CHAPTER – 1
INTRODUCTION TO MOBILE OSES

Introduction to Mobile OSes:

An **operating system** (**OS**) is system software that manages computer hardware and software resources and provides common services for computer programs. The operating system is a component of the system software in a computer system. Application programs usually require an operating system to function.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or be interrupted by it. Operating systems are found on many devices that contain a computer from cellular phones and video game consoles to web servers and supercomputers.

Different Types Of Mobile Operating Systems Are:

Android:

- It is an open source mobile operating system.
- It was initially developed by Android Inc. based on Linux Kernel.
- It was bought by Google in 2005 and was unveiled in 2007.
- It was designed for touchscreen mobile devices.
- Android P is the upcoming android version, was first announced by Google on March 7, 2018 and the first developed preview was released on the same day. The second preview, considered beta quality was released on May 8, 2018.
- Android applications can be downloaded from Google play store.

iOS:

- It is a mobile operating system developed for iPhone and extended to iPod touch and iPod.
- It was initially released on June 2007.
- It was programmed in C, C++ and objective C.
- It was derived from Mac OS X.
- It is available in 34 different languages worldwide.
- The kernel type is hybrid kernel (combination of microkernel and monolithic kernel) architecture.

Ubuntu Touch:

- It is a mobile version of the Ubuntu operating system.
- It was developed by Canonical Limited and the Ubuntu community.
- It was designed primarily for touchscreen mobile devices.
- This project was started in 2011.

- On 5th April 2017, Mark shuttleworth announced that the project would terminated due to lack of market interest.
- The project was then picked up by the UBports.

Blackberry:

- It is a mobile operating system.
- It was developed by blackberry limited for blackberry phones.
- It was written in C++.
- Its kernel used Java Virtual Machine.
- The BlackBerry platform is perhaps best known for its native support for corporate email, through MIDP 1.0 and, more recently, a subset of MIDP 2.0, which allows complete wireless activation and synchronization with Microsoft Exchange, Lotus Domino, or Novell GroupWise email, calendar, tasks, notes, and contacts, when used with BlackBerry Enterprise Server.

Tizen:

- It is Linux kernel based operating system.
- It works on smartphones, tablets, smart televisions, personal computers, smart cameras, wearable computing, Blu-ray players, printers and smart home appliances.
- Project within Linux foundation is governed by Technical Steering Group (TSG).
- It was written in HTML5, C, C++.
- Its latest version was released on May 20, 2017.
- Its kernel type is monolithic kernel.

Firefox:

- It is an open source operating system.
- It was designed by Mozilla and external contributors.
- It was first released on 2013.
- It was written in HTML5, CSS, JavaScript, C++.
- Its latest version was released on April 29, 2015. On September 2016 the development was ended.

Symbian:

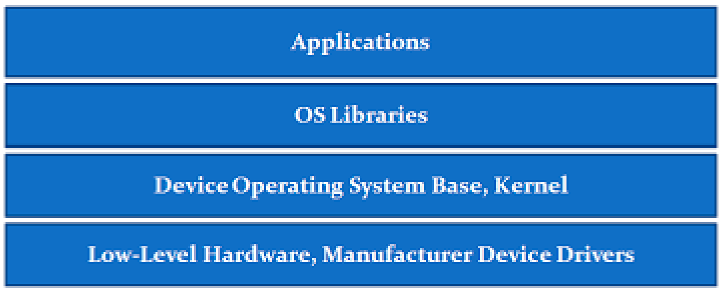
- It is a mobile operating system and computing platform designed for smartphones.
- It was originally developed as closed source operating system for PDAs (Personal Device Assistant) in 1998.
- It was developed by Symbian Limited.
- It was written in C++.
- At present this project is discontinued.
- It was available in multi-lingual.

