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.

This lab contains the following Parts.

1. Set up Android Studio.
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 Android Studio.

In this part you will download and install Android Studio which will be the Integrated Development Environment (IDE) used for this course.

1. Download Android Studio from http://developer.android.com/sdk/index.html. Click on 'Download Android Studio'.
2. Open the executable file android-studio-<xxx>.
3. Once the setup loads, you will see the Welcome Screen.
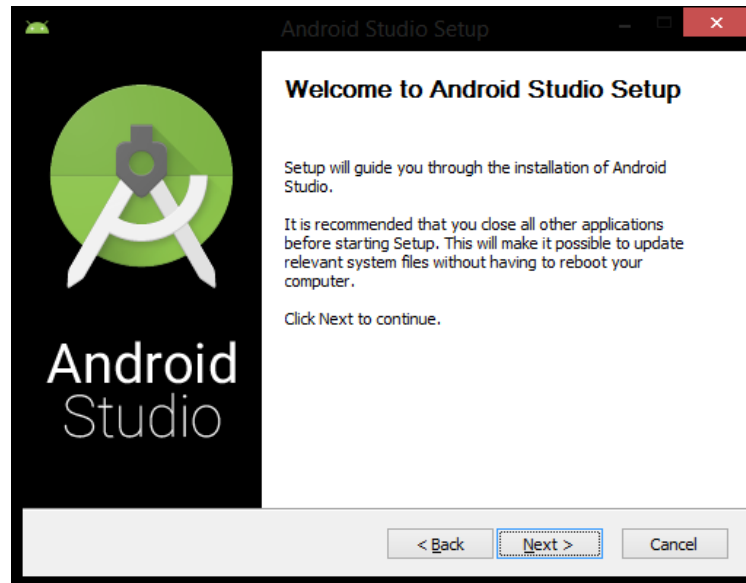
4. Click 'Next >' on the Welcome Screen.
5. When choosing components, ensure all of the checkboxes are checked in for each component to install. Once you are done, click 'Next >'.
6. Agree to the Android Studio and the Intel HAXM License Agreements after reading them.
7. Verify the install locations meet the installation requirements and click 'Next >'.
8. You may or may not see the emulator setup settings, just click 'Next >' after selecting the RAM size.
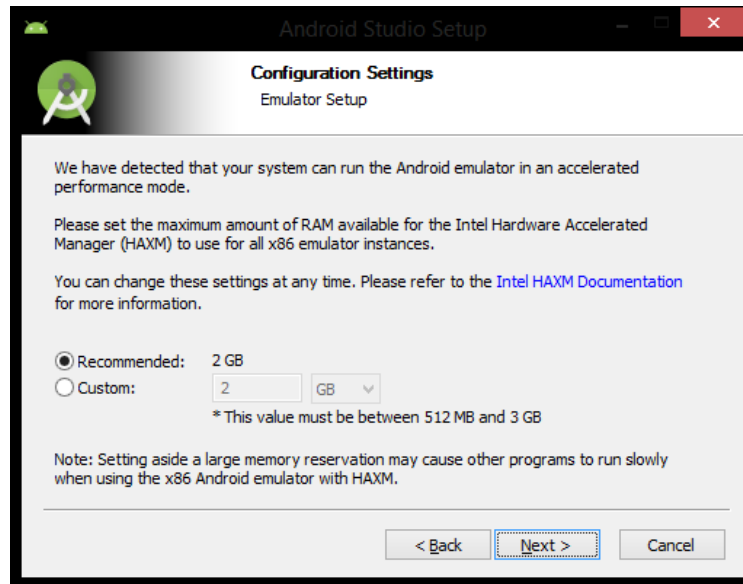
9. Finally, click 'Install'. You will see which operations are currently running in the installation process and a progress bar displaying their progress.
10. Once the installation process is finished click 'Next >'.
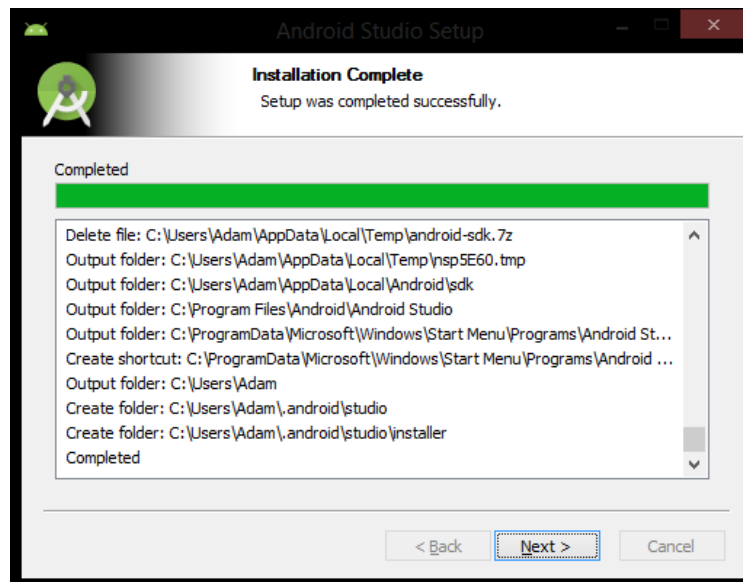
11. Android Studio is now set up. Check on 'Start Android Studio' and click 'Finish'.
12. You will see the Complete Installation screen below.
13. If you had a previous version of Android Studio installed prior, then check either the first of second radio box. Otherwise, check the last radio box and hit 'OK'.
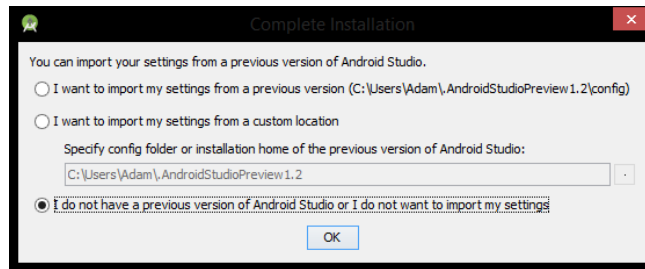
14. As Android Studio starts, the splash screen will appear.

15. After the splash screen you may see some additional setup operations run, such as downloading components.
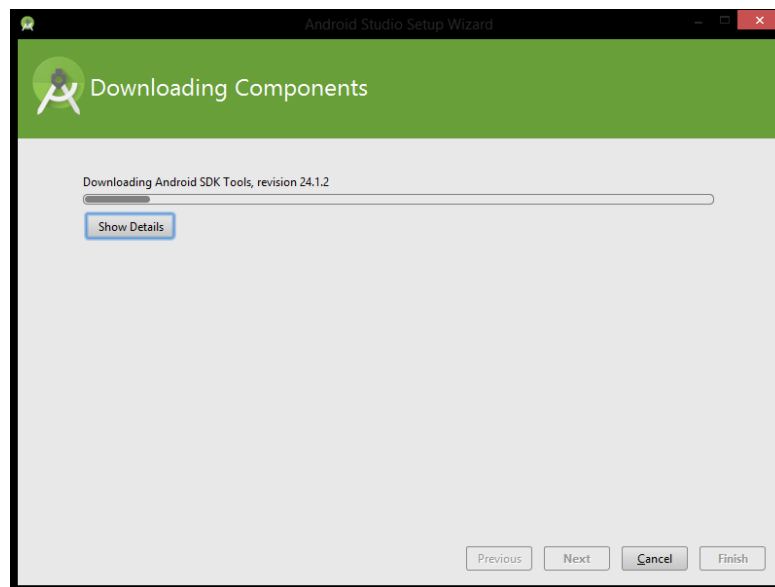
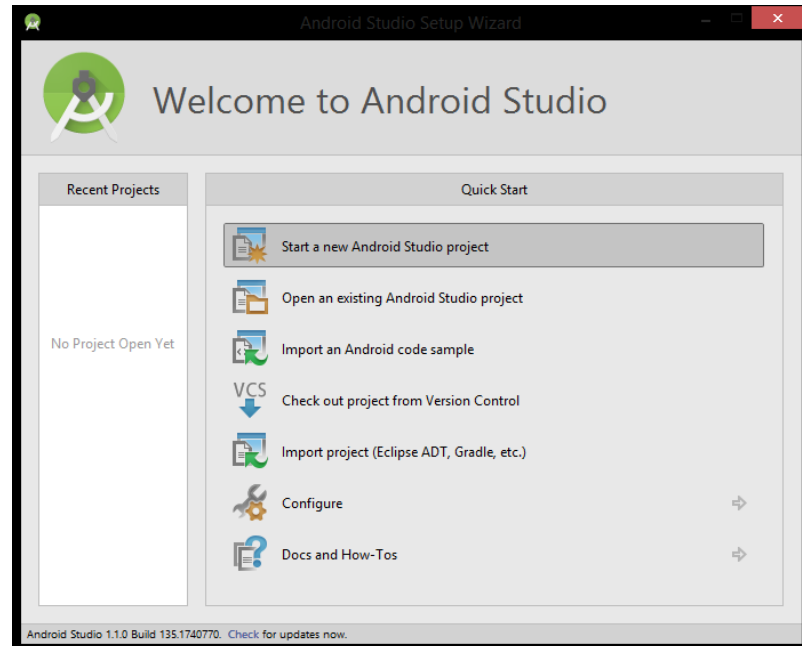16. Once it is finished, click 'Finish'.
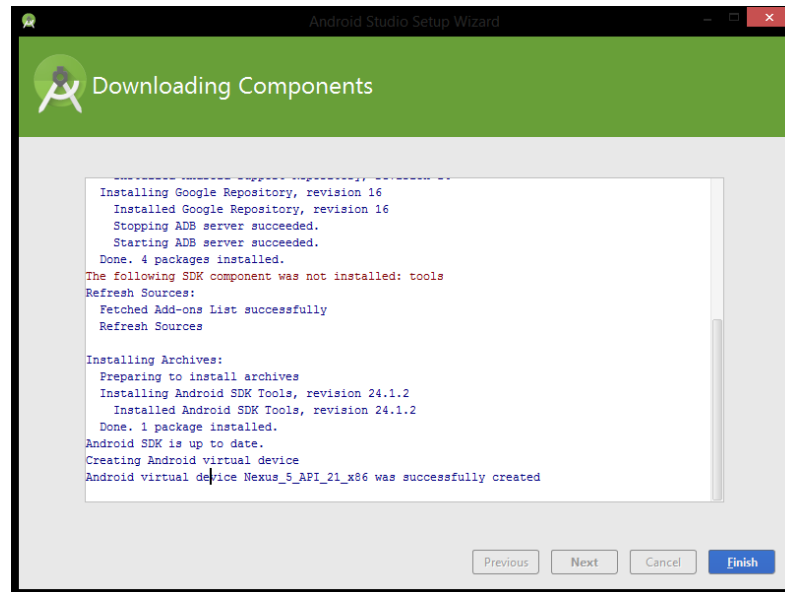
17. Welcome to Android Studio! In the next part we will start our first project.
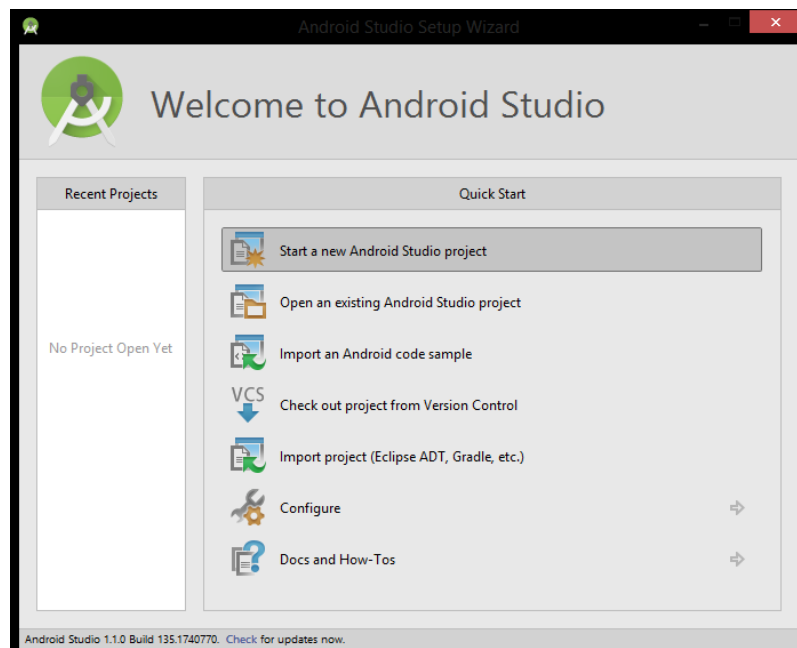
Part 2 – Creating A New Project

