Beginning Your Android Programming Journey

An Introductory Chapter from *EDUmobile.ORG* Android Development Training Program

NOTICE: You Do NOT Have the Right to Reprint or Resell This eBook!

You Also MAY NOT Give Away, Sell or Share the Content Herein

1. Welcome to Android Application Development

Android is one of the most versatile, powerful and elegant platforms coming out of Google in recent years. It was initially developed by Android Inc later purchased by Google and positioned in the Open Handset Alliance.

As per the NPD group the unit sales Android phones is the largest among Smart Phones.

Because it is widely supported by large number of hardware, software and network carriers its market share is growing worldwide in leaps and bounds.

Being an open system based on modified Linux kernel it has been widely accepted by the developer community and presents a golden opportunity to create products and services for this amazing platform.

Android is an open source platform and it is released under open source license. The Android operating system software stack consists of Java applications running on a Java based object oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation. Libraries written in C include the surface manager, OpenCore media framework, SQLite relational database management system, OpenGL ES 2.0 3D graphics API, WebKit layout engine, SGL graphics engine, SSL, and Bionic libc.

Some of the popular Android Phones:

Motorola Droid

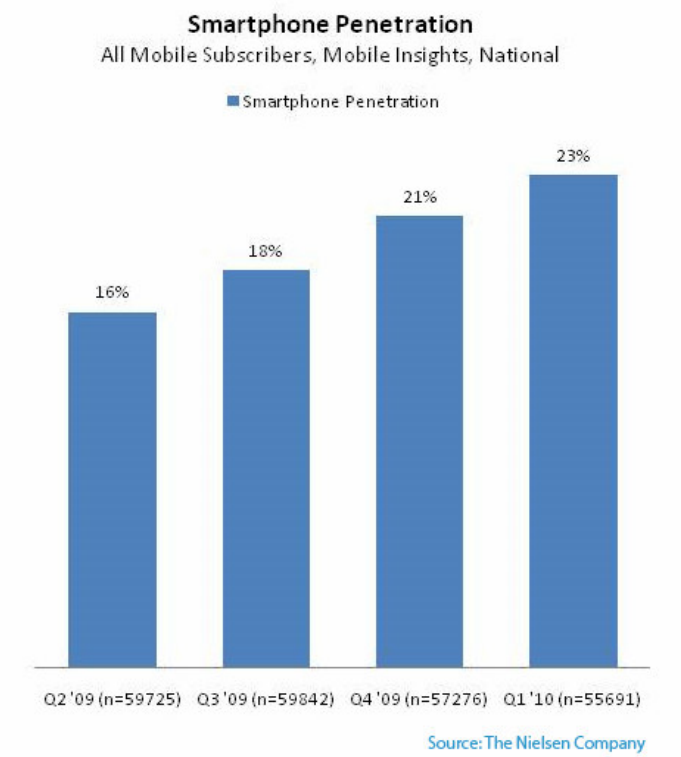
HTC Evo

Nexus One

1.1 The SmartPhone Landscape

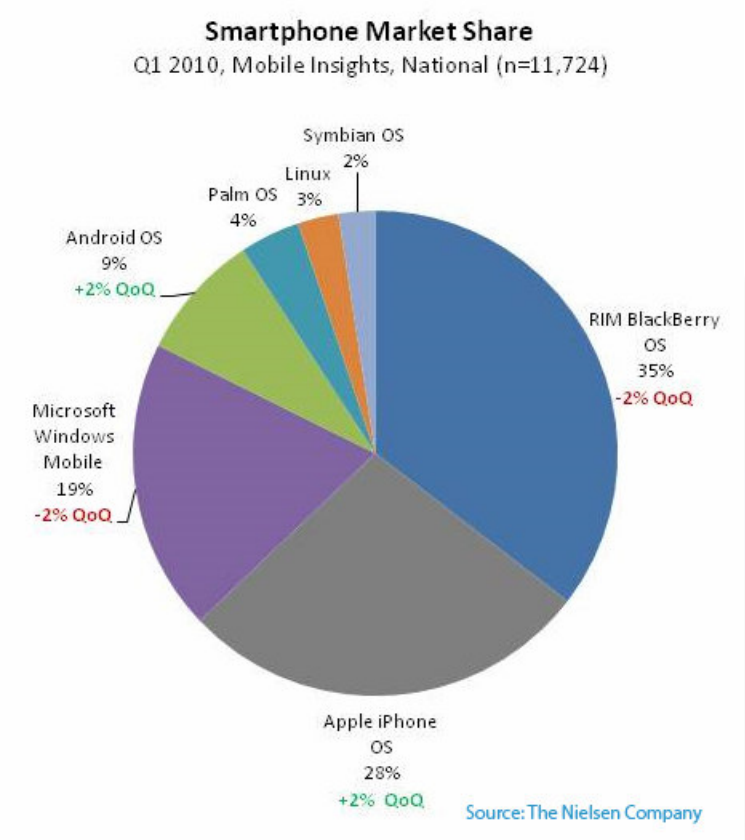
Nokia started the SmartPhone revolution earlier this decade, since then the SmartPhones market has grown from strength to strength. However, with Nokia's Symbian OS quickly loosing market share - Blackberry, Android and iPhone are now are the biggest players in the industry. Android being open source and a wide hardware support is quickly gaining ground for smart phones as well as netbooks and recently, tablet PCs

Generic pattern of SmatPhone penetration



Google's Android operating system has proven itself to be a growing force to be reckoned with as adoption rates among manufactures and consumers continue growing at considerable rates.

Android 2.1 has finally reached a point where it is ready for the mass market. Nexus One with the next release 2.2, AKA Froyo will also soon gain rapid market adoption.



The above data reproduced from Neilsen.com clearly indicates that Smarthone market is one of the rapidly emerging platform.

EDUmobile.ORG is here to help new developers and content enthusiasts to benefit from this great opportunity.

