

Android sdk

Stands for "Software Development Kit." An SDK is a collection of software used for developing applications for a specific device or operating system.

Android Virtual Device (AVD)

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. Lesson 4 shows you how to create an AVD by introducing you to the Android SDK's AVD Manager tool.

Android - Emulator

The Android SDK includes a virtual mobile device emulator that runs on your computer. The emulator lets you prototype, develop and test Android applications without using a physical device.

In this chapter we are going to explore different functionalities in the emulator that are present in the real android device.

Creating AVD

If you want to emulate a real device, first create an AVD with the same device configurations as real device, then launch this AVD from AVD manager.

Changing Orientation

Usually by default when you launch the emulator, its orientation is vertical, but you can change its orientation by pressing Ctrl+F11 key from keyboard.

First launch the emulator. It is shown in the picture below –



Once it is launched, press **Ctrl+F11** key to change its orientation. It is shown below

Sr.No	Command & description
1	Home Shifts to main screen
2	F2 Toggles context sensitive menu
3	F3 Bring out call log
4	F4 End call
5	F5 Search
6	F6 Toggle trackball mode
7	F7 Power button
8	F8 Toggle data network
9	Ctrl+F5 Ring Volume up
10	Ctrl+F6 Ring Volume down



Emulator Commands.

Apart from just orientation commands, there are other very useful commands of emulator that you should keep in mind while using emulator. They are listed below –

Emulator - Sending SMS

You can emulate sending SMS to your emulator. There are two ways to do that. You can do that from DDMS which can be found in Android studio, or from Telnet. (Network utility found in windows).

Sending SMS through Telnet.

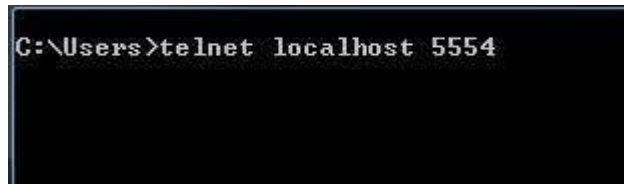


Telnet is not enabled by default in windows. You have to enable it to use it. Once enabled you can go to command prompt and start telnet by typing telnet.

In order to send SMS , note down the AVD number which can be found on the title bar of the emulator. It could be like this 5554 e.t.c. Once noted , type this command in command prompt.

```
telnet localhost 5554
```

Press enter when you type the command. It is shown below in the figure.

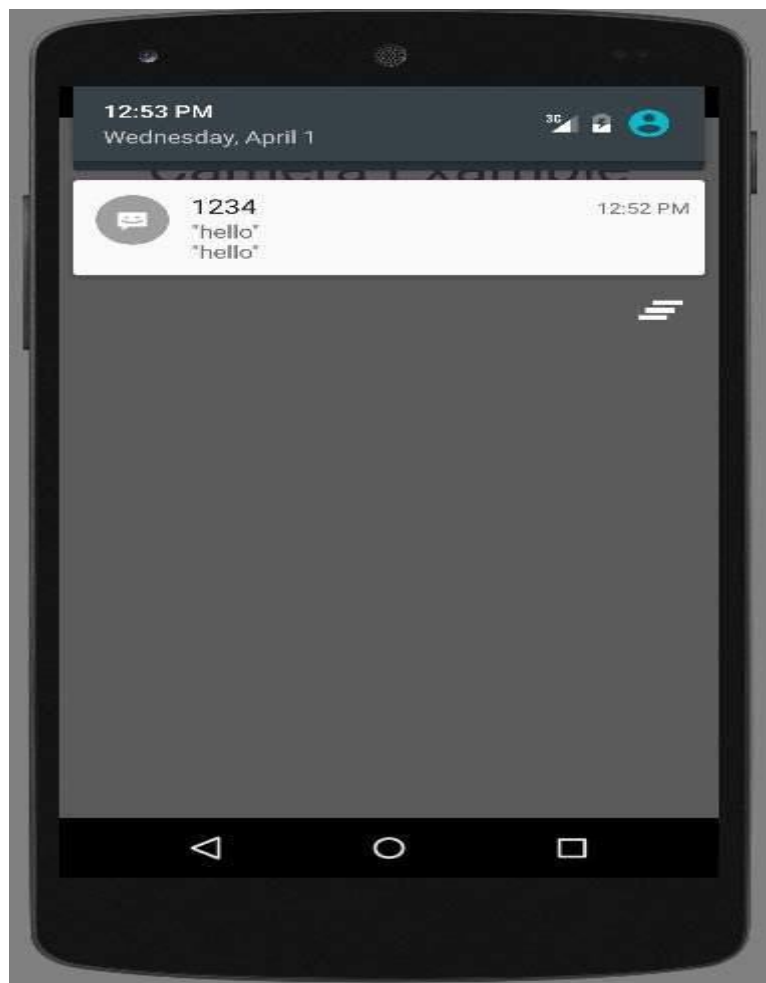


```
C:\Users>telnet localhost 5554
```

You will see that you are now connected to your emulator. Now type this command to send message.

```
sms send 1234 "hello"
```

Once you type this command, hit enter. Now look at the AVD. You will receive a notification displaying that you got a new text message. It is shown below –



Emulator - Making Call

You can easily make phone calls to your emulator using telnet client. You need to connect to your emulator from telnet. It is discussed in the sending sms topic above.