Windows Phone:

- It is also include in the family of mobile operating system but is a closed source.
- It was developed by Microsoft for smartphones as the replacement to successor to windows phone and Zune.
- It was first launched on October 2010 with windows phone 7.
- It was written in C, C++.
- Its latest release is 8.1 update on June 2, 2015.
- It was available in 130 different languages.
- Its kernel type is hybrid (Monolithic in windows phone 7).

BUILD AND STRUCTURE OF MOBILE OSES:

A mobile OS is a software platform on top of which other programs called application programs, can run on mobile devices such as PDA, cellular phones, smartphone and etc.


Applications
OS Libraries
Device Operating System Base, Kernel
Low-Level Hardware, Manufacturer Device Drivers

Low Level Hardware:

Mobile Processors:

Processors use RISC (Reduced Instruction Set Computer) architecture. There are mainly two types of processors and they are:

1. **ARM Processors:**

It is a 32-bit RISC processor architecture and consume low power. Examples: Qualcomm Snapdragon, Marvell's XScale, Texas Instruments OMAP series, etc.

2. **MIPS:**

Microprocessor without Interlocked Pipeline Stages. They are mainly used in embedded systems. The early **MIPS** architectures were 32-bit, with 64-bit versions added later. Examples: TiVo, Playstation1 & 2, etc.

Mobile Memory:

The better the memory management offered by the OS, the wider the options available to applications developers. Mobile devices have two types of memories:

1. **ROM (Read Only Memory):**

For operating system and preinstalled programs. There are different variations on the classic ROM chips which were manufacturer produced and could not change. The most common are:

a. *Programmable Read-Only Memory (PROM):*

This type of ROM can be re-programmed by using a special device called a PROM programmer. Generally, a PROM can only be changed/updated once.

b. *Erasable Programmable Read-Only Memory (EPROM):*

This type of ROM can have its contents erased by ultraviolet light and then reprogrammed by an RPROM programmer. This procedure can be carried out many times; however, the constant erasing and rewriting will eventually render the chip useless.

c. *Electrically Erasable Programmable Read-Only Memory (EEPROM):*

This type of ROM works in a similar way to Flash memory in that it can its contents can be 'flashed' for erasure ad then written to without having to remove the chip from its environment. EEPROMs are used to store a computer system's BIOS, and can be updated without returning the unit to the factory. In many cases, BIOS updates can be carried out by computer users wishing a BIOS update.

2. <u>RAM (Random Access Memory):</u>

For user information. There are various types of RAMs and they are:

a. *Dynamic RAM:*

Loses its stored information in a very short time (for milli sec.) even when power supply is on. D-RAM's are cheaper & lower.

b. *Static RAM*:

Uses a completely different technology. S-RAM retains stored information only as long as the power supply is on. Static RAM's are costlier and consume more power. They have higher speed than D-RAMs. They store information in Hip-Hope.

c. *EDO (Extended Data Output) RAM*:

In an EDO RAMs, any memory location can be accessed. Stores 256 bytes of data information into latches. The latches hold next 256 bytes of information so that in most programs, which are sequentially executed, the data are available without wait states.

d. *SDRAM (Synchronous DRAMS), SGRAMs (Synchronous Graphic RAMs):*

These RAM chips use the same clock rate as CPU uses. They transfer data when the CPU expects them to be ready.

e. *DDR-SDRAM (Double Data Rate – SDRAM):*

This RAM transfers data on both edges of the clock. Therefore the transfer rate of the data becomes doubles.

Kernel:

Responsible for services such as Security, Memory Management, Process Management, etc. It include the following components such as I/O Components, File Systems, Networking Components, etc.

Libraries:

- Media Libraries
- 3D Libraries, etc.

Application Framework:

These Libraries are exposed to developers through the application framework.