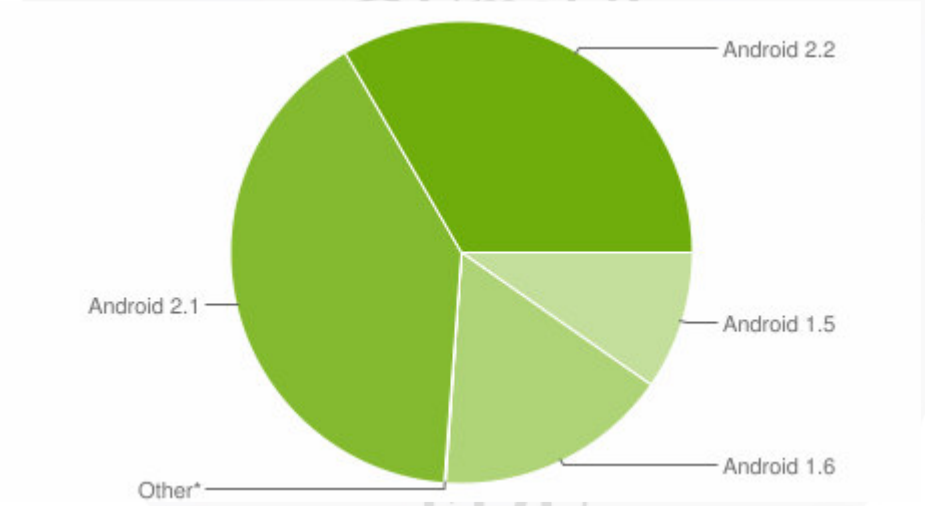
This eBook consists of a brief introduction to Android as a platform. We start by looking at the Android system and the development tools available.

We will go through the installation process of the SDK and we will create some simple programs, that will be side-loaded into a real device. We will finally conclude with way forward for developers to cash in on this opportunity with intuitive and sellable products and services.

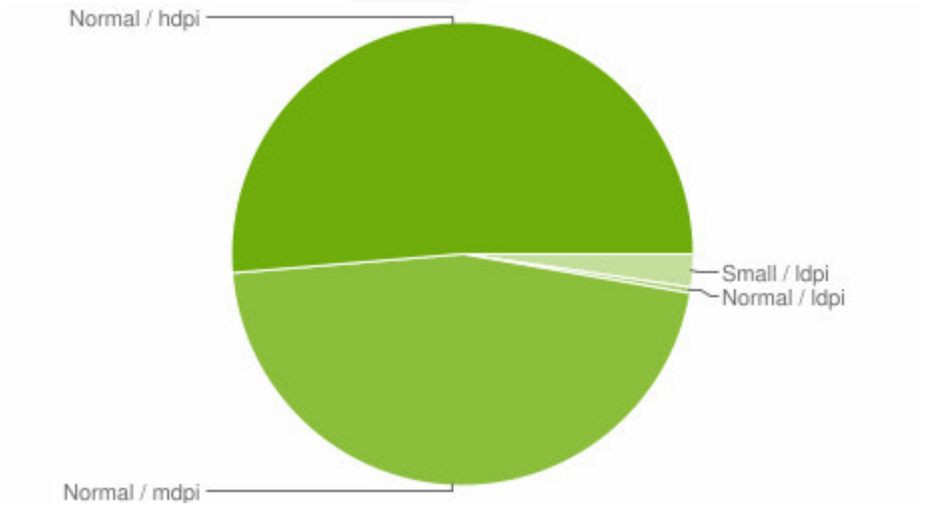
The SDKs distribution currently in the market.

Google has released the Android platform under following versions.

- Android 1.0
- Android 1.1
- Android 1.5
- Android 1.6
- Android 2.1
- Android 2.2



Screen Size Summary



1.2 Android API level

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform. The Android platform provides a framework API that applications can use to interact with the underlying Android system. The framework API consists of:

- A core set of packages and classes
- A set of XML elements and attributes for declaring a manifest file
- A set of XML elements and attributes for declaring and accessing resources
- A set of Intents
- A set of permissions that applications can request, as well as permission enforcements included in the system

Platform Version	API Level
Android 2.2	8
Android 2.1	7
Android 2.0.1	6
Android 2.0	5
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

The above data will give you a peek into the marketplace for which you will create products.

The Android programming is done on two levels

System Level – It involves modifying the Android system code and adapting it to various hardware platforms. It also involves creating additional services and features in the android system itself.

Application level – It involves creating software products and services which sits on top of the Android software stack and interacts with the hardware through the underlying Android platform.

We will limit our discussion in this book only for Application level programming for Android and Android Marketplace. The System level programming is out of the scope of this free ebook and if you want to jump right into it, you should perhaps start with C and Linux kernel programming.

So now that we have defined the scope of our discussions we must outline what is the minimum requirement for you to learn the application programming for Android.

Android applications are built using Java. The Java byte code executes on a efficient and modified virtual machine known as Dalvik Virtual Machine. Dalvik does not align to Java SE nor Java ME Class Library profiles (e.g., Java ME classes, AWT or Swing are not supported). Instead it uses its own library built on a subset of the Apache Harmony Java implementation. So you need to know basic core java but it alone is not sufficient to leverage this great platform for creating great applications. However to follow this book you need to have a basic understanding of Java, XML, Eclipse and object oriented programming principles.

To summarize, you are expected to know -

Core Java – Standard Java programming is necessary to program applications for Android. You should understand operators, loops and should have a basic idea of classes and objects. You can refer here to brush up your java skills

Link - http://download.oracle.com/javase/tutorial/

XML – You should know what the XML standard is all about and how it is used in software applications.

Link - http://www.w3schools.com/xml/default.asp

Eclipse – If you have worked in any IDE for Java you should be ok with eclipse. You can read more about it here.

