LAB : 1

Title: Android Introduction
Aim: Understand the android development environment.

Android is an open source and Linux-based operating system for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Features of Android

1. Open source.
2. Increased Market.
3. Inter App integration.
4. Customizable etc.

Android Application
Many android applications are available in market. The popular categories are:

1. Games.
2. Communication.
3. Social media.
4. Weather.
5. Business etc.

Set-up Java Development Kit (JDK)

The latest version of Java JDK for download is available from Oracle's Java site − Java . Finally set PATH and JAVA_HOME environment variables to refer to the directory that contains **java** and **javac**, right-click on My Computer, select Properties, then Advanced, then Environment Variables. Then, update the PATH value and press the OK button.

Android IDEs

The IDEs are available to develop android application are as follows:

1. Eclipse IDE.
2. Android studio.

Creating and running an Android App in Android Studio: The app creation is showed in the steps 1 to 4.

Step1: Creating a New Android Project

The first step in the application development process is to create a new project within the Android Studio environment. Begin, therefore, by launching Android Studio so that the "Welcome to Android Studio" screen appears as illustrated in Figure 1:

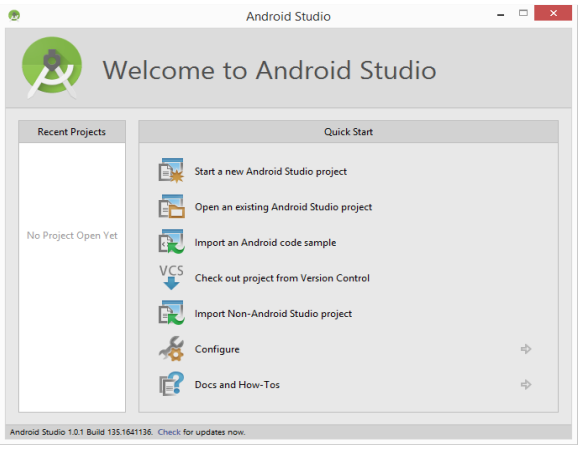

Figure 1

Once this window appears, Android Studio is ready for a new project to be created. To create the new project, simply click on the Start a new Android Studio project option to display the first screen of the New Project wizard as shown in Figure 2:

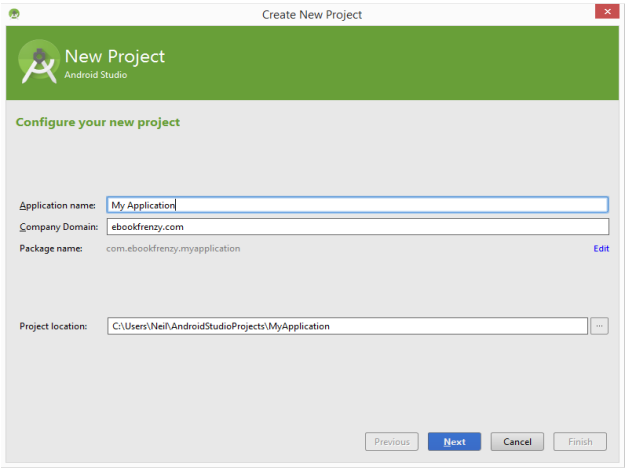

Figure 2

The Project location setting will be default to a location in the folder named AndroidStudioProjects located in the home directory and may be changed by clicking on the button to the right of the text field containing the current path setting.

Click Next to proceed. On the form factors screen, enable the Phone and Tablet option and set the minimum SDK setting to API 8: Android 2.2 (Froyo). The reason for selecting an older SDK release is that this ensures that the finished application will be able to run on the widest possible range of Android devices. The higher the minimum SDK selection, the more the application will be restricted to newer Android devices.

Step 2: Creating an Activity

The next step defines the type of initial activity that is to be created for the application. A range of different activity types is available when developing Android applications. select the option to create a Blank Activity.