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

1. At the Welcome Screen, click on 'Start a new Android Studio project'.

2. Enter the application name 'MyFirstApp' and note where the project is located. The AndroidStudioProjects folder is the default location for new projects.
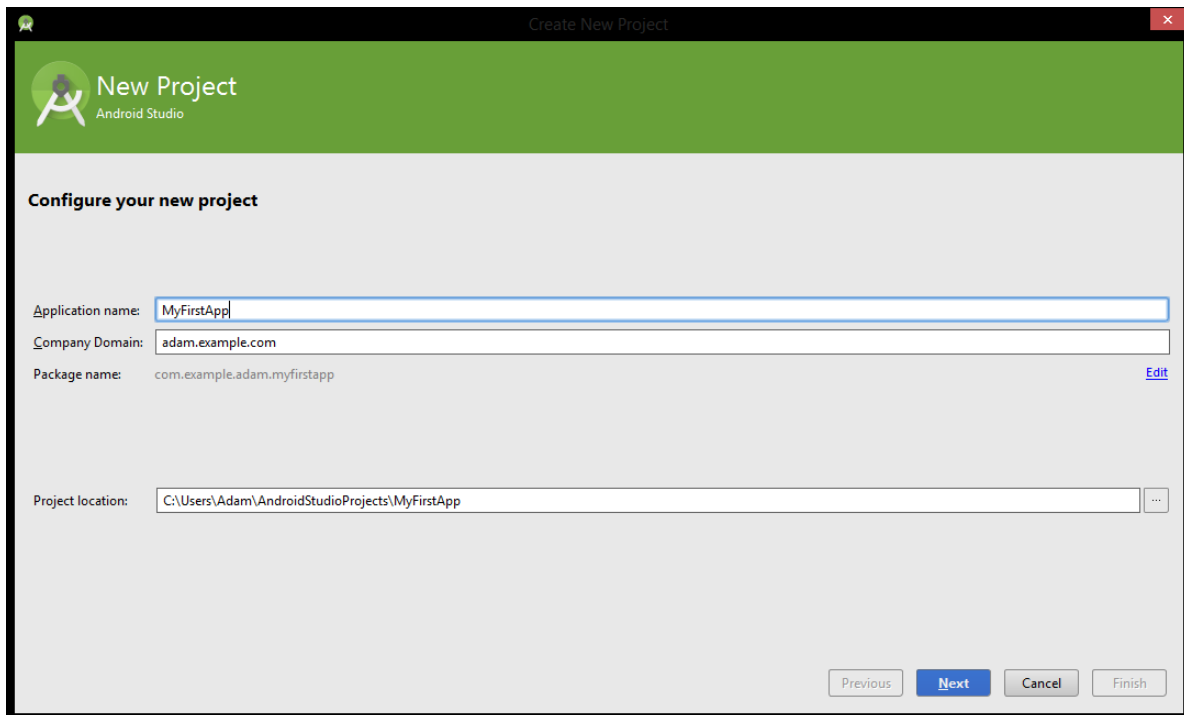
3. Select which devices you would like your app to run on. For now we will be working with 'Phone and Tablet'. Make sure to set the Minimum SDK version to API 18 for this course.
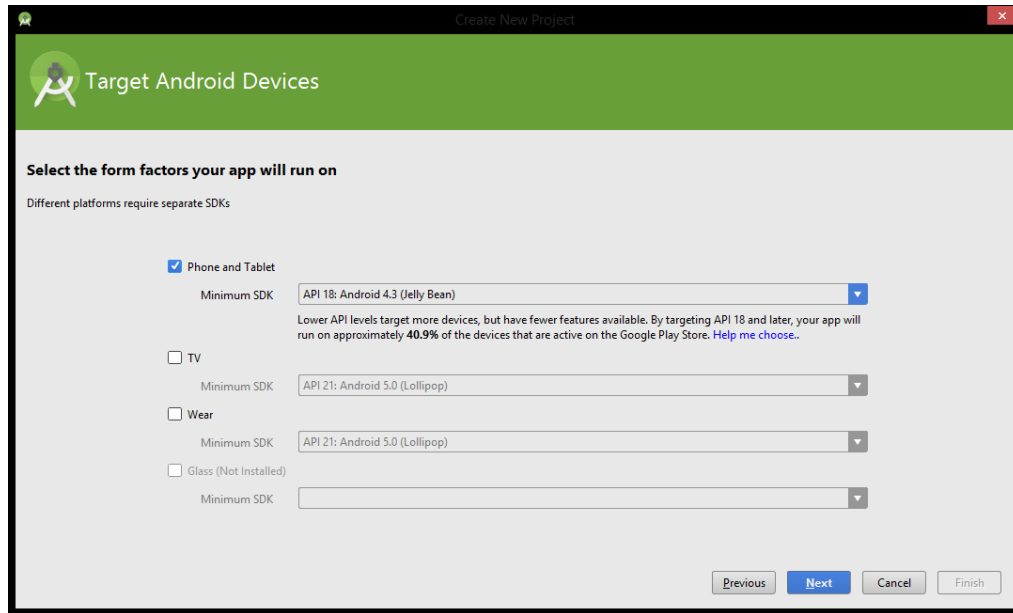
4. Select 'Blank Activity' from the 'Create New Project' dialog box and click 'Next'.
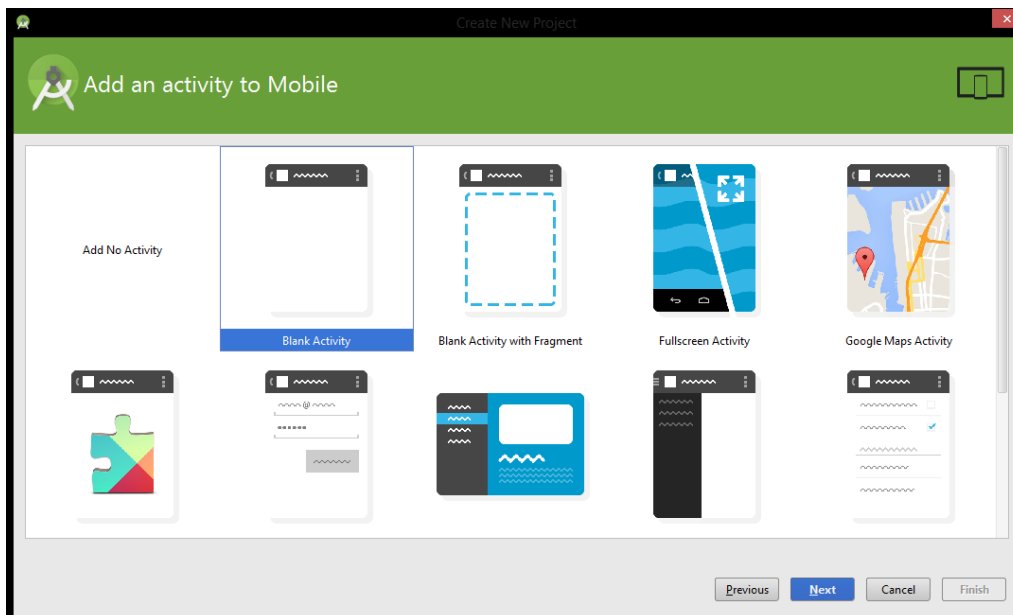
5. In the next window, leave all the settings as default, and then click Finish.
6. Android Studio will now create the project and build it.
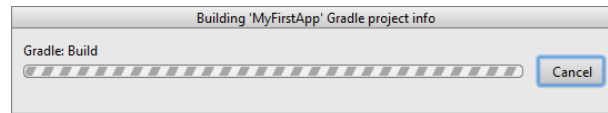
7. You may see a security alert if you are on Windows, click 'Allow access' to continue.


8. Once the Android Studio IDE fully loads, you will see the screen below. If you see text rather than the layout designer make sure the 'Design' tab is selected.
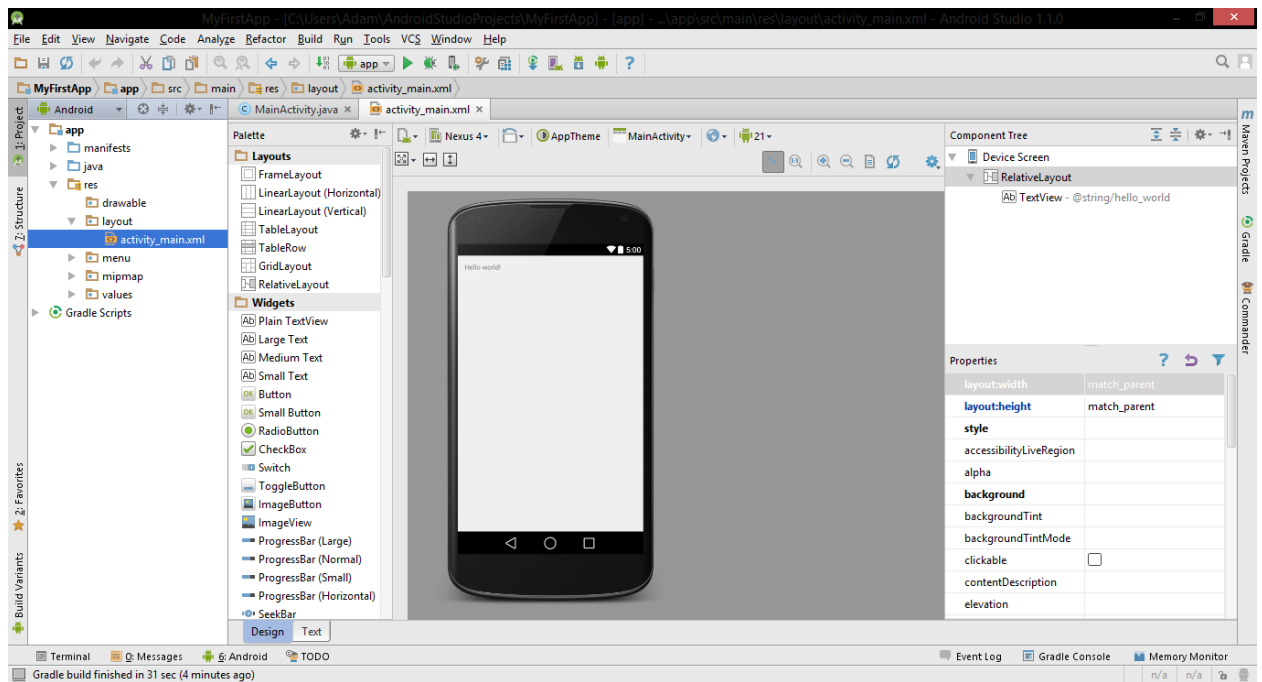
9. The opening screen is the Design View of the activity_main.xml file. You can already see the words "Hello World!" on the App's User Interface.

