Quick Start Guide to Android Development

What are (software) required to develop my first Android application?

- Java JDK
- Eclipse IDE
- Android SDK
- Android Development Tool (ADT) for Eclipse

Steps for installing these softwares are given below.

Setting up the platform

Install latest Java JDK from **Java SE Downloads** (Note: Should install JDK not JRE.)

Install the Eclipse IDE. Version 3.4 or later is recommended

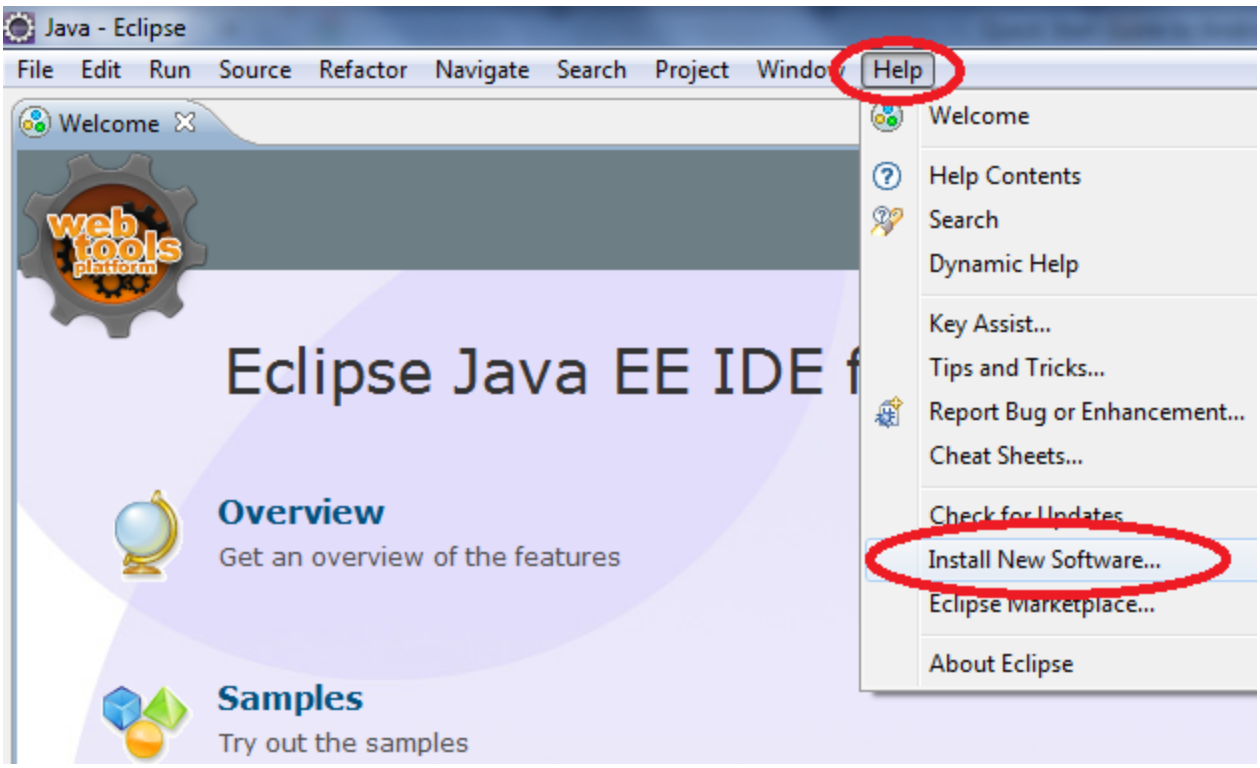
Install the Android SDK.

Installing the ADT Plugin in Eclipse

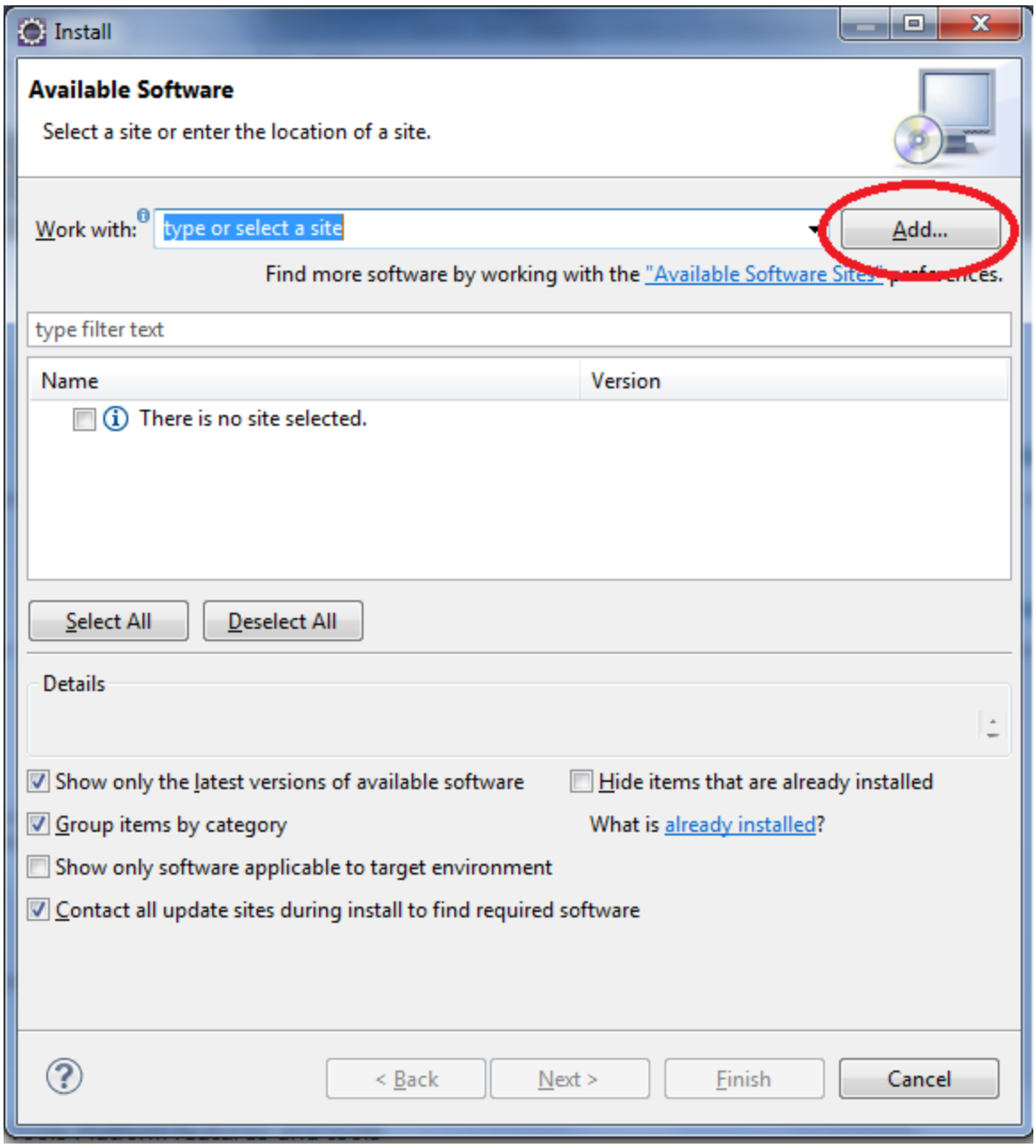
Make sure you have installed the Java JDK, Eclipse and Android SDK properly.

Then follow the steps given below.

1. Start Eclipse
2. Go to **Help** tab, then select **Install New Software**

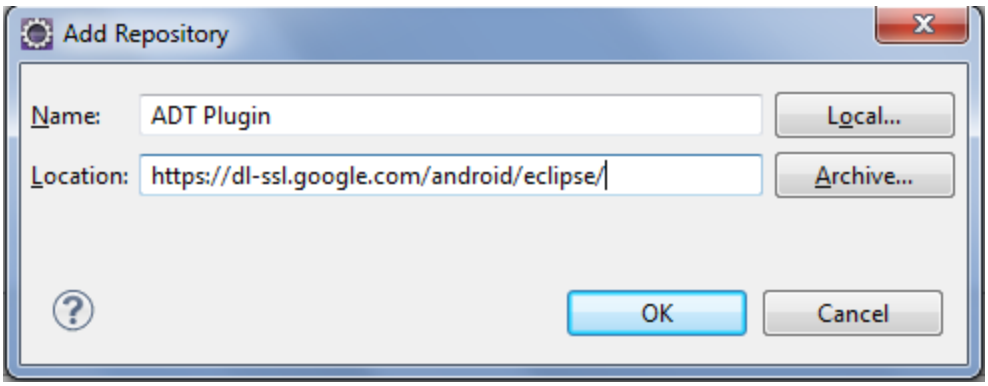


3. Click **Add** button.



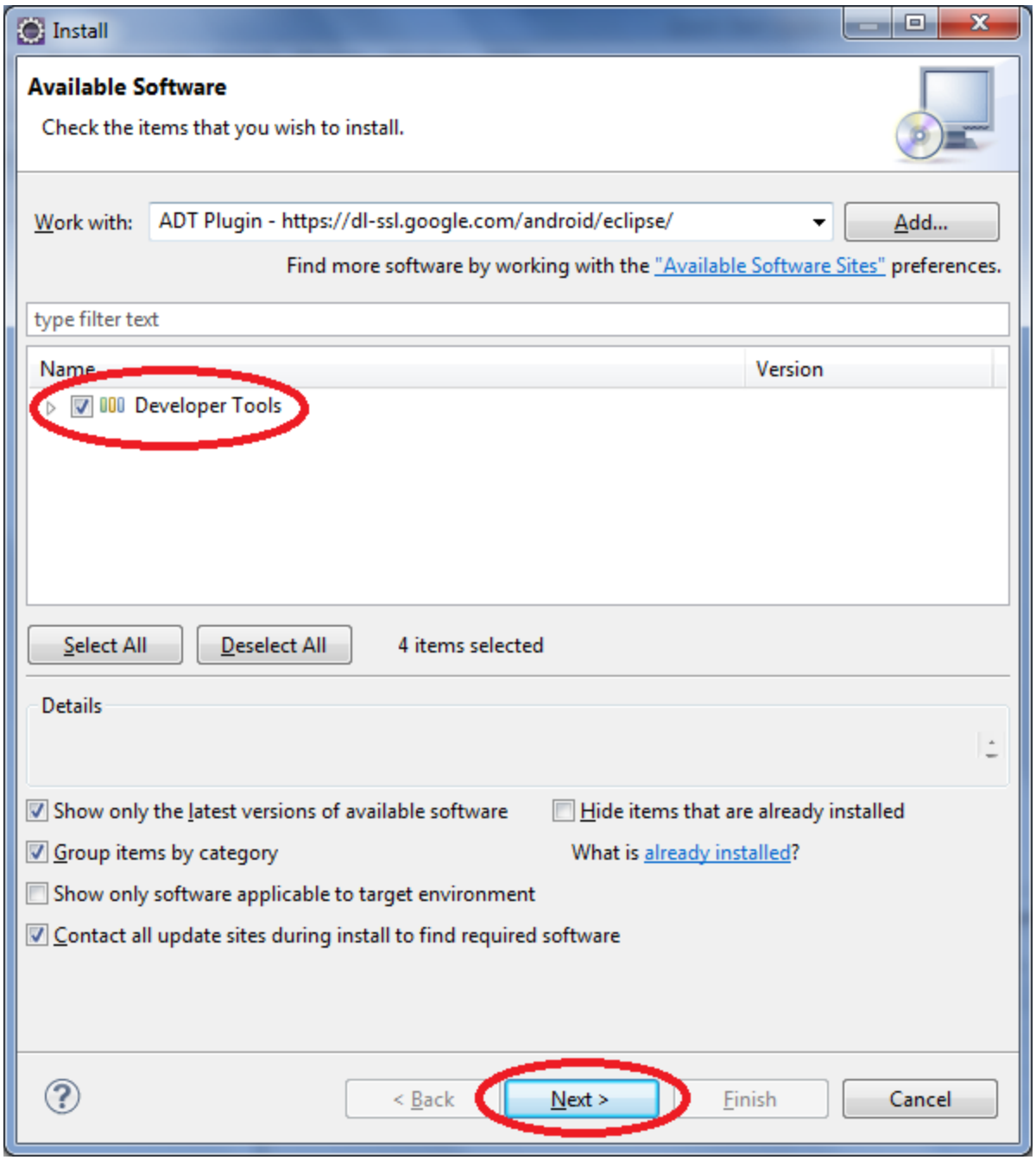
4. In Add dialogue, enter 'ADT Plugin' for name and the URL given below for location.