Link - http://www.vogella.de/articles/Eclipse/article.html

Object fundamentals– You will be able to create elegant and efficient programs if you know the basics of OOPs. Features like Inheritance, Polymorphism, Encapsulation and Overloading help us create efficient and organized software systems and it is expected that you as ajava programmer must know them. Look at the following links for these concepts

Link - http://download.oracle.com/javase/tutorial/java/concepts/

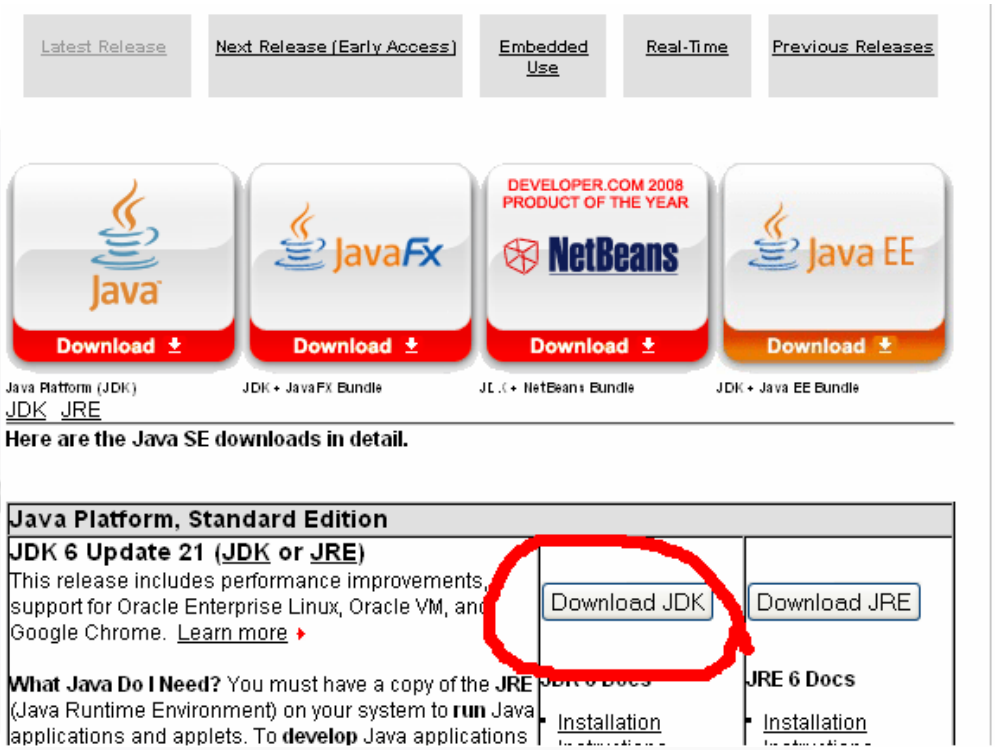
2. Android SDK Installation and Usage.

Please follow the steps mentioned below to install Android SDK and eclipse IDE to get started with application development.

1. **Preparing your development machine**

   Your development system should first download some software before you can program for it. The first one is the JDK which you can download from -

http://www.oracle.com/technetwork/java/javase/downloads/index.html



As this book is targeted towards beginners we suggest using Eclipse and ADT plug-in to develop for android. You can switch to different tool chain once you are familiar with the Android APIs.

You can download eclipse from here - http://www.eclipse.org/downloads/

You can use any eclipse IDE above 3.4 but you should always use the latest version.

Note: Make sure you first install the JDK before installing the Eclipse.

Google Recommends

Operating Systems

- *Windows XP (32-bit), Vista (32- or 64-bit), or Windows 7 (32- or 64-bit)*
- *Mac OS X 10.5.8 or later (x86 only)*
- *Linux (tested on Linux Ubuntu Hardy Heron)*
  - *64-bit distributions must be capable of running 32-bit applications. For information about how to add support for 32-bit applications, see the Ubuntu Linux installation notes.*

Supported Development Environments

Eclipse IDE

- *Eclipse 3.4 (Ganymede) or 3.5 (Galileo)*

  ***Caution**: There are known issues with the ADT plugin running with Eclipse 3.6. Please stay on 3.5 until further notice.*

- *Eclipse JDT plugin (included in most Eclipse IDE packages)*
- *If you need to install or update Eclipse, you can download it from http://www.eclipse.org/downloads/.*

Several types of Eclipse packages are available for each platform. For developing Android applications, we recommend that you install one of these packages:

- *Eclipse IDE for Java EE Developers*
- *Eclipse IDE for Java Developers*
- *Eclipse for RCP/Plug-in Developers*
- *Eclipse Classic (versions 3.5.1 and higher)*

- *JDK 5 or JDK 6 (JRE alone is not sufficient)*
- *Android Development Tools plugin (optional)*
- ***Not** compatible with Gnu Compiler for Java (gcj)*

Other development environments or IDEs

- *JDK 5 or JDK 6 (JRE alone is not sufficient)*
- *Apache Ant 1.6.5 or later for Linux and Mac, 1.7 or later for Windows*
- ***Not** compatible with Gnu Compiler for Java (gcj)*

***Note**: If JDK is already installed on your development computer, please take a moment to make sure that it meets the version requirements listed above. In particular, note that some Linux distributions may include JDK 1.4 or Gnu Compiler for Java, both of which are not supported for Android development.*

Hardware requirements

The Android SDK requires disk storage for all of the components that you choose to install. The table below provides a rough idea of the disk-space requirements to expect, based on the components that you plan to use.

Component type	*Approximate size*	*Comments*
SDK Tools	*50 MB*	*Required.*
Android platform (each)	*150 MB*	*At least one platform is required.*
SDK Add-on (each)	*100 MB*	*Optional.*
USB Driver for Windows	*10 MB*	*Optional. For Windows only.*
Samples (per platform)	*10M*	*Optional.*
Offline documentation	*250 MB*	*Optional.*

***Note** that the disk-space requirements above are in addition to those of the Eclipse IDE, JDK, or other prerequisite tools that you may need to install on your development computer.*

2. Downloading Android Starter package

Once you have your system ready you need to download the Android Starter package. The starter package is not a full development environment — it includes only the core SDK Tools, which you can use to download the rest of the SDK components. You can get the latest version of the SDK starter package from the

Link - http://developer.android.com/sdk/index.html

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r07-windows.zip	23669664 bytes	69c40c2d2e408b623156934f9ae574f0
Mac OS X (intel)	android-sdk_r07-mac_x86.zip	19229546 bytes	0f330ed3ebb36786faf6dc72b8acf819
Linux (i386)	android-sdk_r07-linux_x86.tgz	17114517 bytes	e10c75da3d1aa147ddd4a5c58bfc3646

After downloading, unpack the Android SDK archive to a safe location on your machine. Make a note of the name and location of the unpacked SDK directory on your system — you will need to refer to the SDK directory later, when setting up the ADT plugin or when using the SDK tools.