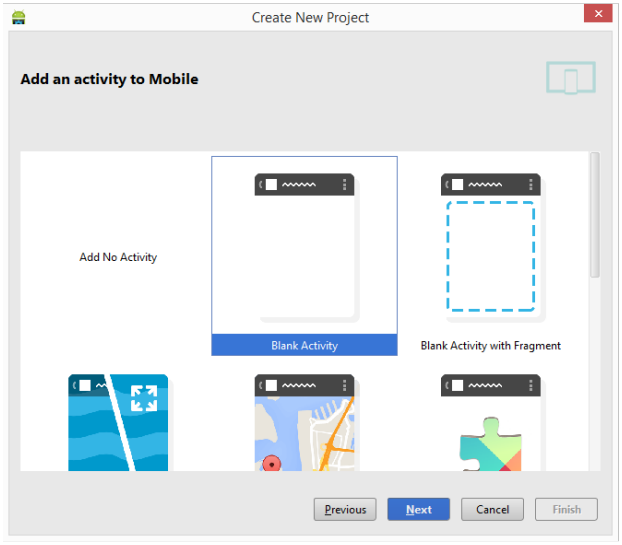

Figure 3

With the Blank Activity option selected, click Next. On the final screen (Figure 4) name the activity and title AndroidSampleActivity. The activity will consist of a single user interface screen layout which, for the purposes of this example, should be named activity_android_sample as shown in Figure 4 and with a menu resource named menu_android_sample.

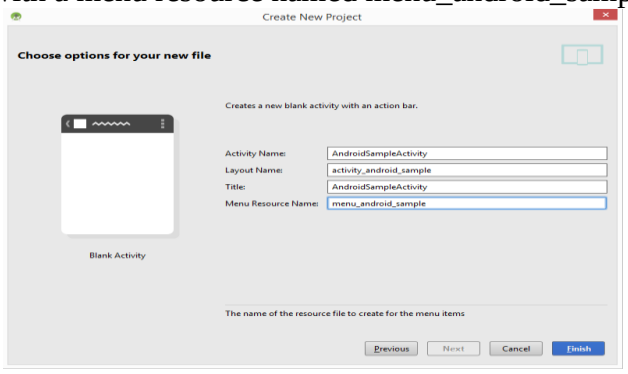

Figure 4

Finally, click on Finish to initiate the project creation process.

Step 3: Creating an Android Virtual Device (AVD) in Android Studio

In the course of developing Android apps in Android Studio it will be necessary to compile and run an application multiple times. An Android application may be tested by installing and running it either on a physical device or in an Android Virtual Device (AVD) emulator environment. Before an AVD can be used, it must first be created and configured to match the specification of a particular device model.

Step 3.1: Creating a New AVD

To create a new AVD, the first step is to launch the AVD Manager. This can be achieved from within the Android Studio environment by selecting the Tools -> Android -> AVD Manager menu option from within the main window.

Once launched, the tool will appear as outlined in Figure 5. Assuming a new Android SDK installation, no AVDs will currently be listed:

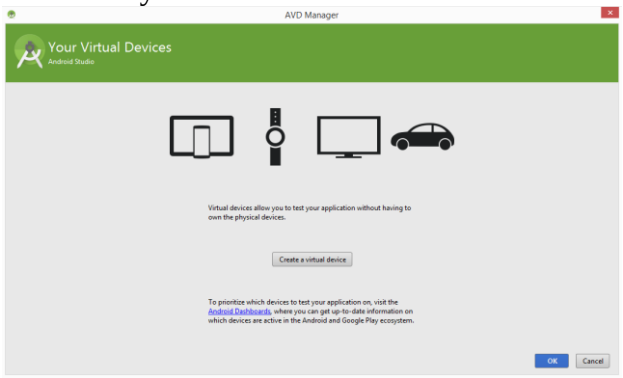

Figure 5

Begin the AVD creation process by clicking on the Create a virtual device button in order to invoke the Virtual Device Configuration dialog:

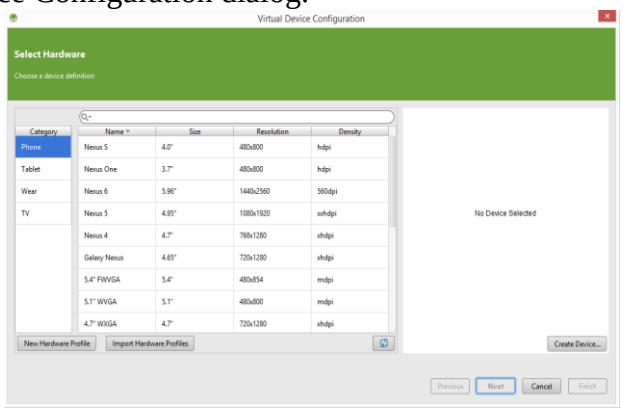

Figure 6

Step 3.2: Within the dialog, perform the following steps to create a first generation Nexus 7 compatible emulator:

1. From the Category panel, select the Tablet option to display the list of available Android tablet AVD templates.
2. Select the Nexus 7 (2012) device option and click Next.
3. On the System Image screen, select the latest version of Android (at time of writing this is Android 5.0.1, Lollipop API level 21) for the armeabi-v7a ABI. Click Next to proceed.
4. Enter a descriptive name (for example Nexus 7) into the name field.
5. Click Finish to create the AVD.