https://dl-ssl.google.com/android/eclipse/

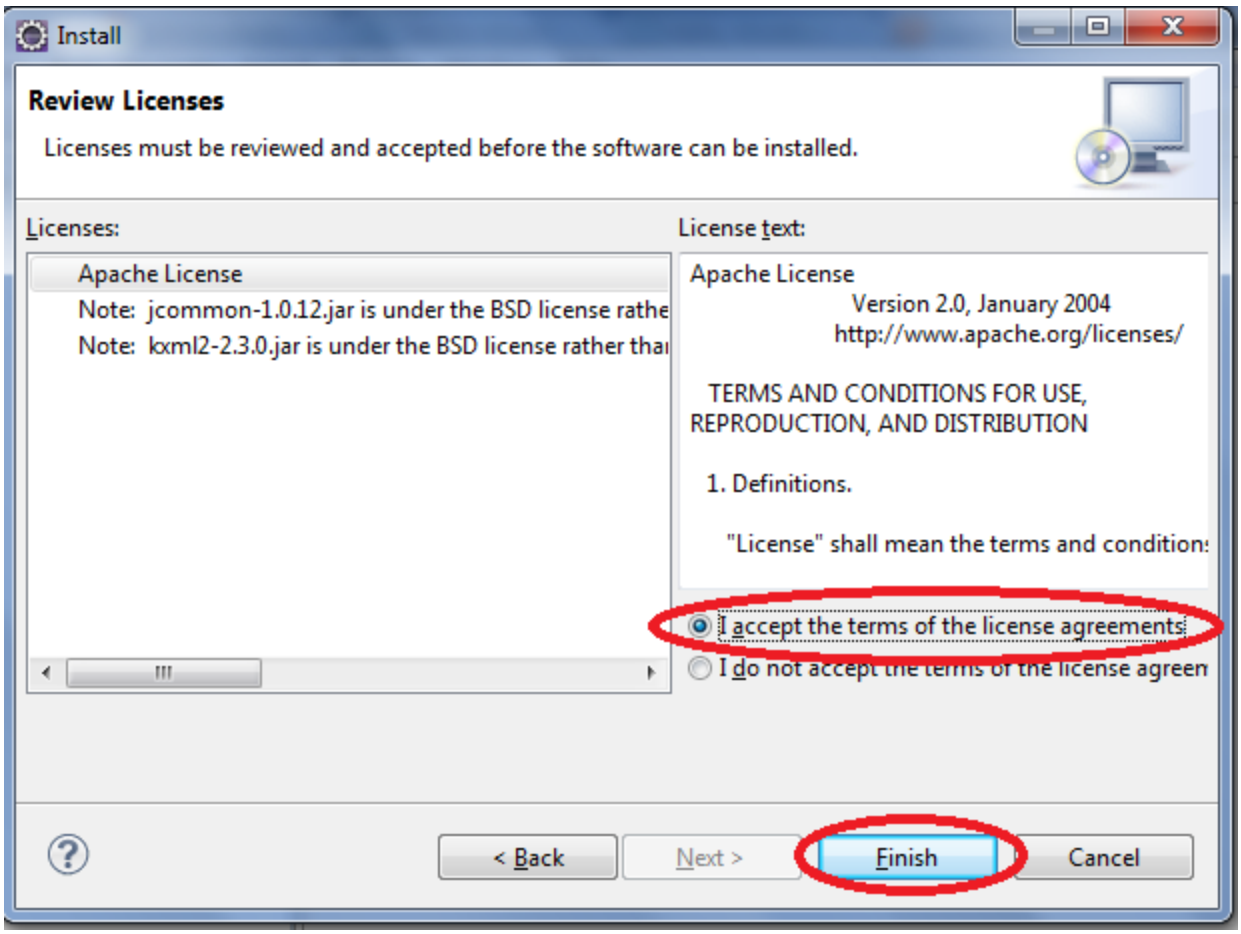


Then Click OK.

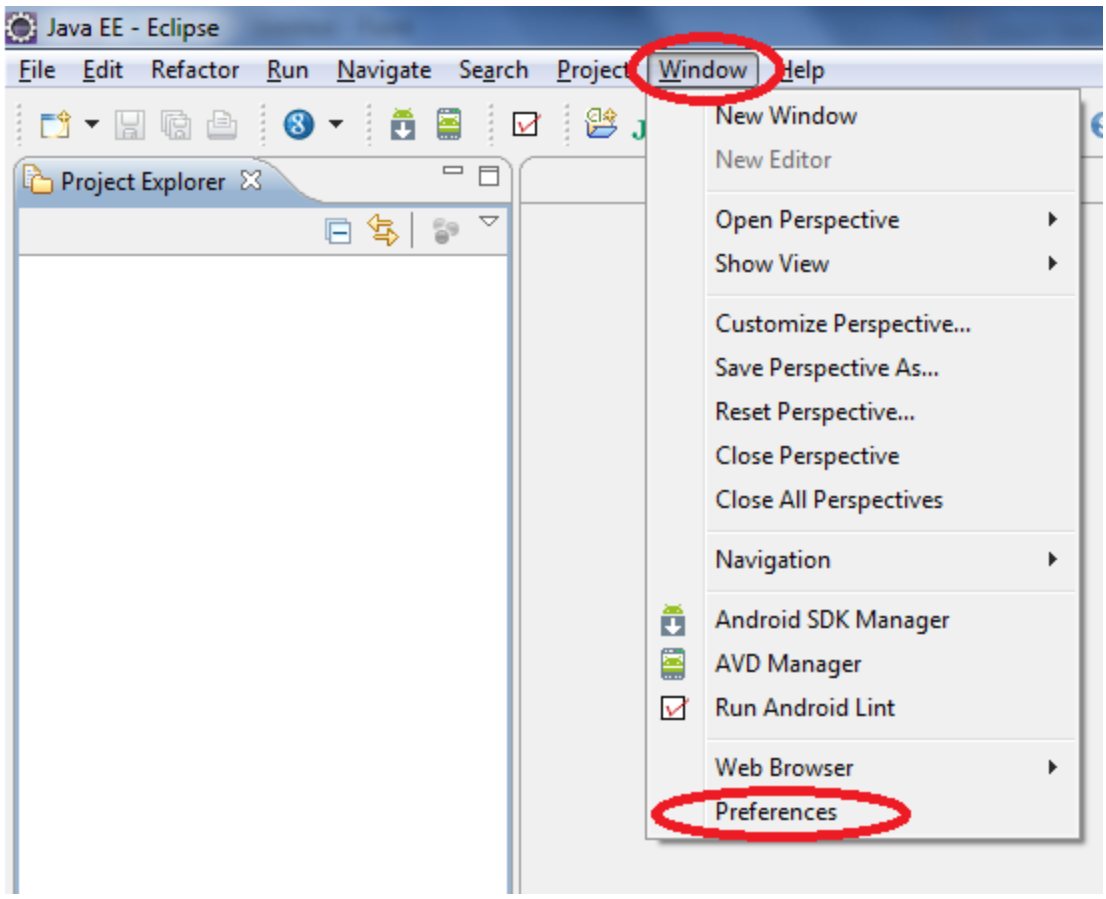
5. Select the checkbox next to ‘Developer Tools’ and click Next.



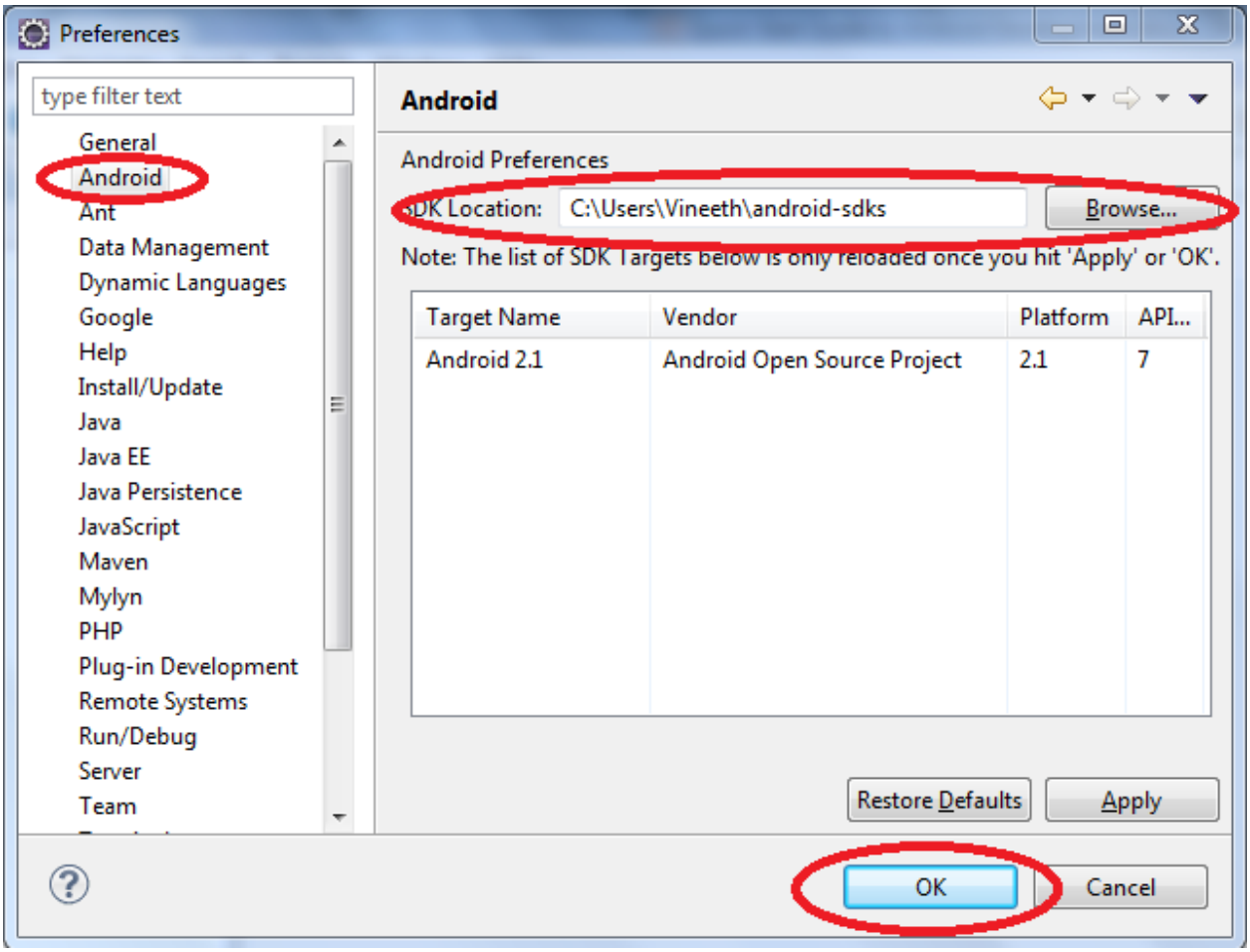
6. In the next window, you'll see a list of the tools to be downloaded. Click **Next**
7. Click finish after reading and accepting the license agreements.