Introduction to Development Environment (Native v/s HTML5):

Native Apps:

Any app installed directly onto a device, usually from its associated app store, can be called a "native" app (although a native app can have HTML5 elements). Native apps are platform dependent i.e. an app made for iPhone can work only on an iPhone, not an Android device. Native apps have distinct advantages and disadvantages, such as:

Pros of Native Apps:

1. **Better Performance:**

A native app has complete access to a phone's hardware resources. The app also interacts directly with the phone without the mediation of a web browser. This lends it significantly better performance, particularly when rendering graphics and animation.

2. **Easier Development:**

Developers can easily leverage platform SDKs to create native apps. This makes easier development and provide us access to experienced engineering talent.

3. **Better Distribution:**

Native apps can be distributed directly through relevant app stores. This can be an incredibly powerful distribution channel that not only makes app installation and updates easy, but can also help cut down on marketing costs through app stores' built-in discovery features.

4. **Better Monetization:**

Since native apps are associated with app stores, they can take advantage of the store's built-in monetization features, such as one-click payments.

Cons of Native Apps:

1. **Increased Development Time And Costs:**

Since developers must build separate apps for each platform, total development time might be significantly higher than an equivalent HTML5 app. This also increases costs since developing for different platforms demands mastery over different languages and development environments. App developers themselves tend to be more expensive than their HTML5/JavaScript counterparts.

2. **Higher Maintenance Costs:**

From the above, it follows that maintenance costs for native apps are also higher. A business might require the services of separate iOS, Android, and Windows Phone developers to keep its apps up and running, eating into profits.

3. **App Store Content Restriction:**

Every app distributed through the app store must adhere to strict content guidelines. For certain businesses (such as gambling), this may be a deal-breaker.

4. **App Store Fees:**

App store fees (up to 30 percent) can eat into our profit margins and make an already costly undertaking prohibitively expensive.

HTML5 Apps:

A web app delivered through an HTML5 enabled browser may be called an "HTML5 app." This means instead of downloading our app from the app store, a user would open her/his browser and navigate to a URL to access our app. HTML5 apps offer various advantages and disadvantages, such as:

Pros of HTML5 Apps:

1. **Platform Independent:**

HTML5 apps are completely platform independent. It does not matter whether our users are on Android, iOS, or Windows Phone, they can all access our app as long as they have a web browser.

2. **Easier Updates:**

Native apps update much like desktop applications, the user has to download each update individually. HTML5 apps, on the other hand, support centralized updates. Every time the user visits our app through her/his browser, she/he sees the latest version of our app with no additional download requirements.

3. **Faster Development:**

HTML5 apps are written in HTML, CSS, JavaScript, and server-side languages such as ASP.NET. This cuts down on development time since there are already vast libraries for these languages (such as jQuery for JavaScript) and a huge body of trained developers.

4. **Cheaper:**

Since HTML5 apps are built on relatively simple web technologies such as HTML5 and JavaScript, we will find it easier to hire affordable HTML5 developers than similarly talented native app developers.

5. **No Content Restrictions:**

Since HTML5 apps need not seek approval from a closed app store, they can carry whatever content we want.

6. **No fees:**

HTML5 apps are delivered directly through the browser. This cuts out the app store and saves us on the 30 percent app store fees.

7. **Better Data:**

Any customer data we collect through a native app is subject to the rules of the app store in question. There are no such restrictions with HTML5 apps; we can collect whatever customer data we want. In fact, the Apple App Store withholding subscriber data was one of the reasons why Financial Times switched to a web app.

8. **Distribution Through App Store:**

Until a couple of years ago, there was no easy method to distribute a HTML5 app through the app store. However, new frameworks such as Phone gap and Trigger.io have eased the distribution problem considerably bundling up HTML5 apps as native apps (which can then be distributed through the app store). Although performance remains a concern and Phone gap still doesn't support every mobile OS fully, it's a capable alternative to building separate native apps for every platform.

Cons of HTML5 Apps:

1. **Poorer Performance:**

An HTML5 app has limited access to a phone's hardware. This leads to poorer performance, especially when dealing with heavy graphics, although new platforms such as Famo.us are trying to mitigate this problem.