On Windows, right-click on My Computer, and select Properties. Under the Advanced tab, hit the Environment Variables button, and in the dialog that comes up, double-click on Path (under System Variables). Add the full path to the tools/ directory to the path.

3. Installing the ADT plug-in

ADT has been created for Android to ease the development of applications using an integrated system for development, compiling and signing and transferring to an android device. It helps us quickly integrate java code with Android API and create applications with it. To simplify ADT setup, we recommend installing the Android SDK prior to installing ADT. When your Eclipse and Android SDK environments are ready, continue with the ADT installation as described in the steps below.

1. Start Eclipse, then select **Help** > **Install New Software**.
2. In the Available Software dialog, click **Add...**.
3. In the Add Site dialog that appears, enter a name for the remote site (for example, "Android Plugin") in the "Name" field.

   In the "Location" field, enter this URL:

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

If you have trouble acquiring the plugin, you can try using "http" in the URL, instead of "https" (https is preferred for security reasons.

Click **OK**.

4. Back in the Available Software view, you should now see "Developer Tools" added to the list. Select the checkbox next to Developer Tools, which will automatically select the nested tools Android DDMS and Android Development Tools. Click **Next**.
5. In the resulting Install Details dialog, the Android DDMS and Android Development Tools features are listed. Click **Next** to read and accept the license agreement and install any dependencies, then click **Finish**.
6. Restart Eclipse.

4. Configuring the ADT Plugin

Once you've downloaded ADT the next step is to modify your ADT preferences in Eclipse to point to the Android SDK directory:

1. Select **Window** > **Preferences...** to open the Preferences panel (Mac OS X: **Eclipse** > **Preferences**).
2. Select **Android** from the left panel.
3. For the *SDK Location* in the main panel, click **Browse...** and locate your downloaded SDK directory.
4. Click **Apply**, then **OK**.

Adding Android platform

The last step is to use AVD manager to install various components into you development environment.

5. Launching from Eclipse/ADT

If you are developing in Eclipse and have already installed the ADT Plugin, follow these steps to access the Android SDK and AVD Manager tool:

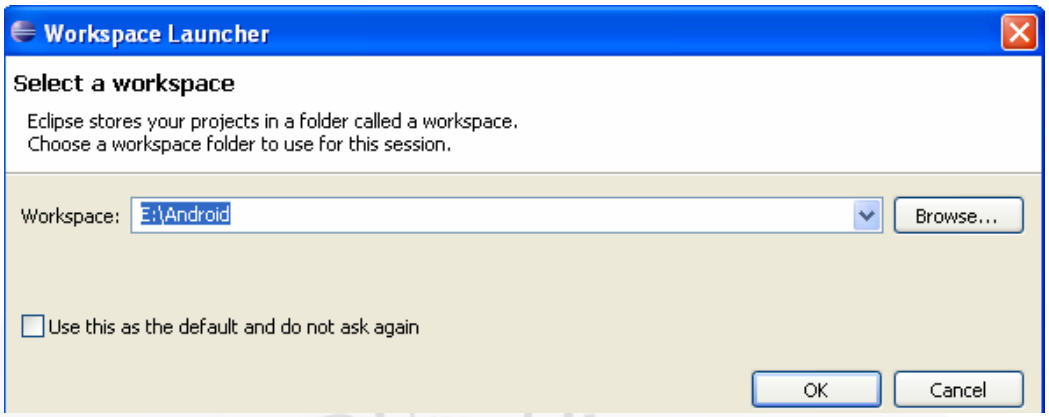
1. Open Eclipse
2. Select **Window** > **Android SDK and AVD Manager**.
3. Select **Available Packages** in the left panel. This will reveal all of the components that are currently available for download from the SDK repository.

4. Select the component(s) you'd like to install and click **Install Selected**.
5. Verify and accept the components you want and click **Install Accepted**. The components will now be installed into your existing Android SDK directories.

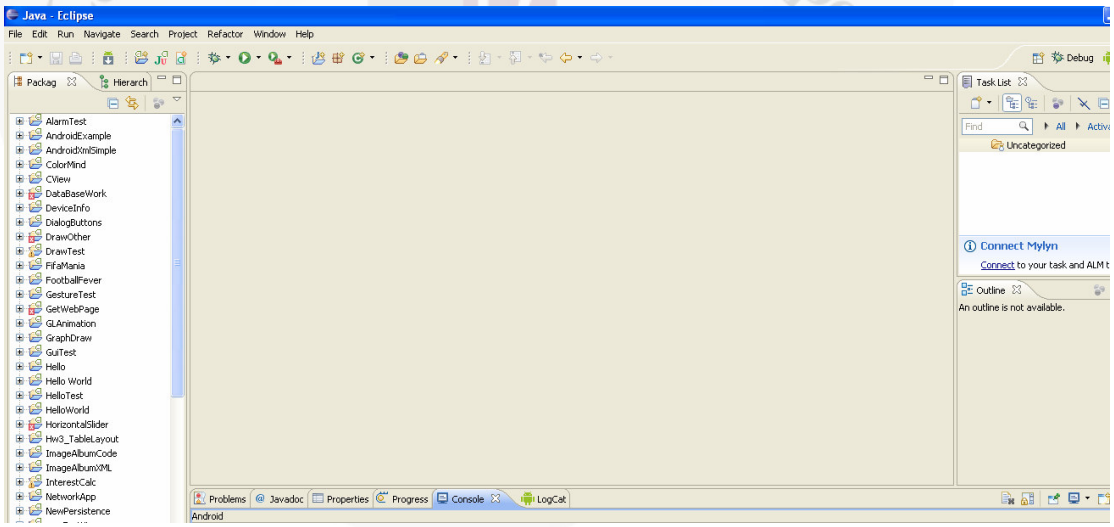
New platforms are automatically saved into the <*sdk*>/platforms/ directory of your SDK; new add-ons are saved in the <*sdk*>/add-ons/ directory; samples are saved in the <*sdk*>/samples/android-<*level*>/; and new documentation is saved in the existing <*sdk*>/docs/ directory (old docs are replaced).

3. Creating a Hello World App for Android

Now our system is ready for work. Start your eclipse and create a work directory if not already created.