8. If you get a security warning saying that the authenticity or validity of the software can't be established, click **OK**
9. When the installation completes, restart Eclipse
10. In **Window** tab, select **Preferences**



11. Select **Android** from the left panel and in the main panel, click **Browse...** and locate your downloaded SDK directory. Then click **OK**



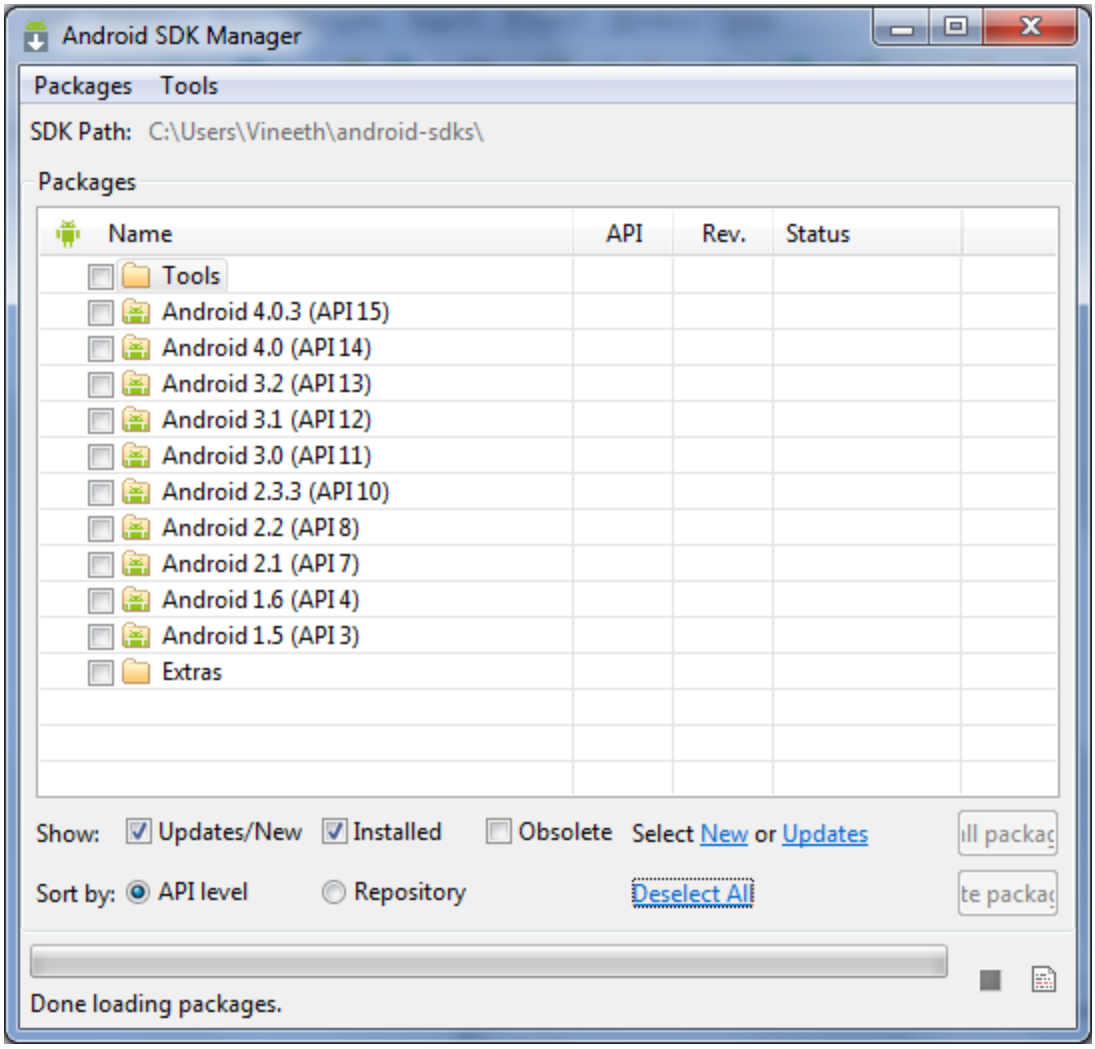
We have installed ADT Plugin. If you face any problem, refer the Android ADT plugin installation guide.

Installing Android Platform

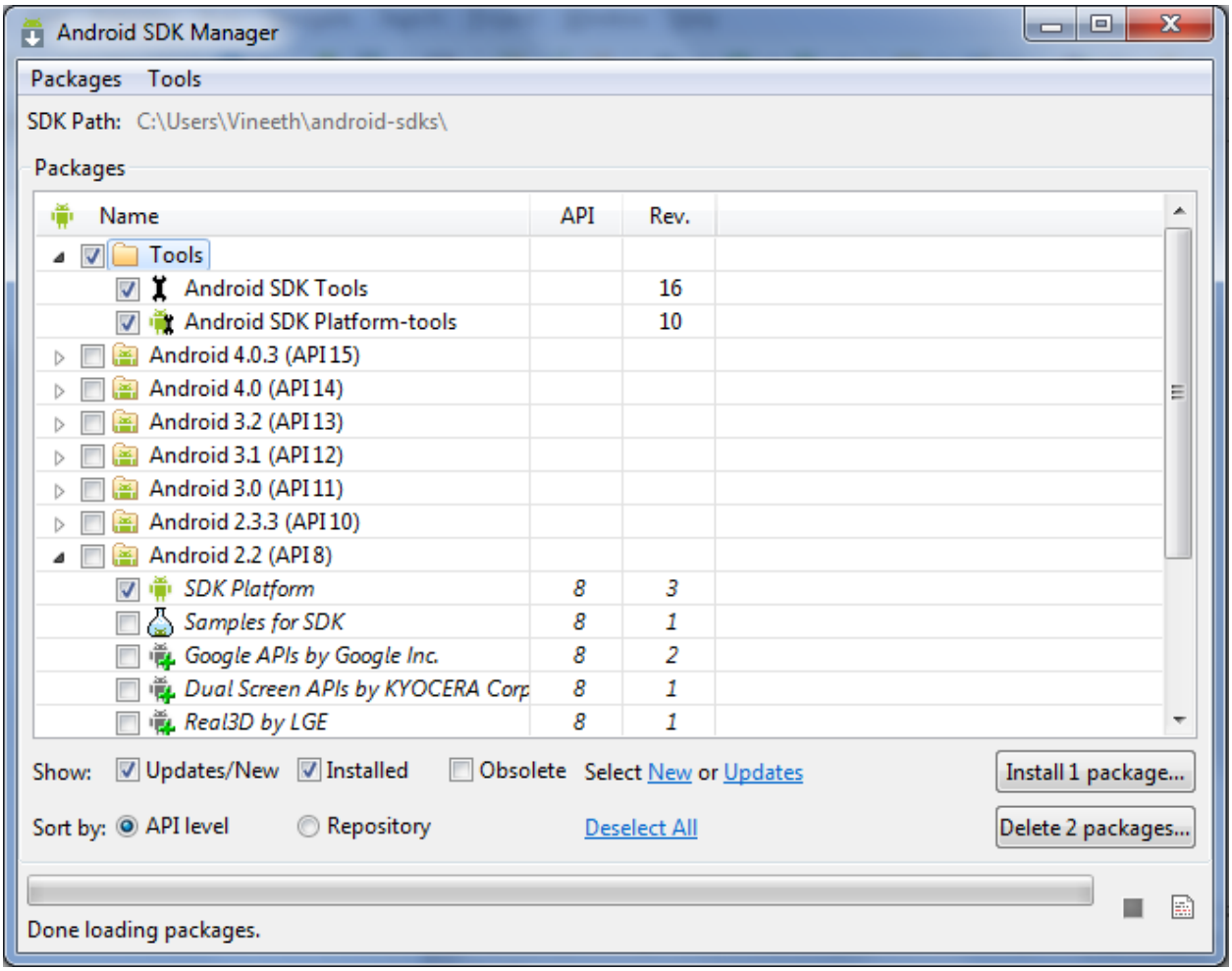
The SDK starter package, which you've already downloaded, includes only a single component: the latest version of the SDK Tools. To develop an Android application, you also need to download at least one Android platform and the associated platform tools. To install it, follow the steps given below.

1. Open Android SDK and AVD Manager

   To open it within Eclipse, go to **Window** tab and select **Android SDK and AVD Manager**
   Otherwise you can go to SDK installed location and open the Android SDK and AVD Manager.



2. Now select Tools (which contains Android SDK Tools and Android SDK Platform-tools) and at least one SDK Platform. I am installing Android 2.2(API8) SDK Platform only. (You can choose any other platform(s) also).



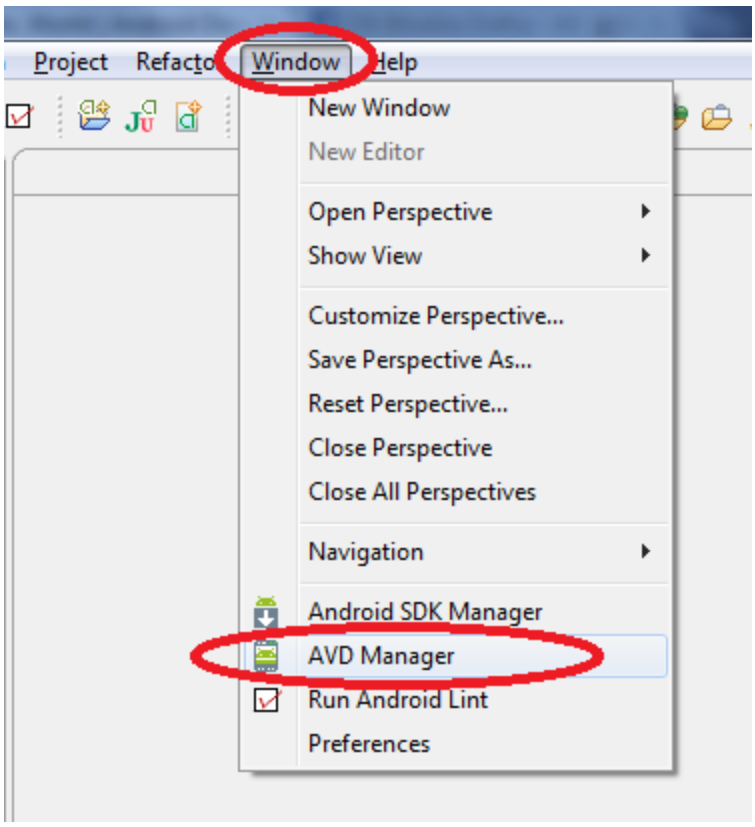
3. Now select ‘Install packages’ (The number of packages shown in button may vary based on the selection you made)
4. In the new window, select ‘Accept All’ and click Install.

