Unit 1
Introduction to Android

Syllabus: The Android 4.1 jelly Bean SDK, Understanding the Android Software Stack, installing the Android SDK, Creating Android Virtual Devices, Creating the First Android Project, Using the Text view Control, Using the Android Emulator, The Android Debug Bridge(ADB), Launching Android Applications on a Handset.

INTRODUCTION

When we talked about operating systems few years ago, the most common answers were Windows, Linux, and mac operating system. However, with the undying competition in the mobile phones market, the next big thing entered was ANDROID, which in no time became the heart of smart phones. Android provides a rich application framework that allows you to build innovative apps and games for mobile devices in a Java language environment.

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touch screen mobile devices such as smart phones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

The Android Operating System is a Linux-based OS developed by the Open Handset Alliance (OHA). The Android OS was originally created by Android, Inc., which was bought by Google in 2005. Google teamed up with other companies to form the Open Handset Alliance (OHA), which has become responsible for the continued development of the Android OS.

The android is a powerful operating system and it supports large number of applications in Smart phones. These applications are more comfortable and advanced for the users. The hardware that supports android software is based on ARM architecture platform. The android is an open source operating system means that it’s free and any one can use it.

The android has got millions of apps available that can help you managing your life one or other way and it is available low cost in market at that reasons android is very popular.

Each time the OHA releases an Android version, it names the release after a dessert. Android 1.5 is known as Cupcake, 1.6 as Donut, 2.0/2.1 as Eclair, 2.2 as Froyo and 2.3 is dubbed Gingerbread. Once a version is released, so is its source code.

The Android OS is designed for phones. The important features of android are given below:

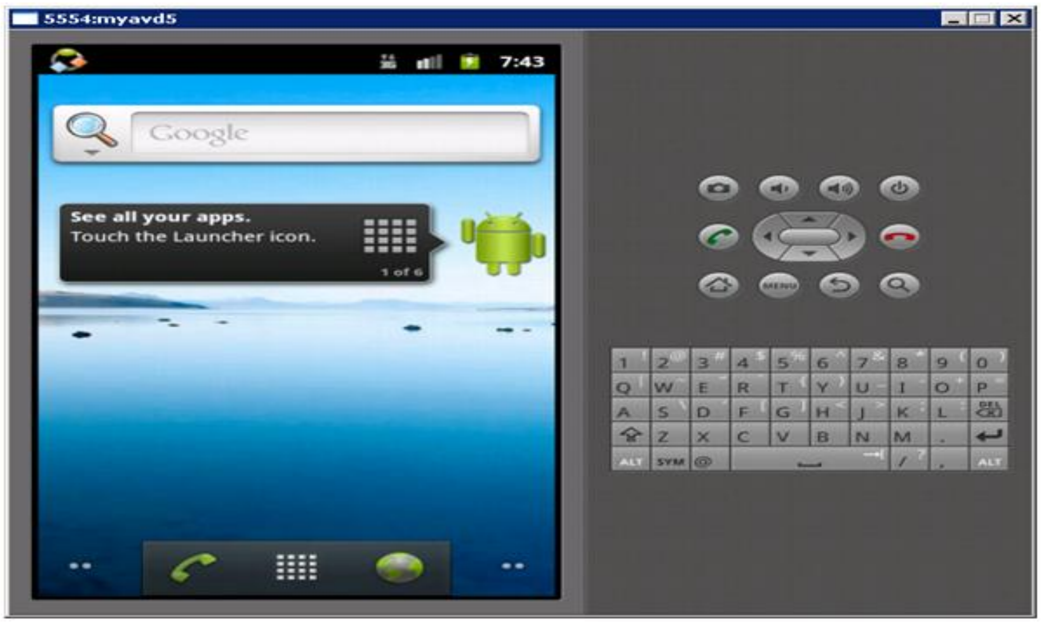
1) It is open-source.
2) Anyone can customize the Android Platform.
3) There are a lot of mobile applications that can be chosen by the consumer.
4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

It provides support for messaging services(SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

Software developers who want to create applications for the Android OS can download the Android Software Development Kit (SDK) for a specific version. The SDK includes a debugger, libraries, an emulator, some documentation, sample code and tutorials. For faster development, interested parties can use graphical integrated development environments (IDEs) such as Eclipse to write applications in Java.

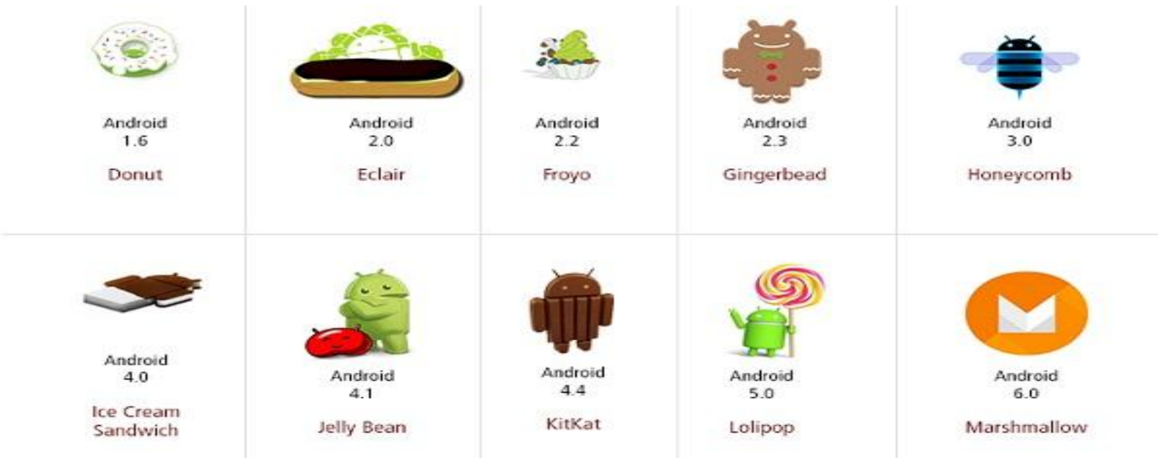
Android Emulator:

The Emulator is a new application in android operating system. The emulator is a new prototype that is used to develop and test android applications without using any physical device.



The android emulator has all of the hardware and software features like mobile device except phone calls. It provides a variety of navigation and control keys. It also provides a screen to display your application. The emulators utilize the android virtual device configurations. Once your application is running on it, it can use services of the android platform to help other applications, access the network, play audio, video, store and retrieve the data.

Android versions: Google did not attach any high-calorie code name to its initial versions 1.0 and 1.1 of the Android Operating System. The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence.



THE ANDROID 4.1 JELLY BEAN SDK

The Android 4.1 Jelly Bean SDK was released with new features for developers in July 2012. It improves the beauty and simplicity of Android 4.0 and is a major platform release that adds a variety of new features for users and app developers. A few of the big features of this release include the following:

• **Project Butter**—Makes the Jelly Bean UI faster and more responsive. Also CPU Touch Responsiveness is added, which increases CPU performance whenever the screen is touched. It uses the finger's speed and direction to predict where it will be located after some milliseconds, hence making the navigation faster.

• **Faster speech recognition**—Speech recognition is now faster and doesn't require any network to convert voice into text. That is, users can dictate to the device without an Internet connection.

• **Improved notification system**— The notifications include pictures and lists along with text. Notifications can be expanded or collapsed through a variety of gestures, and users can block notifications if desired. The notifications also include action buttons that enable users to call directly from the notification menu rather replying to email.

• **Supports new languages**—Jelly Bean includes support for several languages including Arabic, Hebrew, Hindi, and Thai. It also supports bidirectional text.

• **Predictive keyboard**—On the basis of the current context, the next word of the message is automatically predicted.

• **Auto-arranging Home screen**—Icons and widgets automatically resize and realign as per the existing space.

• **Helpful for visually impaired users**—The Gesture Mode combined with voice helps visually impaired users to easily navigate the user interface.

• **Improved Camera app**—The Jelly Bean Camera app includes a new review mode of the captured photos. Users can swipe in from the right of the screen to quickly view the captured photos. Also, users can pinch to switch to a new film strip view, where they can swipe to delete photos.

• **Better communication in Jelly Bean**—Two devices can communicate with Near Field Communication (NFC); that is, two NFC-enabled Android devices can be tapped to share data. Also, Android devices can be paired to Bluetooth devices that support the Simple Secure Pairing standard by just tapping them together.

• **Improved Google Voice search**—Jelly Bean is equipped with a question and answer search method that helps in solving users' queries similar to Apple's popular Siri.

• **Face Unlock**—Unlocks the device when the user looks at it. It also prevents the screen from blacking out. Optionally "blink" can be used to confirm that a live person is unlocking the device instead of a photo.

• **Google Now**—Provides users "just the right information at just the right time." It displays cards to show desired information automatically. For example, the Places card displays nearby restaurants and shops while moving; the Transit card displays information on the next train or bus when the user is near a bus stop or railway station; the Sports card displays live scores or upcoming game events; the Weather card displays the weather conditions at a user's current location, and so on.