2. **Device Fragmentation:**

Device fragmentation within web browsers is a real thing. Different devices might render the same app differently. This means our developers will need extensive testing and fine-tuning to get the UI right, especially when working on more complex apps.

3. **Technical Limitations:**

An HTML5 does not have complete access to a device's features and events. This poses a limit on what we can do with an HTML5 app. The UI options for HTML5 apps are also limited.

4. **Limited Monetization Opportunities:**

A native app can make money through direct app sales and in-app purchases. These monetization options are not available for HTML5 apps. This can be particularly challenging for "freemium" apps.

Hybrid Apps:

Hybrid apps are web apps based on a mix of HTML5 and native development. They're usually built with HTML5 and JavaScript. The code is then wrapped in a native container which allows accessing some native features.

Pros of Hybrid apps:

1. **Fast Development Process**:

Hybrid apps do not necessitate the presence of too many developers to build the app. This is a good thing in case speed is a vital factor in our app development program.

2. **Apt For Apps With Small Release Cycles**:

In case we wish to regularly incorporate changes in our app, Hybrid app development is a very good alternative. Only if the change is huge will the users require updating the app. Updates can be put on the page which the user loads from the server. This way the users will find it easy to navigate through the app.

3. **Optimum for BYOD (Bring Your Own Device) Programs**:

Hybrid apps are ideal for BYOD programs. They work across various platforms as well as devices, this way we are freed from the tension that our employees will not be able to access the app just because it is not compatible with their operating systems or devices.

4. **Reduced Development Cost**:

It falls cheaper to build a hybrid app than a native app. You do not have to undergo the problem of getting separate apps developed for separate platforms i.e. iOS and Android. Rather you will have an app which is ready for both platforms.

5. **Minimum Requirements**:

The developers do not have to learn separate languages in order to build a hybrid app. Hybrid apps look as well as feel quite like a native app and can be created with just one language that is C# which is bolstered with frameworks such as PhoneGap wrapper or Cordova.

Cons of Hybrid apps

1. **Not Impressive UX**:

One of the biggest flaws of hybrid apps is that they fail to present a complete native-like feel. And if users do not get an impeccable first experience, their likelihood of trying the app again falls down drastically.

2. **Reduced Efficiency**:

As compared to native apps, Hybrid apps are generally slower. Because of this we are not able to have meaningful as well as smooth transitions.

3. **Presence Of Lags As Well As UX Issues**:

As hybrid apps do not present a seamless experience that is characteristic of native apps it can create serious performance issues in high-performing apps as well as games. Users will face lags and because of this, they might even leave our app.

Introduction to Android:

Android Inc. was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White. It is a platform and an operating system for mobile devices based on the Linux operating system. It allows developers design applications in a java-like language using Google-developed java libraries.

API Levels and Versions of Android:

The version history of the Android mobile operating system began with the release of the Android beta in November 2007. The first commercial version, Android 1.0, was released in September 2008. Android is under ongoing development by Google and the Open Handset Alliance (OHA), and has seen a number of updates to its base operating system since its initial release.

Code Name	Version Number	Initial Release Date	API Level
Beta	1.1	February 9, 2009	2
Cupcake	1.5	April 27, 2009	3
Donut	1.6	September 15, 2009	4
Éclair	2.0 – 2.1	October 26, 2009	5 – 7
Froyo	2.2 – 2.2.3	May 20, 2010	8
Gingerbread	2.3 – 2.3.7	December 6, 2010	9 – 10
Honeycomb	3.0 – 3.2.6	February 22, 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	July 9, 2012	16 – 18
KitKat	4.4 – 4.4.4	October 31, 2013	19 – 20
Lollipop	5.0 – 5.1.1	November 12, 2014	21 – 22
Marshmallow	6.0 – 6.0.1	October 5, 2015	23
Nougat	7.0 – 7.1.2	August 22, 2016	24 – 25
Oreo	**8.0 – 8.1**	August 21, 2017	26 – 27
Android P	9	June 6, 2018 (beta 2)	28
Android Q	10?	Future Release	

Pros of Android:

- Android is an open source, and it is free for improvement.

