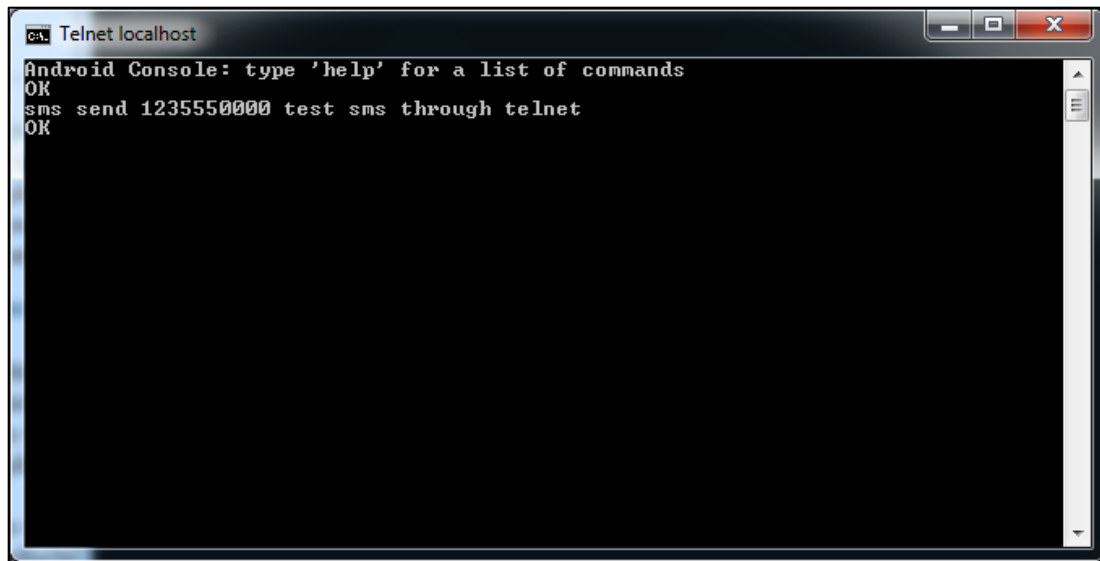
12. Return to the emulator and open up the browser to "http://www.google.com" and measure how long it takes for the page to fully load. The measurement doesn't need to be precise. After each time you load the page, clear the Browser's cache by going to the Browser's menu button and selecting Settings> Privacy & Security > Clear cache.

13. Return to the command line and this time switch the network speed to full. Do this by entering "network speed full"

14. Return to the emulator and open up the browser to "http://www.google.com" again measuring how long the page takes too load, this time on full network speed.

15. Repeat steps 12-15 again four more times. What was the average load time under each network speed? Does the emulator really seem to affect network speed?

16. **7.5 Sending SMS Messages** – For this part of the assignment you'll be sending an SMS message to your emulator through the command line. Return to the command line and enter: "sms send <sender's phone number> <message>"

17. In the emulator, you will receive a text message and see a notification for a text message, from the phone number you entered with the message you entered.
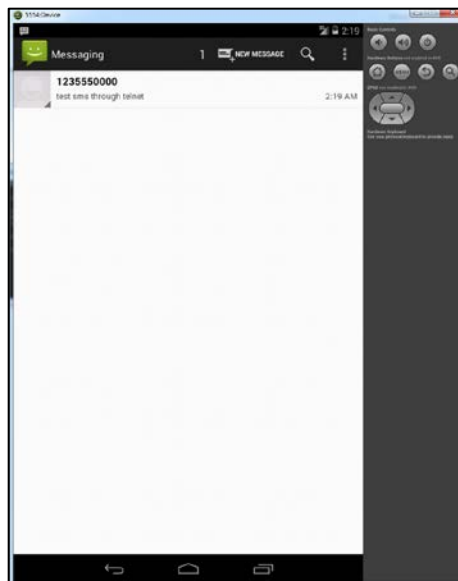
18. **7.4 Modified Hello World** - Remember the first app you made? Let's return to that!

