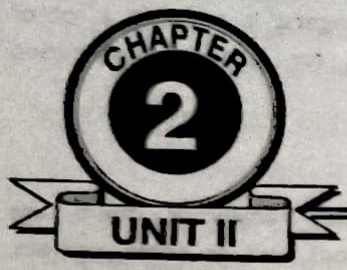


Installation and Configuration of Android

Syllabus :

2.1	Operating System, Java JDK, Android SDK
2.2	Android Development Tools(ADT)
2.3	Android Virtual Devices(AVDs)
2.4	Emulators
2.5	Dalvik Virtual Machine, Difference between JVM and DVM
2.6	Steps to install and configure Android Studio and SDK

2.1 Operating System

- An Operating System (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources. An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.
- An operating system is a software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

2.1.1 Types of Operating Systems

- An operating system performs all the basic tasks like managing file, process, and memory. Thus operating system acts as manager of all the resources, i.e. resource manager. Thus operating system becomes an interface between user and machine.
- Some of the widely used operating systems are as follows-

1. Batch Operating System

- This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having same requirement and group them into batches. It is the responsibility of operator to sort the jobs with similar needs.
- Examples of batch based operating system are : Payroll System, Bank Statements etc.

2. Time-Sharing Operating System

- Each task is given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems. The task can be from single user or from different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.
- Examples of time-sharing OSs are: Multics, Unix etc.

3. Distributed Operating System

- These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all-over the world and, that too, with a great pace. Remote access is enabled within the devices connected in that network.
- Example of distributed operating system is : LOCUS.

4. Network Operating System

- These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network.
- Examples of network operating system are : Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD etc.

5. Real-Time Operating System

- These types of OSs serve the real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called response time. Real-time systems are used when there are time requirements are very strict like missile systems, air traffic control systems, robots etc.
- Two types of real-time operating system which are as follows:
  1. Hard real - time systems
  2. Soft real - time systems
- Examples of real-time operating systems are: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems etc.

2.1.2 Java JDK

- The JDK is a development environment for building applications, applets, and components using the Java programming language.
- The JDK is a software package that contains a variety of tools and utilities that make it possible to develop, package, monitor and deploy applications that build for any standard Java platform, including Java Platform, Standard Edition (Java SE); Java Platform, Micro Edition (Java ME); and Java Platform, Enterprise Edition (Java EE).

1. JVM

- JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java byte code.
- JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent. There are three notions of the JVM: specification, implementation, and instance.
- The JVM performs the following main tasks:
  a. Loads code
  b. Verifies code
  c. Executes code
  d. Provides runtime environment

2. JRE

- JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries plus other files that JVM uses at runtime.
- The implementation of JVM is also actively released by other companies besides Sun Micro Systems.

3. JDK

- JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.
- JDK is an implementation of any one of the below given Java platforms released by Oracle Corporation:
  1. Standard Edition Java Platform
  2. Enterprise Edition Java Platform
  3. Micro Edition Java Platform
- The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

2.1.3 Android SDK

- Android SDK Tools is a component for the Android SDK. It includes the complete set of development and debugging tools for Android. It is included with Android Studio.
- A software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linux kernel.
- The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes the following :
  1. Required libraries
  2. Debugger
  3. An emulator
  4. Relevant documentation for the Android application program interfaces (APIs)
  5. Sample source code
  6. Tutorials for the Android OS.

2.2 Android Development Tools

1. Android Studio

No list of Android development tools would be complete without Android Studio. This is the official IDE (Integrated Development Environment) for Android, making it the number one choice for the majority of developers looking to make basic apps in-keeping with Google's Material Design and with access to all the advanced features of the platform.

2. Visual Studio with Xamarin

Visual Studio is Microsoft's IDE which supports a range of languages including C#, VB.net, JavaScript and more with extensions. Using Xamarin, which now comes bundled in, it's also possible to create cross-platform apps using C# and then test on multiple devices connected to the cloud.

3. Eclipse

The Eclipse Foundation - home to a global community, the Eclipse IDE, Jakarta EE and over 350 open source projects, including runtimes, tools and frameworks which used to develop android application following are steps to develop application using eclipse IDE.

- Install the JDK.
- Download and install the Eclipse for developing android application.
- Download and Install the android SDK.
- Install the ADT plugin for eclipse.
- Configure the ADT plugin.
- Create the AVD.
- Create the hello android application.

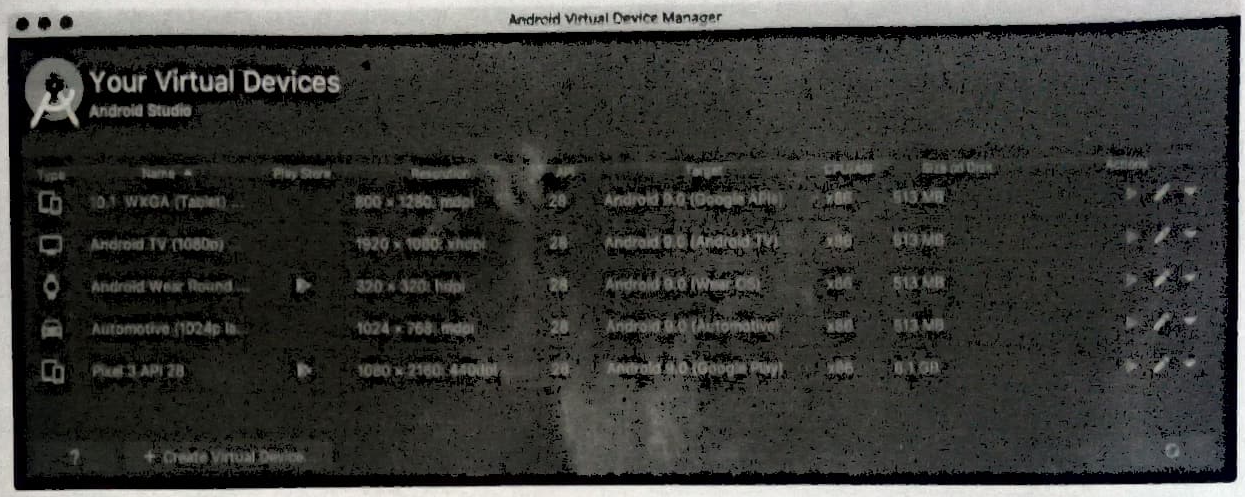
2.3 Android Virtual Devices

2.3.1 Create and Manage Virtual Devices

- An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator. The AVD Manager is an interface you can launch from Android Studio that helps you create and manage AVDs.
- An Android Virtual Device (AVD) is a device configuration that is run with the Android emulator. It works with the emulator to provide a virtual device-specific environment in which to install and run Android apps. Following part shows how to create an AVD by introducing you to the Android SDK's AVD Manager Tool.
- To open the AVD Manager, do one of the following:

  Select Tools → AVD Manager

  Click AVD Manager AVD Manager icon in the toolbar.
- Now click the + Create Virtual Device... button. This will bring up Virtual Device Configuration Dialog
- Select any device you want, then click Next:
- Now you need to choose an Android version for your emulator. You might also need to download it first by clicking Download. After you've chosen a version, click Next



- Enter a name for your emulator, initial orientation, and whether you want to display a frame around it. After you chosen all these, click Finish.
- Now you got a new AVD ready for launching your apps on it.