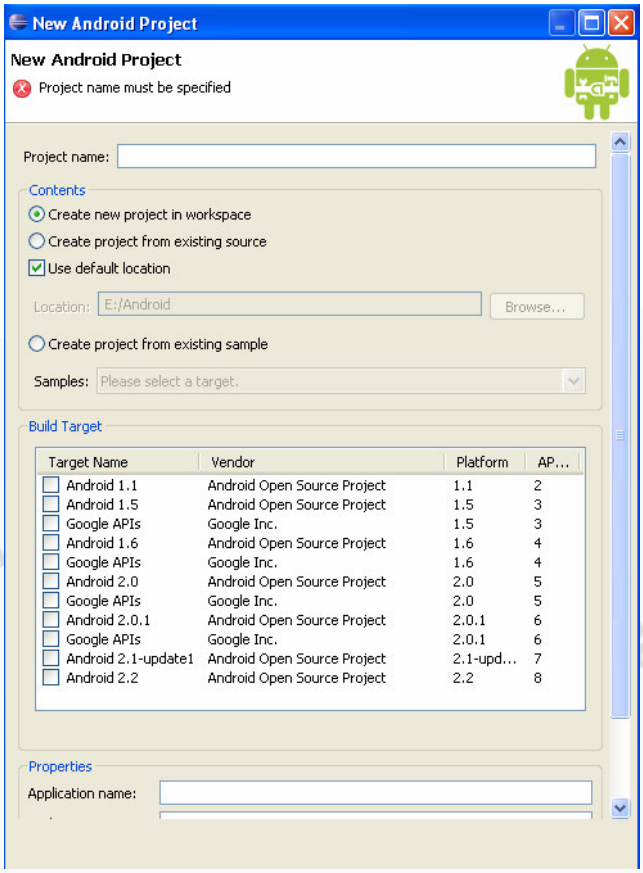
You will see the eclipse ID as shown in the figure below.



The left side of the UI shows the projects created by you. The central UI will show the code written by you and the right end of the figure shows the task lists. The lower end of it shows the console and logger which is very useful while debugging.

Steps to create Hello World Example

1. Open the eclipse IDE in a work space and click
   File > New > Other > Android Project and click Next

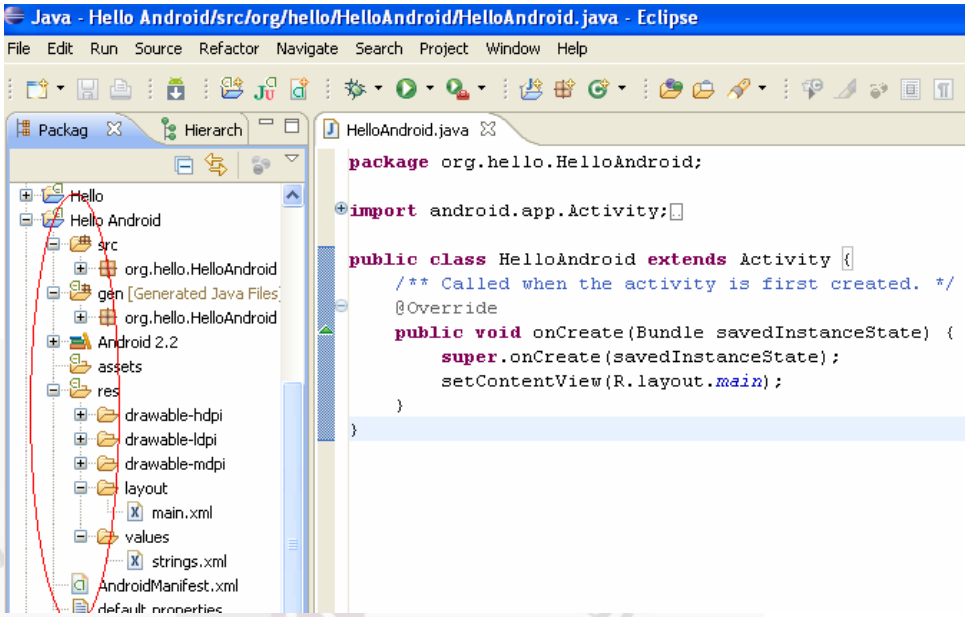


2. Fill the details with project name as “Hello Android”. Under contents Select “Create new Project in Workspace” and tick “Use Default Location”.

    2. Select build target as the latest SDK available. Fill in the properties as follows
       Application Name – Hello Android
       Package Name – org.hello.HelloAndroid
       Create Activity – HelloAndroid

    Click “Finish” to create the project in your workspace.

    3. You will see the following files created automatically by the SDK.



A new activity java file gets created as shown above contains the onCreate() method which is the first method to be called when the Application starts. Let us look into the file structure. The Master folder is same as Application name it is Hello Android in our case. It contains four subfolders including src, gen, res and Android SDK files.

src – It contains the source packages and java source files. In our src folder it currently contains the package or.hello.HelloAndroid. The package further contains the jave file "HelloAndroid.java".

```
package org.hello.HelloAndroid;

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

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

Let us spend some time understanding the code that got auto generated. If you have programmed in java before you can make out most of the code

above. However if you are new to java and programming, let me clarify few things for you.

- The first line declares the package **"org.hello.HelloAndroid"**. A package is a namespace that organizes a set of related classes and interfaces. So all the folders for this project will be contained in this package and they will contain different elements like images, sound files and java source files.

- The next two lines are importing standard packages for the Android specific java code. Import is the key word which is used to access the standard and no standard packages inside a java file.

- We then create our class HelloAndroid which inherits the Activity class. Activity is a standard class of Android which we will discuss in detail a bit later.

- Inside the class we define a method onCreate() which is called when the activity is starting. This is where most initialization happens. The setContentView() inflates the activity's UI and in our example it is calling the main xml discussed below to draw the user Interface.

To sum up this class imports standard definitions and create a class which is a subclass(inherited) of an Activity. The class further has a method called as onCreate() which initializes and paints the UI from a main file. So now that it makes sense lets move ahead.

gen – It contains the auto generated java files. As these files you should make any changes in them. If you make changes in source code the code in this folder will get modified automatically.

Android – It contains the particular SDK libraries being used for the current project.

res – It is one of the other important content folders. It contains three subfolders for images namely drawable –hdpi, drawable –ldpi, drawable –

mdpi. The other two subfolders are layout and values. The layout contains the main.xml which is called when the application is started. If you are familiar with c or java programming you know the function main which is called when ever the programs first starts and in a similar way the main.xml draws its content as soon as the application starts. The strings.xml contained in the values folder is used to define strings to be used within the applications.

Let us look into these two important XML files in detail.

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/r
es/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
 />
</LinearLayout>
```


It starts with the LinearLayout tag which implies that we want to put some components on the screen in a linear fashion. There are many other layouts also defined in Android which you will learn later. The orientation, width and height describe how the layout should look. We follow it with a "TextView" component which is used to display texts on the screen. In this example it is taking the text from the "hello" string defined in the "strings.xml".