After that you will type this command in the telnet window to make a call. Its syntax is given below –

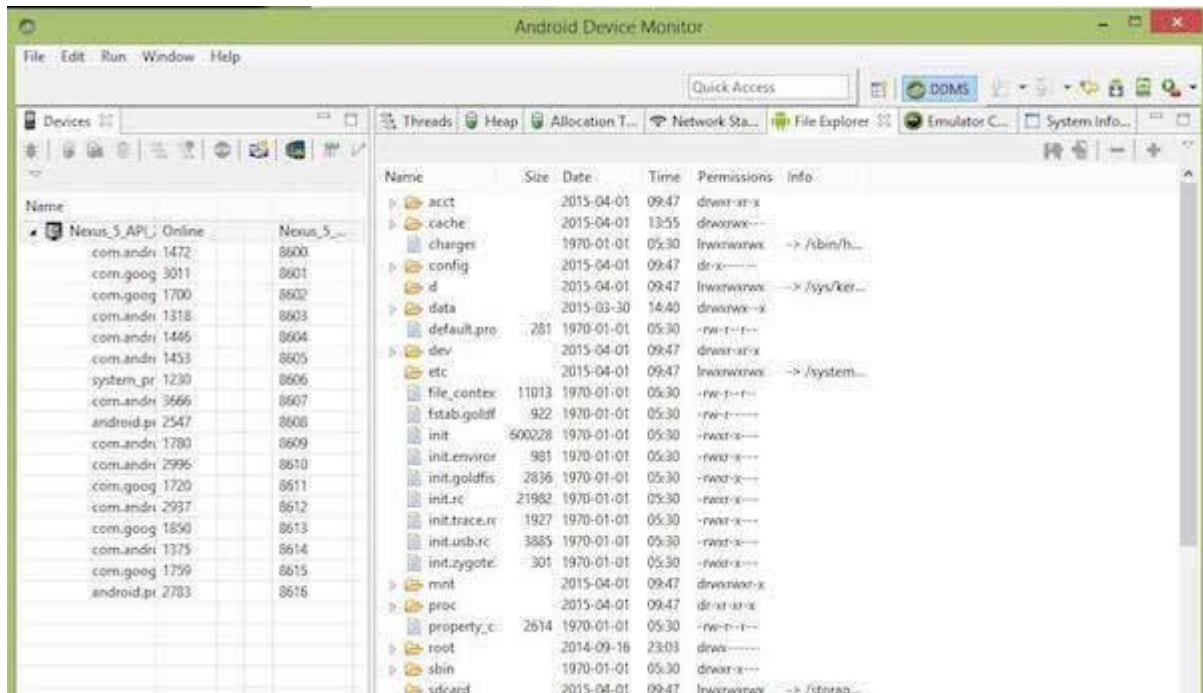
```
gsm call 1234
```

Once you type this command , hit enter. Now look at the AVD. You will receive a call from the number your put in the command. It is shown below –



Emulator - Transferring files

You can easily transfer files into the emulator and vice versa. In order to do that, you need to select the DDMS utility in Android studio. After that select the file explorer tab. It is shown below –



Browse through the explorer and make new folder , view existing contents e.t.c.

Dalvik Virtual Machine | DVM

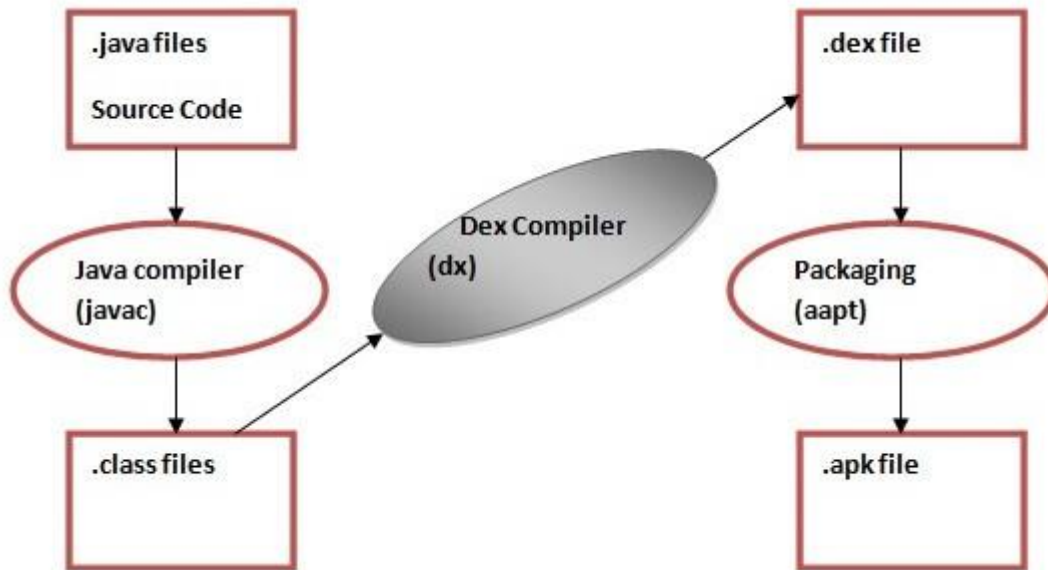
As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.

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:



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.

Runtime Environment

Runtime Environment consists of software instructions (generated from the code) that are used while executing the programming.

JVM is the component that is used to convert bytecode into machine code in order to run Java-based programs.

Why is a Virtual Machine need to run any Application?

- A Virtual Machine isolates the execution of the program from the OS. Thus protecting malicious code from affecting the system files.
- Virtual Machines execute code independent of the CPU architecture

Dalvik Virtual Machine (DVM) was specifically designed to run Android applications initially.

Why Android uses DVM and not JVM?