- It is Google's the search engine leader, which provides many application and games free for android users.
- It uses friendly and easy to access with numerous tools.
- This operating system has high compatibility that can go with nay applications.
- Unlike iOS of Apple, Android is acceptable to many vendors like HTC, Samsung, Sony, etc.
- Endless games and entertainment apps for free and paid.
- Its USB compatibility is great compare to its competitors.

Cons of Android:

- The major complaints come with its battery life which drains out easily.
- Without Wi-Fi or internet, we cannot access many features of it. We should have a Wi-Fi connected at home or at least GPRS.
- As it offers many free applications there are unavoidable advertisements, which make the users tired by making their time wasted.
- There are chances of getting malwares or viruses which can affect our device while accessing the free app like games and entertainment.

Comparison of Android with Other OSes:

- The number of apps of Android in its store (Google play, Mobango) is about 1.5 million, apple's store is about 1.4 million and windows has got about 0.4 million till date.
- Android was the first to support Wi-Fi Direct, multiple user accounts and screen mirroring support.
- The number of OEM for android OS is more than 100, for windows only about 25 while iOS has only one i.e. Apple Inc.
- Android is open source and customizable where iOS and windows are closed source.

Open Hand Alliance:

- The OHA was established on 5 November 2007, led by Google with 34 members, including mobile handset makers, application developers, some mobile carriers and chip makers.
- The Open Hand Alliance (OHA) is a business alliance of firm to develop open standard for mobile devices.
- Devote to advancing open standards for mobile devices.
- Developed technologies that will significantly lower the cost of developing and distributing mobile devices and services.
- Member firms include HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Google, Samsung Electronics, LG Electronics, T-Mobile, Sprint Corporation, Nvidia, and Wind River Systems.

Introduction to Android VM and Runtime (Delvik and ART):

A virtual machine is like a software implementation of a physical computer which works like real physical computer. It means a virtual machine can compile and run any program just like physical computer does for us. But there is a dark-side of virtual machines as it less efficient than real physical computer and provide unstable performance when multiple virtual machine exist on the same machine.

Dalvik Virtual Machine:

Dalvik is a purpose built virtual machine designed specifically for android which was developed by Dan Bornstein and his team. Strictly it was developed for mobile devices. While developing Dalvik Virtual Machine Dan Bornstein and his team realize the constraints specific to the mobile environment which is not going to change in future at least, like battery life, processing power and many more. So they optimized the dalvik virtual machine. Dalvik virtual machine uses register based architecture. With this architecture dalvik virtual machine has few advantages over JAVA virtual machine such as:

- Dalvik uses its own 16 bit instruction set than java 8 bit stack instructions, which reduce the dalvik instruction count and raised its interpreter speed.
- Dalvik use less space, which means an uncompressed .dex file is smaller in size(few bytes) than compressed java archive file(.jar file).

Role of Dalvik Virtual Machine:

The role of dalvik virtual machine is that, In java we write and compile java program using java compiler and run that bytecode on the java virtual machine. On the other side, In android we still write and compile java source file(bytecode) on java compiler, but at that point we recompile it once again using dalvik compiler to dalvik bytecode(dx tool converts java .class file into .dex format) and this dalvik bytecode is then executed on the dalvik virtual machine.

Note: Dalvik team have added Just In Time (JIT) compiler to the Dalvik Virtual Machine. The JIT is a software component which takes application code, analyzes it, and actively translates it into a form that runs faster, doing so while the application continues to run.