10. If you click on the Text tab you can see the layout file underlying the user interface.
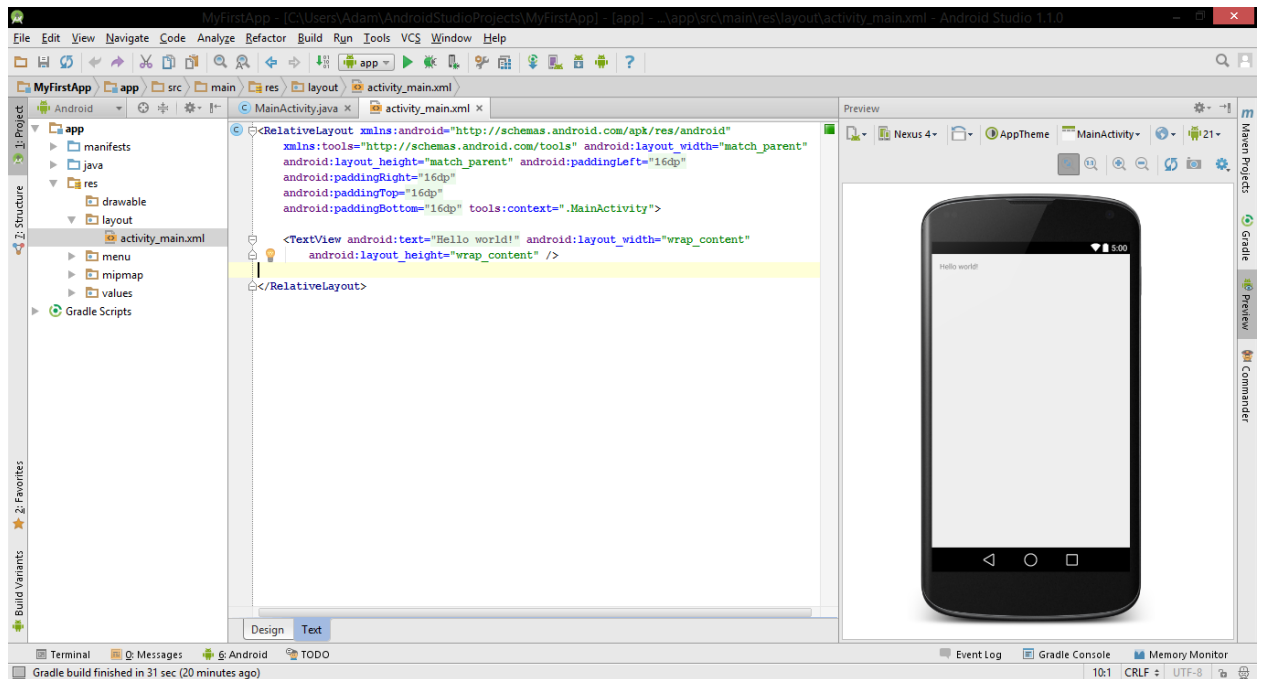
11. To view the backing code for this activity, double click on 'MainActivity' inside of the Project directory tree. This file is located in: 'java' > 'com.example.adam.myfirstapp' > 'MainActivity'.
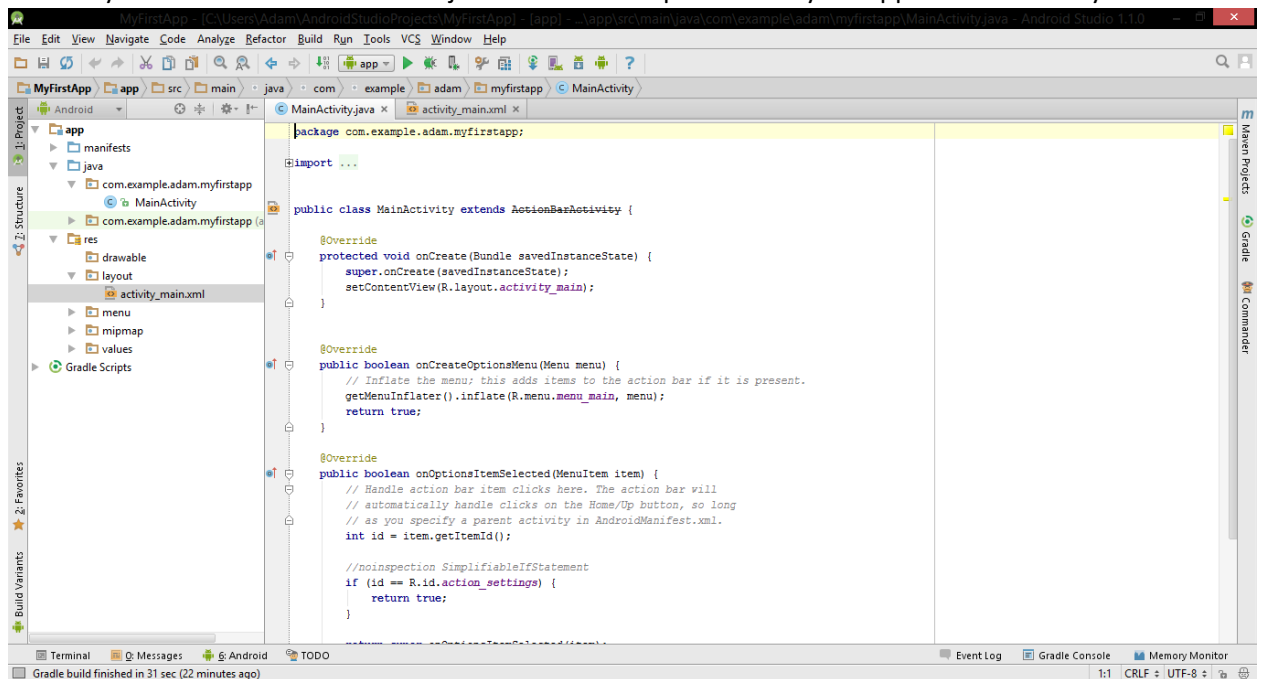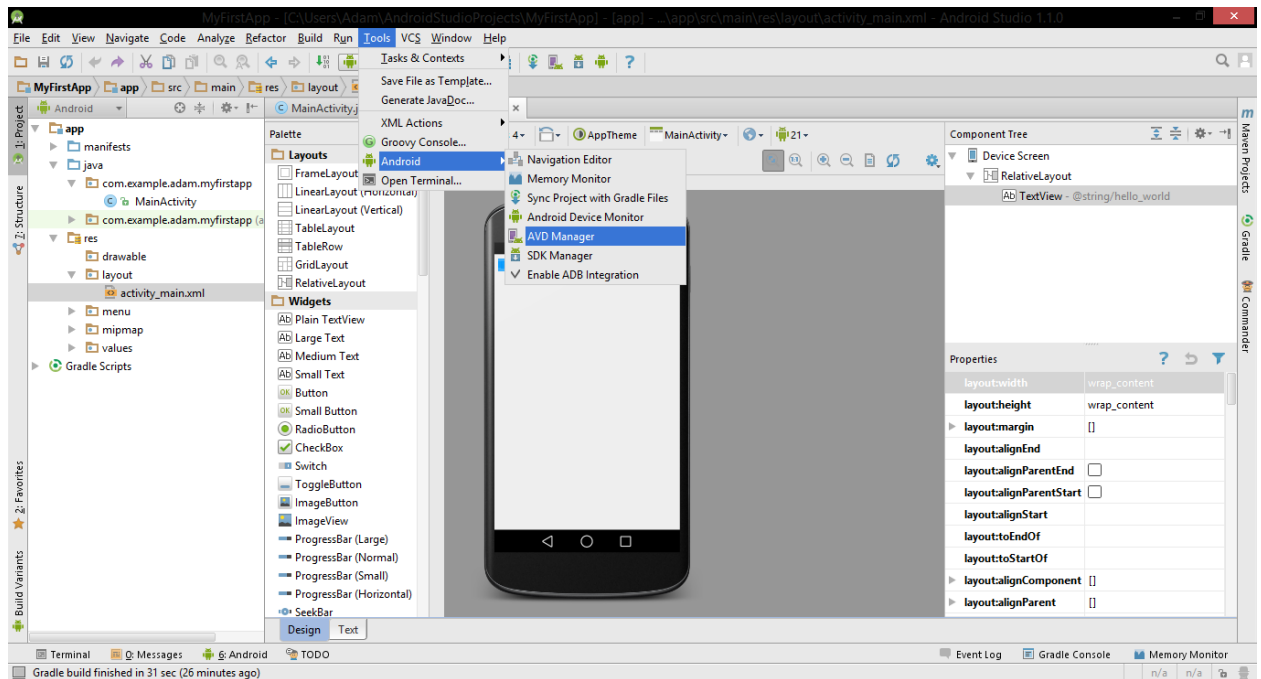

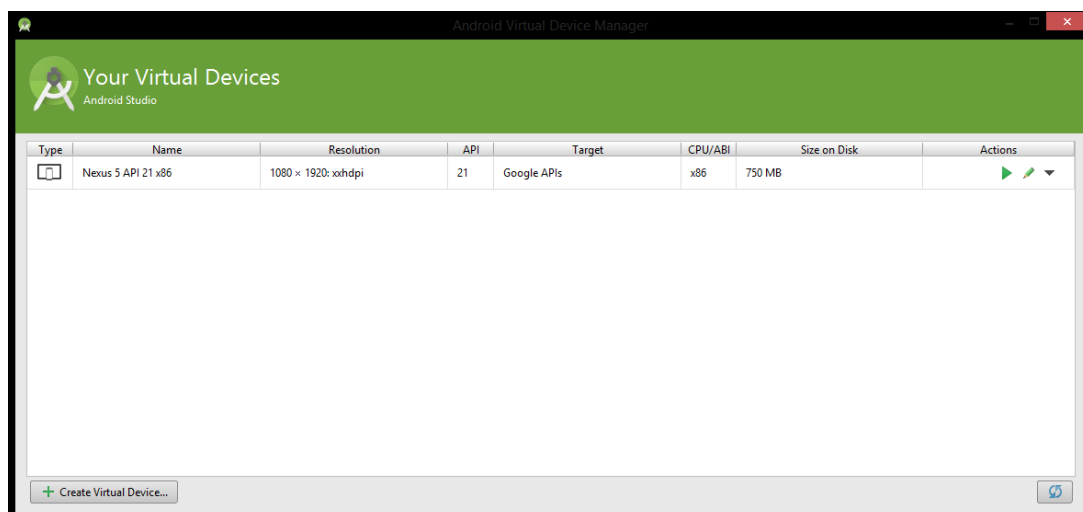In Part 4 we will show you how to run this app in the Android Emulator.

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 Tools > Android > AVD Manager from the Android Studio menu bar.