2.4 Emulators

- The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.
- The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.
- Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.
- The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

1. Requirements and Recommendations

- The Android Emulator has additional requirements beyond the basic system requirements for Android Studio, which are described below:
  - SDK Tools 26.1.1 or higher.
  - 64-bit processor.
  - Windows : CPU with UG (Unrestricted Guest) support.
  - HAXM 6.2.1 or later (HAXM 7.2.0 or later recommended).
- The use of hardware acceleration has additional requirements on Windows and Linux:
  - **Intel processor on Windows or Linux :** Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality.
  - **AMD processor on Linux :** AMD processor with support for AMD Virtualization (AMD-V) and Supplemental Streaming SIMD Extensions 3 (SSSE3).
  - **AMD processor on Windows :** Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for Windows Hypervisor Platform (WHPX) functionality
- To work with Android 8.1 (API level 27) and higher system images, an attached webcam must have the capability to capture 720p frames.

2. Launch the Android Emulator without first running an app

- To start the emulator:
  - Open the AVD Manager.
  - Double-click an AVD, or click Run.
  - The Android Emulator loads.

- While the emulator is running, you can run Android Studio projects and choose the emulator as the target device. You can also drag one or more APKs onto the emulator to install them, and then run them.

2.5 Dalvik Virtual Machine (DVM)

- As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.
- The Dalvik Virtual Machine (DVM) is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for memory, battery life and performance. Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein. The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.
- Let's see the compiling and packaging process from the source file:

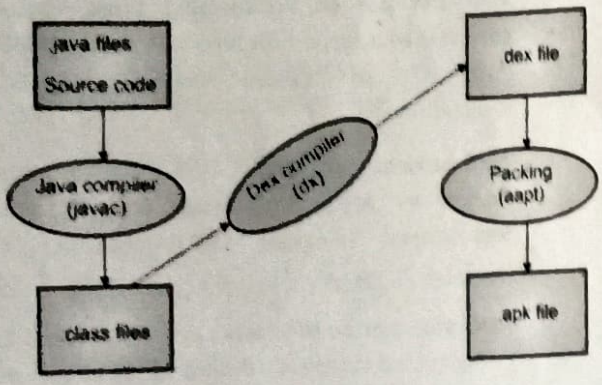


Fig. 2.5.1

- The javac tool compiles the java source file into the class file. The dx tool takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.
- The Android Assets Packaging Tool (AAPT) handles the packaging process.

2.5.1 Role of Dalvik Virtual Machine (DVM)

The role of Dalvik Virtual Machine(DVM) 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.

2.5.2 Difference between JVM and DVM

Sr. No.	DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
1.	It is register based which is designed to run on low memory.	It is stack based.
2.	DVM uses its own byte code and runs ".Dex" file. From Android 2.2 SDK Dalvik has got a Just in Time compiler	JVM uses java byte code and runs ".class" file having JIT (Just In Time).
3.	DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	Single instance of JVM is shared with multiple applications.
4.	DVM supports Android operating system only.	JVM supports multiple operating systems.
5.	For DVM very few re-tools are available.	For JVM many re-tools are available.
6.	There is constant pool for every application.	It has constant pool for every class.
7.	Here the executable is APK.	Here the executable is JAR.

2.6 Steps to Install and Configure Android Studio and SDK

Step 1 : System Requirements

- You will be delighted, to know that you can start your Android application development on either of the following operating systems :
  - Microsoft® Windows® 10/8/7/Vista/2003 (32 or 64-bit)
  - Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks)
  - GNOME or KDE desktop.

- Second point is that all the required tools to develop Android applications are open source and can be downloaded from the web. Following is the list of software's you will need before you start your Android application programming.
  - Java JDK5 or later version
  - Java Runtime Environment (JRE) 6
  - Android Studio