For more information or troubleshooting, check adding SDK Components guide.

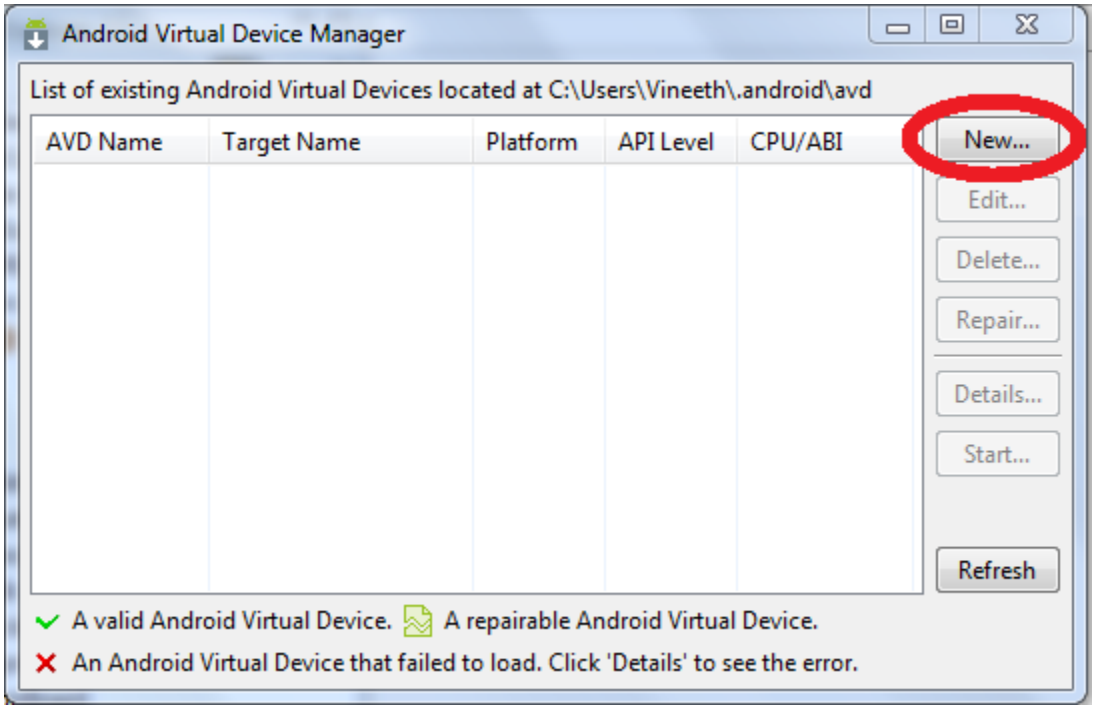
Creating Android Virtual Device (AVD)

Android Virtual Device is an Android Emulator, which is normally used to test the applications during the development. To create a new AVD, follow the steps given below.

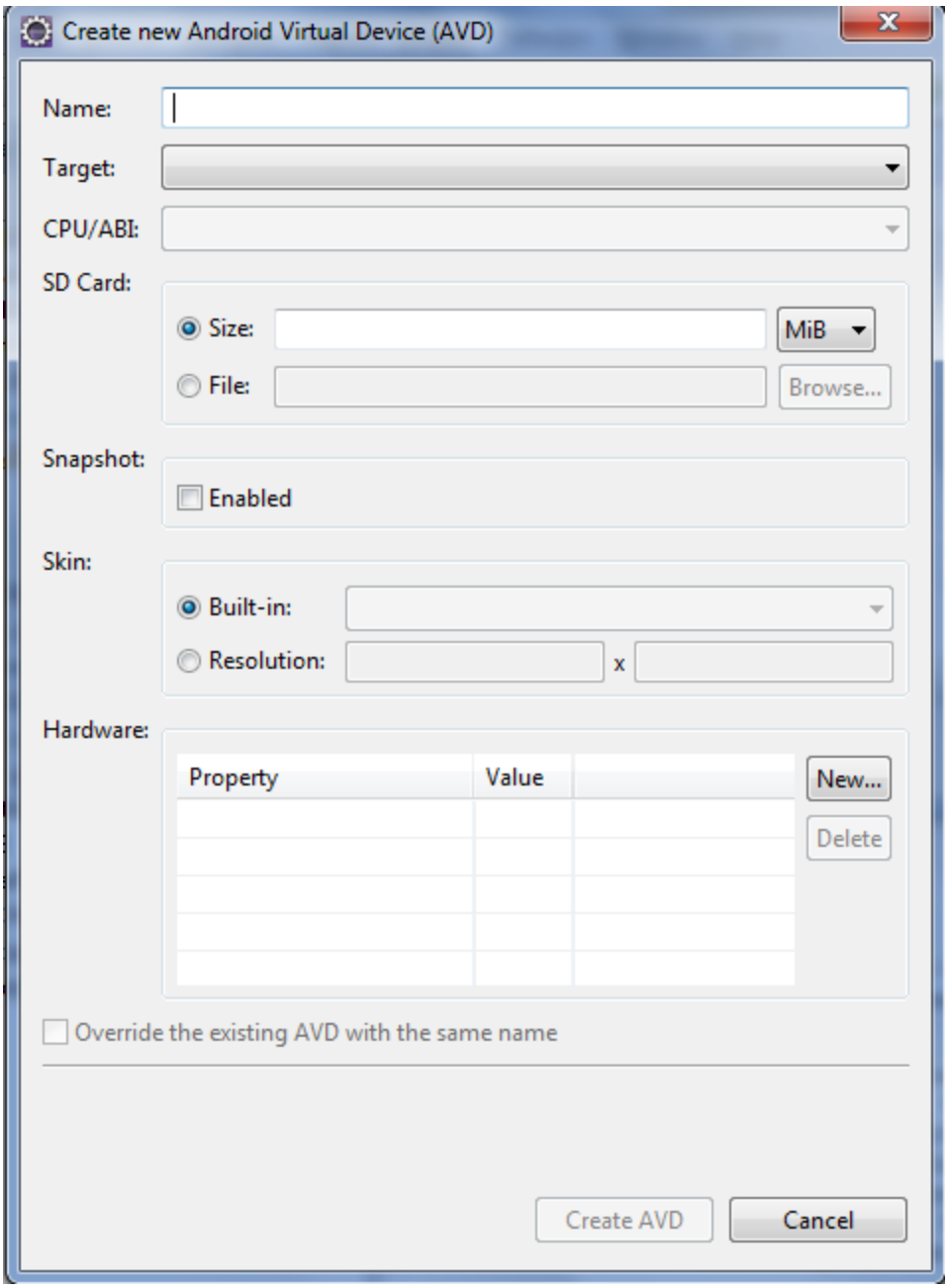
1. In ‘Window’ tab, select ‘AVD Manager’.



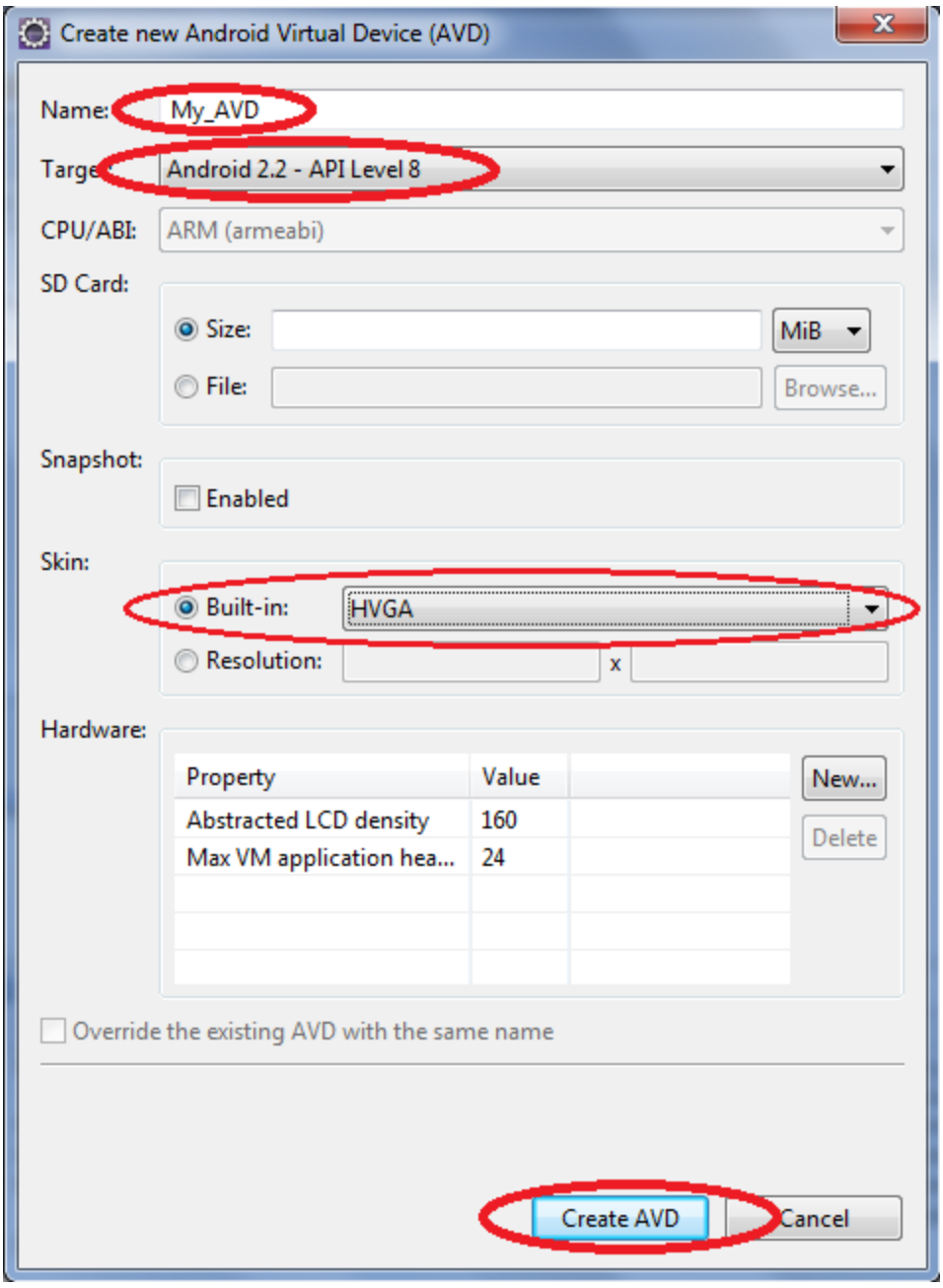
2. Click ‘New’



3. ‘Create New AVD’ dialog appears.