2. A new dialog box will pop up.

3. Click "Create Virtual Device" to create a new Android Virtual Device (AVD).
4. Another dialog box will pop up displaying various pre-made AVD templates. Select whichever device you would like to emulate and click 'Next'. For example, select the Nexus 5.
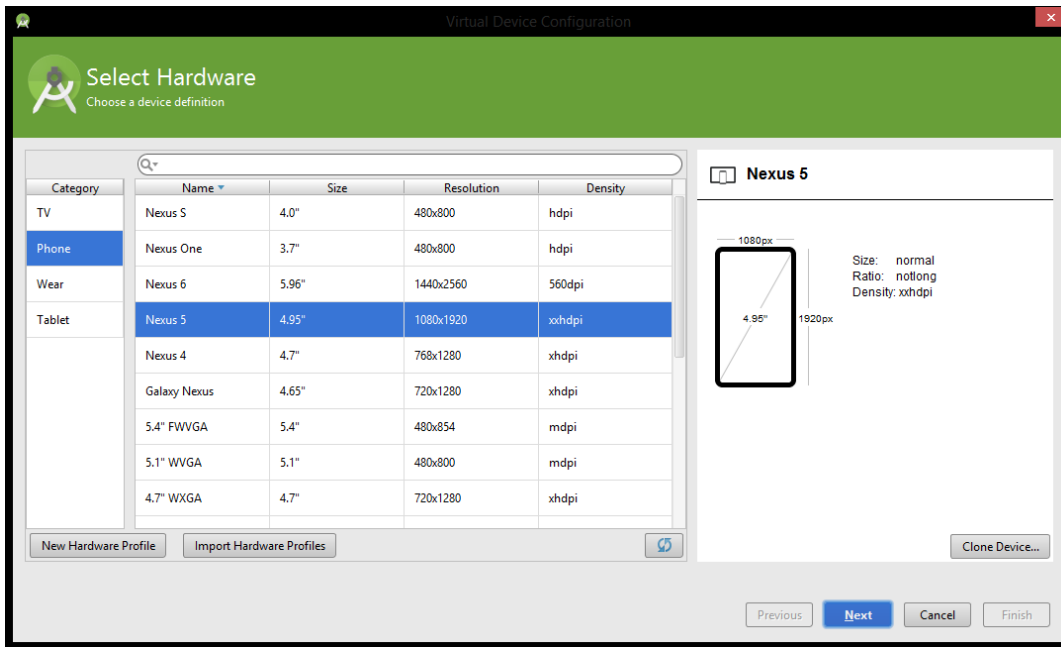
5. Select the appropriate System Image for the virtual machine. To allow users with limited computer memory to participate, all of the class projects will be tested against API level 18. If you haven't downloaded that already, make sure to download it now, by clicking on the "Download" link.
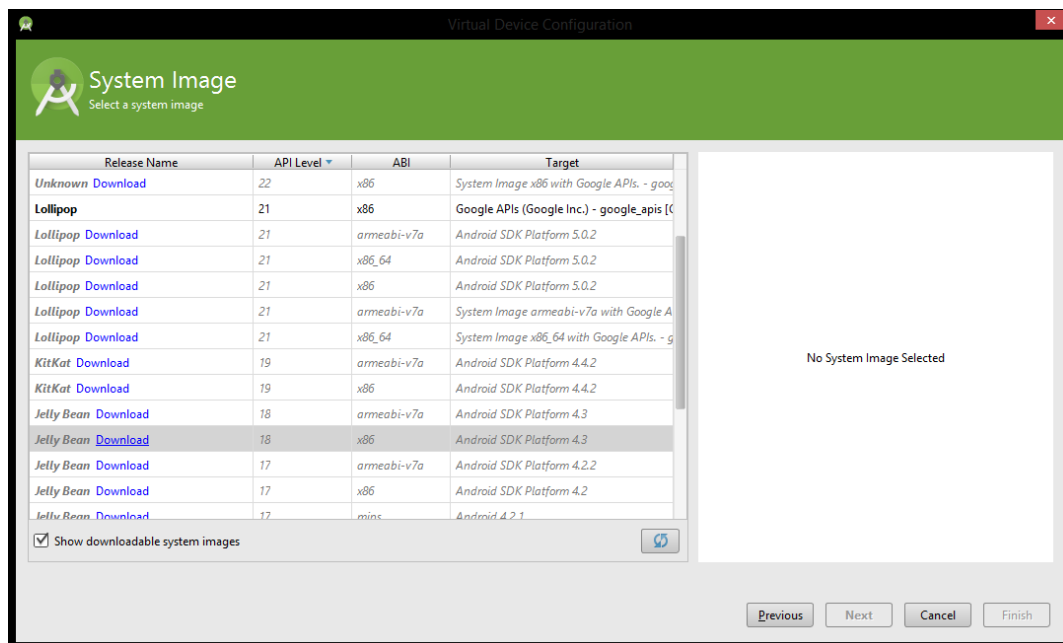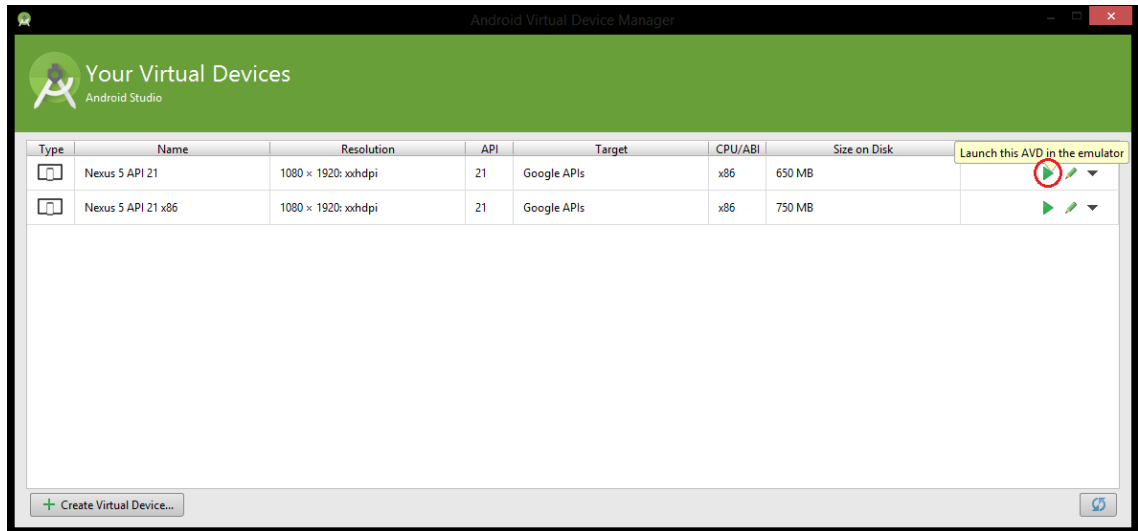
6. Once the Download is finished, click 'Finish'.
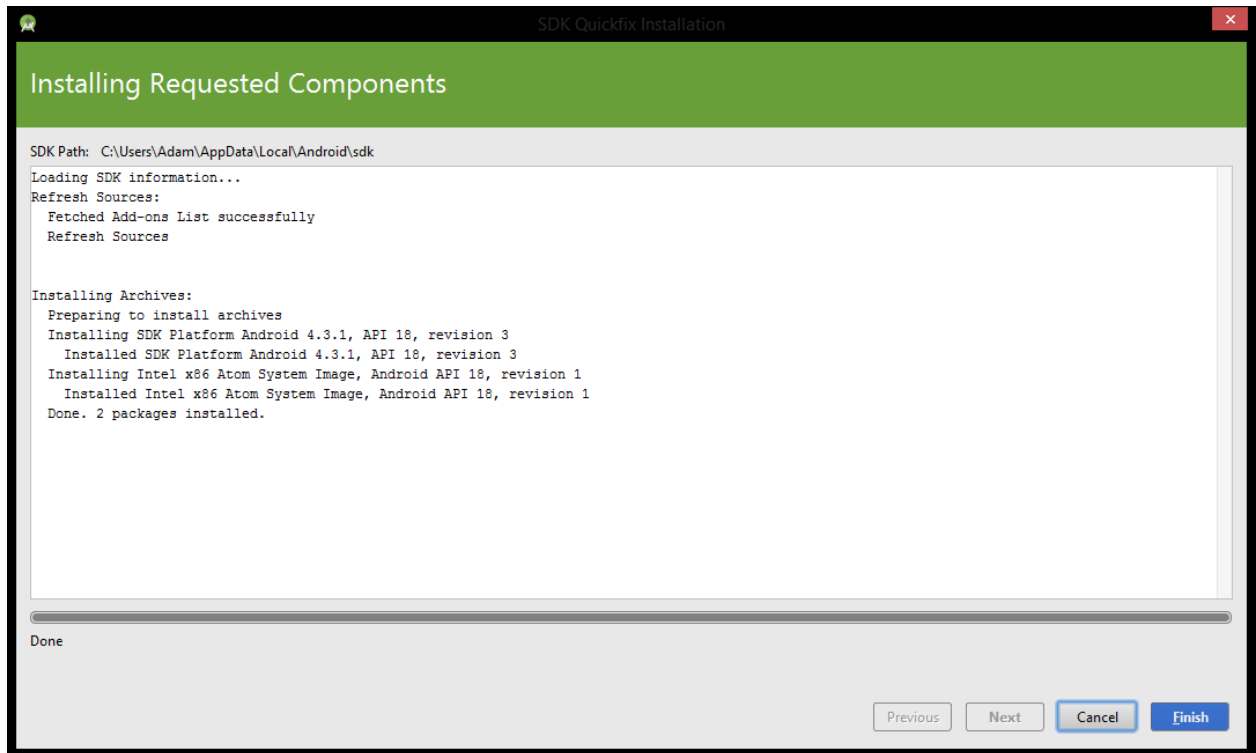
7. Click 'Next' once you have returned to the previous screen.
8. You can keep all of the default selections in the next screen and hit 'Finish'.
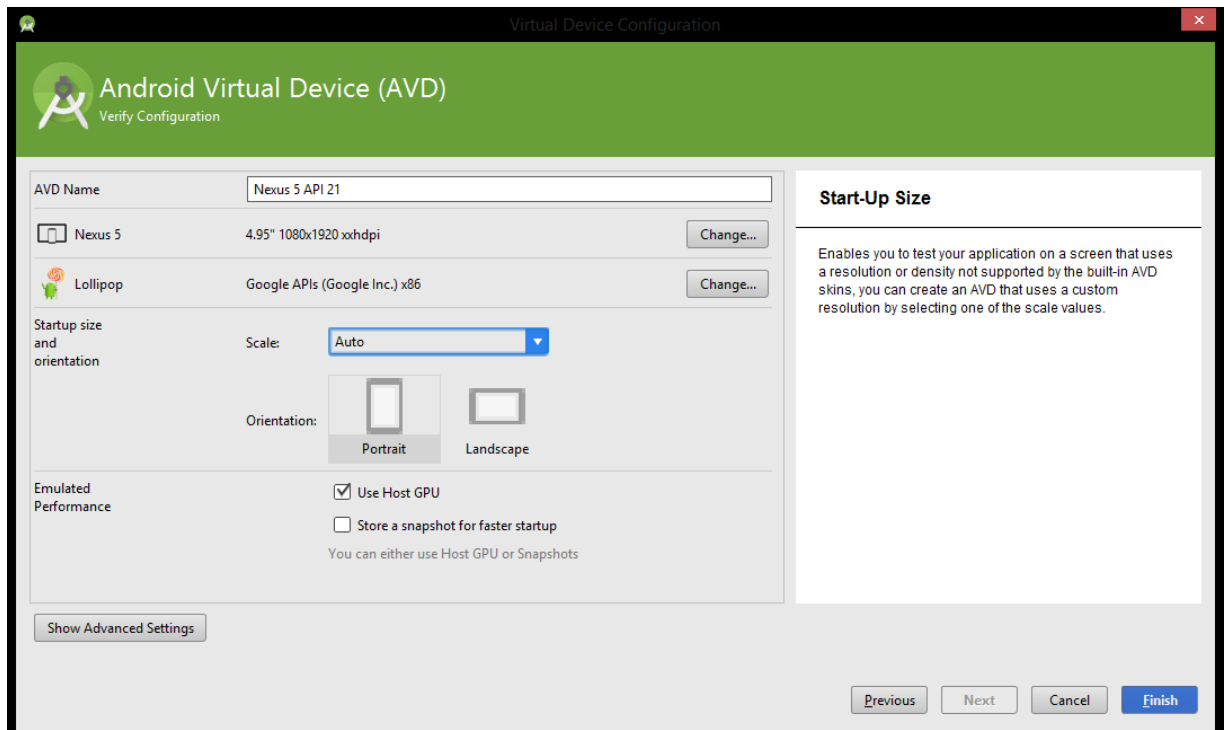
9. Now click on the green 'Play' icon to start the emulator, after clicking on it you can close the Android Virtual Device Manager.

10. As the emulator starts up, you will see a progress dialog appear in Android Studio.

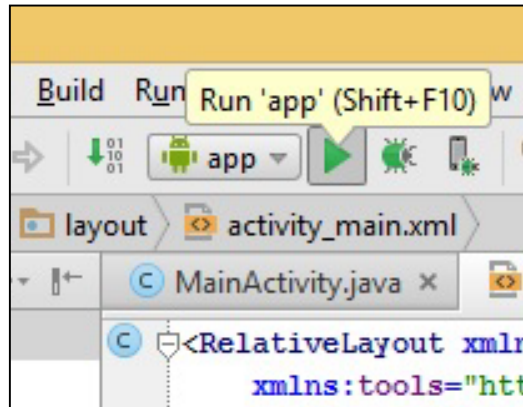11. Next, the emulator will appear and start its boot sequence.