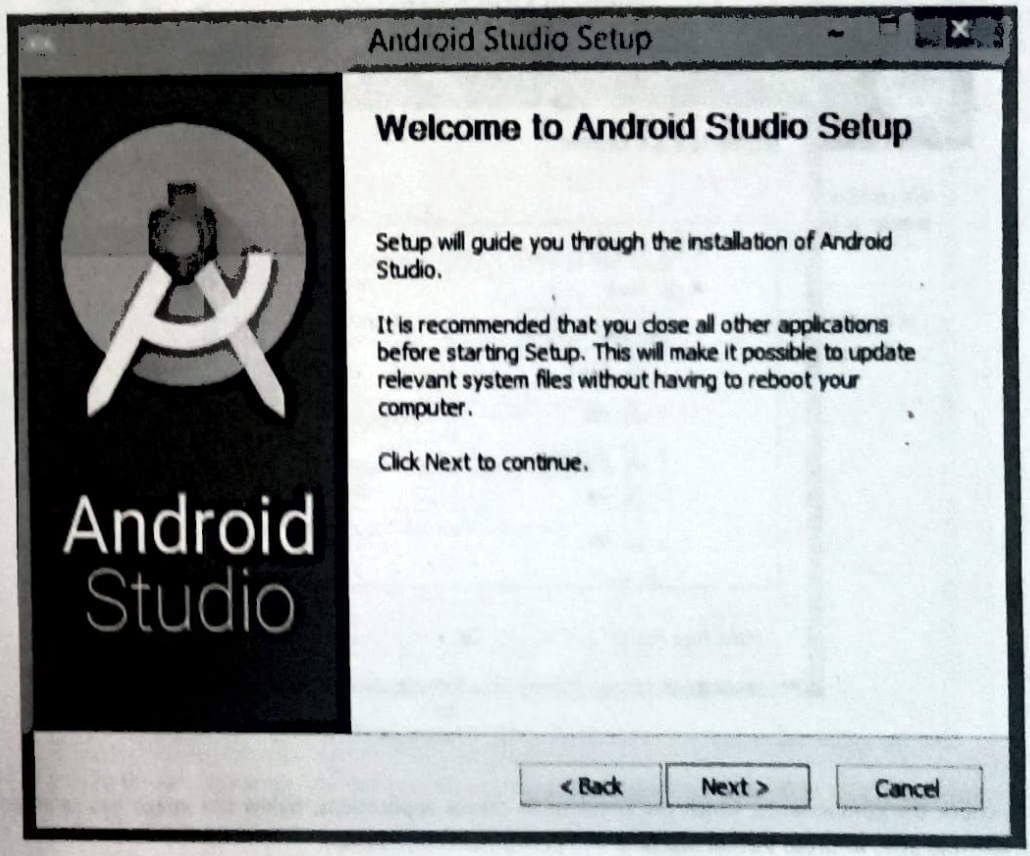
Step 2 : Setup Android Studio

Overview

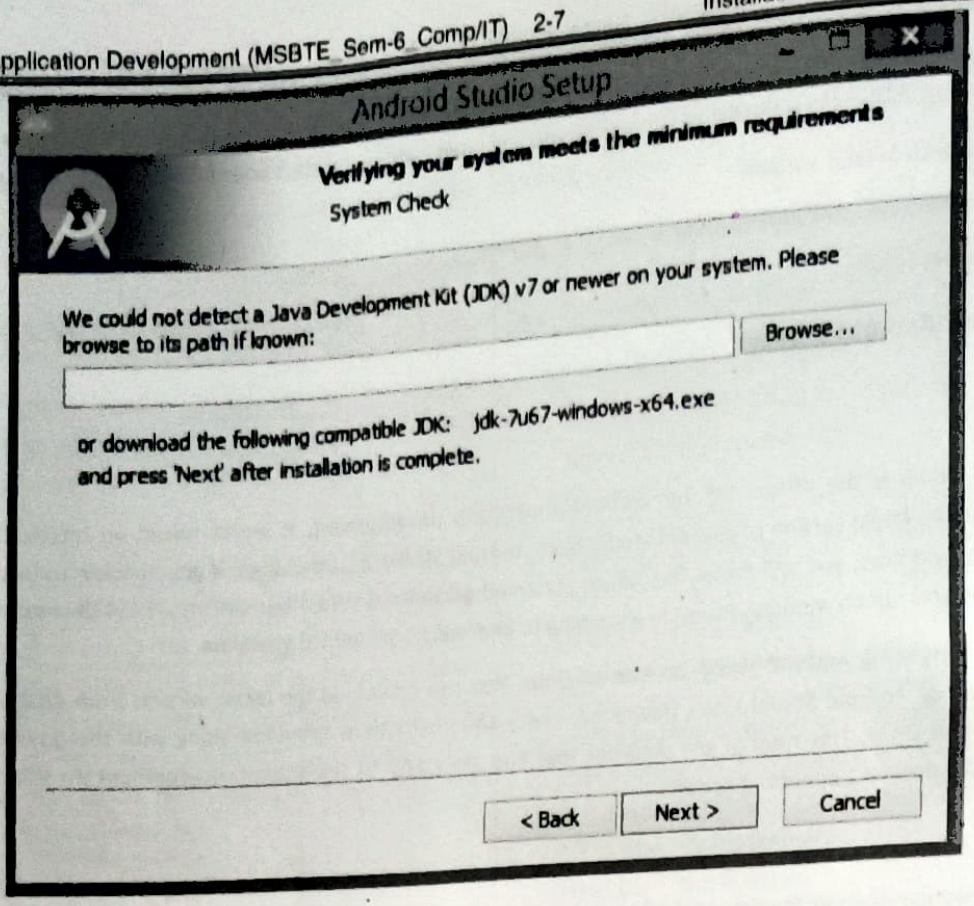
- Android Studio is the official IDE for android application development. It works based on IntelliJ IDEA, You can download the latest version of android studio from Android Studio 2.2 Download, If you are new to installing Android Studio on windows, you will find a file, which is named as android-studio-bundle-143.3101438-windows.exe.So just download and run on windows machine according to android studio wizard guideline.
- If you are installing Android Studio on Mac or Linux, You can download the latest version from Android Studio Mac Download, or Android Studio Linux Download, check the instructions provided along with the downloaded file for Mac OS and Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows 8.1 operating system.

Installation

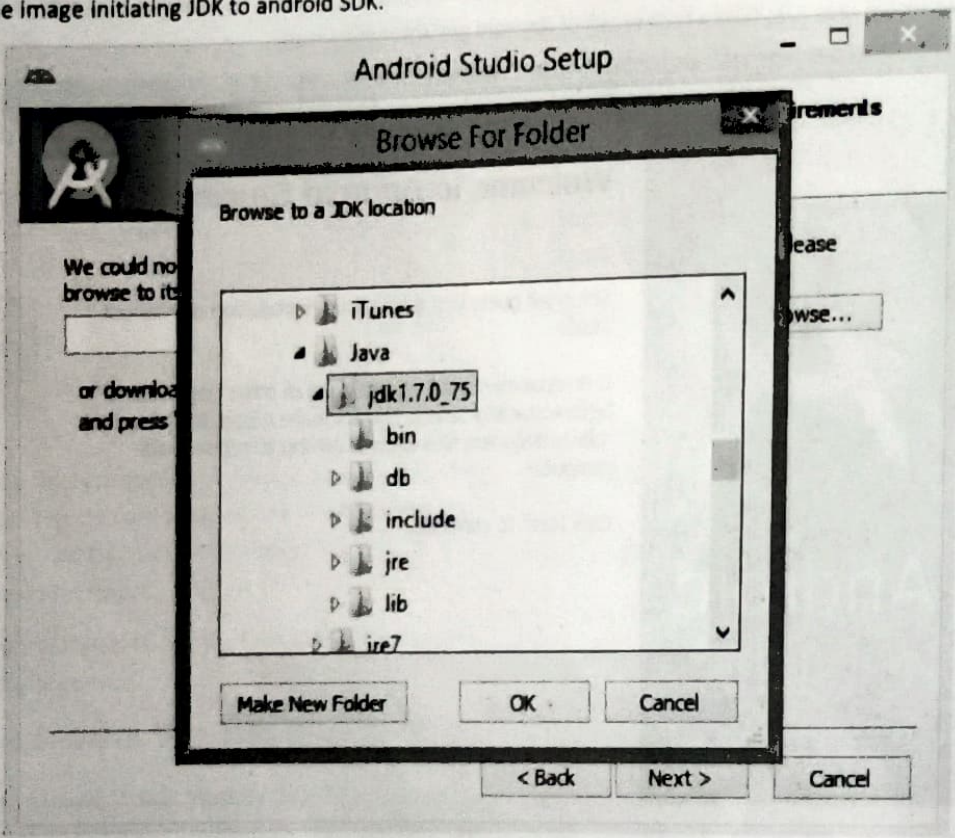
- So let's launch *Android Studio.exe*, Make sure before launch Android Studio, Our machine should require installed Java JDK. To install Java JDK, take a references of Android environment setup.