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="hello">Hello World,
HelloAndroid!</string>
<string name="app_name">Hello Android</string>
</resources>
```

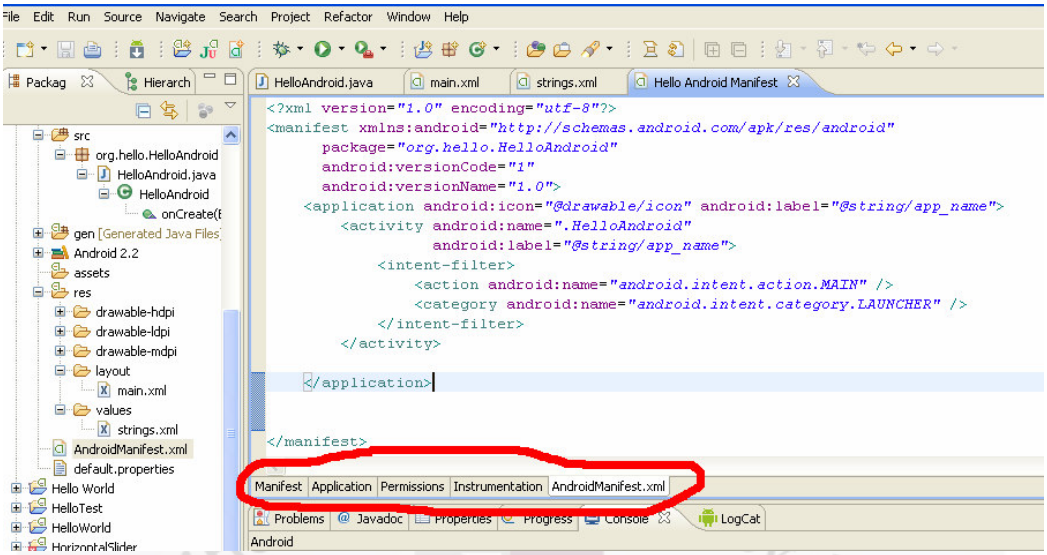
From the above content you can easily make out that the hello string corresponds to the actual string “Hello World, HelloAndroid! Which is same as the application name we gave.

Let us now look into another important xml file created by the SDK for us. It is the android manifest file.

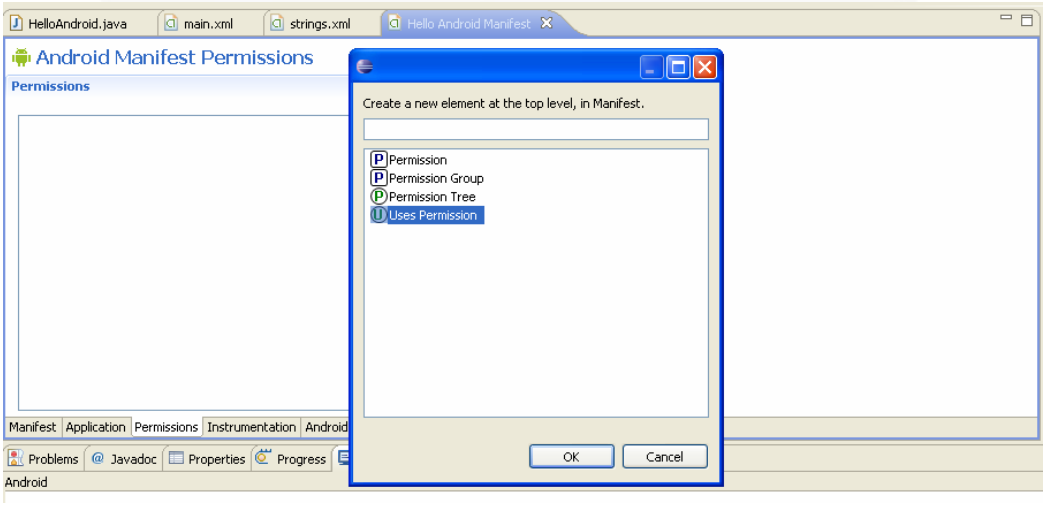
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/r
es/android"
package="org.hello.HelloAndroid"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon"
android:label="@string/app_name">
<activity android:name=".HelloAndroid"
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>
```

This file outlines the main xml and the activity (type of process) which should start after loading the application. You can define specific permissions for the application like Network access and SMS inbox access. The package name, version number and other details about the application is also contained in this file. It primarily consists of four important parts which come together as a XML file. These four components are Manifest, Application, Permission and Instrumentation.