19. In this part you'll modify the original "Hello world!" message of your first app. To do this you need to modify the string value in \res\value\string.xml. Make sure you're back in the Java Perspective.
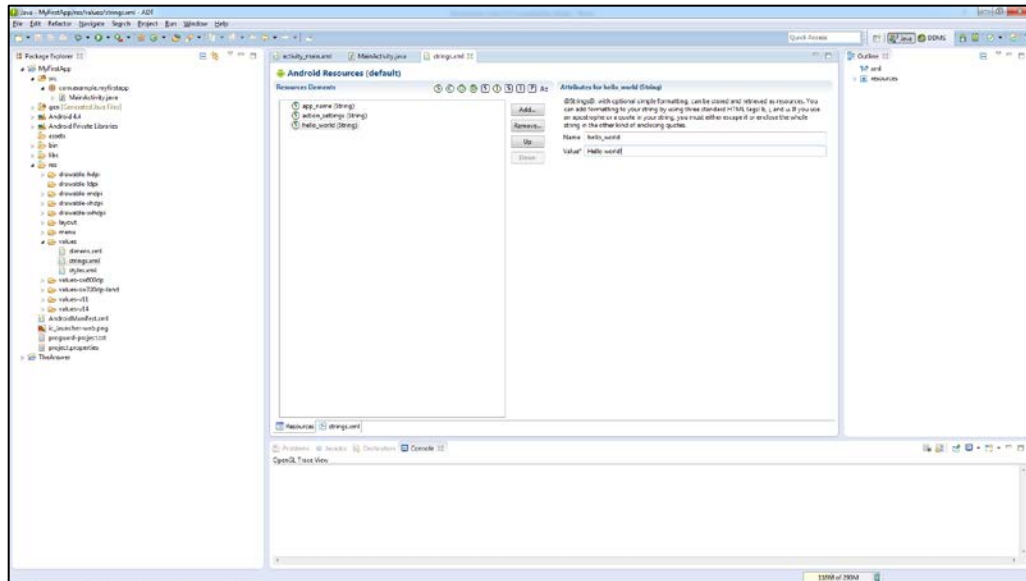
20. Change the value of the string to "Hello world! My name is [yourname]."

21. Now run the app and see the change!

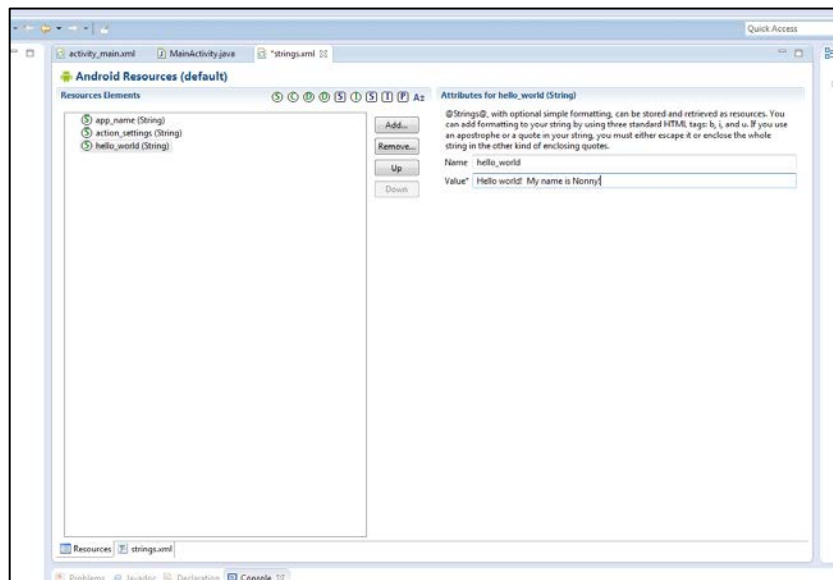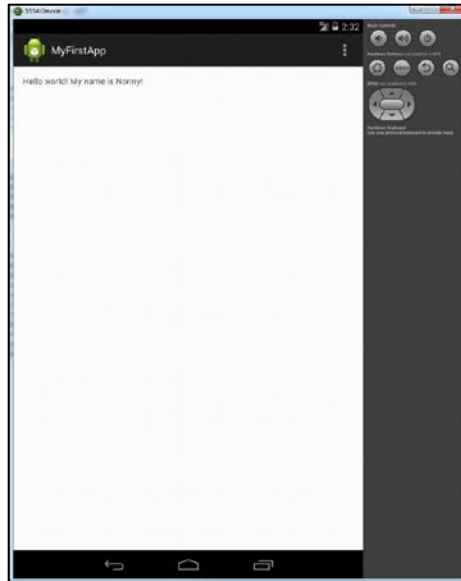22. Now add support for another language such as Spanish! To do this, you'll need to create appropriate string file, run your app, change the emulator instance's default language to Spanish, and then run the app. Your Spanish string, could be: "Hola Mundo! Me llamo [yourname]."