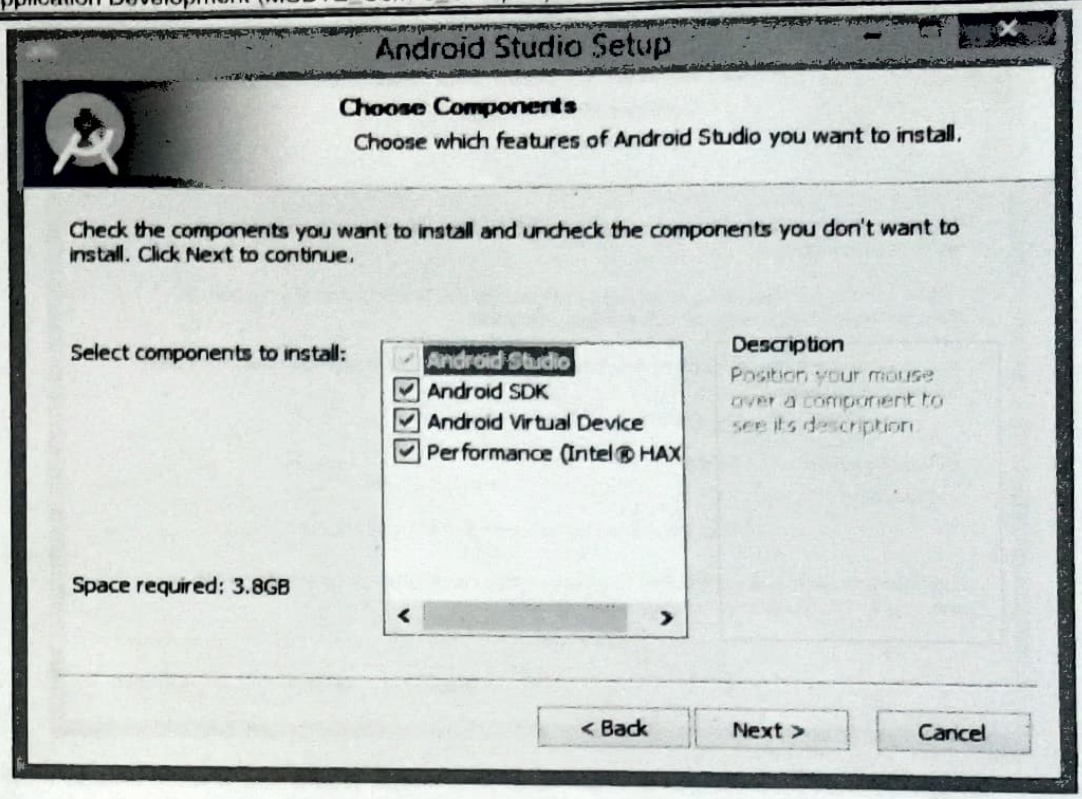
- Once you launched Android Studio, its time to mention JDK7 path or later version in android studio installer.



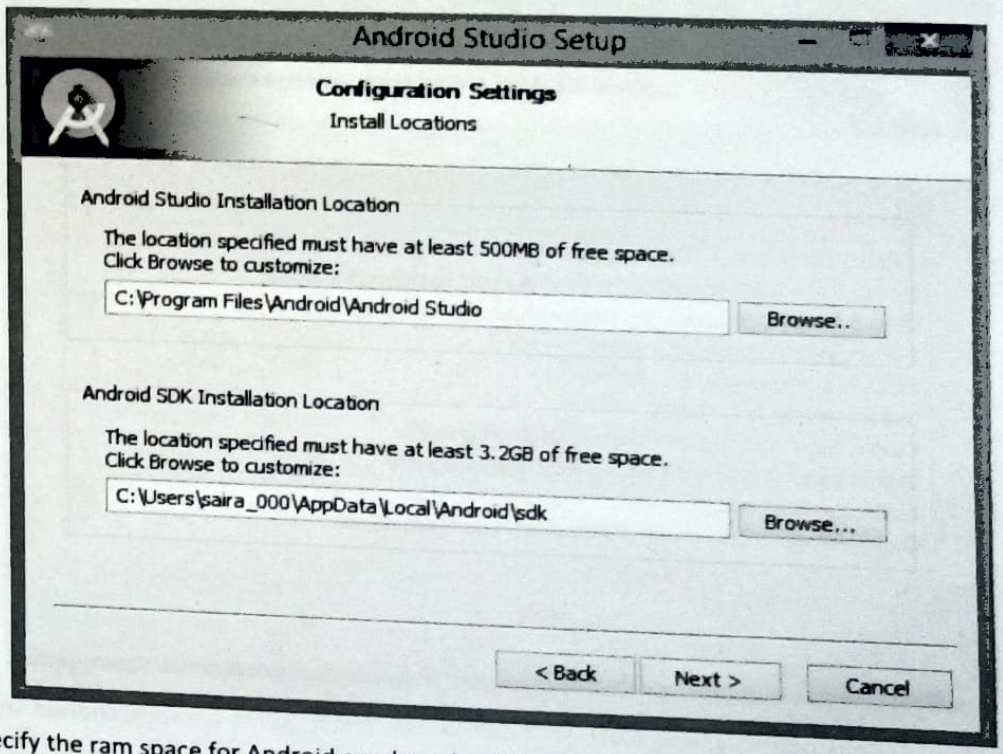
- Below the image initiating JDK to android SDK.



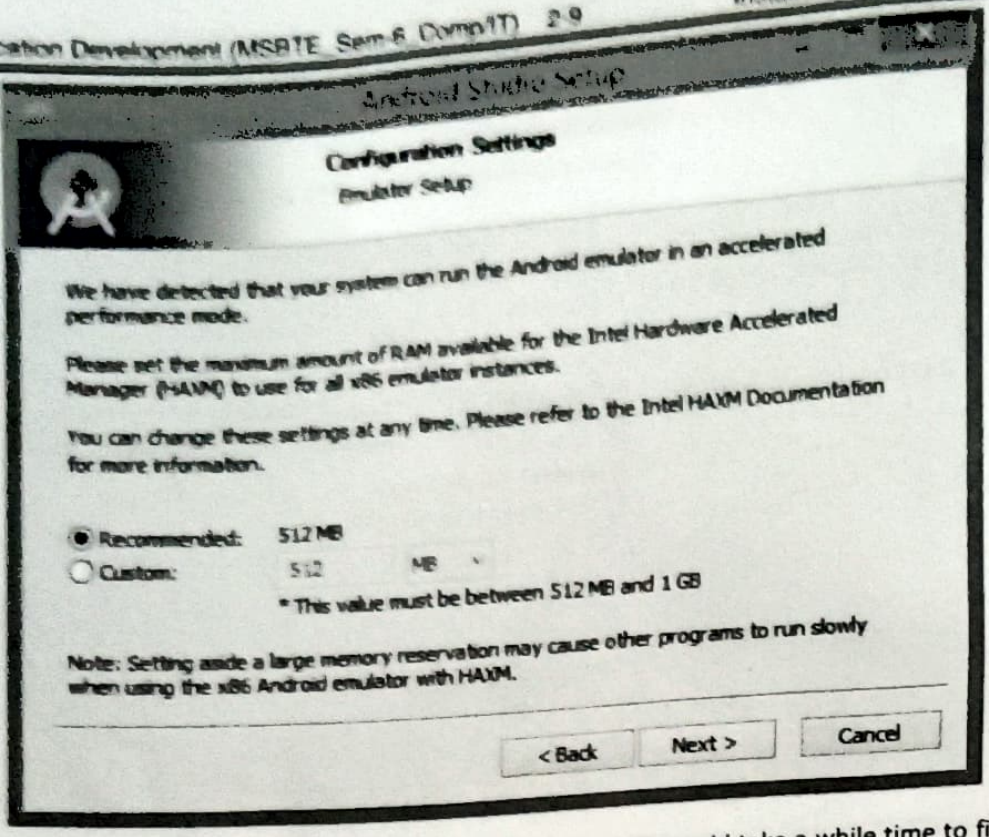
- Need to check the components, which are required to create applications, below the image has selected **Android Studio, Android SDK, Android Virtual Machine** and **performance(Intel chip).**