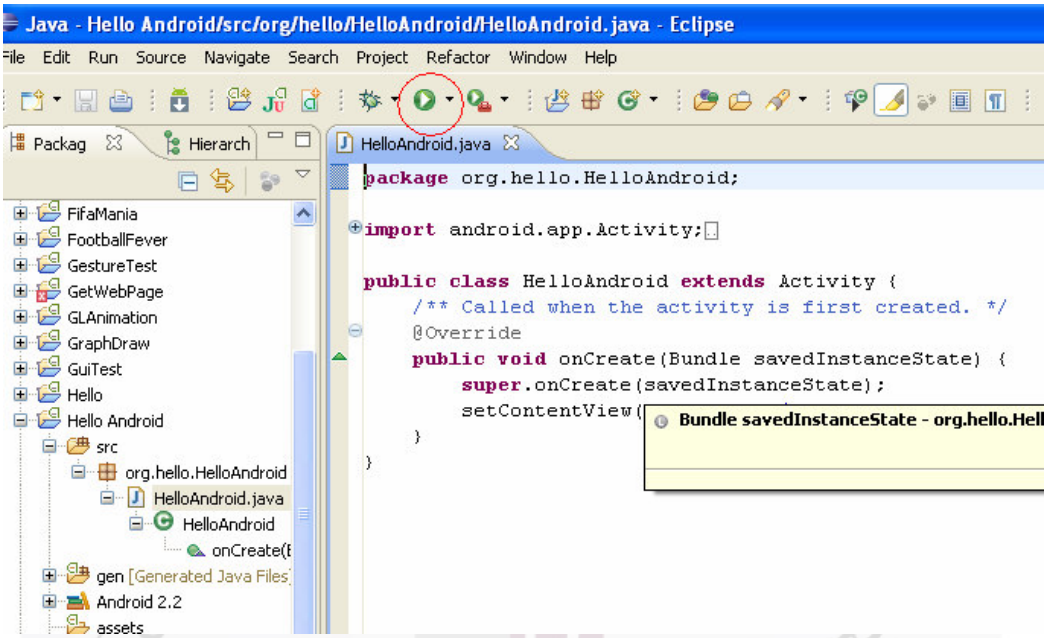


You will be making changes in the four components and the XML will get modified automatically. Like if we are to add permission we will go to the permission tab and add the permission as shown below.

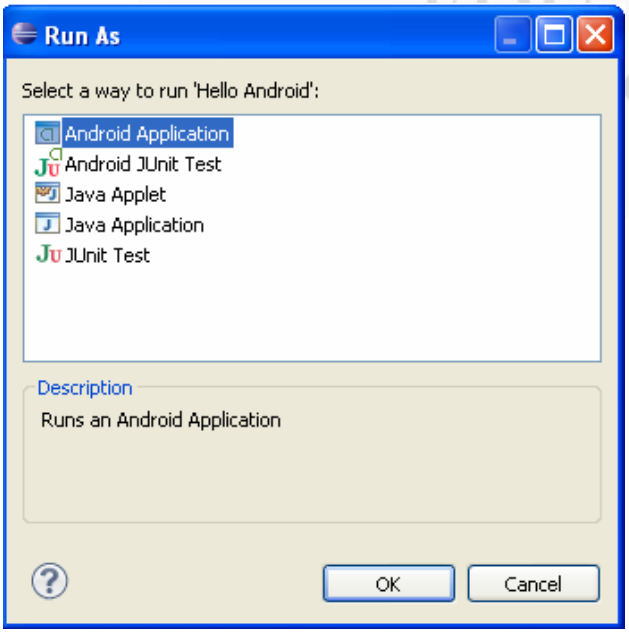


Compiling and executing our Hello World.

It is easy to compile the code and run the device simulator associated with the SDK.

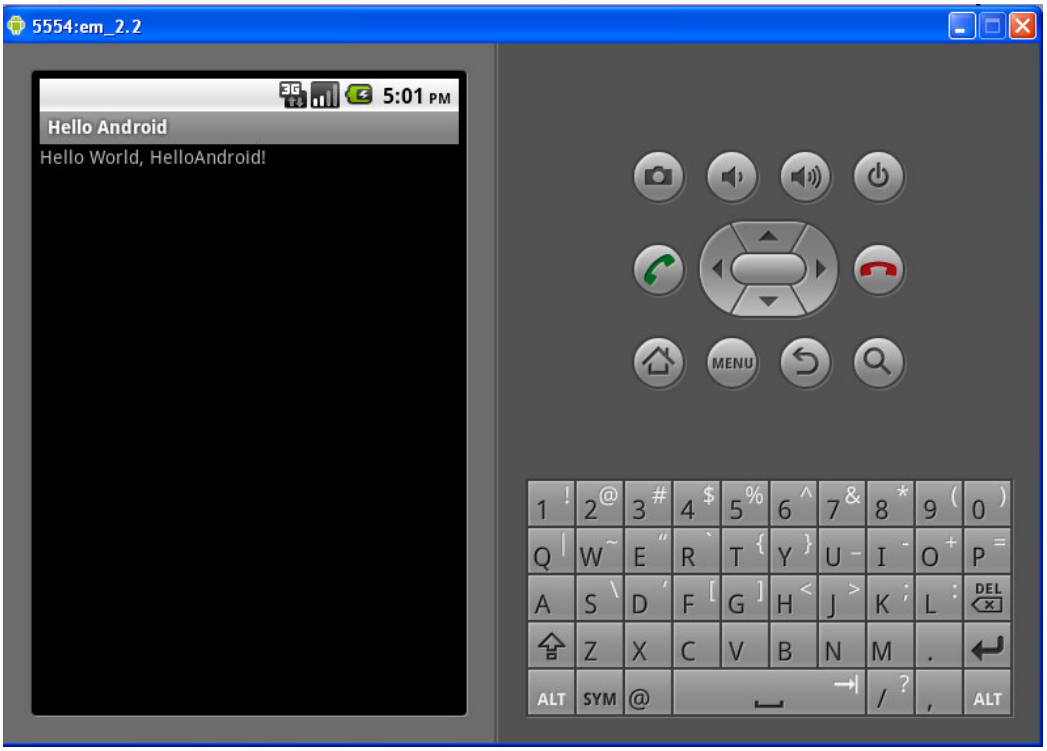


Click on the “Run As” as shown in the figure above. A selection window appears as shown below. Select the Android Application from it.



Once you press ok your code will start compiling and your simulator will start.

Click on the “menu” key on the simulator and you can see the result of the application.