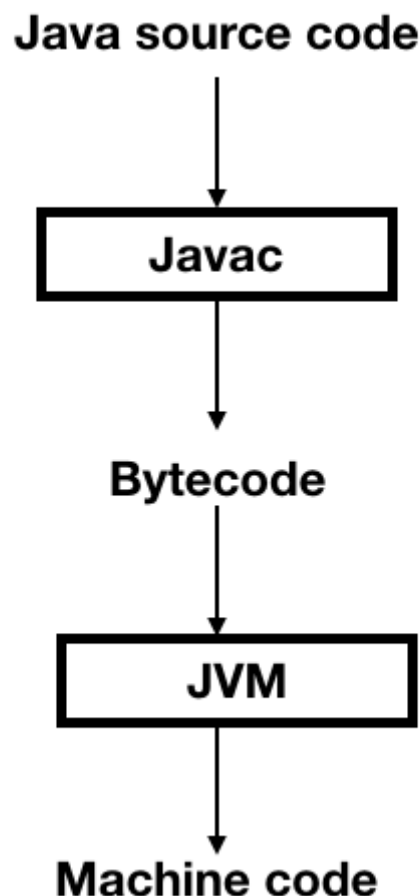
Mobile Environment is not as powerful as your computer systems(majorly). There are battery and ram constraints. DVM was specifically optimized in order to run on Android.

Mobile Environment is not as powerful as your computer systems(majorly). There are battery and ram constraints. DVM was specifically optimized in order to run on Android.

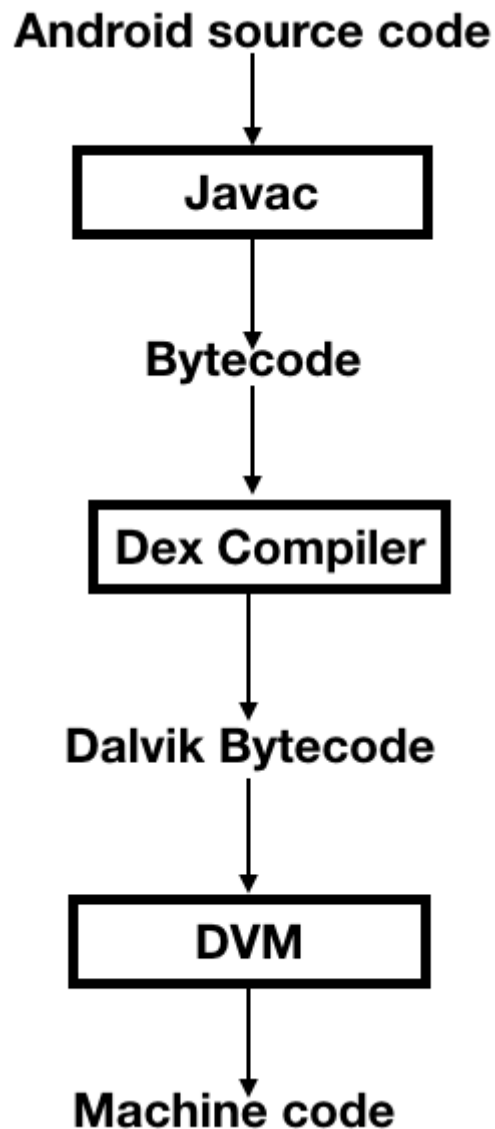
Android DVM

Before looking at the Dalvik Virtual Machine, let's look at the JVM:

Following is the flow of JVM:



Following is the flow of DVM:



We know that DVM is specifically used for low memory devices.

The Dex compiler (`dx` tool) converts the .class files generated from the javac compiler to .dex file.

These .dex files are then converted to machine code.

Note: `dexopt` tool which is a part of the DVM converts the dex file into .odex file format.

JVM runs the bytecode of pure java classes while DVM runs the bytecode of the dex formatted .dex file that was recompiled from the Java bytecode.

JVM dynamically loads the bytecode for each class from the corresponding .class file. While Dalvik bytecode is only composed of one .dex file, containing all the classes of the application.

How is the dex bytecode converted into machine code?

Using **JIT**(Just In Time).

Just In Time is a component that takes application code, analyzes it, and actively translates it into a form that runs faster, doing so while the application continues to run. This leads to increased launch time for applications since it needs to be done everytime the application is launched.

As the execution progresses, more bytecode is compiled and cached. This leads to faster boot times.

DVM and JIT were replaced by ART and AOT respectively since Android Lollipop.



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.

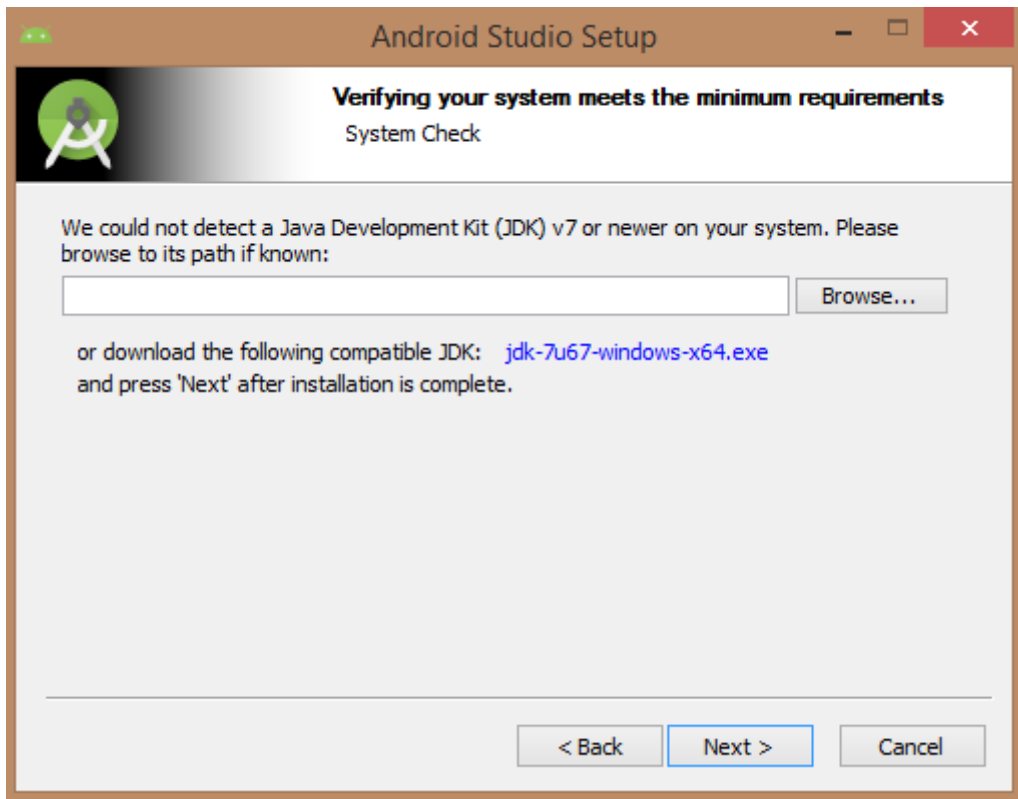
If you are installing Android Studio on Mac or Linux, You can download the latest version from [Android Studio Mac Download](#), or [Android Studio Linux Download](#), check the instructions provided along with the downloaded file for Mac OS and Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows 8.1 operating system.