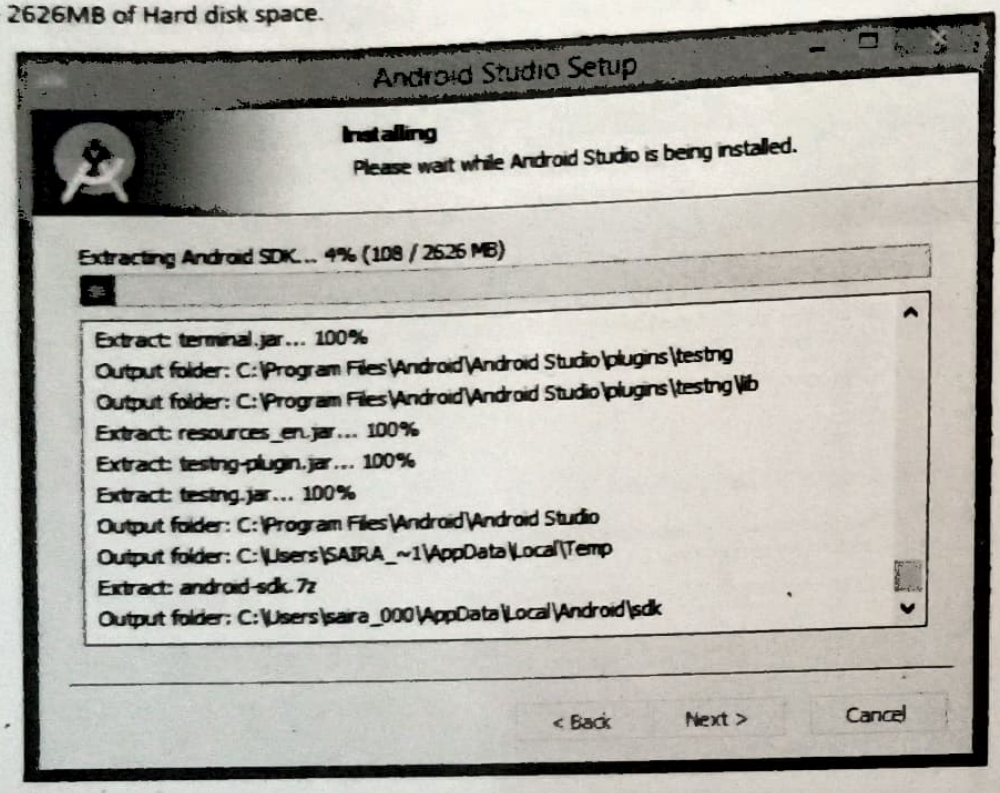
- Need to specify the location of local machine path for Android studio and Android SDK, below the image has taken default location of windows 8.1 x 64 bit architecture.



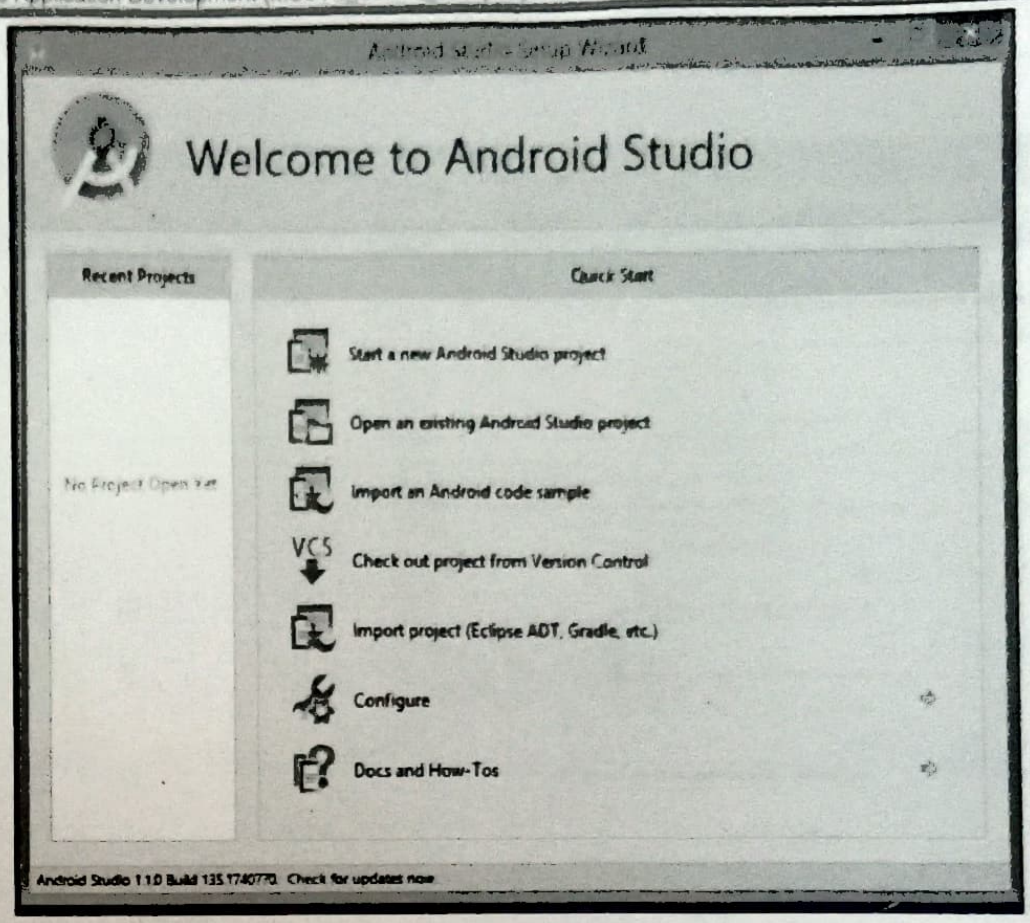
- Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM.