• **Google Play Widgets**—Provides quick and easy access to movies, games, magazines, and other media on the device. It also suggests new purchases on Google Play.

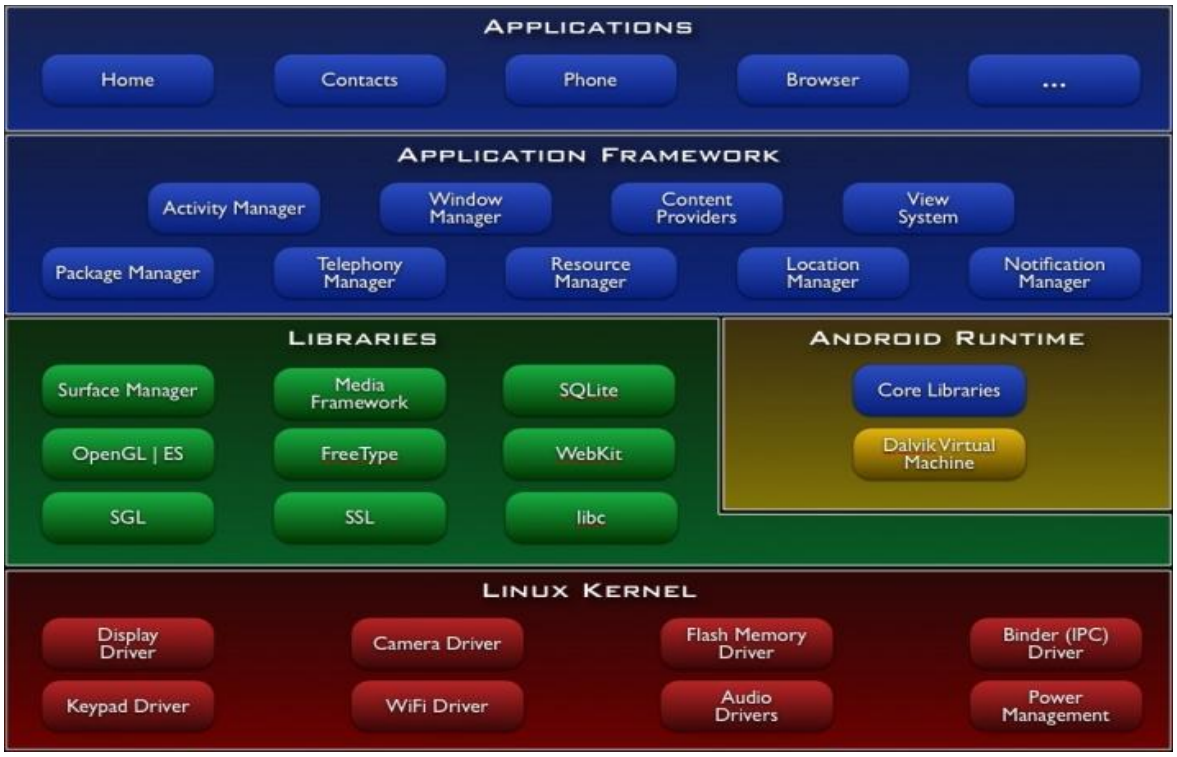
• **Faster Google Search**—Google Search can be opened quickly, from the lock screen and from the system bar by swiping up and also by tapping a hardware search key if it is available on the device.

• **Supports antipiracy**—This feature supports developers in the sense that the applications are encrypted with a device-specific key making it difficult to copy and upload them to the Internet.

UNDERSTANDING THE ANDROID SOFTWARE STACK/ Android Architecture

The Android operating system is built on top of a modified Linux kernel. The software stack contains Java applications running on top of a virtual machine. Components of the system are written in Java, C, C++, and XML. Android operating system is a stack of software components which is roughly divided into five sections

1) Linux kernel
2) Native libraries (middleware),
3) Android Runtime
4) Application Framework
5) Applications



1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. Linux kernel is responsible for device drivers, power management, memory management, device management and resource access. This layer is the foundation of the Android Platform.

- Contains all low level drivers for various hardware components support.
- Android Runtime relies on Linux Kernel for core system services like,
- Memory, process management, threading etc.
- Network stack
- Driver model
- Security and more.

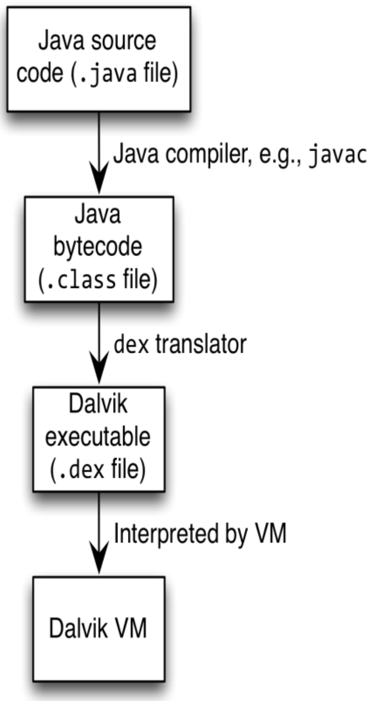
2) Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

- **SQLite** Library used for data storage and light in terms of mobile memory footprints and task execution.
- **WebKit** Library mainly provides Web Browsing engine and a lot more related features.
- The **surface manager** library is responsible for rendering windows and drawing surfaces of various apps on the screen.
- The **media framework** library provides media codecs for audio and video.
- The **OpenGl** (Open Graphics Library) and **SGL**(Scalable Graphics Library) are the graphics libraries for 3D and 2D rendering, respectively.
- The **FreeType** Library is used for rendering fonts.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.


Java source code (.java file)
Java compiler, e.g., javac
Java bytecode (.class file)
dex translator
Dalvik executable (.dex file)
Interpreted by VM
Dalvik VM

- Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.
- Android apps execute on Dalvik VM, a "clean-room" implementation of JVM
- Dalvik optimized for efficient execution
- Dalvik: register-based VM, unlike Oracle's stack-based JVM
- Java .class bytecode translated to Dalvik EXecutable (DEX) bytecode, which Dalvik interprets

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes Android API's such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

- **Activity Manager**: manages the life cycle of an applications and maintains the back stack as well so that the applications running on different processes has smooth navigations.
- **Package Manager**: keeps track of which applications are installed in your device.
- **Window Manager** : Manages windows which are java programming abstractions on top of lower level surfaces provided by surface manager.
- **Telephony Managers**: manages the API which is use to build the phone applications

- **Content Providers:** Provide feature where one application can share the data with another application. like phone number , address, etc
- **View Manager** : Buttons , Edit text , all the building blocks of UI, event dispatching etc.

5) Applications

- On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernel. Any applications that you write are located at this layer.

Android Studio Installation:

1)First of all, Download android studio from this link: https://developer.android.com/studio/index.html

2)JDK 8 is required when developing for Android 5.0 and higher (JRE is not enough). To check if you have JDK installed (and which version), open a terminal and type javac -version. If the JDK is not available or the version is lower than 6, download it from this link.

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

3)To set up Android Studio on **Windows**:

1. Launch the .exe file you just downloaded.

2. Follow the setup wizard to install Android Studio and any necessary SDK tools.

On some Windows systems, the launcher script does not find where Java is installed. If you encounter this problem, you need to set an environment variable indicating the correct location.

Select Start menu > Computer > System Properties > Advanced System Properties. Then open Advanced tab > Environment Variables and add a new system variable JAVA_HOME that points to your JDK folder, for example C:\Program Files\Java\jdk1.8.x.(where x is version number).

<u>To set up Android Studio on Mac OSX</u>:

1. Launch the .dmg file you just downloaded.

2. Drag and drop Android Studio into the Applications folder.

3. Open Android Studio and follow the setup wizard to install any necessary SDK tools.

Depending on your security settings, when you attempt to open Android Studio, you might see a warning that says the package is damaged and should be moved to the trash. If this happens, go to System Preferences > Security & Privacy and under Allow applications downloaded from, select Anywhere. Then open Android Studio again.

If you need use the Android SDK tools from a command line, you can access them at:

/Users/<user>/Library/Android/sdk/ To set up Android Studio on **Linux**:
1. Unpack the downloaded ZIP file into an appropriate location for your applications.

2. To launch Android Studio, navigate to the android-studio/bin/ directory in a terminal and execute studio.sh. You may want to add android-studio/bin/ to your PATH environmental variable so that you can start Android Studio from any directory.

3. Follow the setup wizard to install any necessary SDK tools.

Android Studio is now ready and loaded with the Android developer tools, but there are still a couple packages you should add to make your Android SDK complete.

4)The SDK separates tools, platforms, and other components into packages you can download as needed using the Android SDK Manager. Make sure that you have downloaded all these packages.

To start adding packages, launch the Android SDK Manager in one of the following ways:

- In Android Studio, click SDK Manager in the toolbar.
- If you're not using Android Studio:

Windows: Double-click the SDK Manager.exe file at the root of the Android SDK directory.

Mac/Linux: Open a terminal and navigate to the tools/ directory in the Android SDK, then execute android sdk.