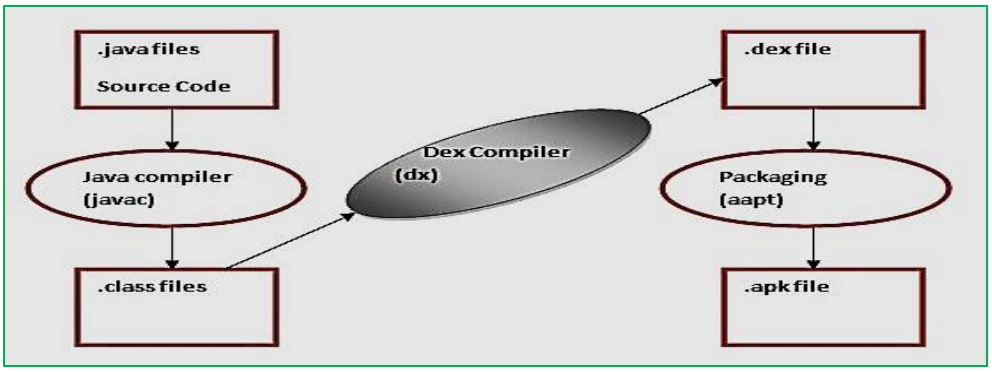


Fig: Compiling and Processing of DVM

Android Runtime (ART):

Android Runtime (ART) is an application runtime environment used by android mobile operating system. It replaces Dalvik, which is the process virtual machine originally used by android, and performs transformation of the applications bytecode into native instructions that are later executed by device's runtime environment. Unlike Dalvik, which since Android "froyo" uses Just-In-Time (jit) compilation to compile the bytecode everytime an application is launched.

ART introduces the use of Ahead-Of-Time (AOT) compilation by performing it upon the installation of an applications. By reducing the overall amount of the compilation that needs to be performed across the operation of an application, a mobile device's processor usage is reduced and battery runtime is improved. At the same time, ART brings improvement in performance, garbage collection, applications debugging and profiling.

To maintain backward compatibility, ART uses the same input bytecode as Dalvik, supplied through standard .dex files as part of APK files, while the .odex files are replaced with Executable and Linkable Format (ELF) executable.

Once an application is compiled by using ART's on device dex2oat utility, it is run solely from the compiled ELF executable; this approach eliminates various overheads involved with jit compilation, but it requires additional time for compilation when an application is installed, and applications take up slightly larger amounts of space to store compiled code.

Installation and Configuration of Android SDKs and Eclipse IDE:

Installing Android software is probably the most challenging part. It takes times - from 30 minutes to *n* hours to forever - depending on our luck (in fact, our IT knowledge) and our PC. We probably need a fairly decent PC (with 8GB RAM) and 10GB of free disk space to run the Android emulator!!! Running on actual Android devices (phone, tablet) requires much lesser resources.

Step 1: Pre-Installation Check List:

1. Before installing Android SDK, we need to install Java Development Kit (JDK). Ensure that JDK is at or above 1.8. We can check our JDK version with command "javac -version".
2. Uninstall older version(s) of "Android Studio" and "Android SDK", if any.
3. We need to install:
   - Android Studio IDE, which is an Integrated Development Environment (IDE) based on IntelliJ (a popular Java IDE); and
   - Android Software Development Kit (SDK).

Step 2: Install "Android Studio IDE":

1. Check that environment variable JAVA_HOME is set to the JDK installation directory via command "set JAVA_HOME".
2. Check the system requirements for Android Studio/SDK. For example: Windows 7/8/10, recommended 8GB of RAM and 4GB of disk space.
3. Goto "Android Developer" ⇒ Select "Get Android Studio" ⇒ "Download Android Studio 3.x.x for Windows (683MB)", e.g., android-studio-ide-171.xxxxxxx-windows.exe.
4. Run the downloaded installer ⇒ In "Choose Components", select "Android Studio" and "Android Virtual Device". ⇒ Follow the on-screen instruction and accept the defaults to complete the installation. We need about 3 GB to 4GB of free disk space! Take note (and take photo) on the installation locations of "Android Studio" (by default "C:\Program Files\Android\Android Studio") and the "Android SDK" (by default C:\Users\username\AppData\Local\Android\Sdk).