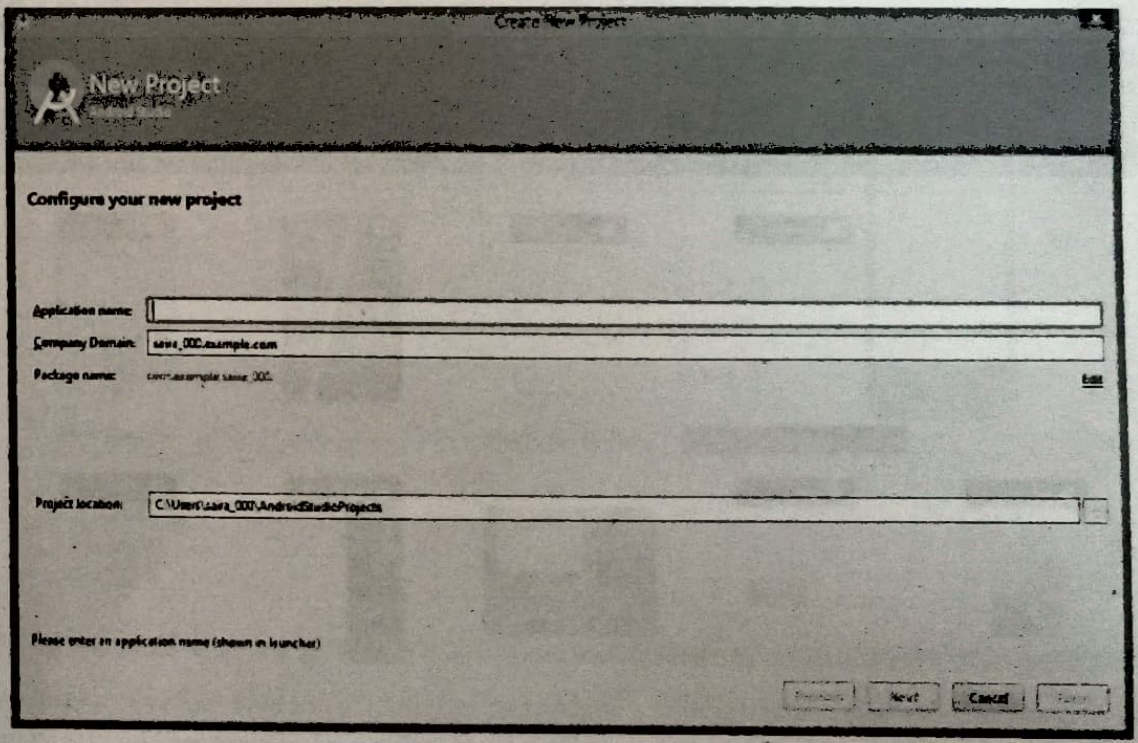
- At final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space.



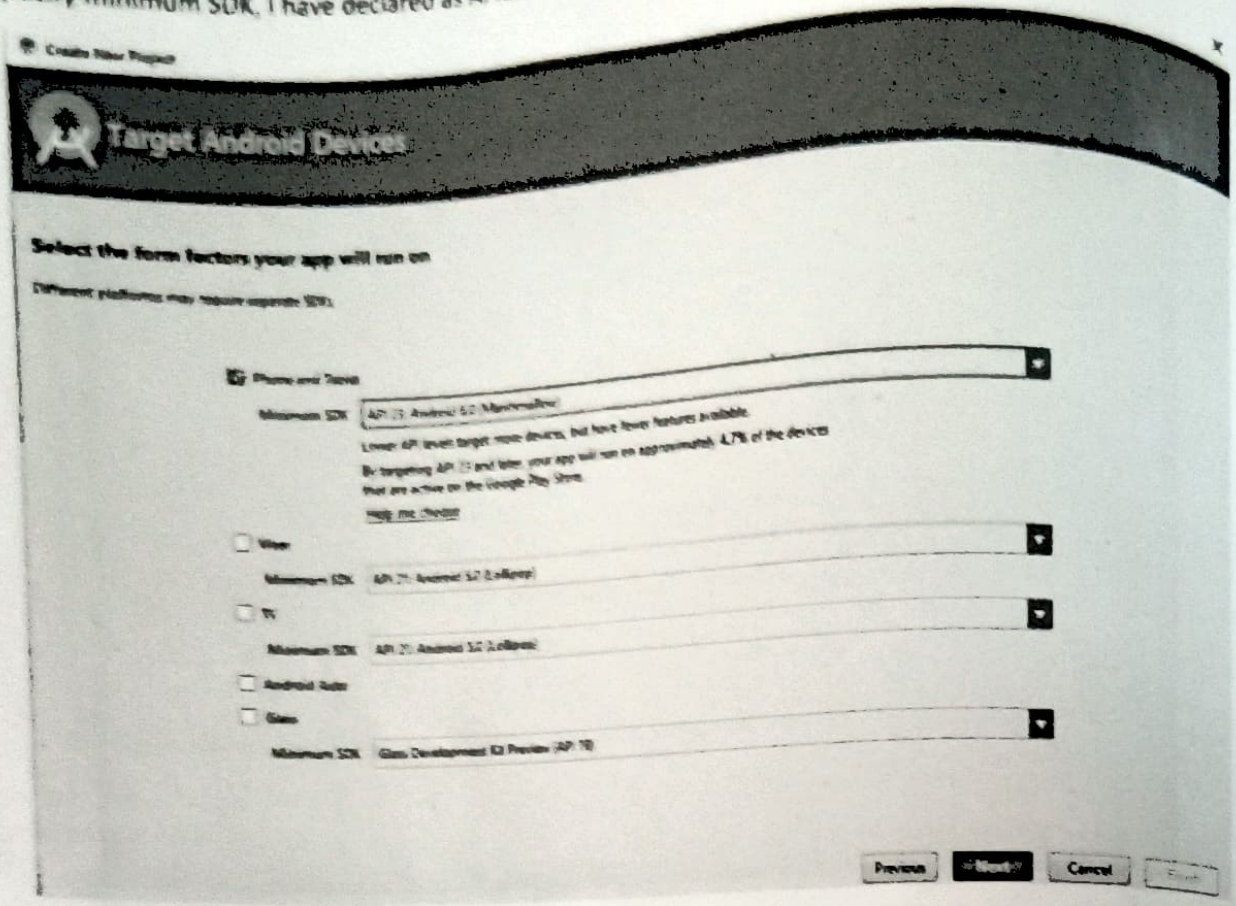
- After done all above steps perfectly, you must get finish button and it gonna be open android studio project with Welcome to android studio message as shown below.