5)Now get all the SDK tools, support libraries for additional APIs, Google Play services (if you need to use them).
6)Once you've selected all the desired packages, continue to install:

Click Install X packages.

In the next window, double-click each package name on the left to accept the license agreement for each.

Click Install.

The download progress is shown at the bottom of the SDK Manager window. Do not exit the SDK Manager or it will cancel the download.

7)Now that we have downloaded and installed everything we need. Enjoy your experience with Android. Best of luck from Internshala.com

INSTALLING THE ANDROID SDK

For developing native Android applications that you can publish on the Google Play marketplace, you need to install the following four applications:

• **The Java Development Kit (JDK)** can be downloaded from http://oracle.com/technetwork/java/javase/downloads/index.html.

• **The Eclipse IDE** can be downloaded from http://www.eclipse.org/downloads/.

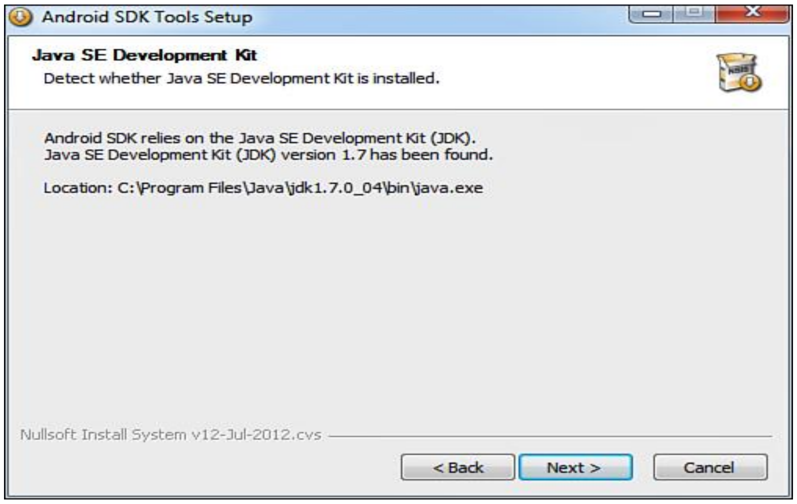
• **The Android Platform SDK Starter Package** can be download from http://developer.android.com/sdk/index.html.

• **The Android Development Tools (ADT) Plug-in** can be downloaded from http://developer.android.com/sdk/eclipse-adt.html. The plug-in contains project templates and Eclipse tools that help in creating and managing Android projects.

The Android SDK is not a full development environment and includes only the core SDK Tools, which are used to download the rest of the SDK components. This means that after installing the Android SDK Tools, you need to install Android platform tools and the other components required for developing Android applications. Go to http://developer.android.com/sdk/index.html and download the package by selecting the link for your operating system.

The first screen is a Welcome screen. Select the Next button to move to the next screen. Because the Android SDK requires the Java SE Development Kit for its operation, it checks for the presence of JDK on your computer.

If Java is already installed on your computer before beginning with Android SDK installation, the wizard detects its presence and displays the version number of the JDK found on the machine, as shown in Figure.


Android SDK Tools Setup
Java SE Development Kit
Detect whether Java SE Development Kit is installed.

Android SDK relies on the Java SE Development Kit (JDK).
Java SE Development Kit (JDK) version 1.7 has been found.

Location: C:\Program Files\Java\jdk1.7.0_04\bin\java.exe

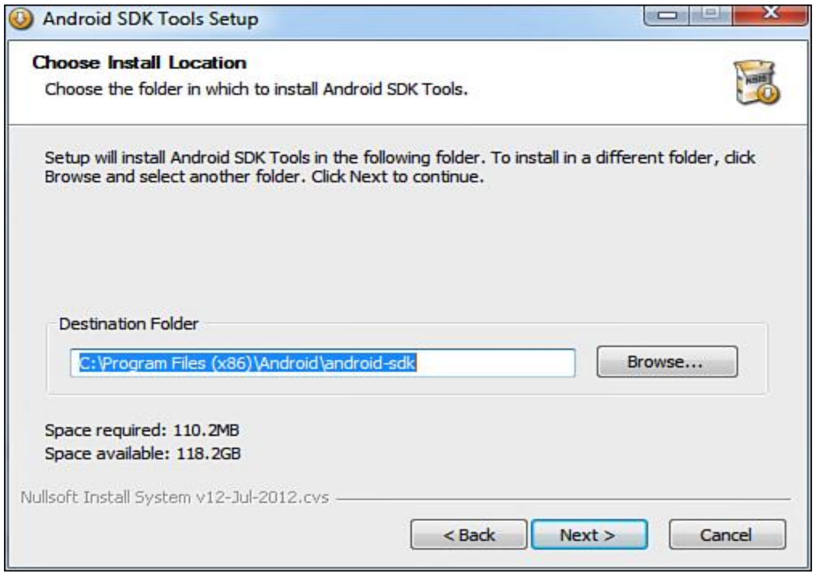
Nullsoft Install System v12-Jul-2012.cvs
< Back
Next >
Cancel

Figure Dialog box informing you that the JDK is already installed on the computer

Select the Next button. You get a dialog box asking you to choose the users for which Android SDK is being installed. The following two options are displayed in the dialog box:

- Install for anyone using this computer
- Install just for me

Select the Install for anyone using this computer option and click Next. The next dialog prompts you for the location to install the Android SDK Tools, as shown in below Figure . The dialog also displays the default directory location for installing Android SDK Tools as C:\Program Files (x86)\Android\android-sdk, which you can change by selecting the Browse button. Keep the default directory for installing Android SDK Tools unchanged; then select the Next button to continue.


Android SDK Tools Setup

Choose Install Location

Choose the folder in which to install Android SDK Tools.

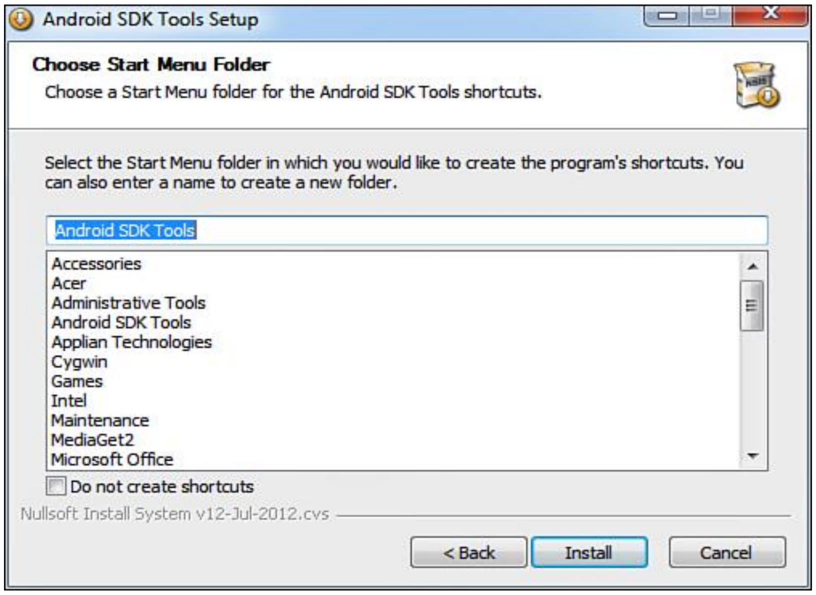
Setup will install Android SDK Tools in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Program Files (x86)\Android\android-sdk

Space required: 110.2MB
Space available: 118.2GB

Nullsoft Install System v12-Jul-2012.cvs

The next dialog box asks you to specify the Start Menu folder where you want the program's shortcuts to appear, as shown.


Android SDK Tools Setup

Choose Start Menu Folder

Choose a Start Menu folder for the Android SDK Tools shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Android SDK Tools

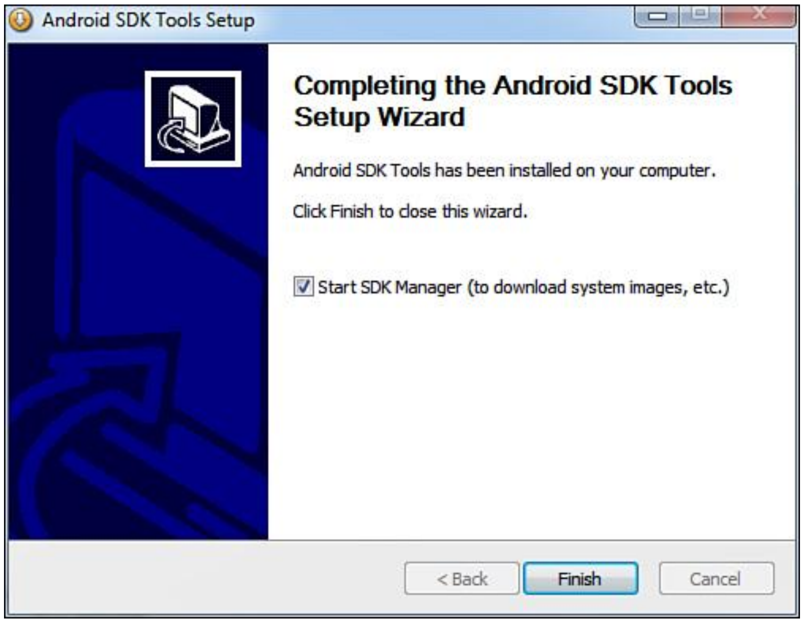
Accessories
Acer
Administrative Tools
Android SDK Tools
Applian Technologies
Cygwin
Games
Intel
Maintenance
MediaGet2
Microsoft Office

☐ Do not create shortcuts

Nullsoft Install System v12-Jul-2012.cvs

Figure: Dialog box to select the Start menu shortcut folder

A default folder name appears called Android SDK Tools. If you do not want to make a Start Menu folder, select the Do not create shortcuts check box. Let's create the Start Menu folder by keeping the default folder name and selecting the Install button to begin the installation of the Android SDK Tools. After all the files have been downloaded and installed on the computer, select the Next button. The next dialog box tells you that the Android SDK Tools Setup Wizard is complete and the Android SDK Tools have successfully installed on the computer. Select Finish to exit the wizard, as shown in Figure


Android SDK Tools Setup

Completing the Android SDK Tools Setup Wizard

Android SDK Tools has been installed on your computer.