For more information, take a look at:
http://developer.android.com/training/basics/supporting-devices/languages.html

23. **7.5 HierarchyView Tool** - Use the HierarchyView tool (or the Dump View Hierarchy to UI Automator button in DDMS) to display and examine the us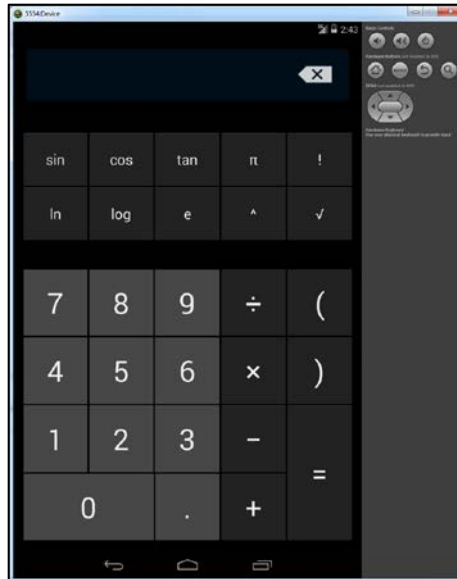er interface of the default Calculator App in your AVD. To do this, start the emulator and open the Calculator App. Be aware that some people report problems with this when running on an actual device.

24. In Eclipse, open the DDMS Perspective, then select "Dump View Hierarchy to UI Automator" button, and select Calculator activity. Or use the hierarchyviewer program from the command line.