After the device has booted, the emulator will be ready for user interaction.
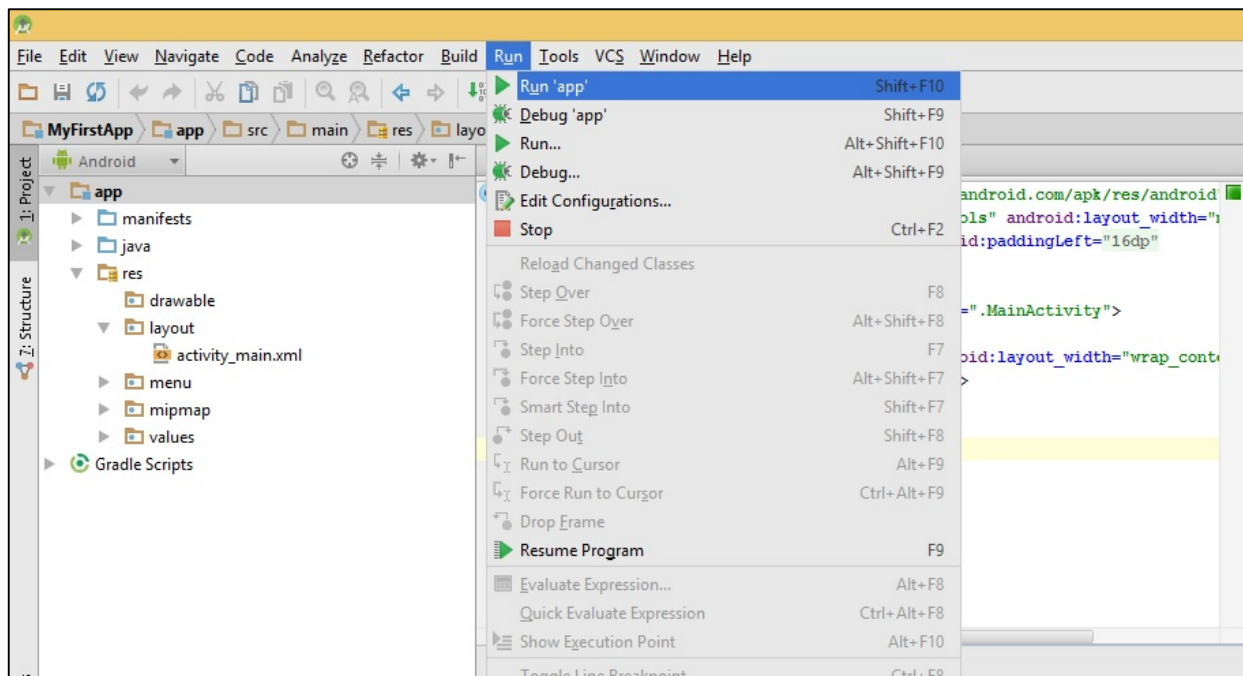
Part 4 – Running Your First App

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

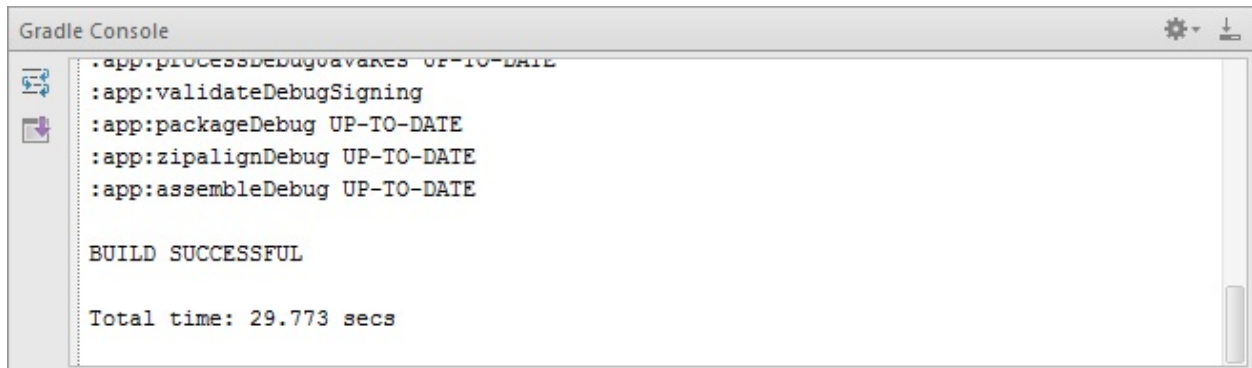1. There are two ways to run the app.

Method 1: Return to Android Studio and simply click on the "Run 'app'" Button (Shortcut: Windows - Shift + F10, Mac - Ctrl + R)

Method 2: Return to Android Studio and select Run > Run 'app'.

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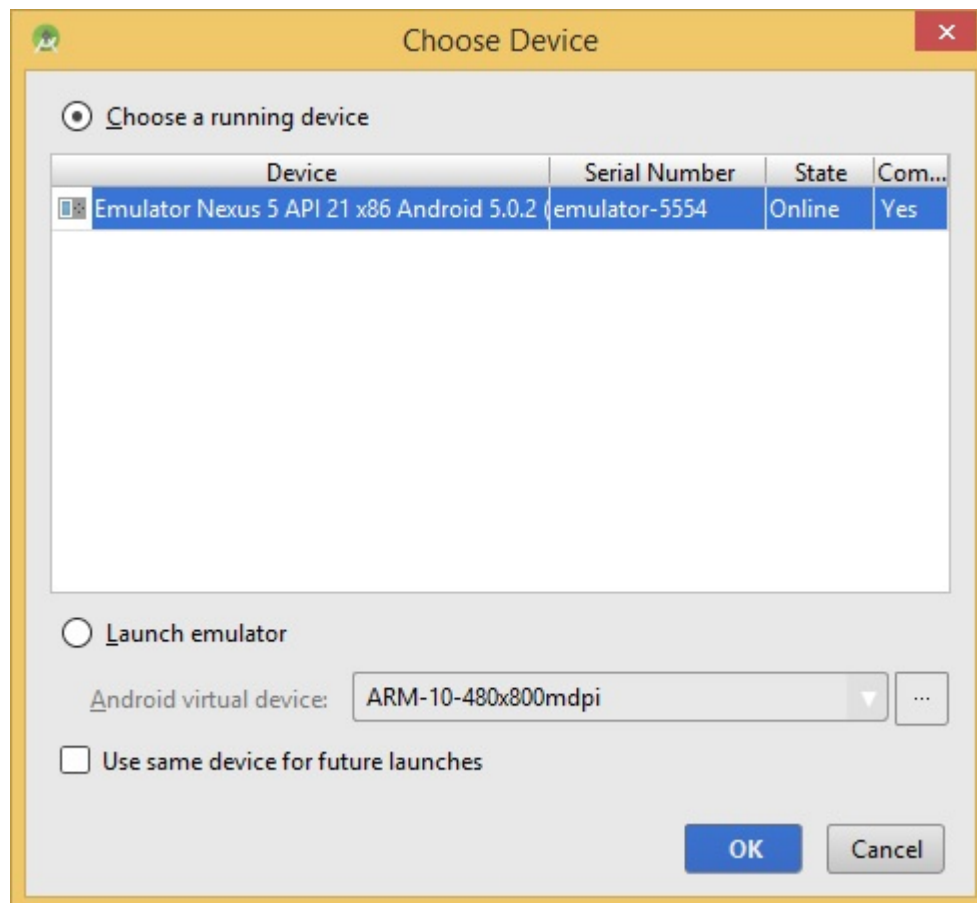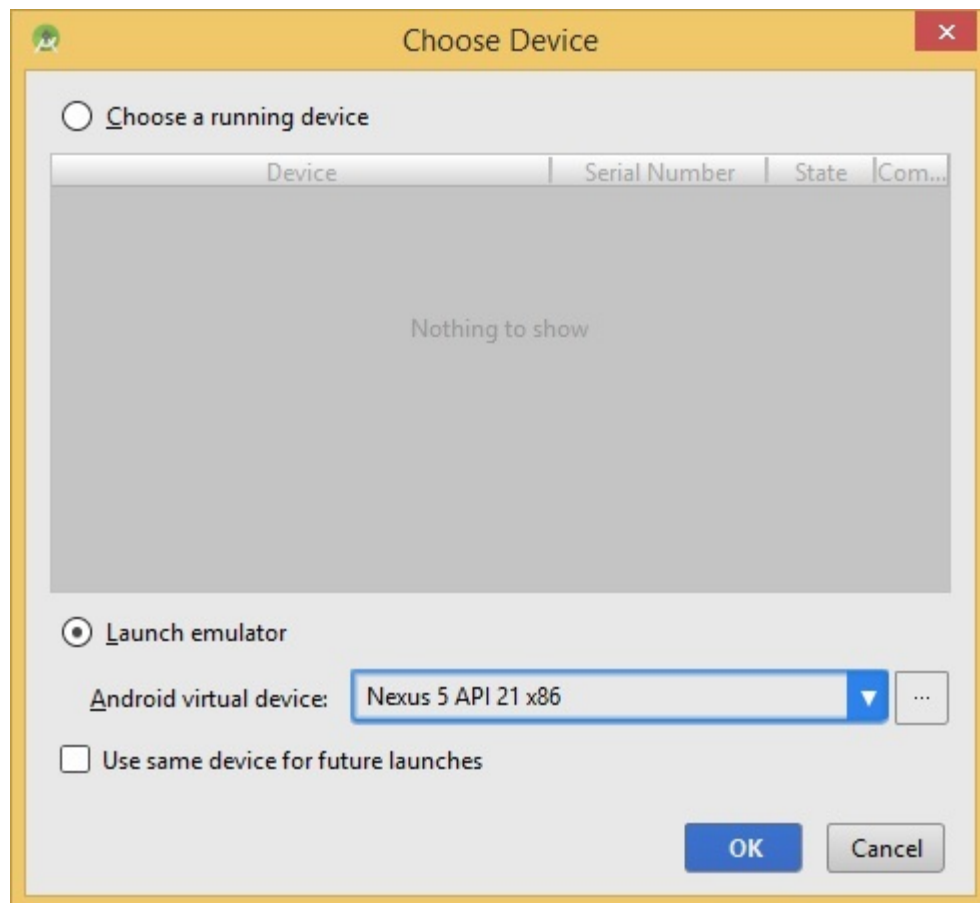

3. After you see the words, BUILD SUCCESSFUL, a dialog box will pop up asking you to choose a device on which to run the app.
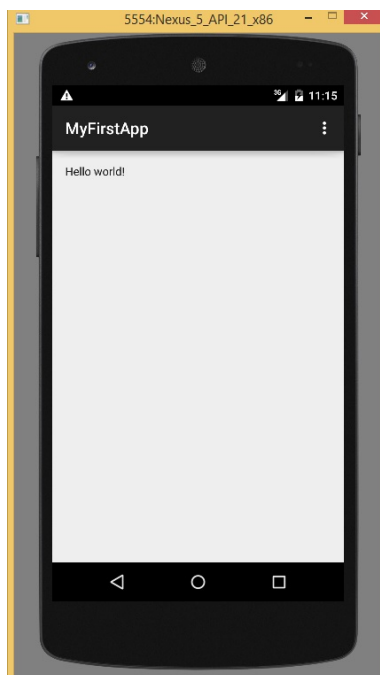
If you do not have a running device, you can choose an AVD to launch in the emulator. The system will start an emulator and run the app in it.