Click Finish to close this wizard.

☑ Start SDK Manager (to download system images, etc.)

Figure: Successful installation of the Android SDK Tools dialog box

Note that the check box Start SDK Manager (to download system images) is checked by default. It means that after the Finish button is clicked, the Android SDK Manager, one of the tools in the Android SDK Tools package, will be launched. Android SDK is installed in two phases. The first phase is the installation of the SDK, which installs the Android SDK Tools, and the second phase is installation of the Android platforms and other components.

An Android application is a combination of several small components that include Java files, XML resource and layout files, manifest files, and much more. It would be very time-consuming to create all these components manually. So, you can use the following applications to help you:

• **Eclipse IDE**—An IDE that makes the task of creating Java applications easy. It provides a complete platform for developing Java applications with compiling, debugging, and testing support.

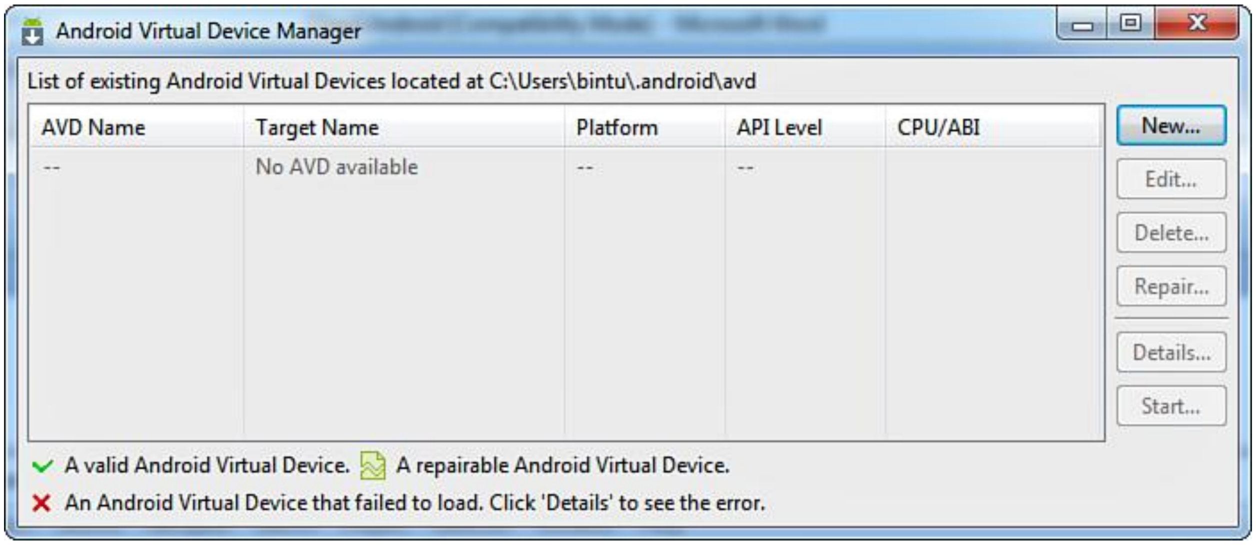
• **Android Development Tools (ADT) plug-in**—A plug-in that's added to the Eclipse IDE and automatically creates the necessary Android files so you can concentrate on the process of application development.

Before you begin the installation of Eclipse IDE, first set the path of the JDK that you installed, as it will be required for compiling the applications. To set the JDK path on Windows, right-click on My Computer and select the Properties option. From the System Properties dialog box that appears, select the Advanced tab, followed by the Environment Variables button. A dialog box, Environment Variables, pops up. In the System variables section, double-click on the Path variable. Add the full path of the JDK (C:\Program Files\Java\jdk1.7.0_04\bin\java.exe) to the path variable and select OK to close the windows.

CREATING ANDROID VIRTUAL DEVICES

An Android Virtual Device (AVD) represents a device configuration. There are many Android devices, each with different configuration. To test whether the Android application is compatible with a set of Android devices, you can create AVDs that represent their configuration. For example, you can create an AVD that represents an Android device running version 4.1 of the SDK with a 64MB SD card. After creating AVDs, you point the emulator to each one when developing and testing the application. AVDs are the easiest way of testing the application with various configurations.

To create AVDs in Eclipse, select the Window, AVD Manager option. An Android Virtual Device Manager dialog opens, as shown in Figure. The dialog box displays a list of existing AVDs, letting you create new AVDs and manage existing AVDs. Because you haven't yet defined an AVD, an empty list is displayed.



List of existing Android Virtual Devices located at C:\Users\bintu\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
--	No AVD available	--	--	

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure: The AVD Manager dialog

Select the New button to define a new AVD. A Create new Android Virtual Device (AVD)dialog box, appears The fields are as follows:

• Name—Used to specify the name of the AVD.
• Target—Used to specify the target API level. Our application will be tested against the specified API level.
• CPU/ABI—Determines the processor that we want to emulate on our device.
• SD Card—Used for extending the storage capacity of the device. Large data files such as audio and video for which the built-in flash memory is insufficient are stored on the SD card.
• Snapshot—Enable this option to avoid booting of the emulator and start it from the last saved snapshot. Hence, this option is used to start the Android emulator quickly.
• Skin—Used for setting the screen size. Each built-in skin represents a specific screen size. You can try multiple skins to see if your application works across different devices.
• Hardware—Used to set properties representing various optional hardware that may be present in the target device.

In the AVD, set the Name of the AVD to demoAVD, choose Android 4.1—API Level 16 for the Target, set SD Card to 64 MiB, and leave the Default (WVGA800) for Skin.

In the Hardware section, three properties are already set for you depending on the selected target. The Abstracted LCD density is set to 240; the Max VM application heap size is set to 48, and the Device RAM size is set to 512.

You can select these properties and edit their values, delete them, and add new properties by selecting the New button. New properties can include Abstracted LCD density, DPad support, Accelerometer, Maximum horizontal camera pixels, Cache partition size, Audio playback support, and Track-ball support, among others.

Note

Note

The larger the allocated SD Card space, the longer it takes to create the AVD. Unless it is really required, keep the SD Card space as low as possible. I would recommend keeping this small, like 64MiB.

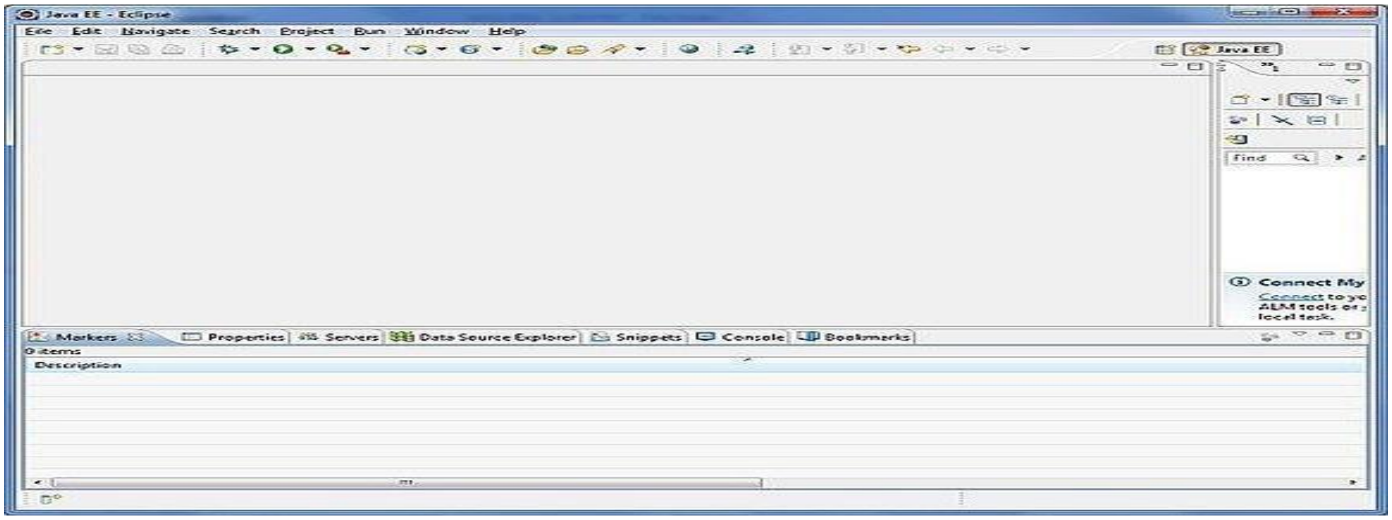
Finally, select the Create AVD button (see Figure 1.20—right) to see how to create the virtual device called demoAVD.