Step 3: Installing Android SDK:

Notes: Adding too many SDK packages, especially the so-called system images for emulating different device (e.g., various phone/tablet), will take an extremely LONG time, especially if everyone is downloading and jamming up the network. The system images also take up a lot of disk space - a few GB per API level!!!

Check if it is possible to copy the SDK instead of downloading the 1GB during installation?

1. Launch Android Studio ⇒ It will run the "setup" wizard for the first launch ⇒ do not import previous settings ⇒ In "Installation Type", choose "Standard" ⇒ Check the SDK folder, by default c:\Users\username\AppData\Local\Android\Sdk ⇒ Finish.

   This step will download another 1GB of SDK package and take times to complete.

   Note: In Windows, "AppData" is a hidden directory. We need to choose "View" ⇒ Check "Show Hidden Items" to see this directory.

2. (Optional) We can check the SDK packages installed by selecting "Configure" ⇒ "SDK Manager":

   Under "SDK Platforms":

   - Android API 27

   Under "SDK Tools":

   - Android SDK Build Tools
   - Android Emulator 27.x.x
   - Android SDK Platform-Tools 27.x.x
   - Android SDK Tools 26.x.x

- Intel x86 Emulator Accelerator (HAXM installer)
- Android Support Repository
- Google Repository

Eclipse:

From the moment the Android SDK was released, Eclipse has been the standard IDE for Android development and remains so to this day. From our perspective, these are the main reasons for this:

- With the release of the Android SDK, Google immediately made available the extensive Android Development Tools (ADT) plugin for Eclipse.
- ADT is used and maintained by the Google Android platform developers themselves.
- Eclipse/ADT, like the Android SDK itself, is open source and available free of charge.

Eclipse Home and Download Area

- *http://www.eclipse.org*
- *http://www.eclipse.org/downloads/*

Android Development Tools Plugin for Eclipse ADT

- *http://developer.android.com/sdk/eclipse-adt.html*

Official Google ADT Eclipse Update Site

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

Installing ADT

- To install the Eclipse ADT plugin, go to the Eclipse Help
- Install New Software menu and click the Add (a New Software Site) button.
- This should display the dialog shown.
- Enter own preferred Name and in the Location use either of the following resource locators:
  https://dl-ssl.google.com/android/eclipse/
  http://dl-ssl.google.com/android/eclipse/
- The Eclipse Available Software (dialog shown) displays with the ADT listed. Select all the tools and click next or Finish. Continue with the setup workflow until the installation is complete.

Running the Emulator:

- With project selected in the Package Explorer pane, click the green "play" button in the Eclipse toolbar to run project.
- At first time we have to go through a few steps to set up a "run configurations", and so Eclipse knows what we want to do.

❖ First, in the "Run As" list, choose "Android Application": If we have more than one emulator AVD or device available, we will then get an option to choose which we wish to run the application on. Otherwise, if we do not have a device plugged in, the emulator will start up with the AVD we created earlier.

❖ Then, Eclipse will install the application on our device or emulator and start it up.

<u>Using ADB Command Lines:</u>

This makes at least two assumptions:

1. We have the Android SDK installed.
2. We have an Android emulator (or physical device) running.

At the adb shell prompt we can enter a variety of commands to interact with your Android emulator or device.

The brief list of frequently used commands is:

Command	Action
adb device	List the installed devices
adb pull <remote> <local>	Copy a file or directory to the emulator or device
adb intall foo.apk	Install the apk file/app
adb intall –r foo.apk	Update the already installed app
adb uninstall <pkg>	Uninstall the app given by pkg
ls	List files and directories
logcat	View the system's log buffers
reboot	Reboot the device

This command lets us uninstall an Android application, where the argument given to the uninstall command is the root package name of the app: ***$adb uninstall com.MyPackageName***