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

5. 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.
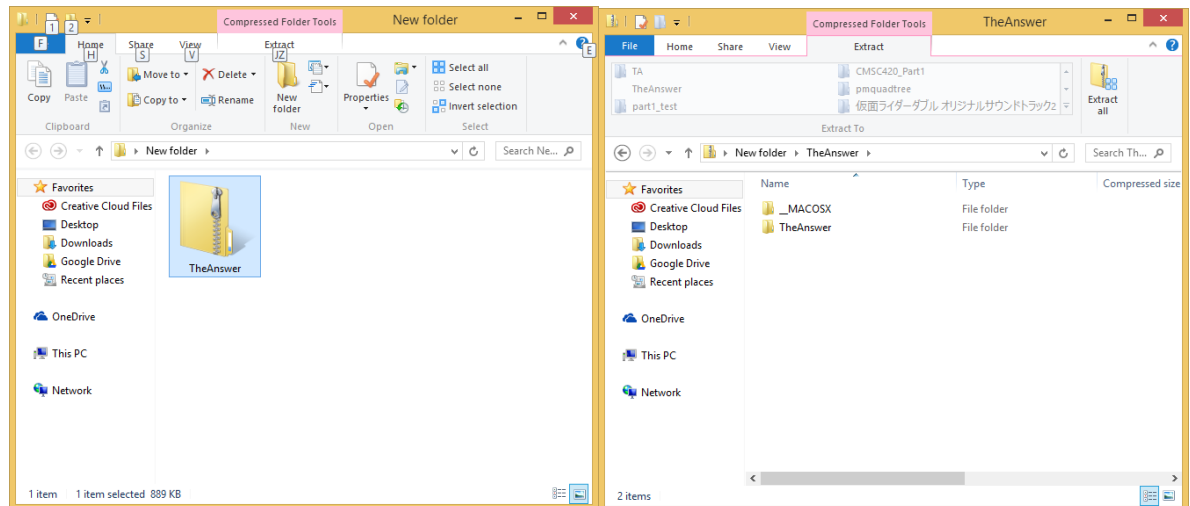

3. Return to Android Studio. Select File > Import Project… from the menu bar to import the application project.
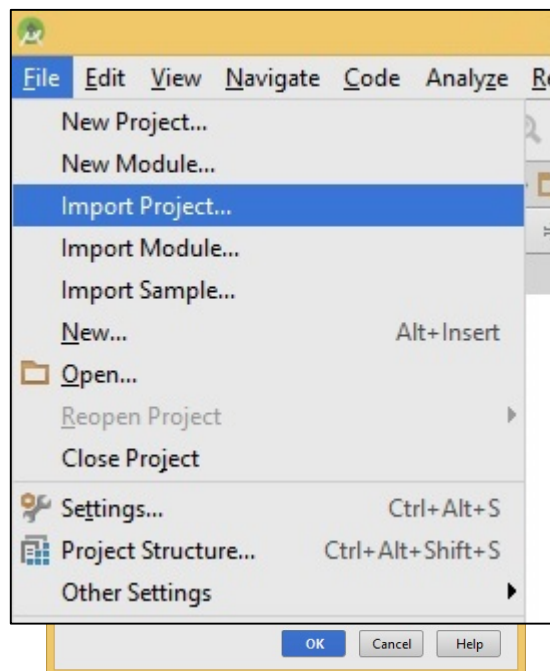

4. Next, in the dialog box that appears, browse and select the Project that you want to import. For this example, select "The Answer". Then press OK Button.

If the project was originally created in Eclipse, a dialog box will show up asking for the Import Destination Directory. For this example, keep the default setting and press the Next button.
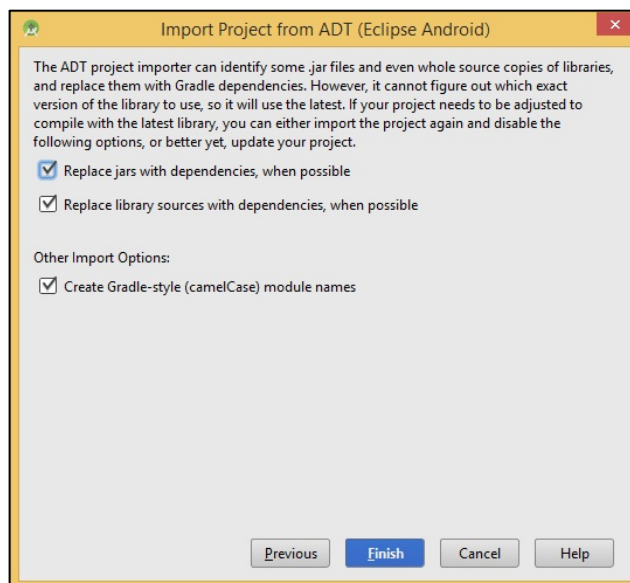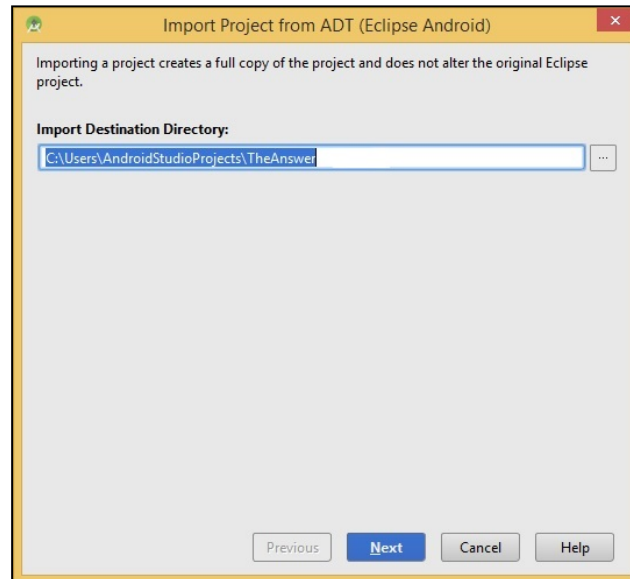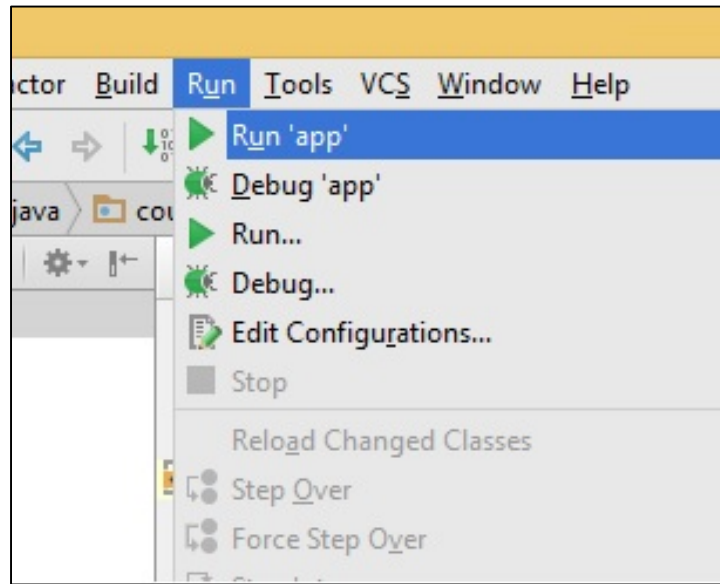

Keep all default settings and press Finish Button.

5. At this point the application should appear in the project window on the left side of the IDE.
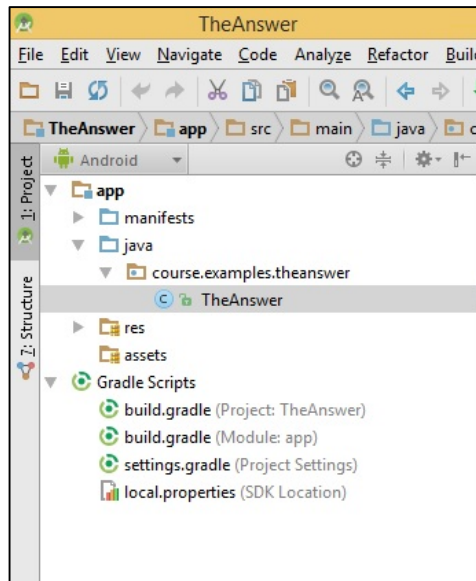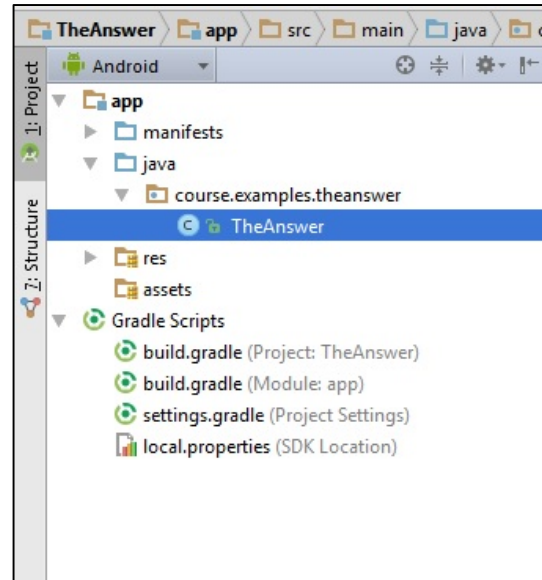
6. Select Run > Run 'app' from the tool bar.

7. 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 Android Studio debugger to debug the TheAnswer application you imported in Part 5.

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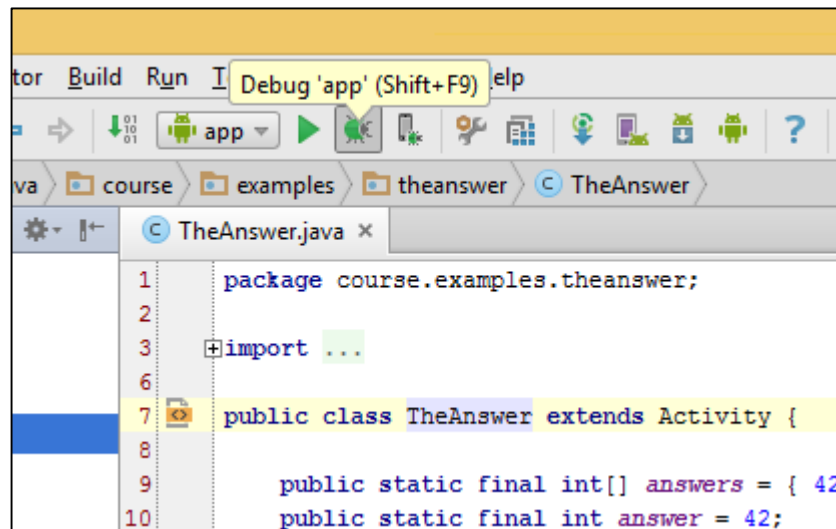

2. On this screen, click the highlighted area next to the line:
   "int val = findAnswer();"

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

```
16          setContentView(R.layout.answer_layout);
17
18          TextView answerView = (TextView) findViewById(R.id.answer_view);
19
20 ●  |     int val = findAnswer();
21          String output = (val == answer) ? "42" : "We may never know";
22          answerView
23                  .setText("The answer to life, the universe and everything is:\n\n"
24                          + output);
25      }
26
```

4. Next, press the Debug button in the Toolbar to start debugging the application (Shortcut: Windows - Shift + F9, Mac - CTRL+D).

```
tor  Build  Run  I  Debug 'app' (Shift+F9)  elp

 ⇒  ↓⁰¹  📱 app ▾  ▶  🐞  📲  ⚙  🖥  ⬇  📦  📥  🤖    ?

va  📁 course  📁 examples  📁 theanswer  © TheAnswer

⚙▾ |←   © TheAnswer.java ×

 1      package course.examples.theanswer;
 2
 3      ⊞import ...
 6
 7 ⊙    public class TheAnswer extends Activity {
 8
 9          public static final int[] answers = { 42
10          public static final int answer = 42;
```

Similar to Step 2 in Part 4, after you see the BUILD SUCCESSFUL message, a dialog box will pop up asking you to choose a device.
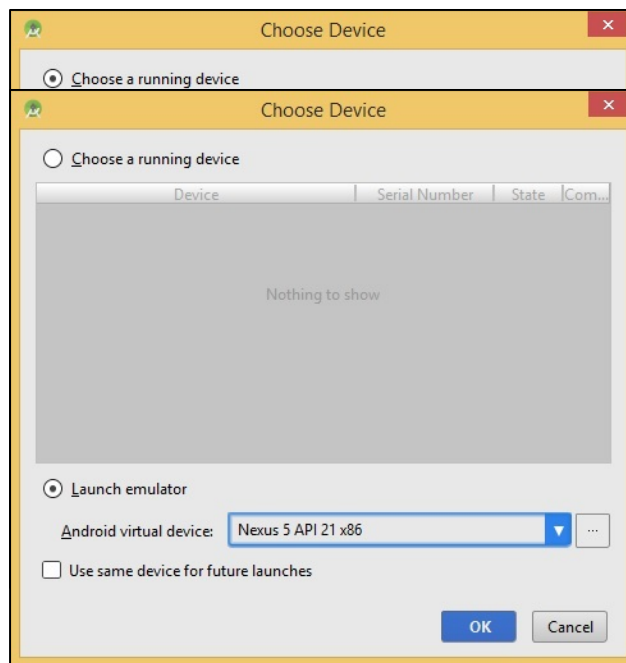

5. If you do not have a running device, you can choose an emulator to launch. The system will start an emulator and run the app in it.