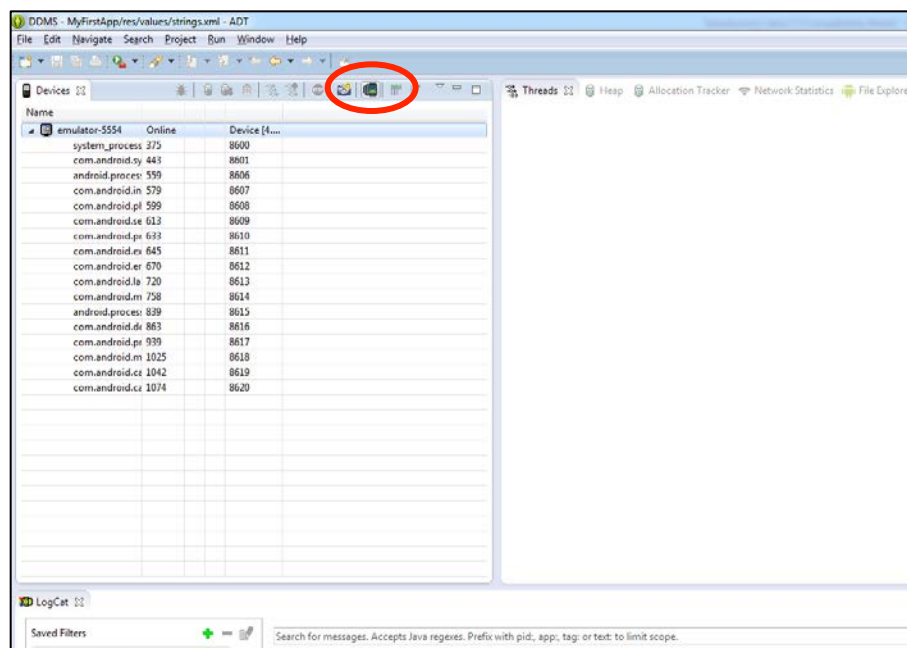
25. A dialog box will appear as the hierarchy viewer does its work.
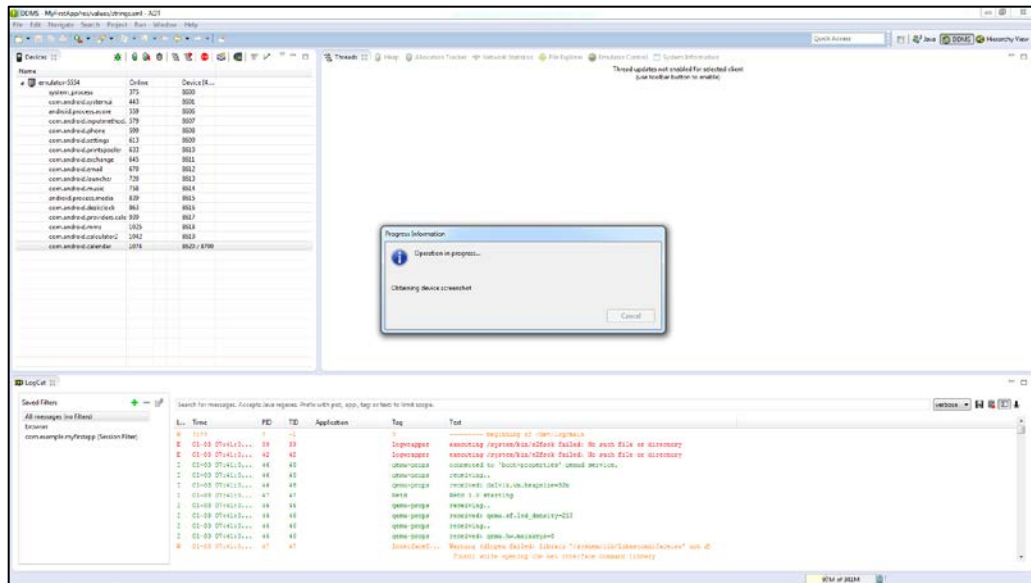
26. Once it's done, you'll see a screen such as the one shown below. See if you can use the Hierarchy Viewer to answer the following questions:
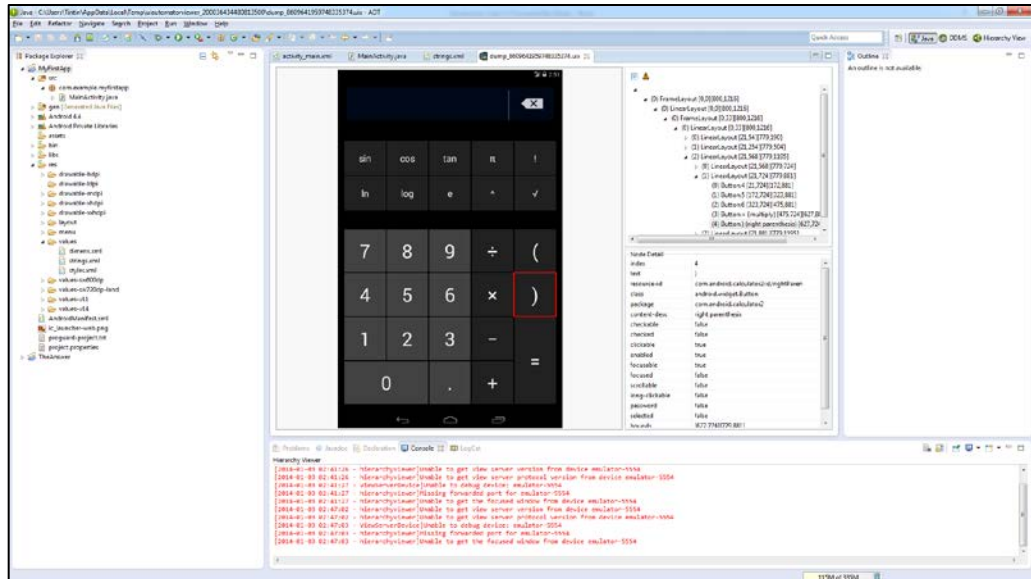
   a. What is the height of the View tree (how many levels deep is the hierarchy)?
   b. Examine the View showing the plus (+) button. What kind of View is its parent? How many children Views does its parent contain? What is the relative index (position) of the plus (+) button View within its parent?