Installation

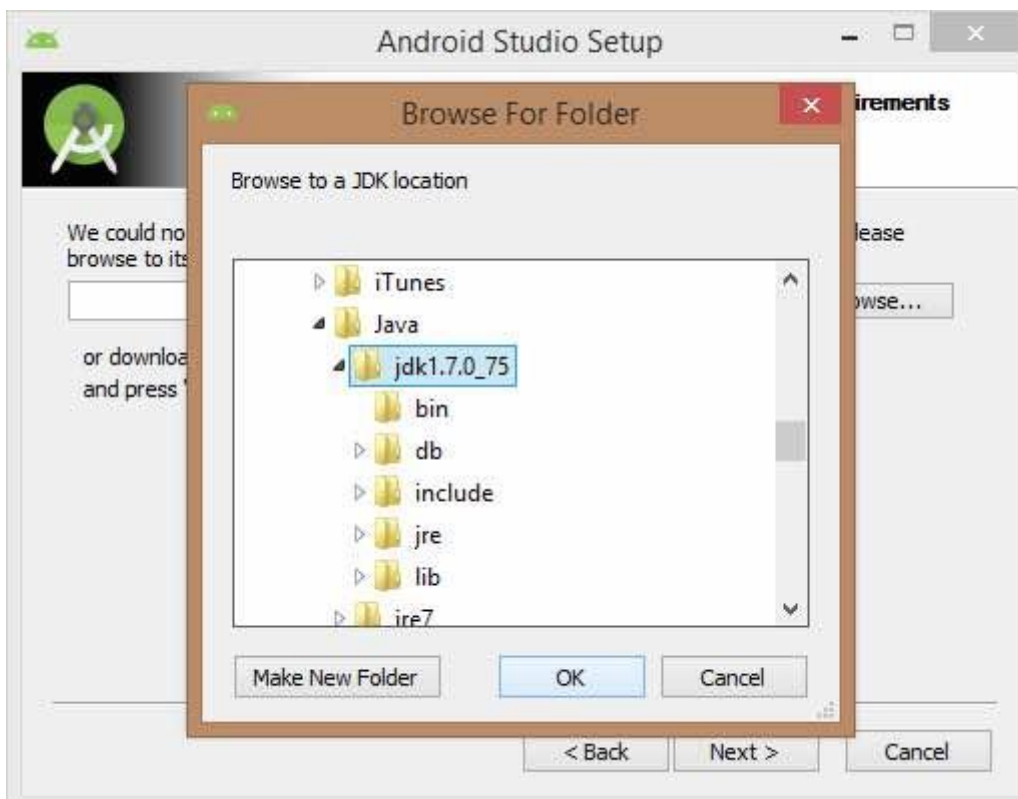
So let's launch *Android Studio.exe*. Make sure before launch Android Studio, Our Machine should required installed Java JDK. To install Java JDK, take a references of [Android environment setup](#)



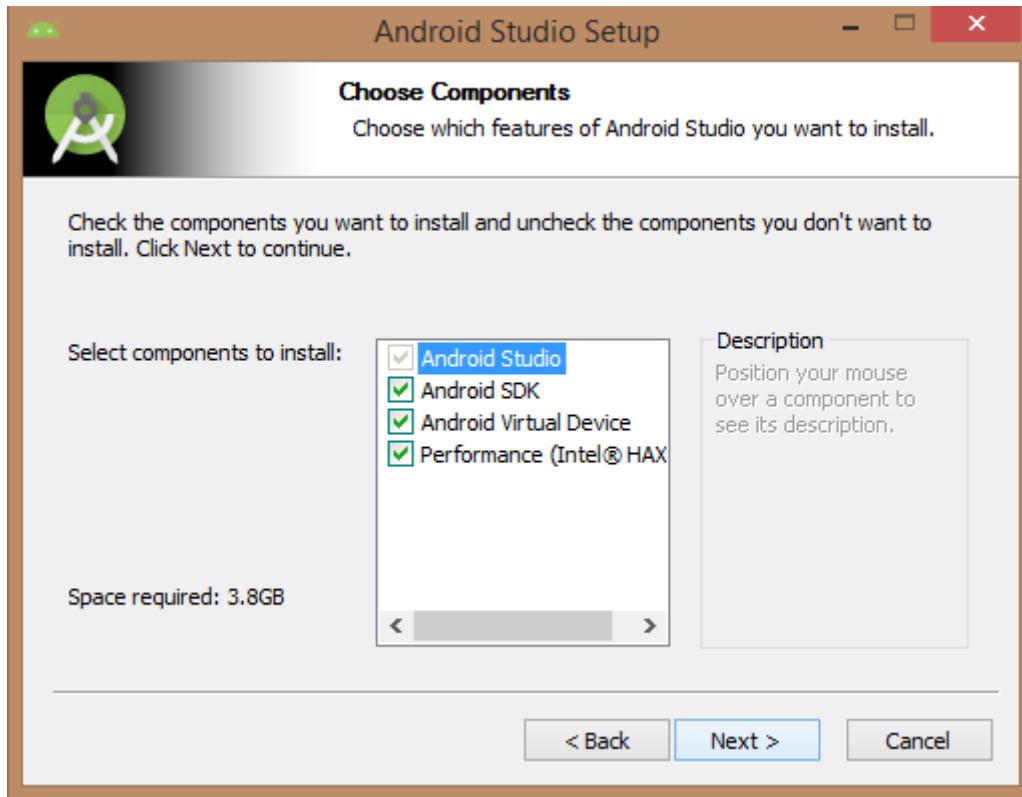
Once you launched Android Studio, its time to mention JDK7 path or later version in android studio installer.