6. Your Emulator should load the App and stop before the words, "The answer to life……" , are displayed on the screen. You can see the Debug Window appears next to Run now. Click on it to show Debug window.
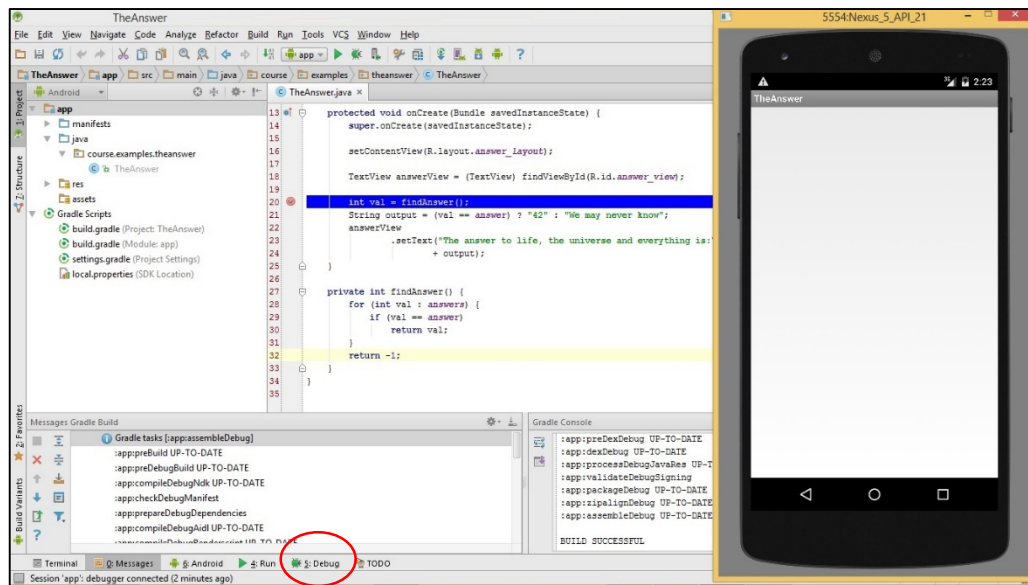

7. Now that the app is stopped, you can examine the app's state and step through the app's execution using the following buttons appearing in the menu bar.

1 – Resume Program (F9)   7 – Step Out (Shift + F8)
2 – Pause Program
3 – Stop (Ctrl + F2)
4 – Step Over (F8)
5 – Step Into (F7)
6 – Force Step Into (Alt + Shift + F7)
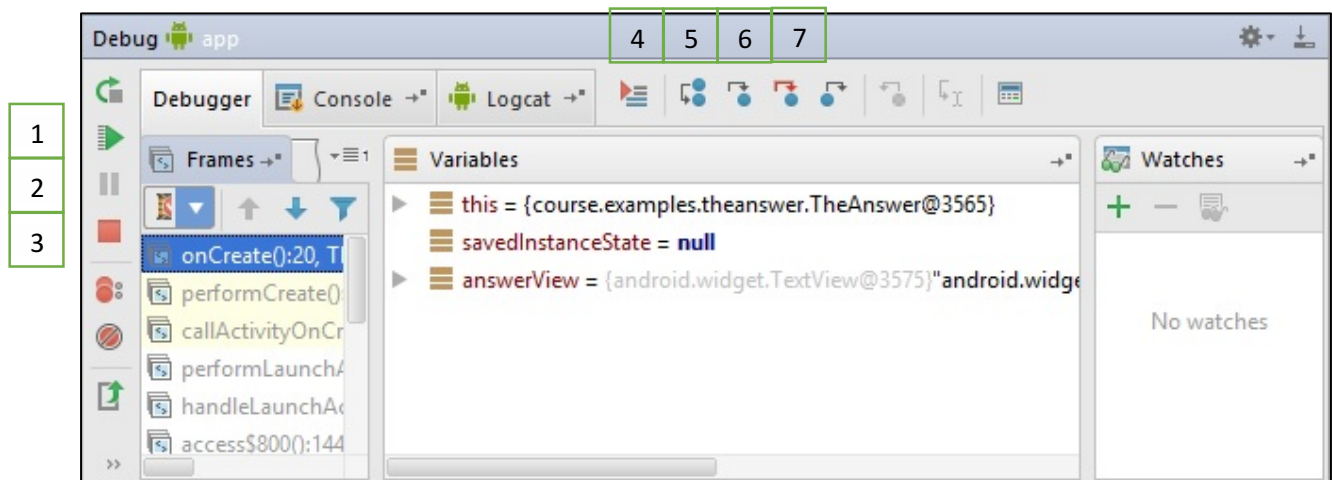
8. Next, press the Resume icon to continue executing the app. The app will finish loading and will display the text.
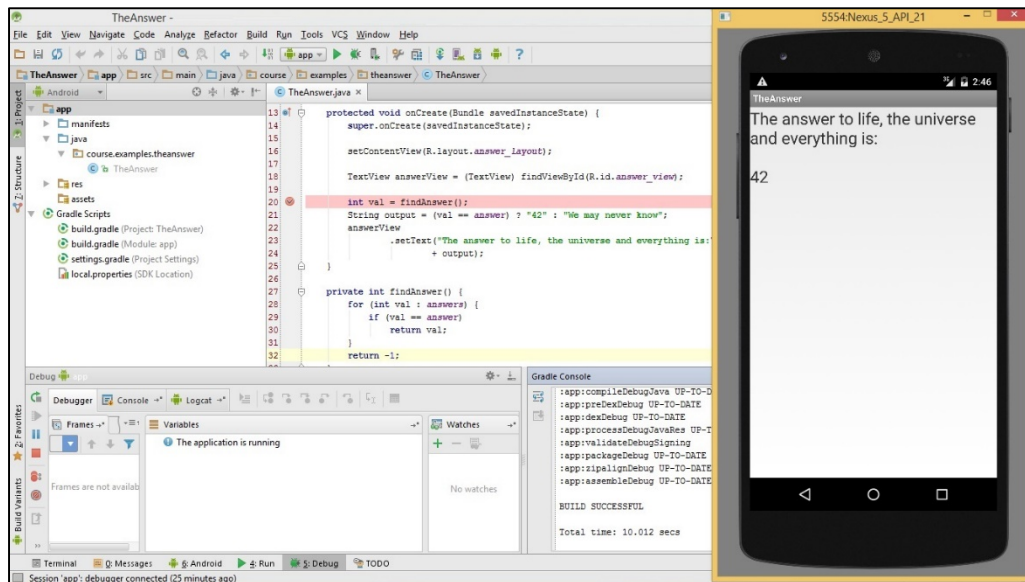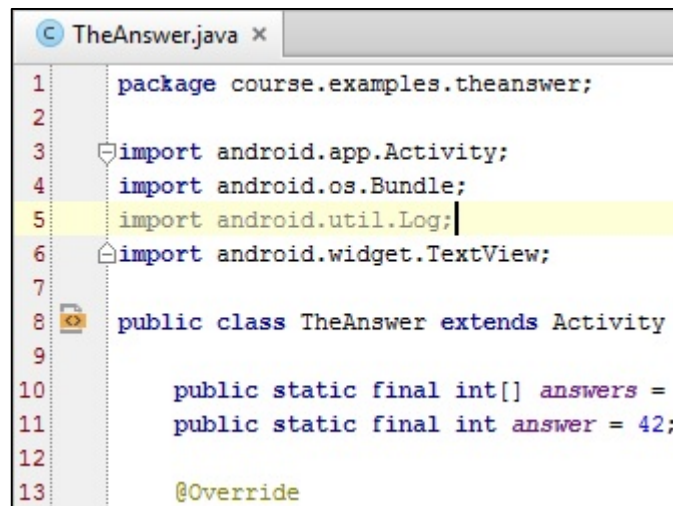

9. The next debugging task will have you create and display informational messages to the LogCat panel, to help you better understand the application's runtime behavior. To generate these messages, you will use methods in the android.util.Log class. You will also need to import this class into your application.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

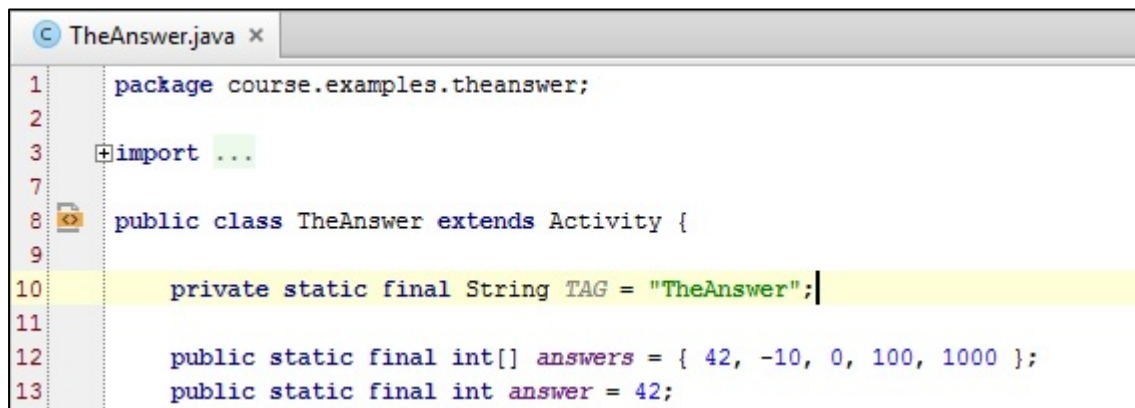See http://developer.android.com/reference/android/util/Log.html for more information.

10. Import the android.util.Log library by typing, "import android.util.Log;" near the beginning of the code for TheAnswer.java.