27. **7.6 DeskClock** – Download the source code for DeskClock application. Study it and try to answer the questions below. You don't have to run this app. Just look at the code.

   a. Unzip the file and then import the project into Eclipse. Don't worry about any compilation or syntax errors that might appear.

    b. Open the project's AndroidManifest.xml file. The file has tags corresponding to all of the four fundamental component classes. Identify one instance of each fundamental component. For each one, identify its corresponding source code file and find the line of code that defines the class as a subtype of one of the fundamental components (e.g., class X extends Activity, class Y extends Service, etc.)

    c. Look in the res directory for this application. How many layout directories are there? Which specific device sizes and/or orientations would use each of the different layout directories you've found. For help with this problem you could read the following blog post: http://android-developers.blogspot.com/2011/07/new-tools-for-managing-screen-sizes.html

    d. Look again in the res directory. How many different values directories are there? Find two examples of resource strings that can be configured for different languages.

    e. Look at R.java file that was generated for this application. There is a field in the id class called "cities". In which function and class does this field get used? How is it referenced in the application code? Hint: Use Eclipse's Open Call Hierarchy feature.

28. **7.7 LogCat** – Download the source code for WikiNotes application and run the application. For this task, you'll be familiarizing yourself further with LogCat.

    a. Using the Logcat feature of DDMS, filter out messages relating to the WikiNotes application.


    b. What is the log tag used by this application's main Activity? Where do you see that in the application's source code?

    c. What is the message associated with the first Log entry emitted when this application starts running? For more information, read the following webpage:
http://developer.android.com/tools/debugging/debugging-log.html

29. **7.8 Heap View** – Use DDMS to view heap usage for the WikiNotes process. In the Devices tab, select the WikiNotes process (com.google.android.wikinotes). And click the Update Heap button.
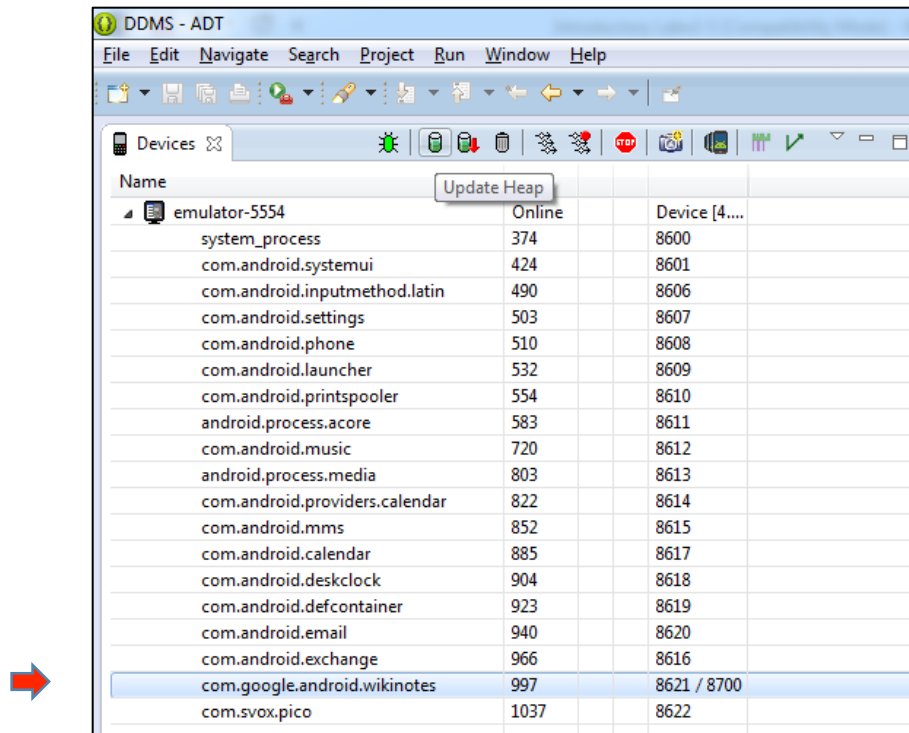

a. In the Heap tab, click CauseGC to enable the collection of heap data.
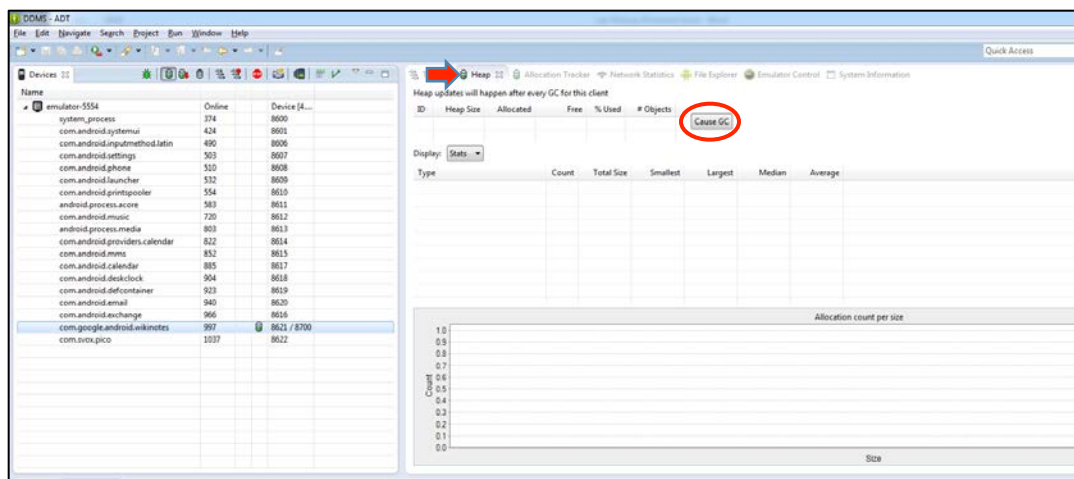
b. How many objects have been created?
c. Which type of Object has by allocated most frequently?
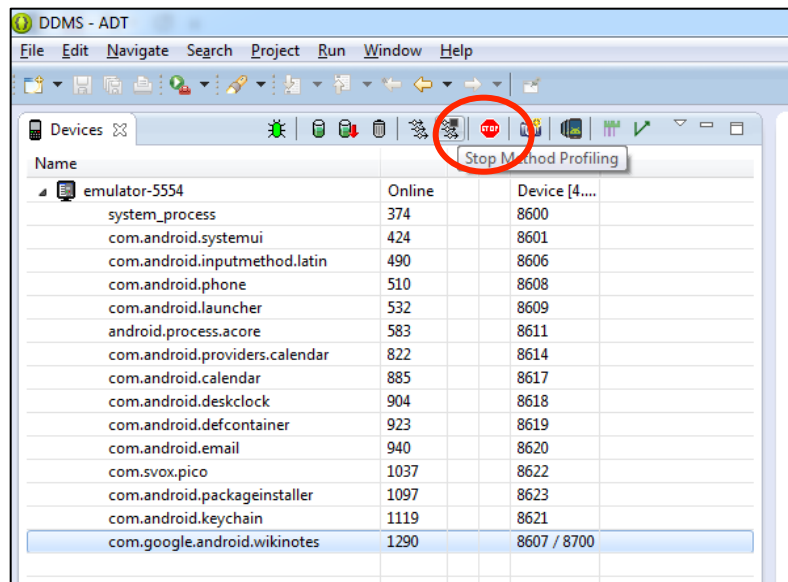d. How much memory has been allocated for this type of Object?