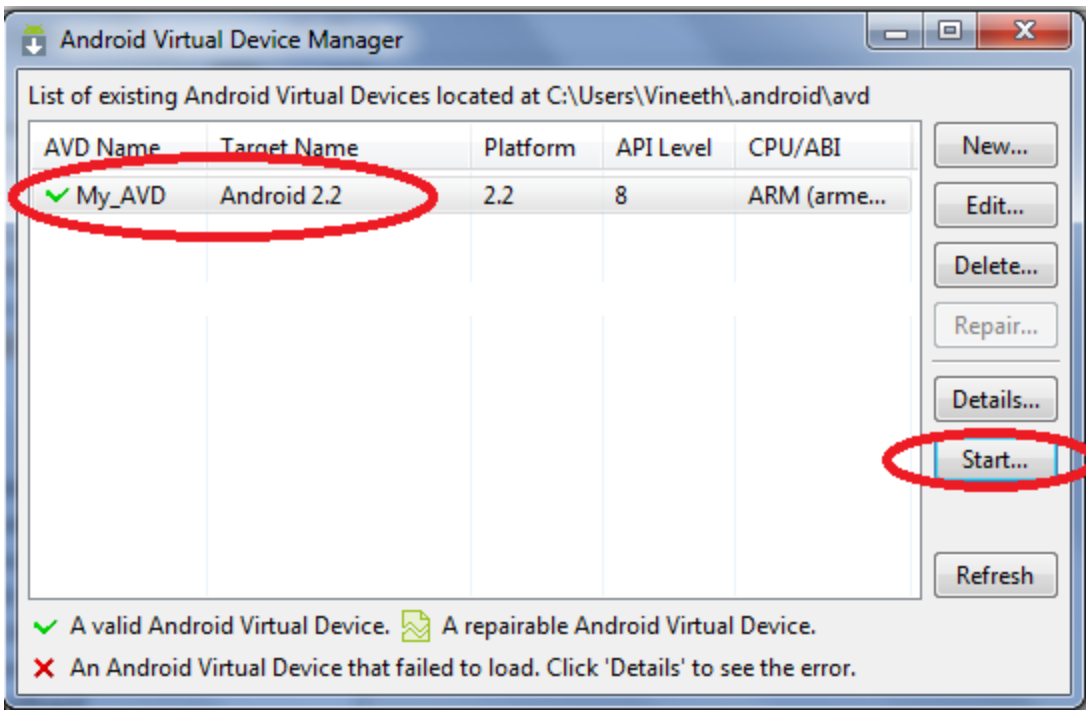
4. Give a name. (like ‘My_AVD’)
5. Select a target Android version (such as ‘Android 2.2 or any other version which you have installed during the installation of android platform).
6. This is an optional but suggested step. Change the ‘Skin’ – ‘Build-in’ to ‘HVGA’. It reduces the AVD screen size. I personally prefer this size because it helps to see the device properly in normal laptop screens)



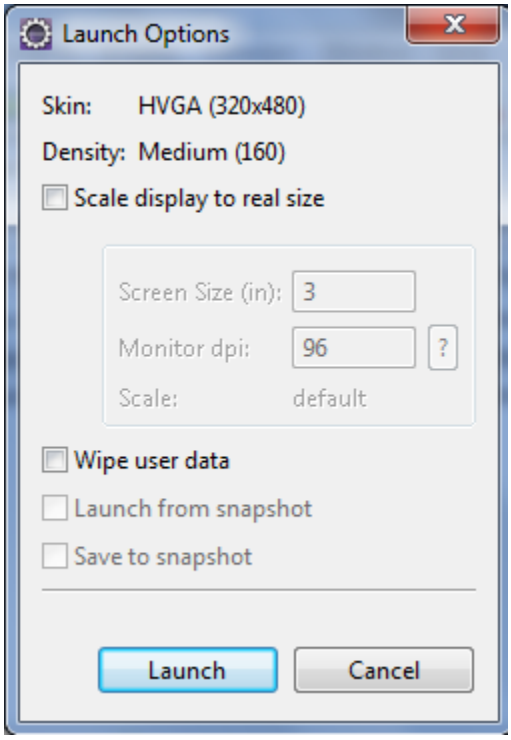
7. Click ‘Create AVD’

We have created the AVD. Let us start it and see how it looks.

8. Select the new AVD (My_AVD) and click ‘Start’.



9. Click ‘Launch’



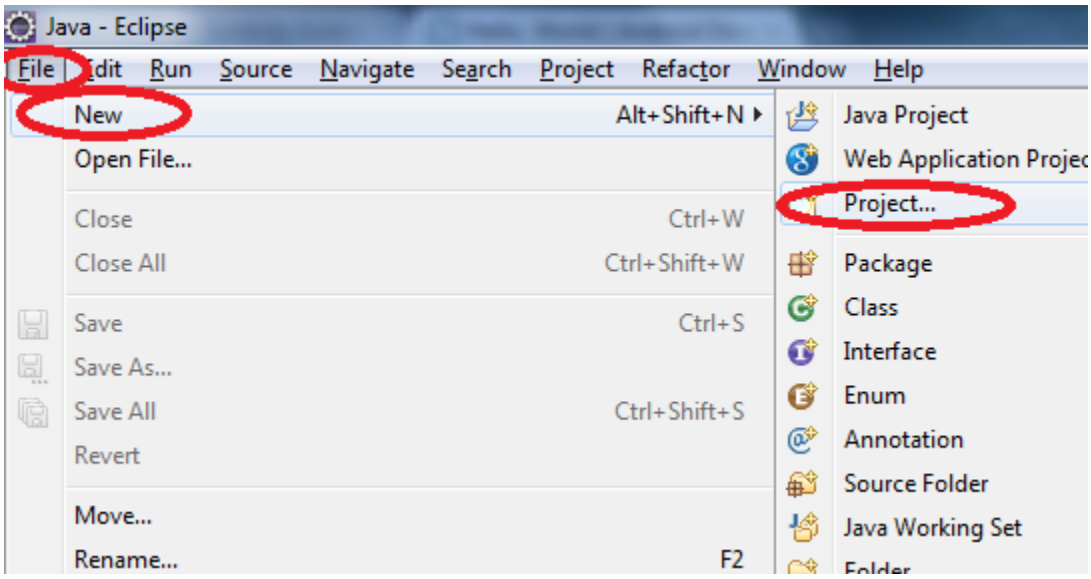
10. After a few minutes, (it takes a few minutes to launch completely) you should be able to see an emulator very similar to the one below.

Now you can close the emulator.

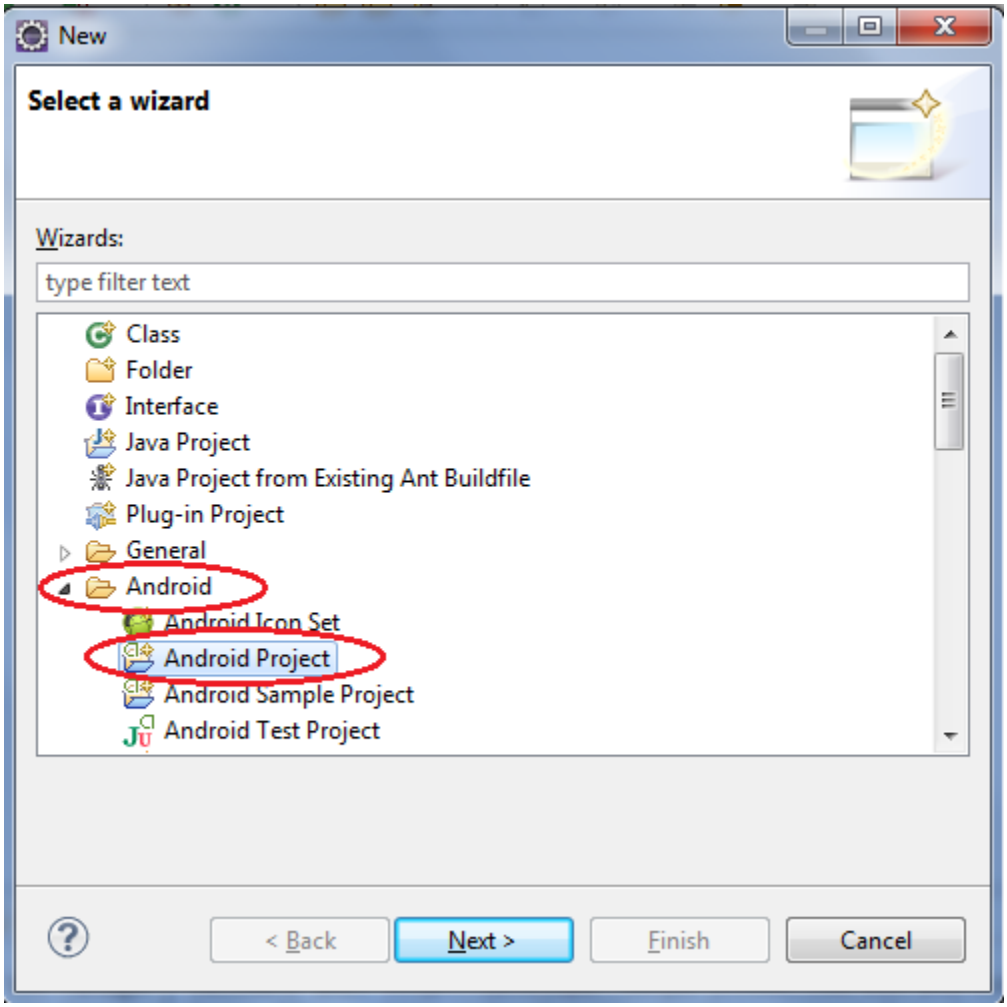
Creating an Android Application

The steps given below will help to create a simple android application.

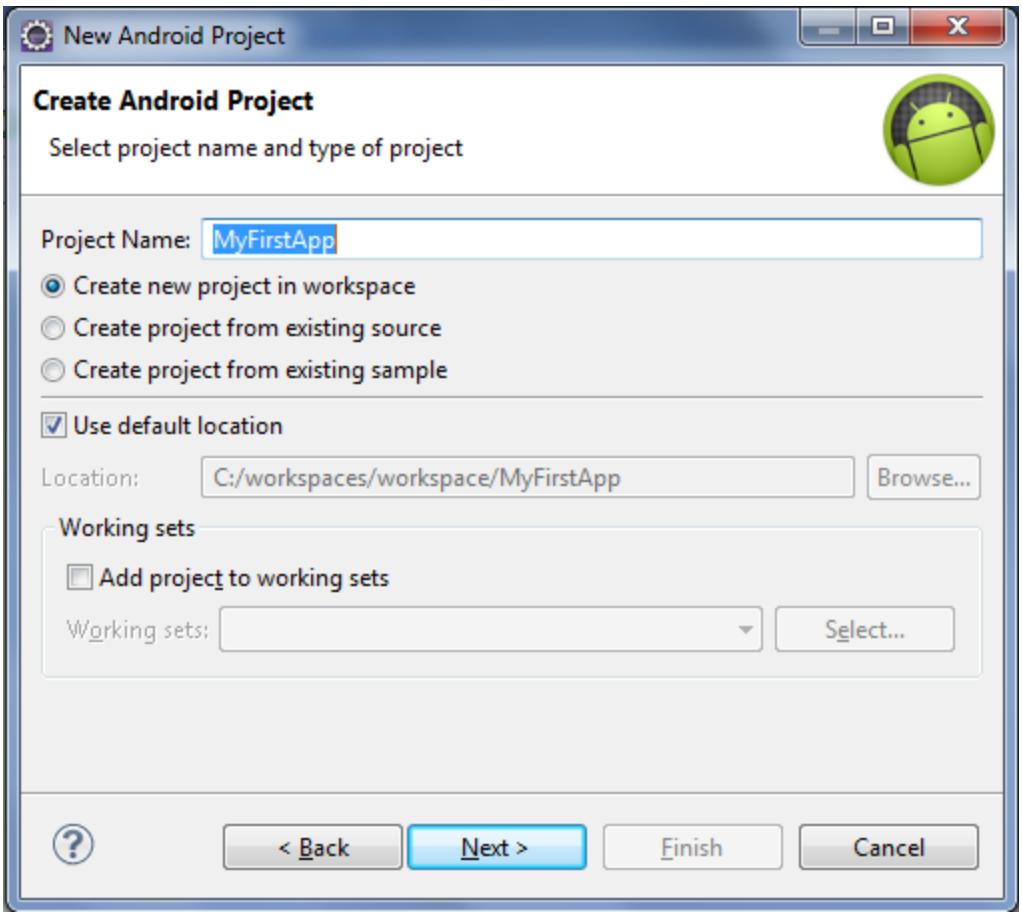
1. In eclipse, select File > New > Project