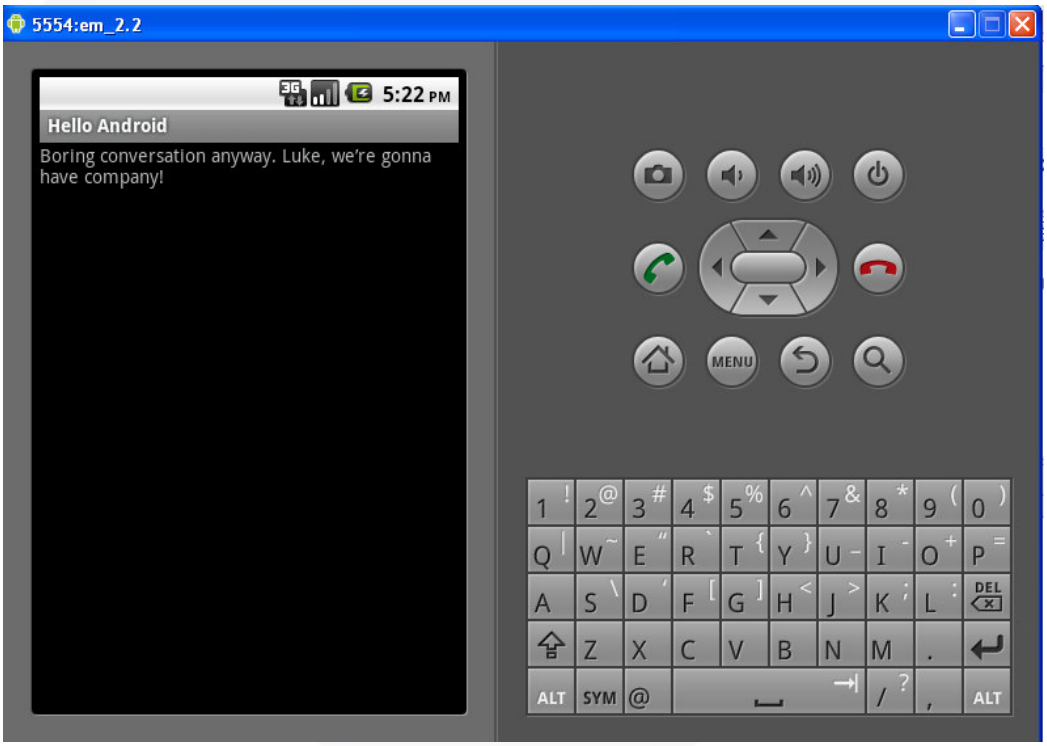


This example of ours was very basic but it still is a complete Android Application and you can take some time to grasp the structure and basic aspects of an Android Application. Let us in the meantime try and modify this program of ours and put some other text in the output screen.

Open the strings.xml of your project and make the following changes

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Boring conversation anyway. Luke, we're
gonna have company!</string>
    <string name="app_name">Hello Android</string>
</resources>
```

Save your project and run your project once again. You should now see the following



Go through all the java and xml files for the project before you move ahead.

4. A Bit Of Theory

Let us now go through some important features of Android and see how they are linked and provides application developers powerful environment to work in. It primarily consists of a stack of operating system, middleware and key applications which can be summarized as follows

- **Application framework** – The application framework is a component driven system where old components are replaced and new components are added thereby keeping the environment dynamic.
- **Dalvik Virtual machine** – it is an optimized java virtual machine.
- **Integrated Browser** – Webkit based internet browser.
- **SQLite** – A relational data base for applications.
- **Media support** – It has codecs for common video and audio support.
- **Telephony components** – These include other GSM and CDMA telephony components along with other phone features.

Structurally Android can be broken down into four major parts -

Applications – The actual application that the user interacts with resides in this layer. All your third party applications you create will belong to this part of the system.

Application Framework – The application framework provides usable components for the application developers on top of which new applications can be built. It include things like Views, Resource manager, content providers and notification manager helping the application use underlying standard libraries.
Libraries – It is the C/C++ libraries used by Android system. These are provided to the developer through Application framework.

Linux Kernel – Android uses linux kernel for hardware management and providing hardware abstraction to the rest of the software stack.

4.1 Application Development Basics

Android application development is done in java programming language. The compiled code is bundled into an Android package which can be signed and installed on the mobile phone. The Android application can be considered as a series of processes and the control moves from one process to another creating an application for the user to interact with. Hence there is no single entry point for Android application like main but rather there are components which can start as the need arises. These components can be classified into four parts:

1. **Activity** – An activity is use to present visual interface to the user. If your application draws a view similar to what we did in our hello world application where we created a Textview is an example of an activity. You can have many activities based on the user interfaces used by your application but each one of them is a subclass of the

Activity class. Each activity has a separate window to draw the visual component but if required it can use more than one window also.

2. **Services** – A services does not have a visual component but runs in the background and carries out some background process while the other visual component s are running in the foreground. A service will inherit Service base class. It runs in the main thread and for resource intensive tasks it can branch out a new thread.
3. **Broadcast receivers** – This component are responsible for receiving and reacting to broadcasted messages. Broadcast can be system generated like the "Low battery" but it can be from other applications running in the background. They must inherit BroadcastReceivers base class.
4. **Content providers** – This provides specific set of data from one application to other applications. The content providers extend ContentProvider base class. These are used in conjunction with Content resolver which provide methods for inter process communication.

To summarize before developing an application you need to decide the structure of your application in terms of these fundamental entities. You need to decide what your visual entities are and what resultant activities are there. You further need to decide if you will be using a back ground process as a service or not and if you want to handle Broadcast messages. These basic decisions will carve out the basic design of your application which can then be designed at a much lower level.

4.2 The Road Ahead

So, this was our introductory lesson on Android application development and the revolutionary it presents to the developers as a platform. So what should be your next steps? I suggest the following

- Master Core Java – You should know basic objective oriented programming and should know how to use java to leverage the power of object oriented programming.
- Start with basic Android APIs like activity and intents to create simple programs using layouts.
- Move next to 2d graphics and multimedia and networking.
- Next inline is persistence and SQLite.
- You can further expand into Opengl-ES based on your needs.

So, What Next ?

This was a small step in our journey towards getting started with Android development. If you are interested in mastering NAdroid Development, we suggest you sign up for our **complete 12 Week Android Programming Course** at the link below. It comes with a full money back guarantee, so there is no risk at all.

You will be assigned your own personal one-on-one tutor, whom you may consult any time with questions and problems you face.

The course is delivered over a period of 12 weeks via Online Video that you may download and view at your convenience, along with material and Weekly Worksheets.

You will also get access to our private forum where you can meet other fellow Android developers.

Click Here To Learn More About The Android Course and To Enroll

Affiliates Join Our High Paying Affiliate Program and Make Tons Of Money!

- Interested in promoting our Courses and making money?
- Get instant $25 bonus for signing up!

Click Here to sign up for our Affiliate Program