30. **7.9 Method Profiling** - Use DDMS to view method execution sequences for the WikiNotes process. In the Devices tab, select the WikiNotes process and then click the Start Method Profiling button.
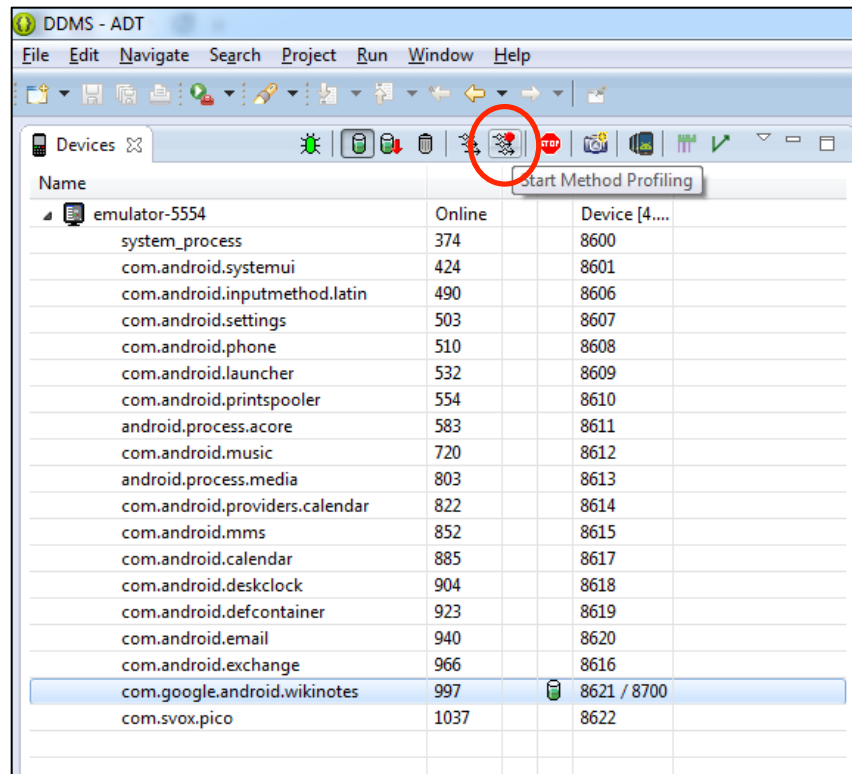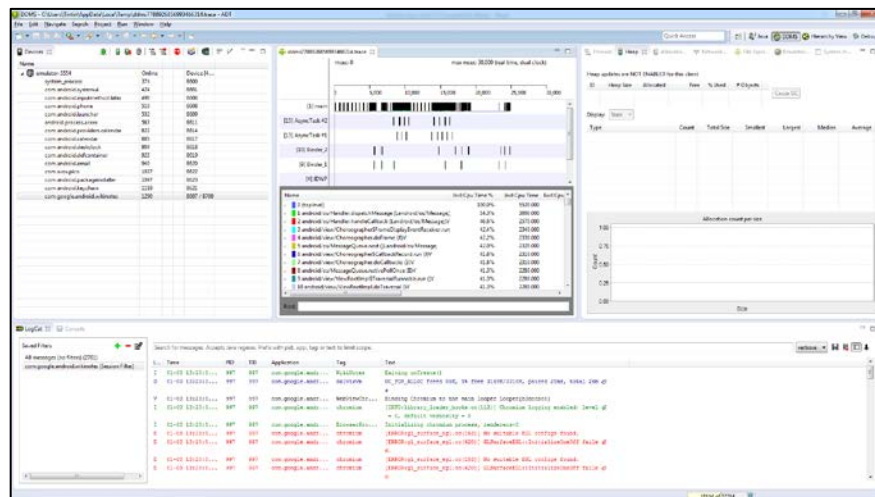