You now have everything ready for developing Android applications—the Android SDK, the Android platform, the Eclipse IDE, the ADT plug-in, and an AVD for testing Android applications. You can now create your first Android application.

Creating the First Android Project

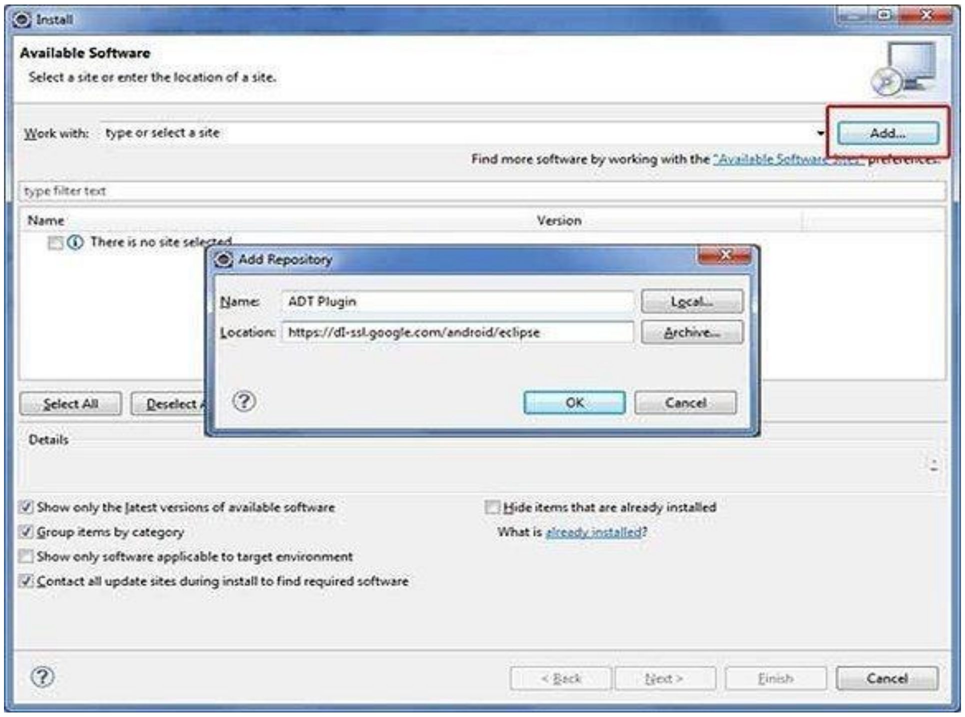
Now let’s go over how to set up your first project so all you’ll have left to do is write! you'll start a new Android Studio project and get to know the project workspace, including the project editor that you'll use to code the app.

Step 1: Setup Eclipse IDE:Install the latest version of Eclipse. After successful installation, it should display a


Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
Java EE
Find
Connect My
Markers Properties Servers Data Source Explorer Snippets Console Bookmarks
0 items
Description

Step 2: Setup Android Development Tools (ADT) Plugin

Here you will learn to install the Android Development Tool plugin for Eclipse. To do this, you have to click on **Help > Software Updates > Install New Software**. This will display the following dialogue box.


Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

There is no site selected

Add Repository

Name: ADT Plugin

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

☑ Show only the latest versions of available software

☑ Group items by category

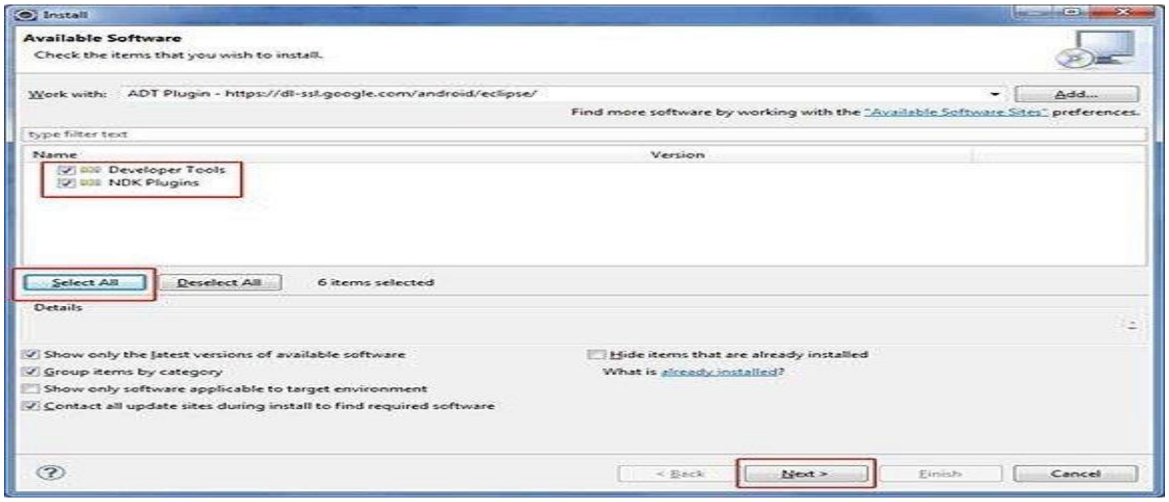
☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

☐ Hide items that are already installed

What is already installed?

Just click on the Add button as shown in the picture and add **https://dl-ssl.google.com/android/eclipse/** as the location. When you press OK, Eclipse will start to search for the required plug-in and finally it will list the found plug-ins.


Install

Available Software

Check the items that you wish to install.

Work with: ADT Plugin - https://dl-ssl.google.com/android/eclipse/

Find more software by working with the "Available Software Sites" preferences.

☑ Developer Tools

☑ NDK Plugins

6 items selected

☑ Show only the latest versions of available software

☑ Group items by category

☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

☐ Hide items that are already installed

What is already installed?

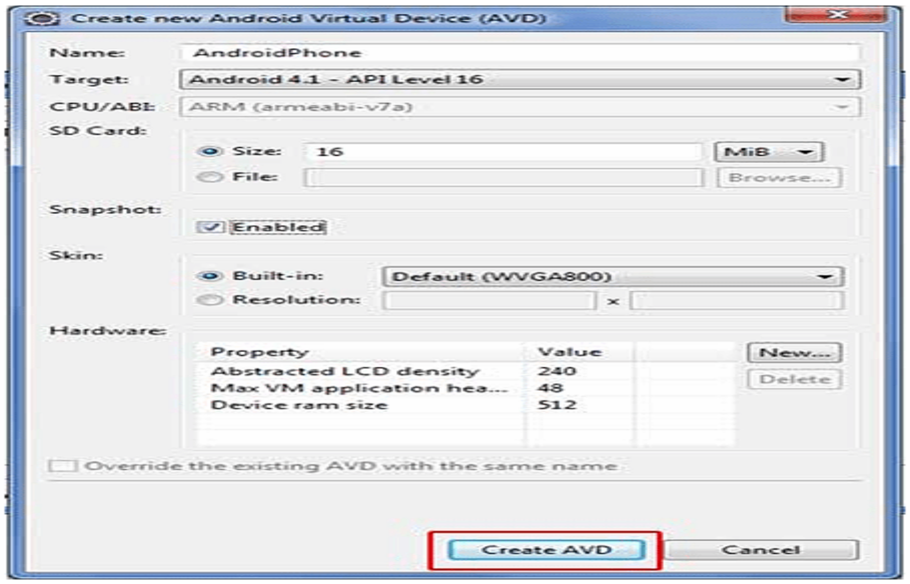
Step 3: Configuring the ADT plugin

After the installing ADT plugin, now tell the eclipse IDE for your android SDK location. To do so:

1. Select the **Window menu > preferences**
2. Now select the android from the left panel. Here you may see a dialog box asking if you want to send the statistics to the google. Click **proceed**.
3. Click on the browse button and locate your SDK directory e.g. my SDK location is C:\Program Files\Android\android-sdk .
4. Click the apply button then OK.

Step 4: Create Android Virtual Device:

The last step is to create Android Virtual Device, which you will use to test your Android applications. To do this, open Eclipse and Launch Android AVD Manager from options **Window > AVD Manager** and click on **New** which will create a successful Android Virtual Device. Use the screenshot below to enter the correct values.



Create new Android Virtual Device (AVD)

Name: AndroidPhone

Target: Android 4.1 - API Level 16

CPU/ABI: ARM (armeabi-v7a)

SD Card:

- Size: 16 MiB
- File: Browse...