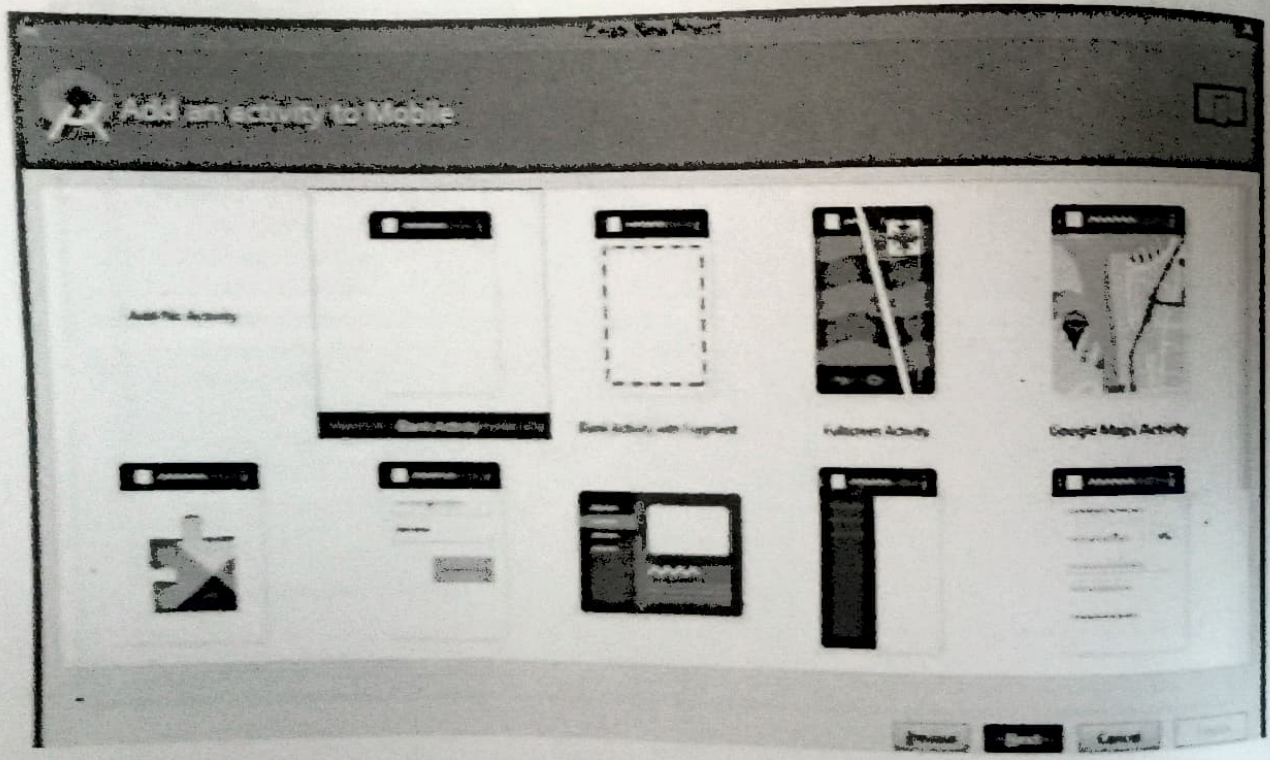
- You can start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.



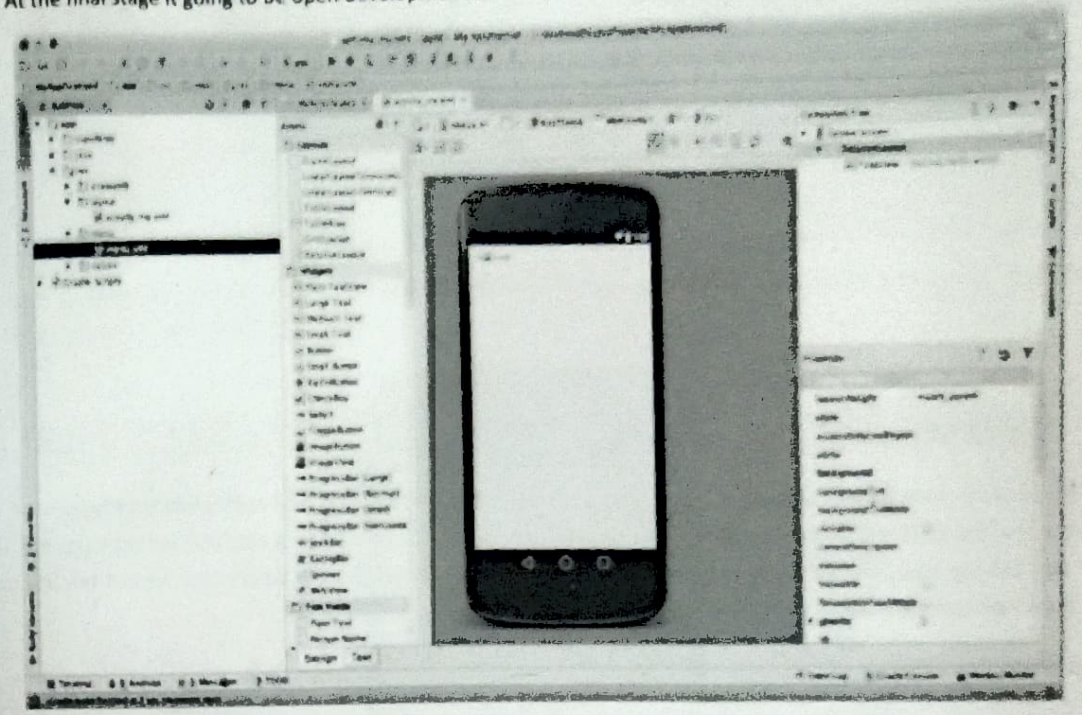
- After entered application name, it going to be called select the form factors your application runs on, here need to specify minimum SDK, I have declared as API23: Android 6.0(Mashmallow).



- The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications.