Interact with the WikiNote application.
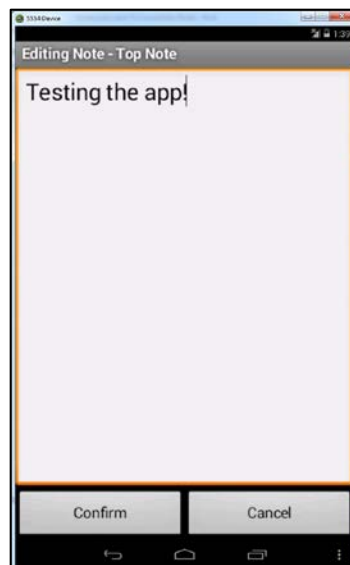
and then click the Stop Method Profiling button.


Your screen should look something like this:

a. Look at the profile panel. Find a method that is part of the WikiNotes application (it's in the com.google. android.wikinotes package).
b. What is the name of the method?
c. In what file is the source code for this method?
d. What are the exclusive real time and the inclusive real time spent in this method? Why are they different?

31. **7.9 Debugging and JUnit Testing** – Download the source code for the simple calculator application with corresponding JUnit Test project at: http://code.google.com/p/simple-calc-unit-testing/downloads/detail?name=FullCodeDownload.zip&can=2&q= (Note – you may need to download a new API level package for this app, using the SDK manager.)