Snapshot: [✓ Enabled]

Skin:

- Built-in: Default (WVGA800)
- Resolution: x

Hardware:

Property	Value
Abstracted LCD density	240
Max VM application hea...	48
Device ram size	512

USING THE TEXTVIEW CONTROL

In android **ui** or **input** controls are the interactive or View components which are used to design the user interface of an application. In android we have a wide variety of UI or input controls available, those are TextView, EditText, Buttons, Checkbox, Progressbar, Spinners, etc.

In android, **TextView** is a user interface control which is used to set and display the text to the user based on our requirements. The TextView control will act as like label control and it won't allow users to edit the text. A good example of TextView control usage would be to display textual labels for other controls, like "Enter a Date:", "Enter a Name:" or "Enter a Password:".

In android, we can create a TextView control in two ways either in XML layout file or create it in Activity file programmatically.

Specific attributes of TextView controls you will want to be aware of:

- Give the TextView control a unique name using the id property.
- Set the text displayed within the TextView control using the text property; programmatically set with the setText() method.
- Set the layout height and layout width properties of the control as appropriate.
- Set any other attributes you desire to adjust the control's appearance. For example, adjust the text size, color, font or other style settings.
- By default, text contents of a TextView control are left-aligned. However, you can position the text using the gravity attribute. This setting positions your text relative to the control's overall width and height and only really makes sense to use if there is whitespace within the TextView control.
- In XML, this property would appear within your TextView control as:

android:gravity="center"

- By default, the background of a TextView control is transparent. That is, whatever is behind the control is shown. However, you can set the background of a control explicitly, to a color resource, or a drawable (picture). In XML, this property would appear within your TextView control as:

  android:background="#0000ff"
- By default, any text contents within a TextView control is displayed as plain text. However, by setting one simple attribute called autoLink, all you can enable automatic detection of web, email, phone and address information within the text. In XML, this property would appear within your TextView control as:

  android:autoLink="all"
- You can control the color of the text within the TextView control by using the textColor attribute. This attribute can be set to a color resource, or a specific color by hex value. In XML, this property would appear within your TextView control as:

  android:textColor="#ff0000"
- You can control the style of the text (bold, italic) and font family (sans, serif, monospace) within the TextView control by using the textStyle and typeface attributes. In XML, these properties would appear within your TextView control as:

  android:textStyle="bold"
  android:typeface="monospace"

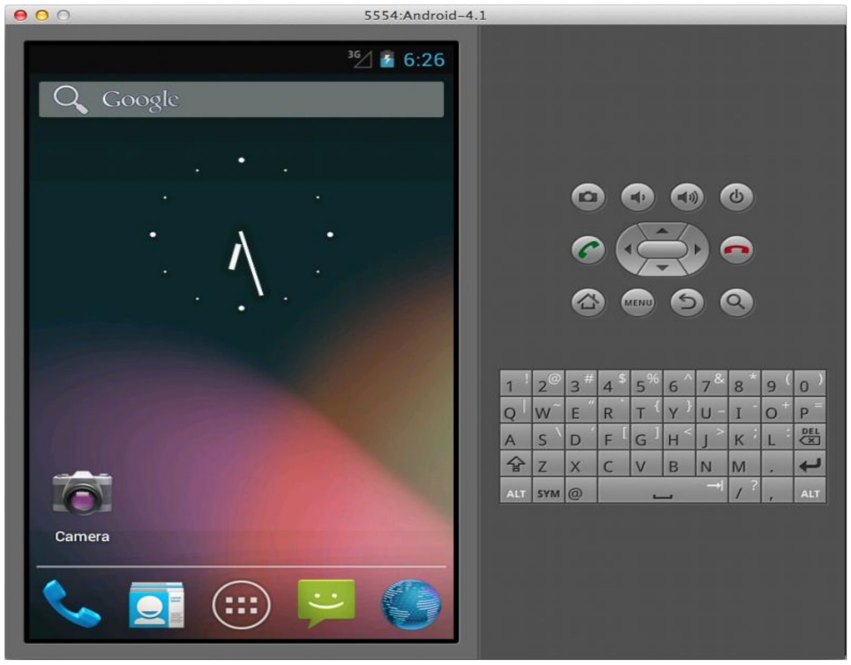
Example:

```
<TextView
   android:id="@+id/message"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   tools:context=".HelloWorldAppActivity"
   android:typeface="serif"
   android:textColor="#0F0"
   android:textSize="25dp"
   android:textStyle="italic"
   android:gravity="center_horizontal" />
```

This code makes the text of the TextView control appear in serif font, green color, 25dp size, italic, and at the horizontal center of the container

USING THE ANDROID EMULATOR

The Android emulator is used for testing and debugging applications before they are loaded onto a real handset. Android emulator is typically used for deploying apps that are developed in your IDE without actually installing it in a device. Android emulators such as Bluestacks can run android apps where in which emulators like AVD and genymotion are used to emulate an entire operating system. The Android emulator is integrated into Eclipse through the ADT plug-in.



Limitations of the Android Emulator

The Android emulator is useful to test Android applications for compatibility with devices of different configurations. But still, it is a piece of software and not an actual device and has several limitations:

- Emulators no doubt help in knowing how an application may operate within a given environment, but they still don’t provide the actual environment to an application. For example, an actual device has memory, CPU, or other physical limitations that an emulator doesn’t reveal.
- Emulators just simulate certain handset behavior. Features such as GPS, sensors, battery, power settings, and network connectivity can be easily simulated on a computer.
- SMS messages are also simulated and do not use a real network.
- Phone calls cannot be placed or received but are simulated.
- No support for device-attached headphones is available.
- Peripherals such as camera/video capture are not fully functional.
- No USB or Bluetooth support is available.

The emulator provides some facilities too. You can use the mouse and keyboard to interact with the emulator when it is running. For example, you can use your computer mouse to click, scroll, and drag items on the emulator. You can also use it to simulate finger touch on the soft keyboard or a physical emulator keyboard. You can use your computer keyboard to input text into UI controls and to execute specific emulator commands. Some of the most commonly used commands are

- Back [ESC button]
- Call [F3]
- End [F4]
- Volume Up [KEYPAD_PLUS, Ctrl-5]
- Volume down [KEYPAD_MINUS, Ctrl-F6]
- Switching orientations [KEYPAD_7, Ctrl-F11/KEYPAD_9, Ctrl-F12]

You can also interact with an emulator from within the DDMS tool. Eclipse IDE provides three perspectives to work with: *Java perspective, Debug perspective,* and *DDMS perspective.* The Java perspective is the default and the one with which you have been working up to now. You can switch between perspectives by choosing the appropriate icon in the top-right corner of the Eclipse environment. The three perspectives are as follows:

• **The Java perspective**—It's the default perspective in Eclipse where you spend most of the time. It shows the panes where you can write code and navigate around the project.

• **The Debug perspective**—Enables application debugging. You can set breakpoints; step through the code; view LogCat logging information, threads, and so on.

• **The Dalvik Debug Monitor Service (DDMS) perspective**—Enables you to monitor and manipulate emulator and device status. It also provides screen capture and simulates incoming phone calls, SMS sending, and GPS coordinates. To manage content in the device or emulator, you can use the ADB (Android Debug Bridge).

THE ANDROID DEBUG BRIDGE(ADB)

The Android-Debug-Bridge (abbreviated as adb) is a software-interface for the android system, which can be used to connect an android device with a computer using an USB cable or a wireless connection. It can be used to execute commands on the phone or transfer data between the device and the computer.[1]

The tool is part of the Android SDK (Android Software Development Kit) and is located in the subdirectory platform-tools. In previous versions of the SDK it was located in the subdirectory tools.

The Android Debug Bridge is a software interface between the device and the local computer, which allows the direct communication of both components. This includes the possibility to transfer files from one component to the other one, as well as executing commands from the computer on the connected device. The ADB can be used through a command line windows, terminal/shell in Linux-based systems, a command line (cmd) for Windows. The main advantage is to execute commands on the phone directly out of the computer, without any direct user interaction to the phone, which makes especially debugging a lot easier.

It is a client-server program that includes three components:

- **A client**, which sends commands. The client runs on your development machine. You can invoke a client from a command-line terminal by issuing an adb command.
- **A daemon (adbd)**, which runs commands on a device. The daemon runs as a background process on each device.
- **A server**, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

When you start an adb client, the client first checks whether there is an adb server process already running. If there isn't, it starts the server process. When the server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients—all adb clients use port 5037 to communicate with the adb server.

The server then sets up connections to all running devices. It locates emulators by scanning odd-numbered ports in the range 5555 to 5585, the range used by the first 16 emulators. Where the server finds an adb daemon (adbd), it sets up a connection to that port. Note that each emulator uses a pair of sequential ports — an even-numbered port for console connections and an odd-numbered port for adb connections.

Once the server has set up connections to all devices, you can use adb commands to access those devices. Because the server manages connections to devices and handles commands from multiple adb clients, you can control any device from any client

Launching Android Applications on a Handset

To load an application onto a real handset, you need to plug a handset into your computer, using the USB data cable. You first confirm whether the configurations for debugging your application are correct and then launch the application as described here:

1. In Eclipse, choose the Run, Debug Configurations option.

2. Select the configuration HelloWorldApp_configuration, which you created for the HelloWorldApp application.

3. Select the Target tab, set the Deployment Target Selection Mode to Manual. The Manual option allows us to choose the device or AVD to connect to when using this launch configuration.