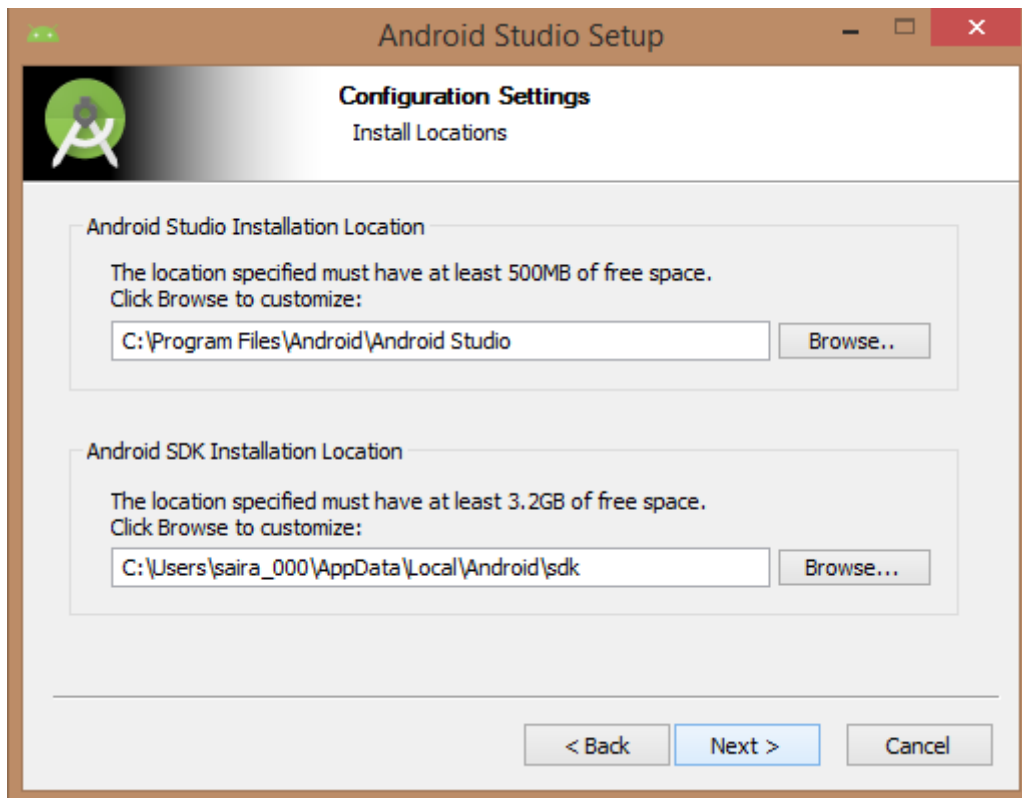
Below the image initiating JDK to android SDK



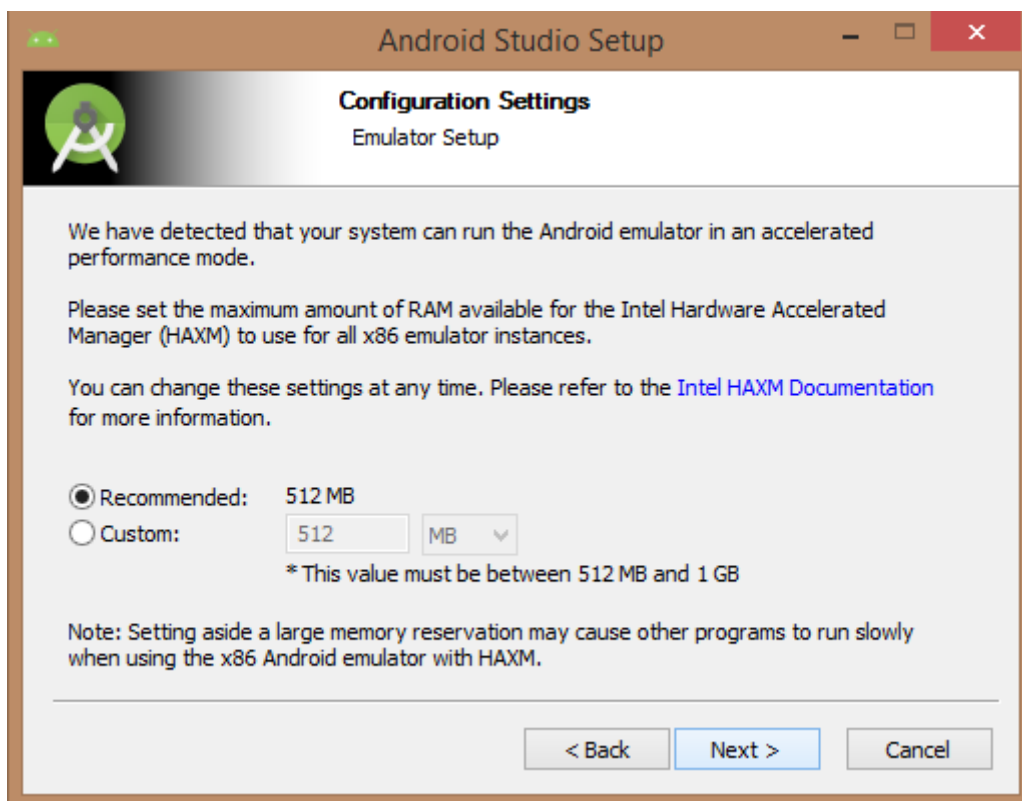
Need to check the components, which are required to create applications, below the image has selected **Android Studio**, **Android SDK**, **Android Virtual Machine** and **performance(Intel chip)**.