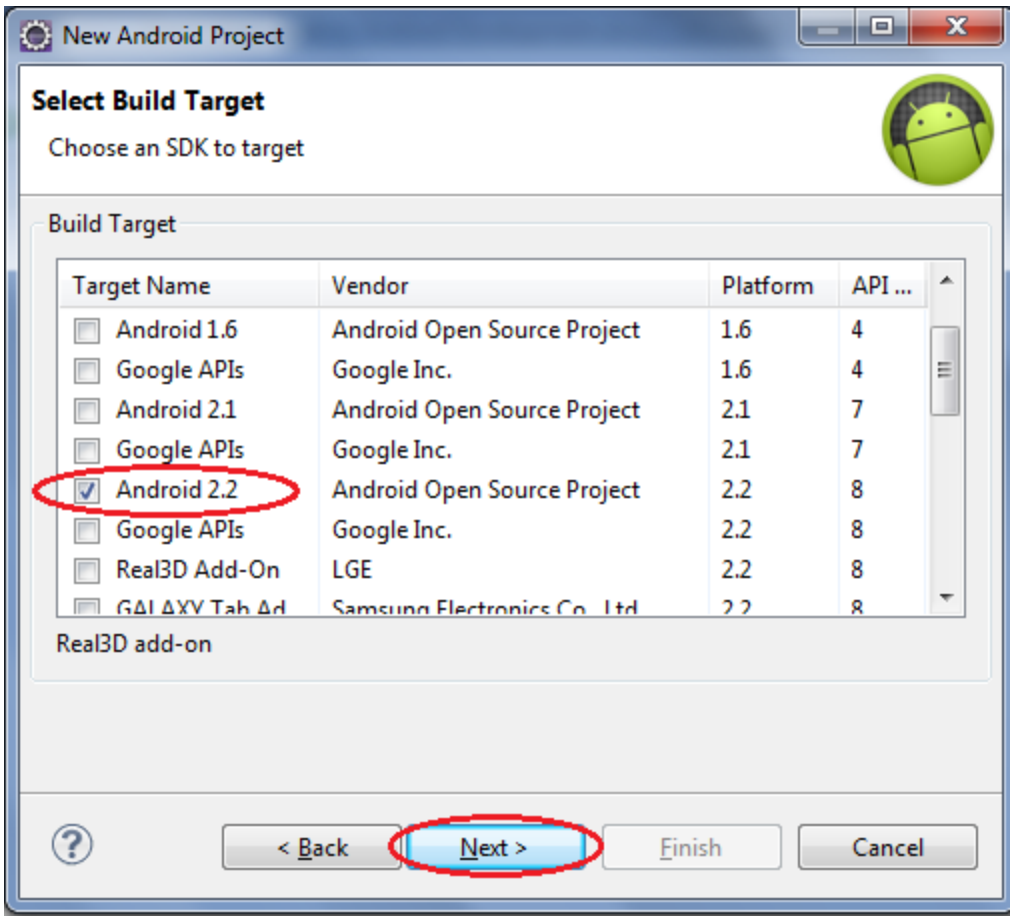
2. Select ‘Android Project’ in ‘Android’ folder.



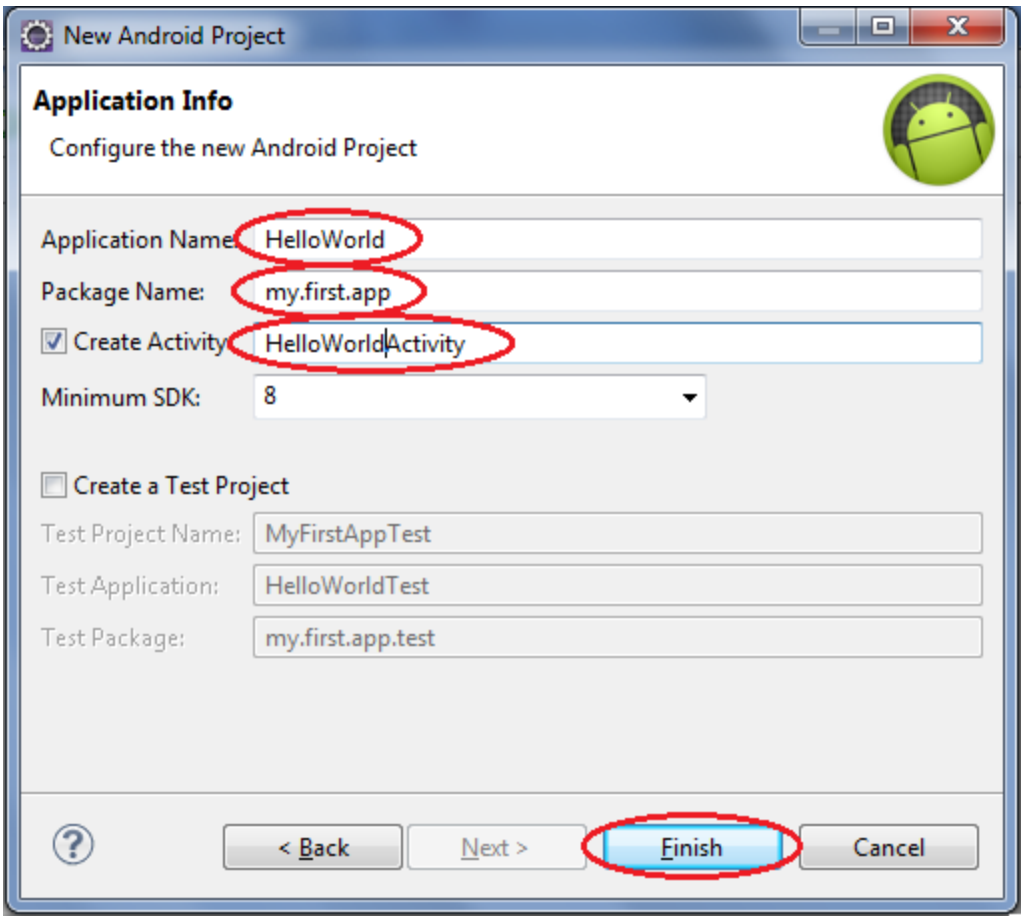
3. Mention a project name (such as ‘MyFirstApp’)



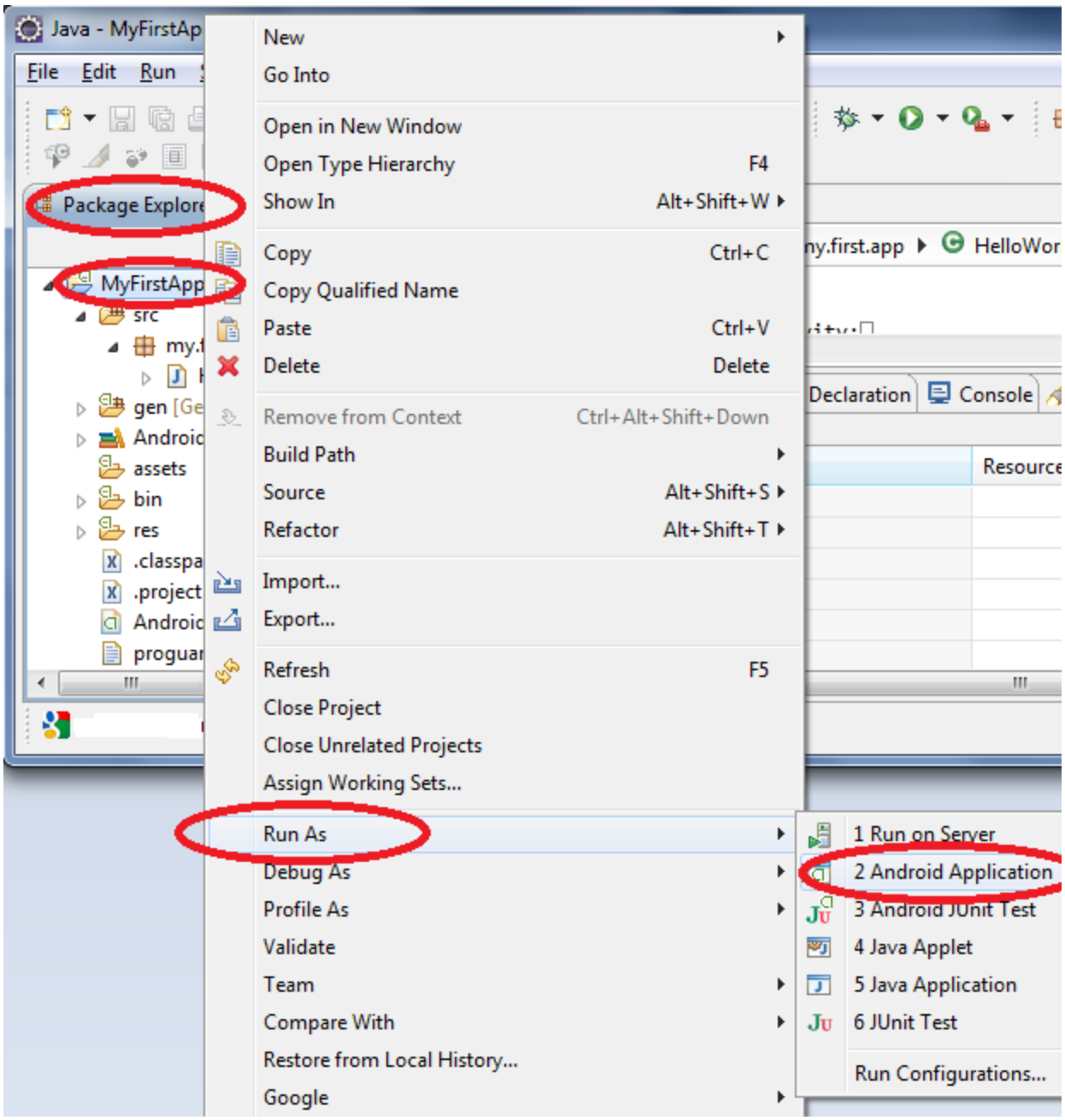
4. Select the target version. (You can select multiple, remember to select AVD version)