32. Import the two projects into eclipse and run the SimpleCalcTest app in the emulator.
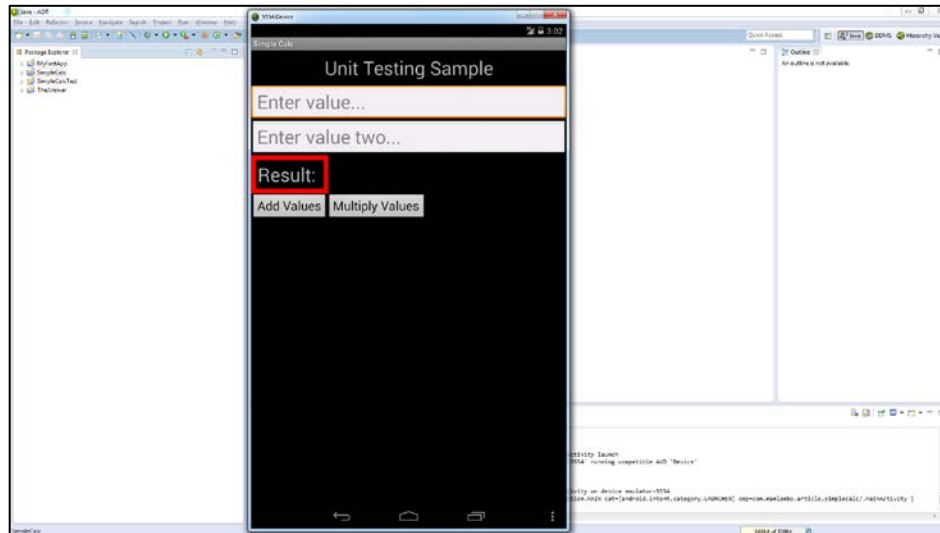
33. Set a breakpoint at the line 91 (Function testSubtractValues() of MathValidation.java) and start Debugging the project.
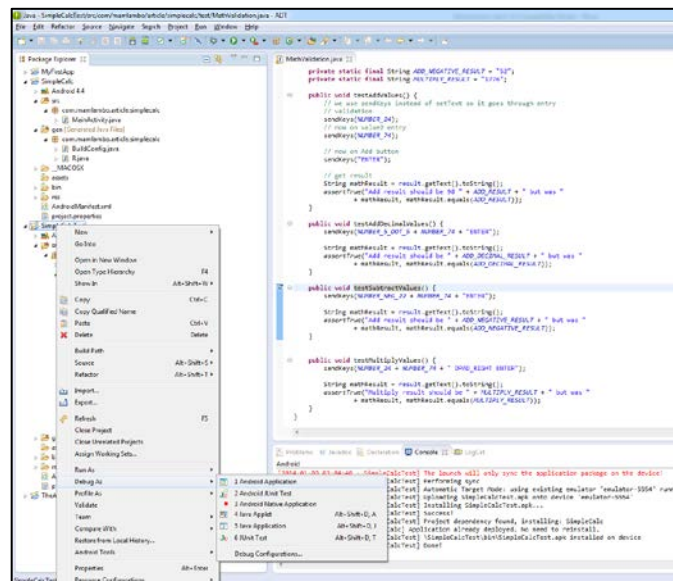
34. Also try debugging the app with Android JUnit Tests. Right-click > Debug As > Android JUnit Test.
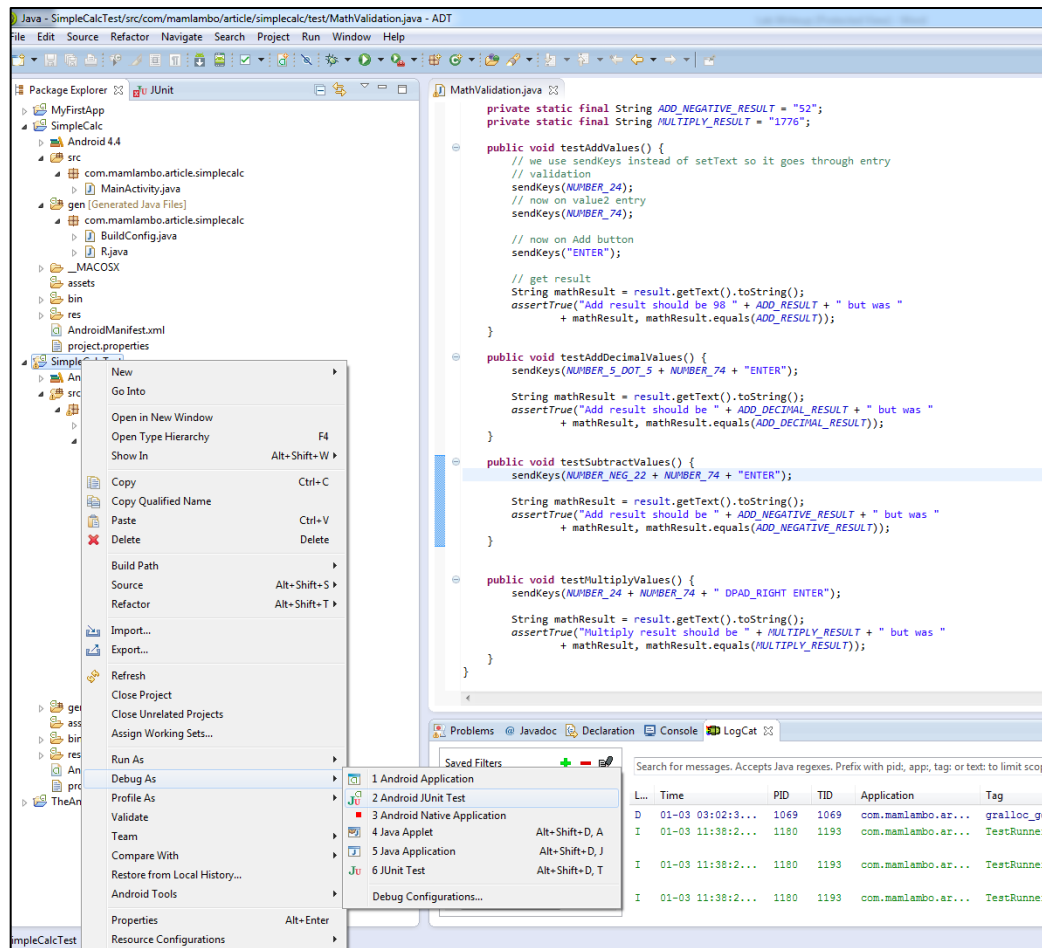
35. Step in the function, debug it and use the watch windows to keep track of the value of the variable "mathResult." Then please try to answer the following questions.

    a. What is the value of "mathResult" after line 94?

    b. Is the value correct?