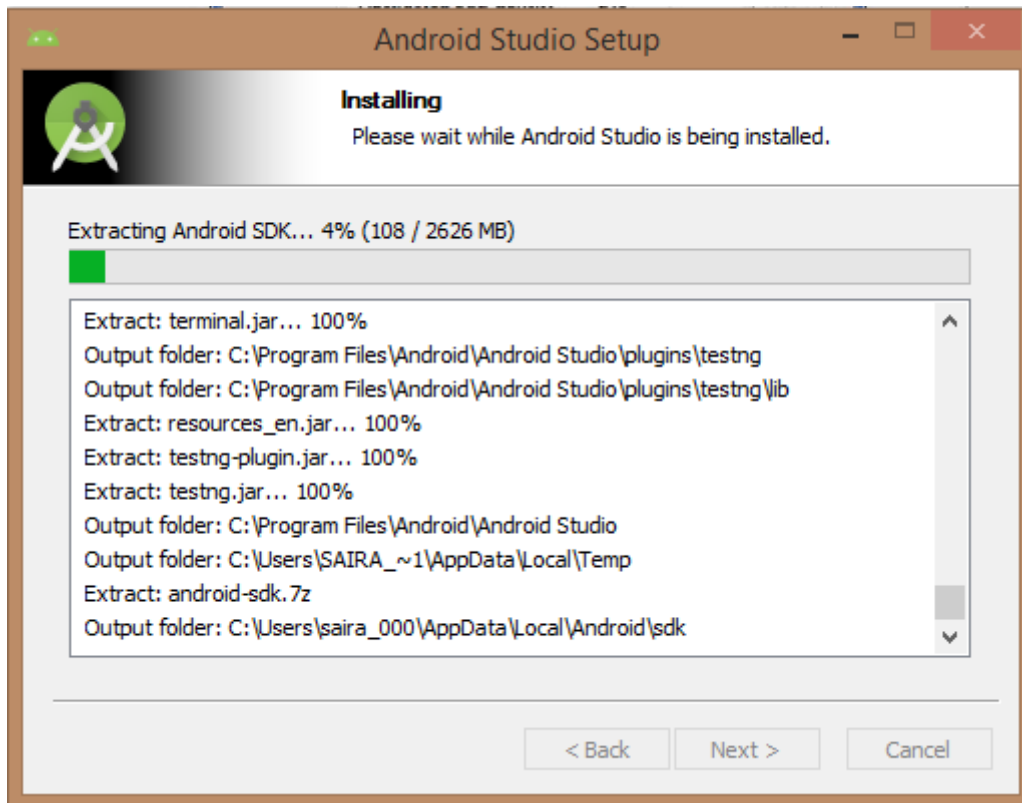
Need to specify the location of local machine path for Android studio and Android SDK, below the image has taken default location of windows 8.1 x64 bit architecture.



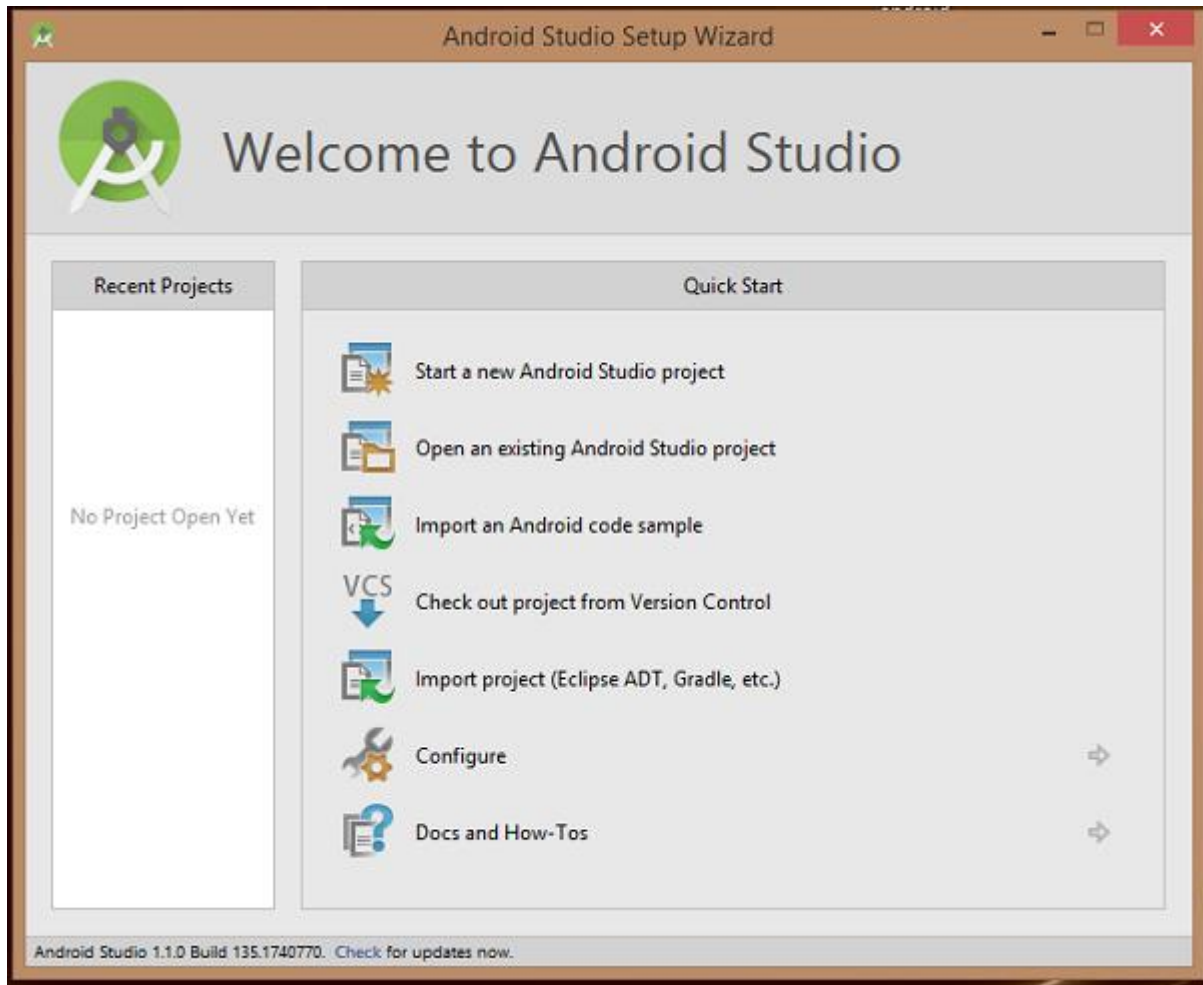
Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM.