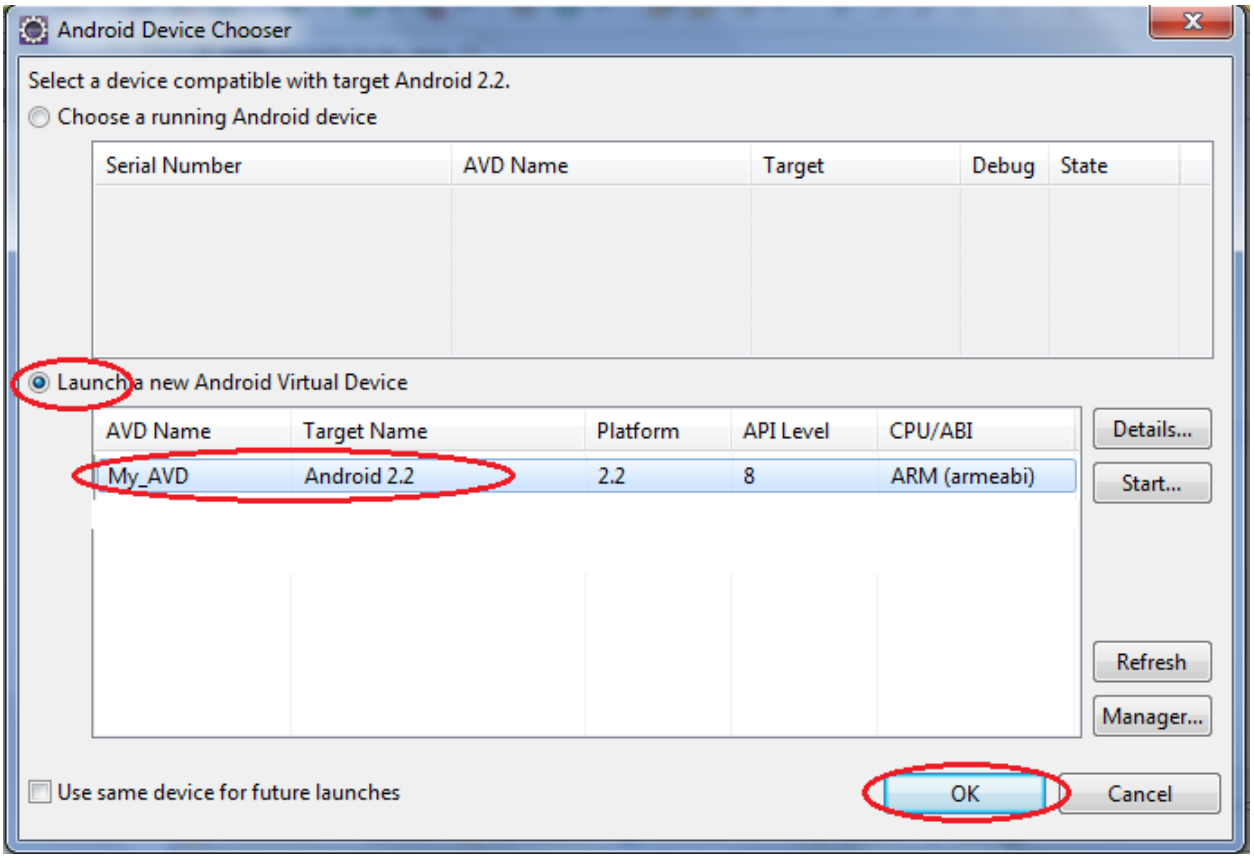
5. Mention application name, package name and activity name and click Finish.



6. Now the first application is ready and we can test it. To test it, we are going to install the app in the emulator 'My_AVD'. Actually eclipse takes care of this installation. What we have to do is, we have to run this project as an android project.
7. To run it, go to 'Package Explorer' or 'Project Explorer' (located in left side middle panel of eclipse). Right click on the project 'MyFirstApp', Go to 'Run As' and select 'Android Application'.

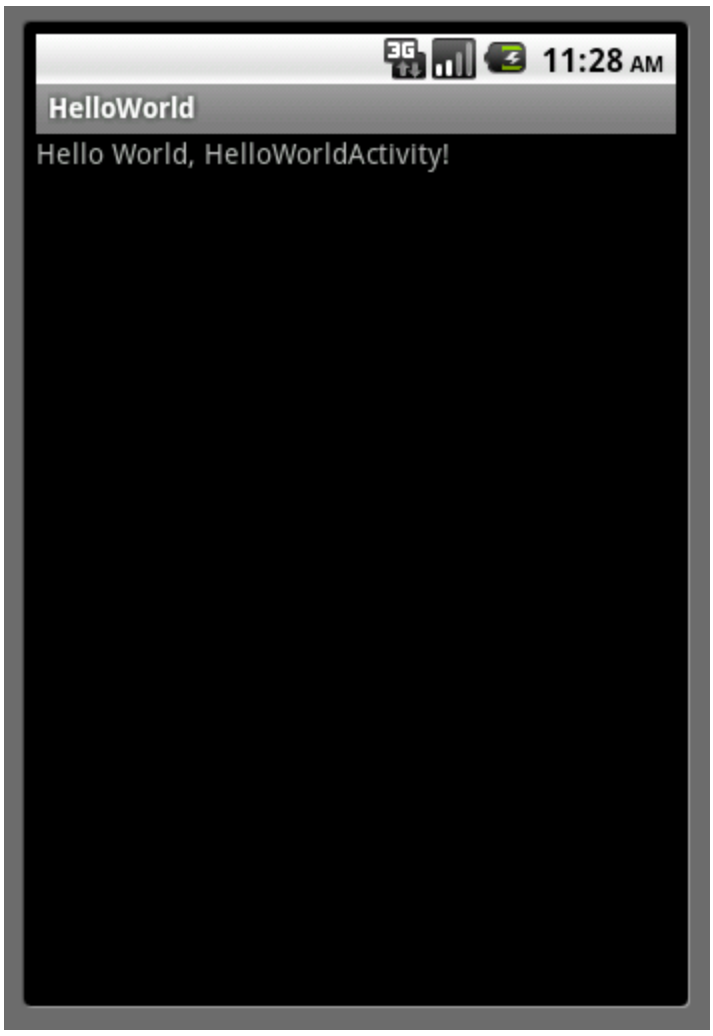


8. If eclipse asks you to select the device, choose ‘My_AVD’. And click OK.



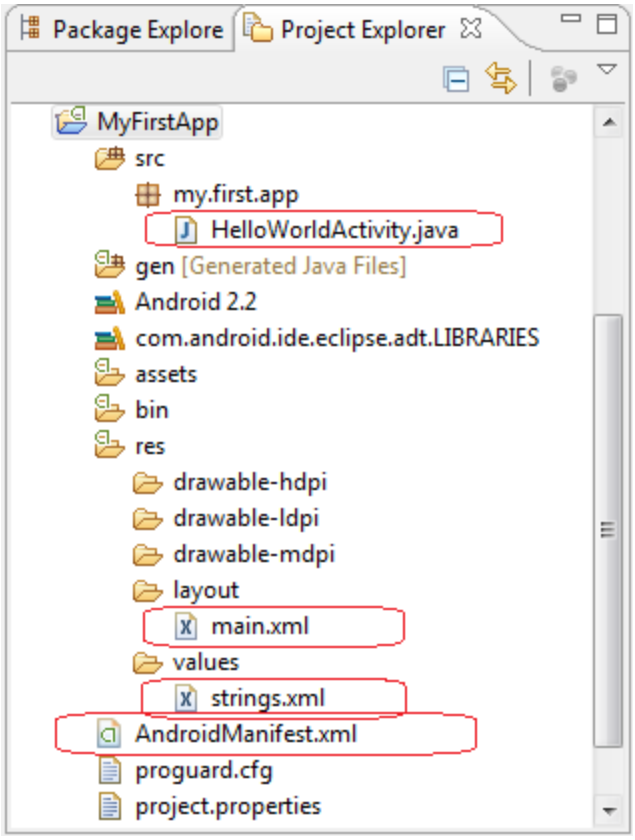
9. It takes a few minutes to load the virtual device. When it is loaded you should be able to see the home screen.

10. Normally the application starts itself, but sometimes it doesn't. If it doesn't, you have to click on the 'MENU' button as shown above.
11. You should be able see the application in the AVD.



Understanding the application ‘MyFirstApp’

The project file structure is shown in ‘Project Explorer’ is given below. Important files are highlighted.



Let us have a quick overview of these folders and files.

- **MyFirstApp**: the root folder.
- **src**: contains the source code (java code).
- **HelloWorldActivity.java**: an android activity class.
- **gen**: contains the eclipse generated java code. We don't need to know more about that right now.
- **Android 2.2**: Android library
- **com.android.ide.eclipse.adt.LIBRARIES**: android eclipse related library.
- **bin**: contains compiled code.
- **res**: contains resources other than the java code.
- **res/drawable-*** folder: contains images of different resolutions.
- **layout**: contains layouts of UI screens (activities).
- **layout/main.xml**: The 'HelloWorld' screen layout.
- **values/string.xml**: contains the "text" strings values. Normally if your app supports different languages, then there will be different files, one for every language. This file contains the string values in English.
- **AndroidManifest.xml**: is the manifest file. The following section explains more about it.

Android Manifest file

The manifest file contains information about the application. The important information are highlighted below.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="my.first.app"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".HelloWorldActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

`android:icon="@drawable/ic_launcher"`: Mentions application icon. It specifies the application icon is the image file in drawable folder with the name 'ic_launcher'.

`android:label="@string/app_name"`: Mentions application name. It specifies the application name is mentioned in string.xml (in values folder) with the name 'app_name'

`<activity android:name=".HelloWorldActivity"`: Defines an activity and mentions that the java activity class for this activity is *HelloWorldActivity.java* in the default project package 'my.first.app'.

The intent filter `<action android:name="android.intent.action.MAIN" />` defines this activity as the main activity and it will be launched when the application is launched.

So, when the application is launched, the activity 'HelloWorldActivity' will be started. In the following section, we will see more about the 'HelloWorldActivity' activity.