4. Apply the changes to the configuration file by clicking the Apply button.

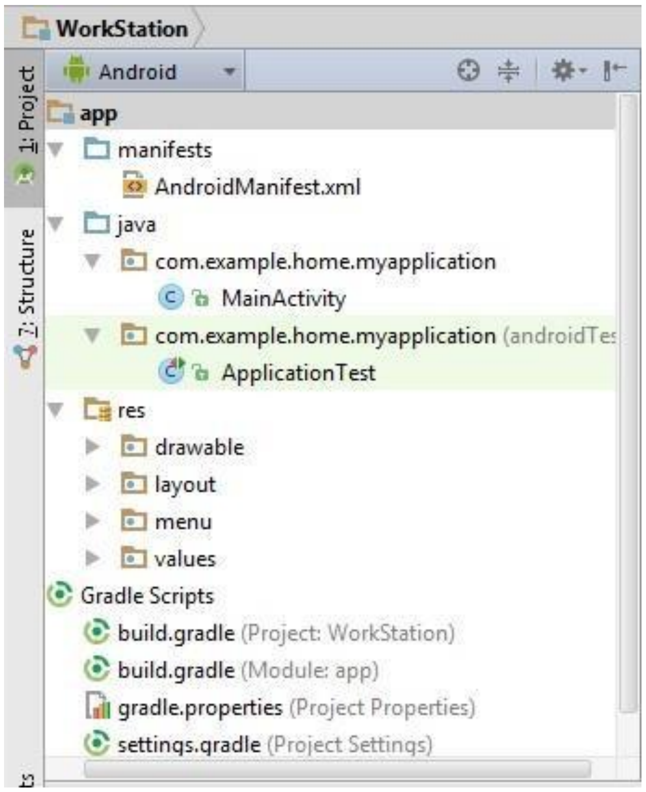
5. Plug an Android device into your computer, using a USB cable.

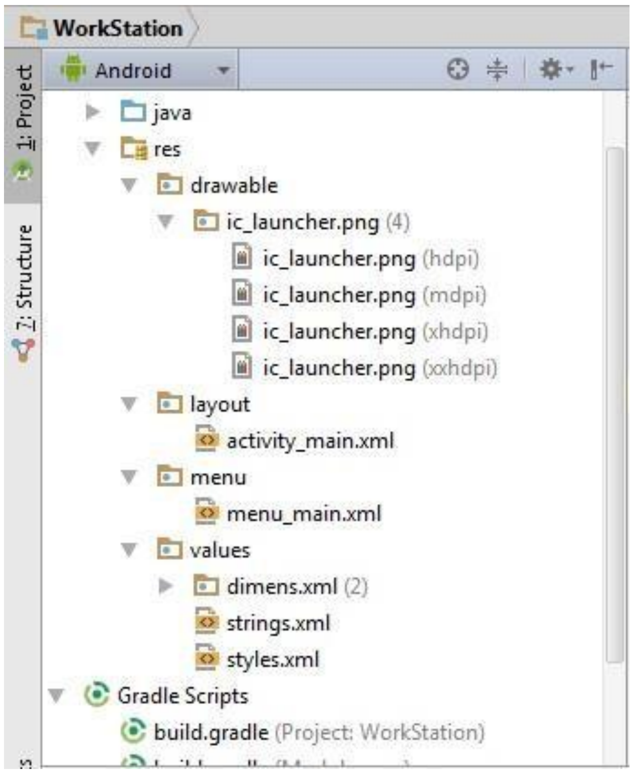
6. Select Run, Debug in Eclipse or press the F11 key. A dialog box appears, showing all available configurations for running and debugging your application. The physical device(s) connected to the computer are also listed. Double-click the running Android device. Eclipse now installs the Android application on the handset, attaches a debugger, and runs the application.

Android Studio Project Structure

Android studio shows a structured view of the project files and also provides a quick access to source files also. Android studio groups all the required source files, build files, resource files of all the modules at the top of project view.

Let`s first understand the anatomy of the Android Studio:


WorkStation
Android
1: Project
7: Structure
app
manifests
AndroidManifest.xml
java
com.example.home.myapplication
MainActivity
com.example.home.myapplication (androidTes
ApplicationTest
res
drawable
layout
menu
values
Gradle Scripts
build.gradle (Project: WorkStation)
build.gradle (Module: app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)



1) **.idea:** This folder contains the directories (subfolders) for IntelliJ IDEA settings.

2) **app:** It contains the actual application code(source files, resource file and manifest files). It further contains the following sub-directories :
a) **bulid:** This folder has sub-folders for the build-variants. The app/build/output/apk directory contains packages named app-<flavor>-<built-type>.apk. So different variant of the single app resides here.
b) **libs:** As the name suggests, this folder has all the .jar files and the library files.

3) **src:** Here you will see two sub-directories androidTest and main.
In **androidTest** the application code for testing purpose is created by android automatically.
This helps in building testing packages without modifying the building files and the application code.

main contains the all the source .java files including the stub mainactivity.java.

The **res** directory is where you will find the resources like drawable files, menu, values (style files, string values, dimension values).

<u>Sub-directories of res folder</u>

- anim/: For XML files that are compiled into animation objects.
- color/: For XML files that describe colors.

- drawable/:For image files (PNG, JPEG, or GIF), 9-Patch image files, and XML files that describe Drawable shapes or Drawable objects that contain multiple states (normal, pressed, or focused).
- mipmap/: For app launcher icons. This folder contains additional sub-folders for different screen resolutions. Once an image is imported into this folder, Studio automatically resizes the image into different resolutions and places them in the appropirate folders. This behavior allows launcher apps to pick the best resolution icon for your app to display on the home screen. To see how to import an image into the mipmap folder, please read about using 'Image asset'.
- layout/: XML files that are compiled into screen layouts (or parts of a parent layout).
- menu/: For XML files that define application menus.
- values/: For XML files that define constants that can be accessed in other XML and Java files.

R.java

R.java is an uneditable file auto-generated by Studio, whenever we modify the XML content. This file links the XML values to Java files. Whenever we need to use XML values in a Java file, we call these values using the R class.

The following images show a sample XML file, where a string variable named 'title' is assigned a value of 'Internshala - Tic Tac Toe'. This variable is then accessed in a java file using the R class.
4) **Gradle scripts:** With Android studio, Google switched to the new advanced building system, Gradle. It is a JVM based build system. If you want to make the package building task automatically, then you can write your own script in java or groovy and distribute it. Gradle allows to create different variants apk files for the same application project. We will learn about Gradles in the later chapters.
5) **External Libraries:** This is the place where all the referenced libraries and the information about the targeted platform SDK are stored.

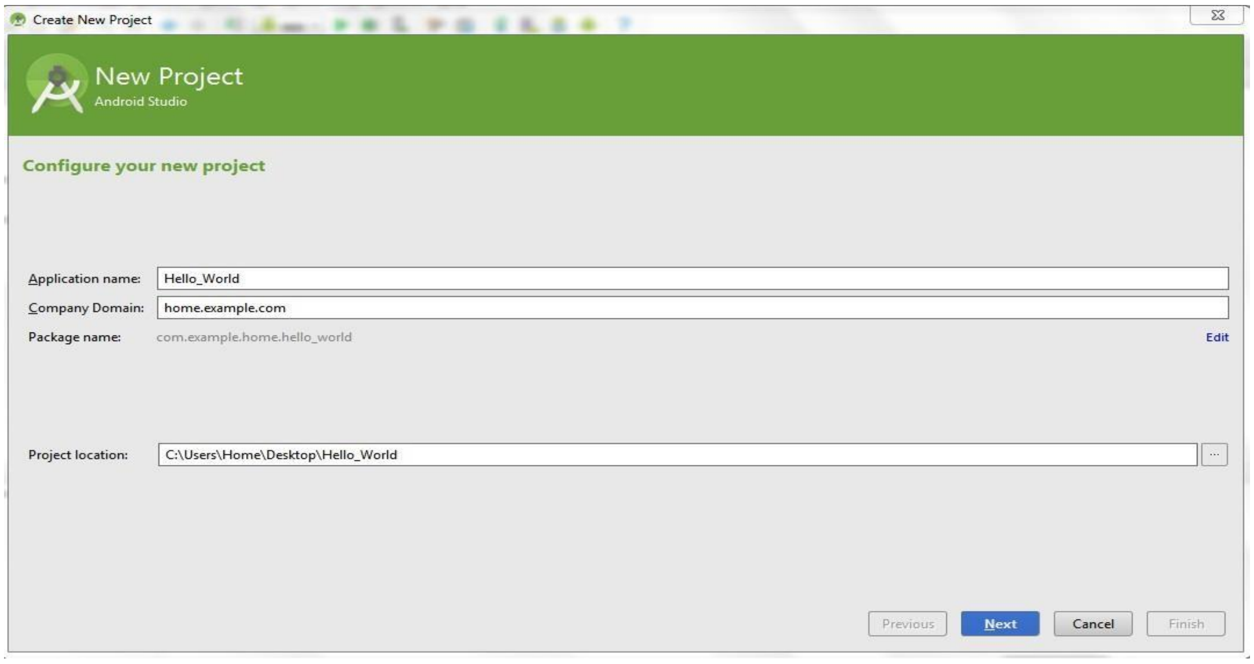
What is Build system and Gradle

The build system is responsible to build, test an android system and also prepare the deployable files for the specified platform. In simple words, build system generates the .apk files for the application project.