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.

The screenshot shows the 'Create New Project' dialog in Android Studio. The title bar reads 'Create New Project'. The main header is 'New Project' with the Android Studio logo. Below this, the section is titled 'Configure your new project'. There are four input fields: 'Application name' (empty), 'Company Domain' (saira_000.example.com), 'Package name' (com.example.saira_000), and 'Project location' (C:\Users\saira_000\AndroidStudioProjects). A red error message at the bottom left says 'Please enter an application name (shown in launcher)'. At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

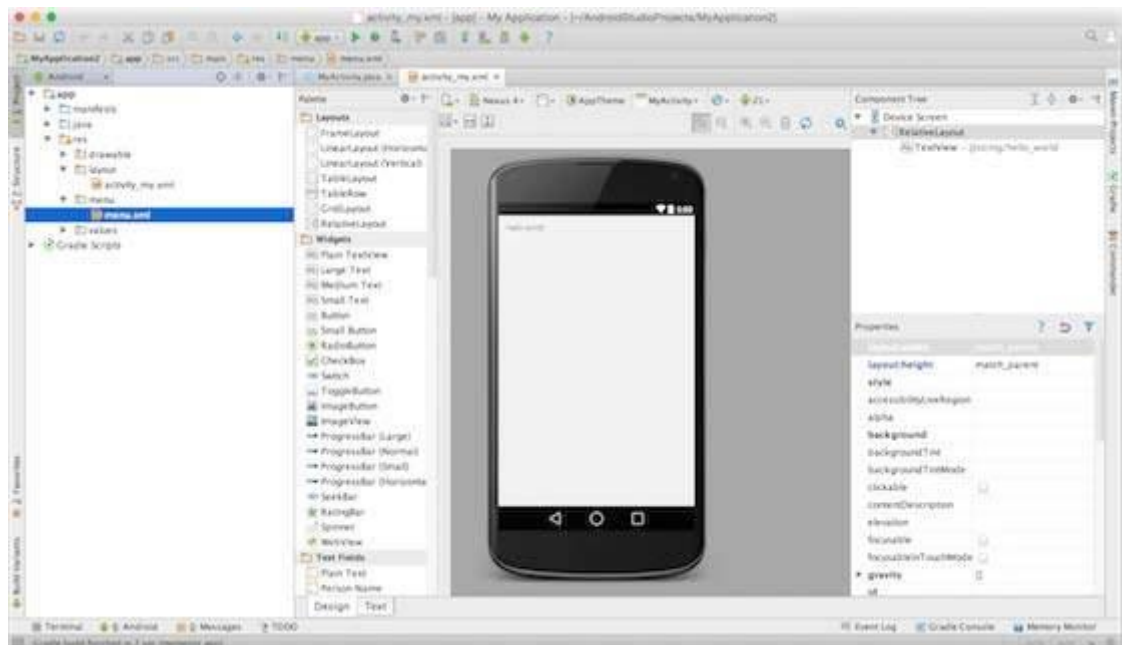
After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK, in our tutorial, I have declared as API23: Android 6.0(Mashmallow)

The screenshot shows the 'Create New Project' dialog in Android Studio, specifically the 'Target Android Devices' step. The title bar reads 'Create New Project'. The main header is 'Target Android Devices'. Below this, the section is titled 'Select the form factors your app will run on'. A subtitle reads 'Different platforms may require separate SDKs'. There are five radio button options: 'Phone and Tablet' (checked), 'Wear', 'TV', 'Android Auto', and 'Glass'. Each option has a 'Minimum SDK' dropdown menu. For 'Phone and Tablet', the dropdown is set to 'API 23: Android 6.0 (Marshmallow)'. Below this dropdown, there is a note: 'Lower API levels target more devices, but have fewer features available. By targeting API 23 and later, your app will run on approximately 4.7% of the devices that are active on the Google Play Store.' and a link 'Help me choose'. At the bottom right, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