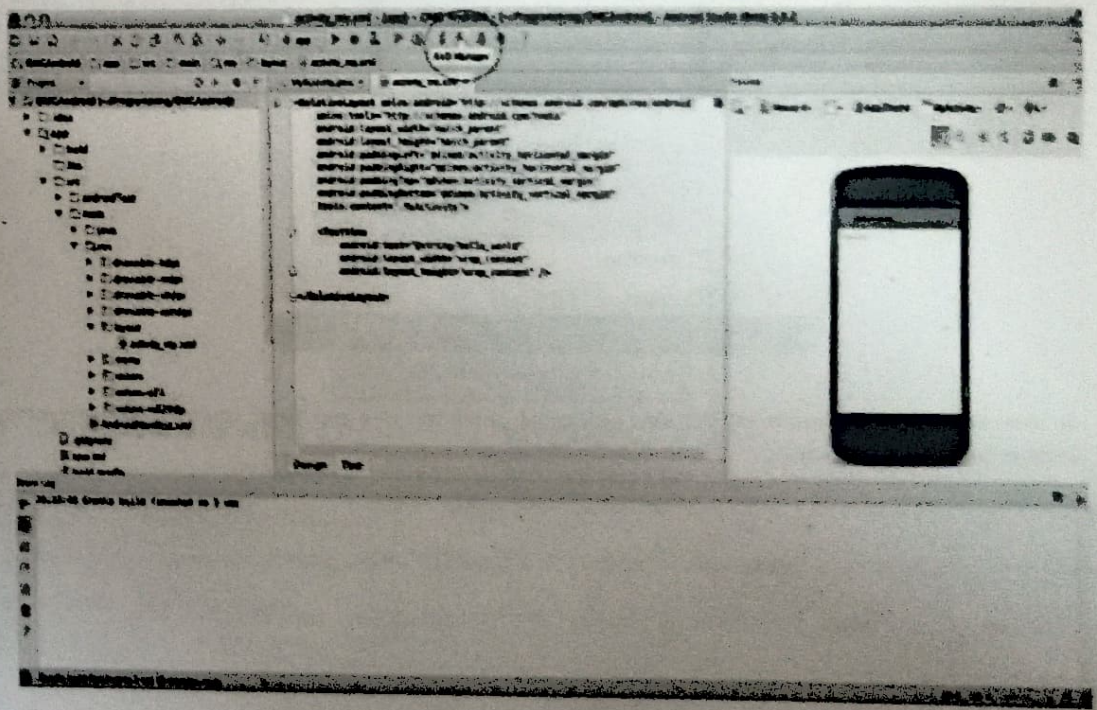


- At the final stage it going to be open development tool to write the application code.

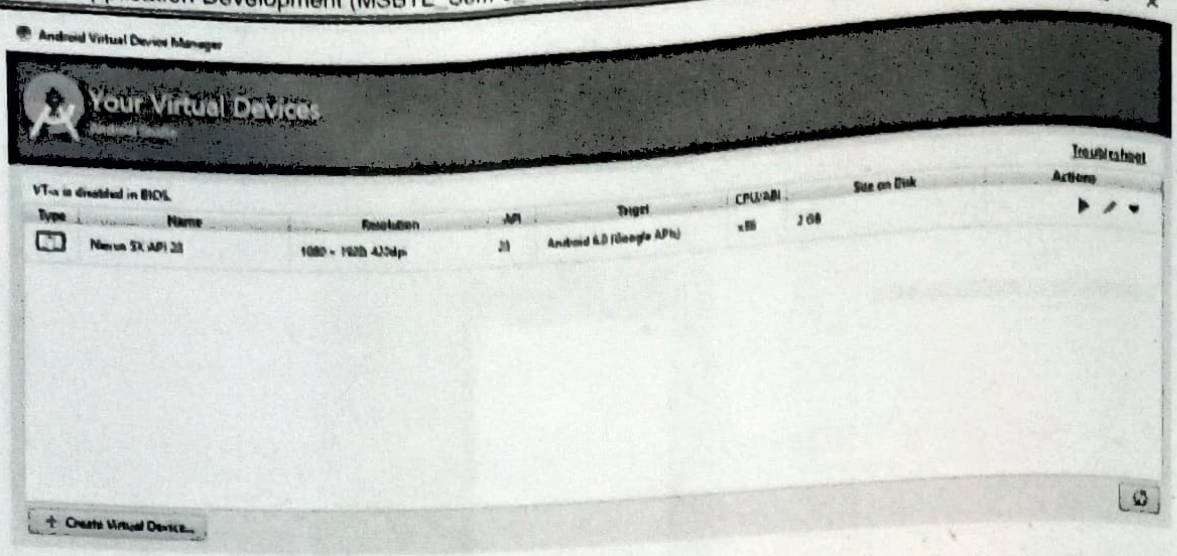


Step 3 : Create Android Virtual Device

- To test your Android applications, you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager Clicking AVD_Manager icon as shown below.



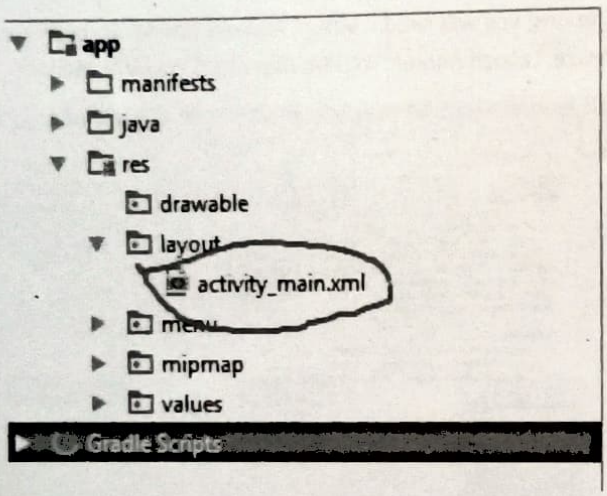
- After click on a virtual device icon, it going to be shown by default virtual devices which are present on your SDK, or else need to create a virtual device by clicking **Create new Virtual device** button



- If your AVD is created successfully it means your environment is ready for Android application development. If you like, you can close this window using top-right cross button. Better you re-start your machine and once you are done with this last step, you are ready to proceed for your first Android example but before that we will see few more important concepts related to Android Application Development.

Hello Word Example

- Before writing a Hello word code, you must know about XML tags. To write hello word code, you should redirect to **App>res>layout>Activity_main.xml**



- To show hello word, we need to call text view with layout (about text view and layout, you must take references at Relative layout and Text view).

```
<RelativeLayoutxmlns:android="http://schemas. android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">
```

```
<TextViewandroid:text="@string/hello_world"
android:layout_width="550dp"
android:layout_height="wrap_content" />

</RelativeLayout>
```

- Need to run the program by clicking **Run>Run App** or else need to call **shift+f10** key. Finally, result should be placed at virtual devices as shown Fig. 2.6.1.

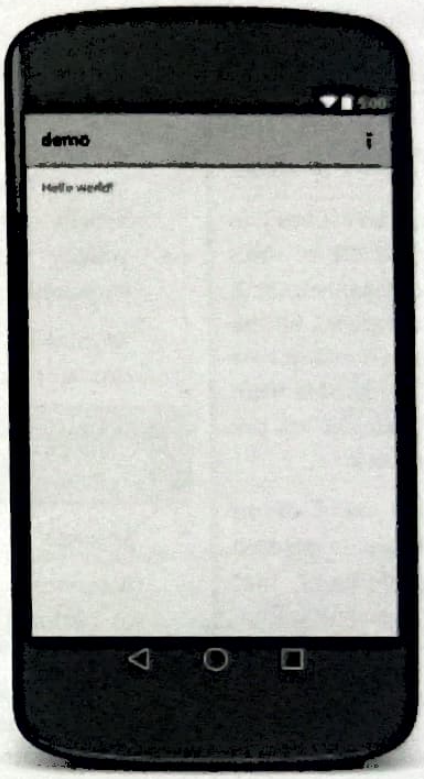


Fig. 2.6.1

Review Questions

Q.1 Describe function of various components required for android IDE.

Q.2 List various tools or IDE used for android application development.

Q.3 Explain Android SDK and Java JDK.

Q.4 Define role of emulators in android application development with example.

Q.5 Compare JVM and DVM.

Q.6 Explain Dalvik virtual machine.

Q.7 Describe various installation steps of android studio and its environment.

□□□