With Android Studio, the advanced build system allows the developers to configure the build systems manually, create different variant APK files from single project (without modifying the execution code) and share the code and resources from other modules. Before Android Studio, Eclipse was the IDE used for android development. Eclipse kept all the .java files (in src directory) and resource files (in res directory) in the same directory. That allows the build system to group all the files into an intermediate code and finally generates .apk file.
Android Studio uses Gradle as its build system. One intriguing feature that Gradle offers is that it allows you to write your own script to automate the task of building an app. As Gradle is plug-in based system, if you have your own programming language then you can write the plug-in in the script using java or groovy and share it with other developers too. Isn`t that cool?

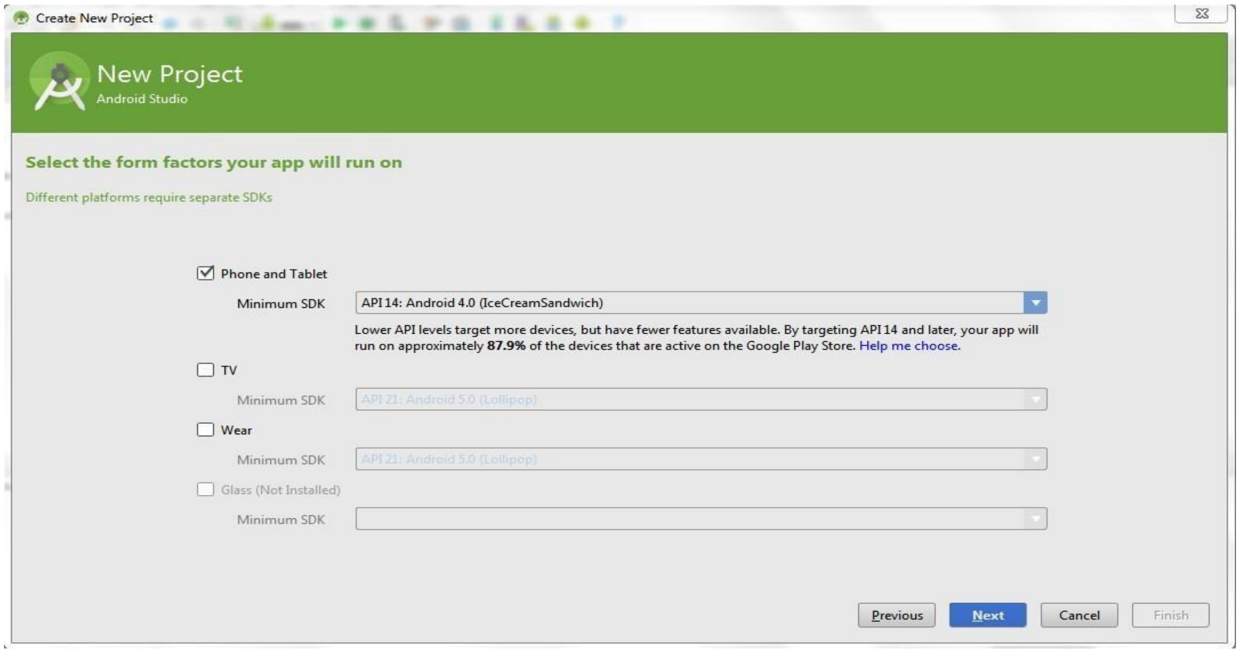
<u>'Hello World' APP</u>

The first you will be creating will be "Hello World"! We will see how to start a project in Android Studio. First of all, launch Android Studio. Goto File-> New Project. Give the name of the application. Here we name it "Hello_World". Hit next.



Application name: Hello_World
Company Domain: home.example.com
Package name: com.example.home.hello_world Edit
Project location: C:\Users\Home\Desktop\Hello_World

Select the category of the target devices and the desire minimum target platform of android:

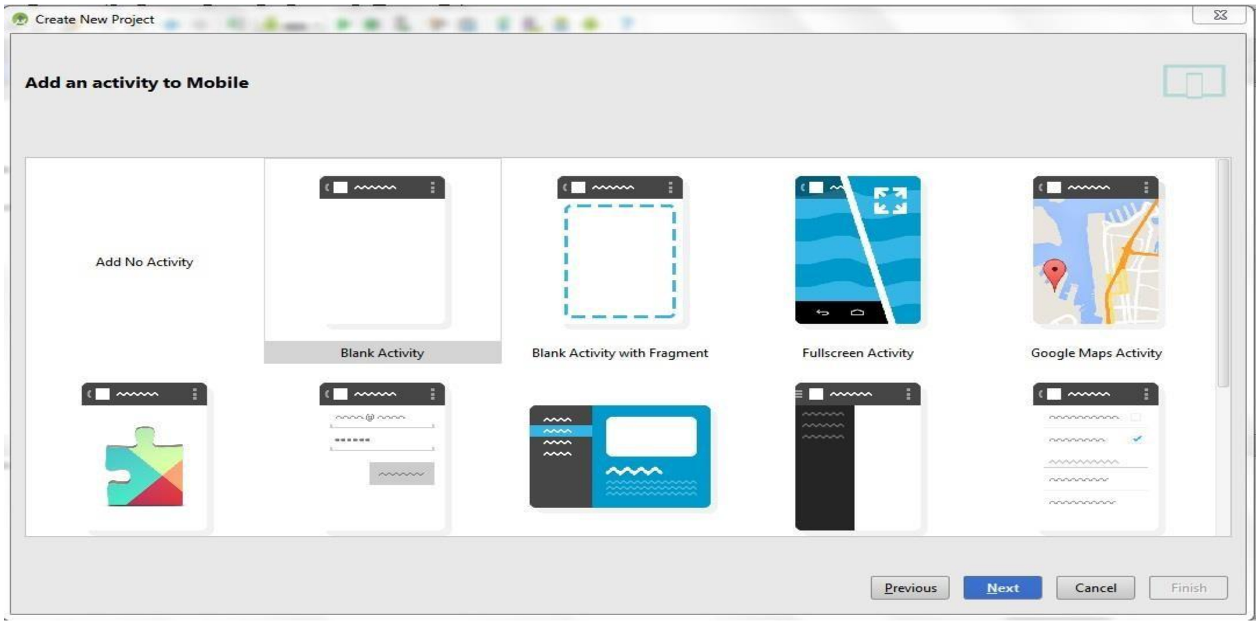

Create New Project
New Project
Android Studio
Select the form factors your app will run on
Different platforms require separate SDKs
Previous
Next
Cancel
Finish

☑ Phone and Tablet
Minimum SDK: API 14: Android 4.0 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available. By targeting API 14 and later, your app will run on approximately **87.9%** of the devices that are active on the Google Play Store. Help me choose.
☐ TV
Minimum SDK: API 21: Android 5.0 (Lollipop)
☐ Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)
☐ Glass (Not Installed)
Minimum SDK:

Now, select the main activity style depending upon the needs. Here for "Hello_world", select blank activity.

Name the main_activity files, which are the first activity to be displayed in the application. Click ‘finish’.
Run the Application.

Building and Running an App



First you need to build your application project before running it on the device. Click on the “build” icon from the toolbar and then select “Make project”.

To Run an application:

Select “Run” from the toolbar and then “run app”.

You can run your app on the emulator or on a real-device from Android Studio. This is done using the debug version of the app. Run configuration defines which module will run, which activity is start, and about all the AVD settings. If you run the android application for the first time, android studio will automatically create a run configuration and choose AVD to run it. Though you can create or modify the run configuration.

Run an app on Emulator:

When you run an app on the emulator, first make sure about the AVD (Android Virtual Device). You can choose from the available AVD or create a new AVD.
Go to Tools >Android > AVD Manager. Click on ‘create virtual device’.
Choose Hardware category and configuration, Click next. Select the desired system version and create the AVD.
To run application on the desired AVD, click on the launch button.

Run an App on the real-device :

First, you have to make sure that the real-device in which you desire to run your app, is debuggable.
1) Check if the android:debuggable attribute is set to true in the build.gradle file. If not then, you cannot debug it. Make it true in case you want to run it on real-device.
2) Please enable the USB Debugging, in the device. In android 4.2 or above, the developer option can be enabled by going to Settings > About Phone and tapping Build Number 7 times. Now you can see the Developer Option in the previous screen.
Now, when you have your AVD set up or the real-devices ready, go to Run > Run (OR Run > Debug). If you don’t see your device in the AVD window, then you need to download appropriate device drivers for your device from the internet.