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

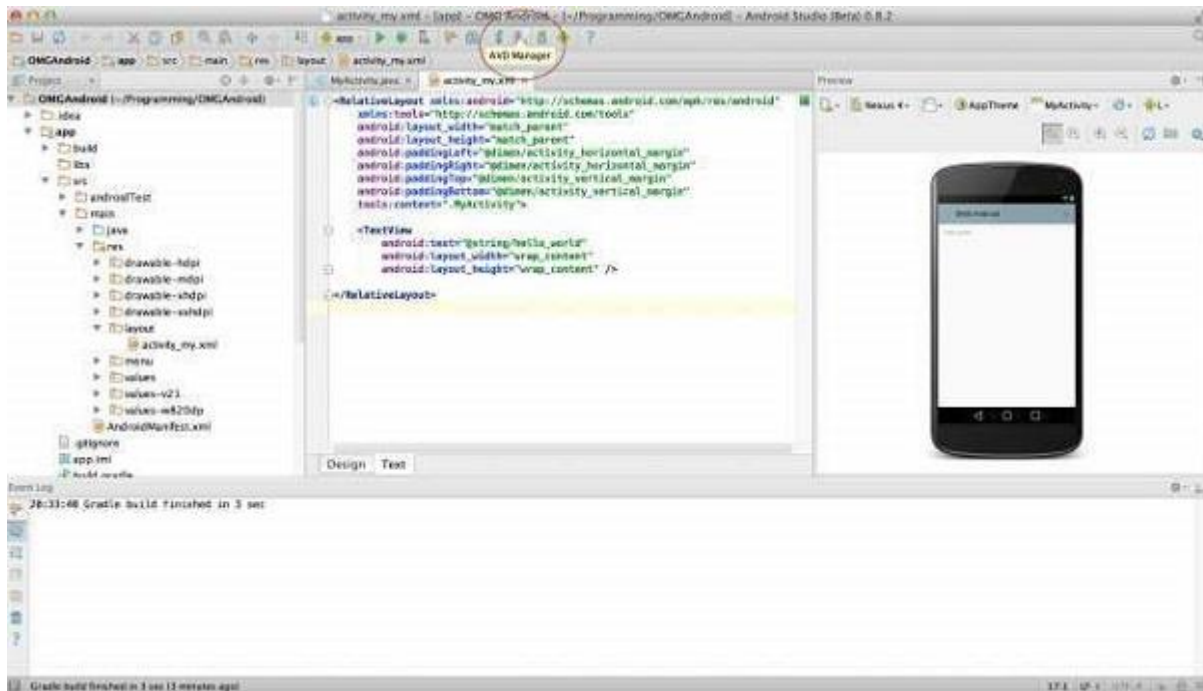


At the final stage it going to be open development tool to write the application code.

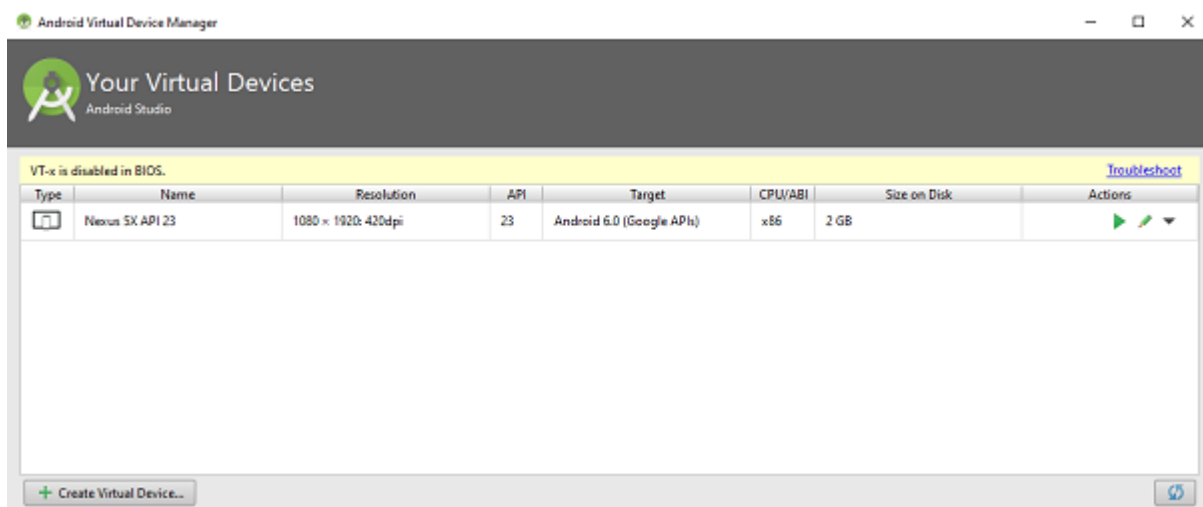


Step 3 - Create Android Virtual Device

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



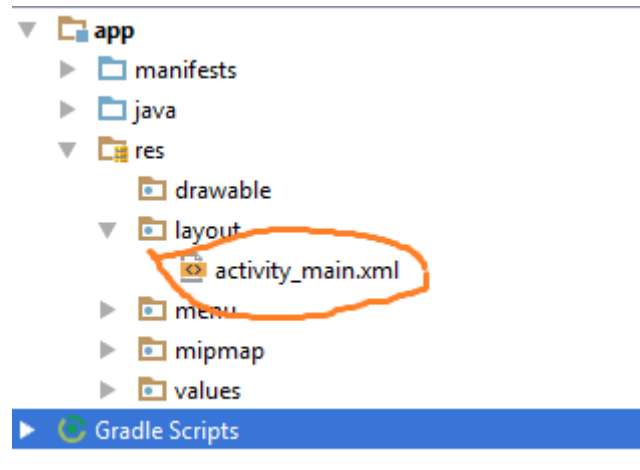
After Click on a virtual device icon, it going to be shown by default virtual devices which are present on your SDK, or else need to create a virtual device by clicking **Create new Virtual device** button



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).

```
<RelativeLayout
xmlns: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">

    <TextView android: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 below