11. 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";"

12. 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");"

```
16      protected void onCreate(Bundle savedInstanceState) {
17          super.onCreate(savedInstanceState);
18
19          setContentView(R.layout.answer_layout);
20
21          TextView answerView = (TextView) findViewById(R.id.answer_view);
22          Log.i(TAG, "Printing the answer to life");
23          int val = findAnswer();
24          String output = (val == answer) ? "42" : "We may never know";
25          answerView
26              .setText("The answer to life, the universe and everything is:\n\n"
27                  + output);
28      }
```
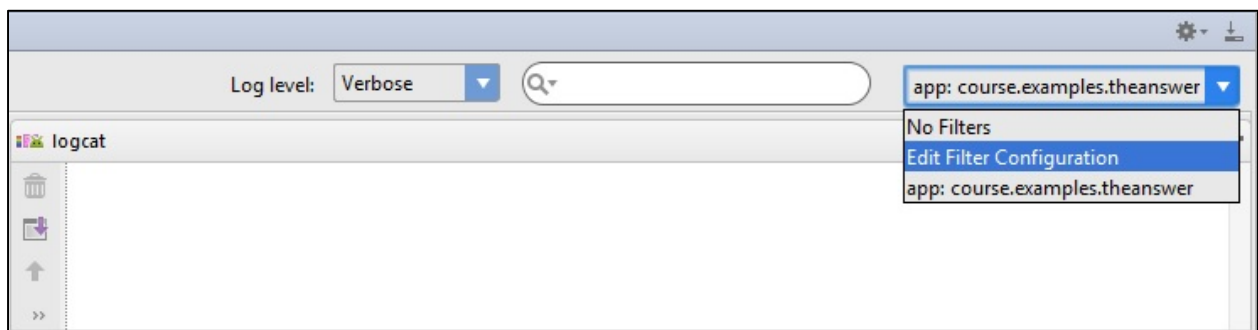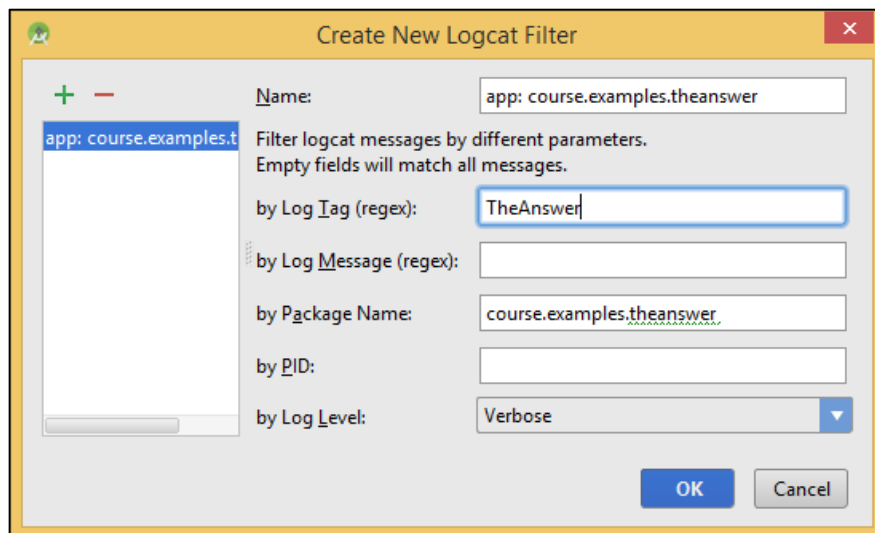
13. Save your changes.
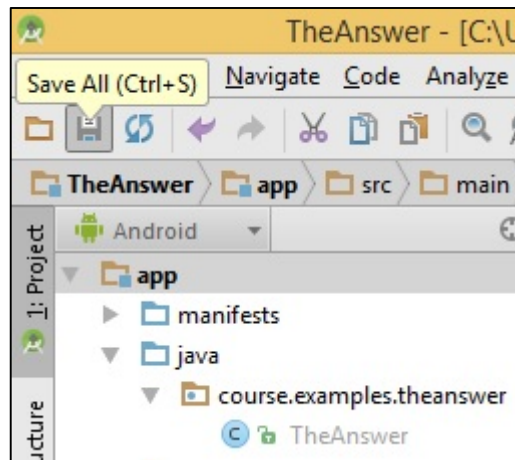
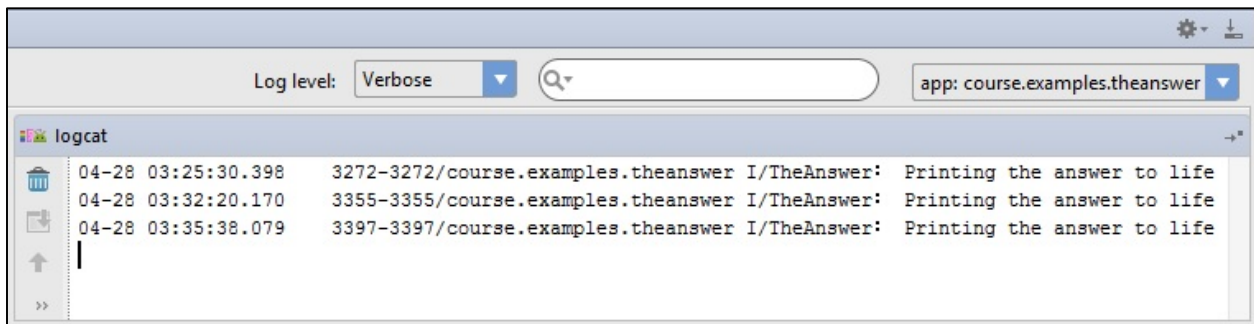14. Run the application. (See Part 4 for more details on Running App).
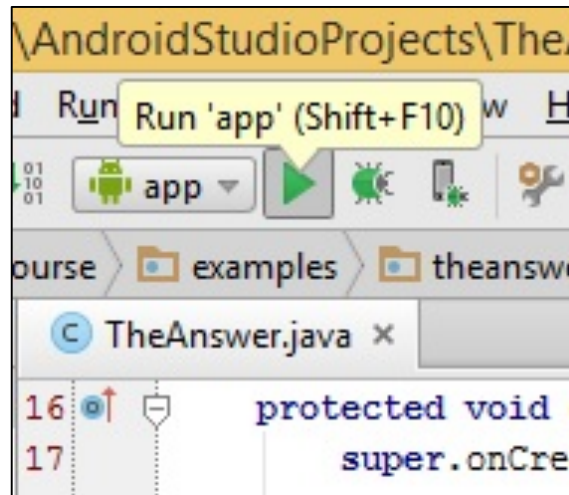
15. Once the app is running, open the LogCat panel at the bottom. Look for drop down menu and select Edit Filter Configuration.

16. Enter "TheAnswer" in LogTag and hit Return.

17. You will now see the log message from the TheAnswer application in the LogCat panel.


Extra Challenge

If you finish all the work above in class, then do the follow challenge activity as well.
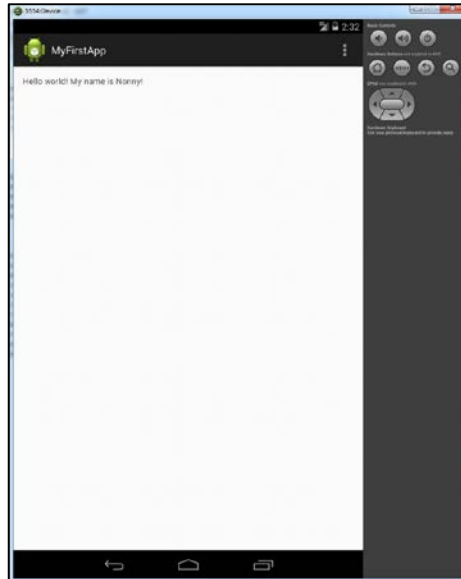
1. **Modified Hello World** - Remember the first app you made? Let's return to that!

2. 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.
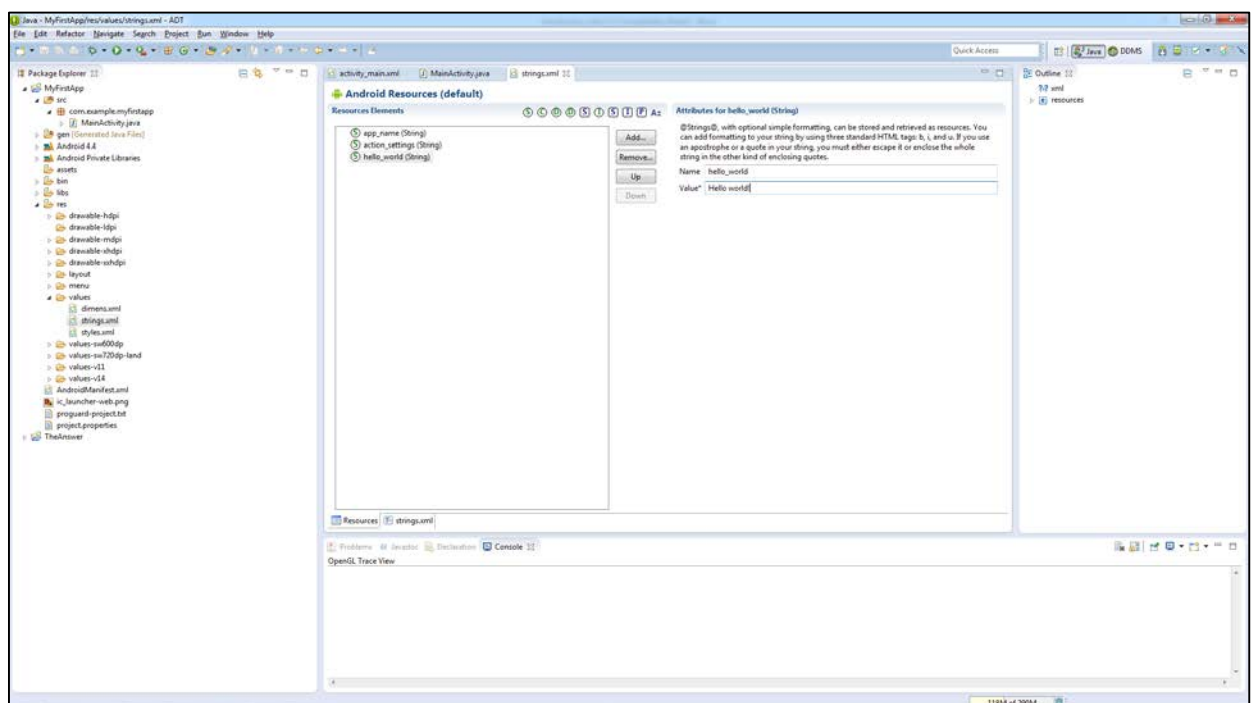
3. Change the value of the string to "Hello world! My name is [yourname]."
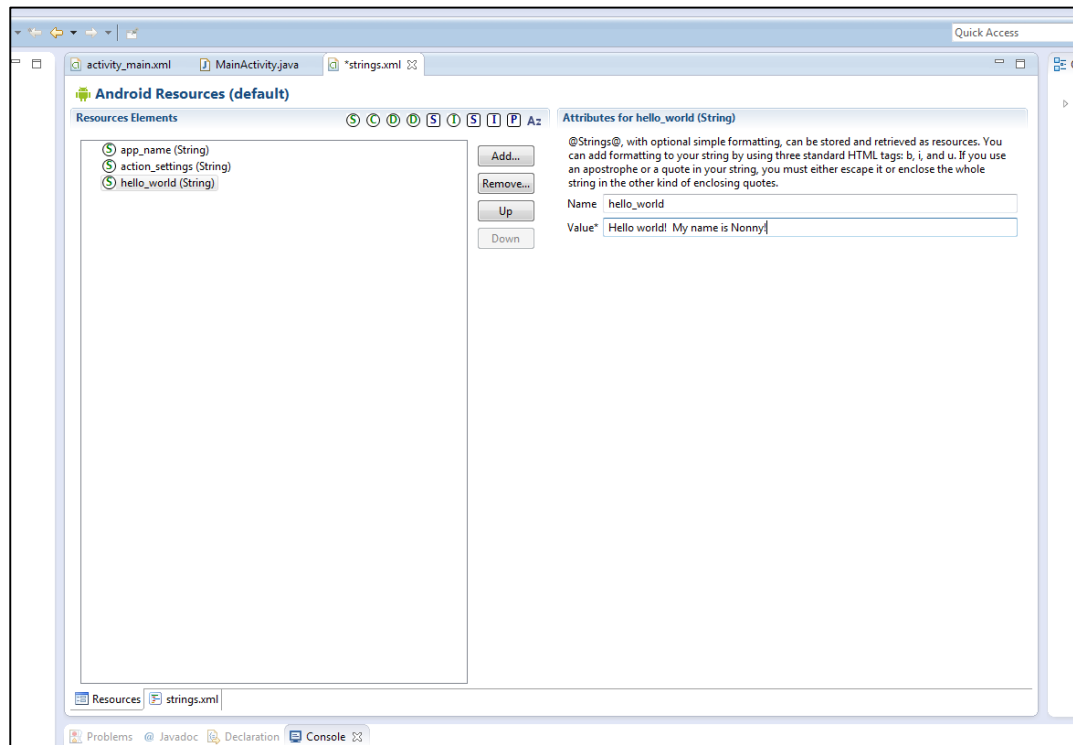
4. Now run the app and see the change!

5. Now add support for another language such as Spanish! To do this, you'll need to create an appropriate string file, run your app, change the emulator instance's default language to Spanish, and then rerun 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