HelloWorldActivity.java

```
package my.first.app;

import android.app.Activity;
```

```
import android.os.Bundle;

public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

- This is a java class.
- It should extend the android.app.Activity to become an Activity.
- It overrides the onCreate(…) method (this method will be called by the framework when the activity is created. So the code for customizing activity goes here)
- In this case, the only major operation done in the onCreate method is setting the content view to the layout 'main'.
- `R.layout.main` refers to the main.xml file in res/layout folder.

The 'main' layout xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

This is the xml view (not 'Graphical Layout') of main.xml in eclipse.

- The layout (orientation) followed is linear vertical.
- It contains only one element in it i.e. a `TextView`.
- 'TextView' displays a simple text and the text value is mentioned in `@string/hello`
- `@string/hello` refers to the text named 'hello' in strings.xml

Customizing MyFirstApp

Let us modify this application to a greeting application which greets the user using his name. So, before greeting, application should get the user name from user. Steps which we will be following to do that are given below.

- Customize the main layout xml to get name from user.

- Create new activity (java class and layout xml) for greeting the user.
- Add the new activity to the manifest file.
- Modify the HelloWorldActivity.java to read the user name and pass the information to the greeting activity.

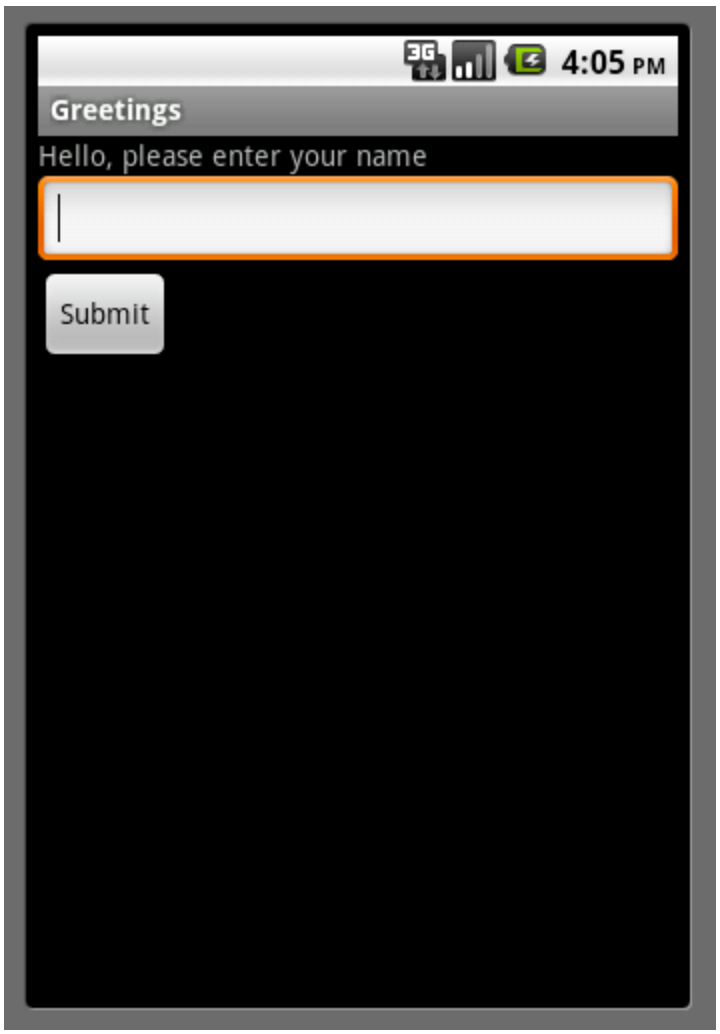
Customizing the main layout xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </EditText>

    <Button
        android:id="@+id/btnNameSubmit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit" />

</LinearLayout>
```

- An ‘EditText’ (text box) and a ‘Button’ are added to the layout.
- Ids are mentioned for both elements. Id is not mandatory but it should be added if you need to access it from activity class.
- For the button, the text is mentioned directly i.e. not using the strings.xml. This is not the standard or preferred way. I did so just to show you that this is possible.
- In strings.xml, modify strings ‘app_name’ and ‘hello’.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, please enter your name</string>
    <string name="app_name">Greetings</string>
</resources>
```

Now run the application to see how the new layout will look like.



Creating the new activity

As part of the new activity, you need to create a layout xml and an activity class.

Create the layout xml ‘greetings.xml’ in res/layout folder.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/greetingText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />

</LinearLayout>
```

The layout xml mainly has a ‘TextView’ with id ‘greetingText’.

Now create the activity class ‘GreetingsActivity.java’ in the package ‘my.first.app’.

```java
package my.first.app;

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

public class GreetingsActivity extends Activity {

	private TextView txtGreetings;

	@Override
	protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.greetings);

		// Initializing the textview object
		txtGreetings = (TextView) findViewById(R.id.greetingText);

		String name = getUserName();
		String greetings = getGreetings(name);
		txtGreetings.setText(greetings);
	}

	/**
	 * Creates the greeting string.
	 *
	 * @param name
	 *            user name
	 * @return the greeting text.
	 */
	private String getGreetings(String name) {
		String g = "Hello";
		// adding name to greeting
		g += (g == null || g.equals("")) ? "" : " " + name;

		// adding message to greeting
		g += ", welcome to the world of Android.";
		return g;
	}

	/**
	 * Fetches the key 'name' in extras bundle.
	 *
	 * @param name
	 *            the key
	 * @return value in bundle for key 'name'
	 */
	private String getUserName() {
		// Getting the 'name' from extras bundle.
		return getIntent().getExtras().getString("name");
	}
}
```

Adding the activity info to manifest file

Then add 'GreetingsActivity' information in Manifest file. For adding it, create a new node 'activity' under node 'application' and mention the activity name and label.

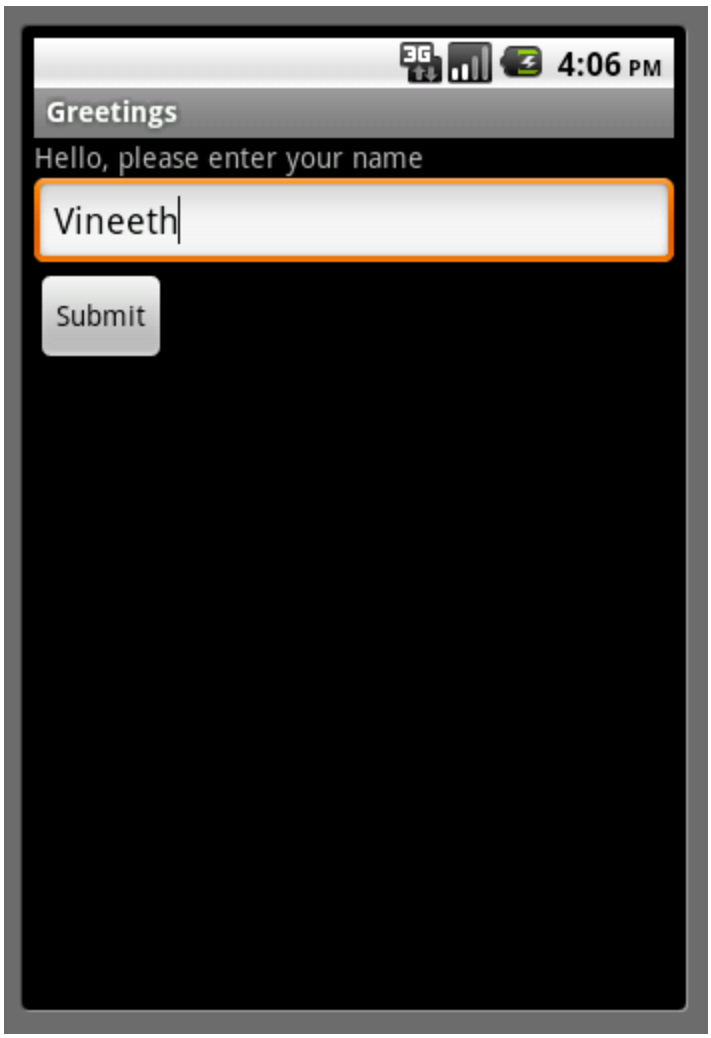
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="my.first.app"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".HelloWorldActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>

        <activity
            android:name=".GreetingsActivity"
            android:label="@string/app_name" >
        </activity>
    </application>

</manifest>
```

Modifying the HelloWorldActivity

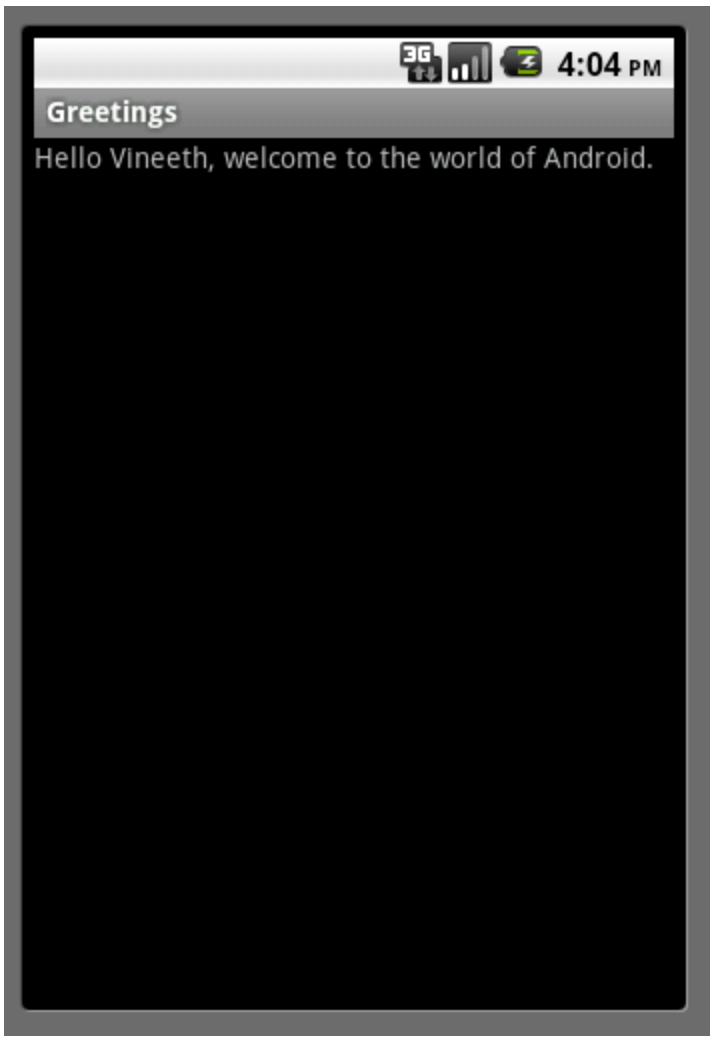
Now modify the HelloWorldActivity.java to read the user name and pass the information to the greeting activity.

```
package my.first.app;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class HelloWorldActivity extends Activity {

	private EditText txtName;
	private Button btnSubmit;

	/** Called when the activity is first created. */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		//Initializing the editText and button.
		txtName = (EditText) findViewById(R.id.editTextName);
		btnSubmit = (Button) findViewById(R.id.btnNameSubmit);

		// Adding click listener to the button.
		btnSubmit.setOnClickListener(new View.OnClickListener() {

			public void onClick(View view) {
				// When the button is clicked, this method will be
called.
				displayGreeting();
			}

		});
	}

	/**
	 * Displays the 'Greetings' activity.
	 */
	private void displayGreeting() {
		String name = txtName.getText().toString();

		// Starts the Greetings activity
		Intent i = new Intent(this, GreetingsActivity.class);
		i.putExtra("name", name);
		startActivity(i);
	}
}
```

Now, run the modified application. Enter a name in the text box and click submit button.



You should be able to see the Greetings activity with the name you mentioned.



What next?

This guide explains only the basic concepts of Android application development. The best way to learn more is by trying more apps and reading more about it. Some related links are given below.

Android Developer home page
http://developer.android.com/index.html

What is Android?
http://developer.android.com/guide/basics/what-is-android.html

Activity and its life cycle
http://developer.android.com/guide/topics/fundamentals/activities.html

User Interface
http://developer.android.com/guide/topics/ui/index.html

XML Layouts
http://developer.android.com/guide/topics/ui/declaring-layout.html

The AndroidManifest.xml File
http://developer.android.com/guide/topics/manifest/manifest-intro.html

The ‘Hello World’ tutorial from Android
http://developer.android.com/resources/tutorials/hello-world.html

SDK Installation guide
http://developer.android.com/sdk/installing.html

References

SDK Installation guide
http://developer.android.com/sdk/installing.html

The ‘Hello World’ tutorial from Android
http://developer.android.com/resources/tutorials/hello-world.html