With the AVD created, the AVD Manager may now be closed. If future modifications to the AVD are necessary, simply re-open the AVD Manager, select the AVD from the list and click on the pencil icon in the Actions column of the device row in the AVD Manager.

Step 4: Running the Application in the AVD

With an AVD emulator configured, the example AndroidSample application created now can be compiled and run. With the AndroidSample project loaded into Android Studio, click on the run button represented by a green triangle located in the Android Studio toolbar as shown in Figure 7, select the Run -> Run… menu option or use the Shift+F10 keyboard shortcut.

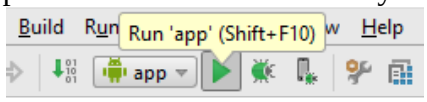

Figure 7

By default, Android Studio will respond to the run request by displaying the Choose Device dialog. This provides the option to execute the application on an AVD instance that is already running, or to launch a new AVD session specifically for this application. The step 3.2 will create Nexus7 AVD as a running device shown in Figure 8. With this device selected in the dialog, click on OK to install and run the application on the emulator.

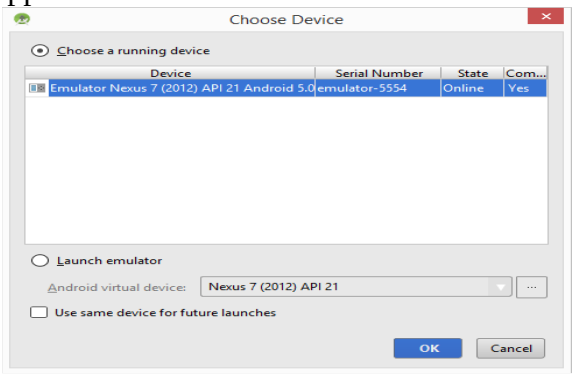

Figure 8

Once the application is installed and running, the user interface for the AndroidSampleActivity class will appear within the emulator.

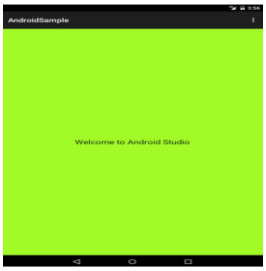

Figure 9

Android folder structure visible in the IDE/ Explorer is shown in figure 10.

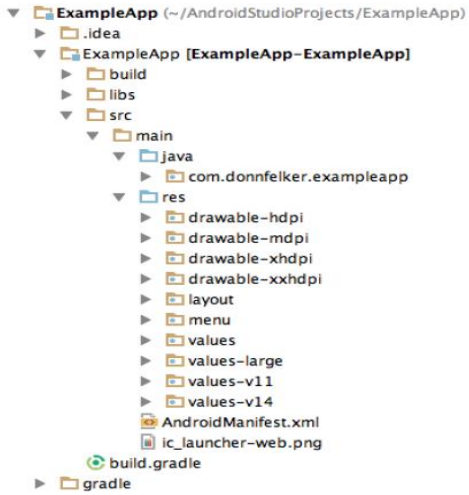

Figure 10

- **JAVA** contains the **.java** source files for your project. By default, it includes an MainActivity.java source file having an activity class that runs when your app is launched using the app icon.
- **res/drawable-hdpi** is a directory for drawable objects that are designed for high-density screens.
- **res/layout** is a directory for files that define your app's user interface.
- **res/values** is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
- **AndroidManifest.xml** is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

Android UI Layouts

The basic building block for user interface is a **View** object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

The **ViewGroup** is a subclass of **View** and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

At third level there are different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run

time using **View/ViewGroup** objects or can declare the layout using simple XML file **main_layout.xml** which is located in the res/layout folder of the project.

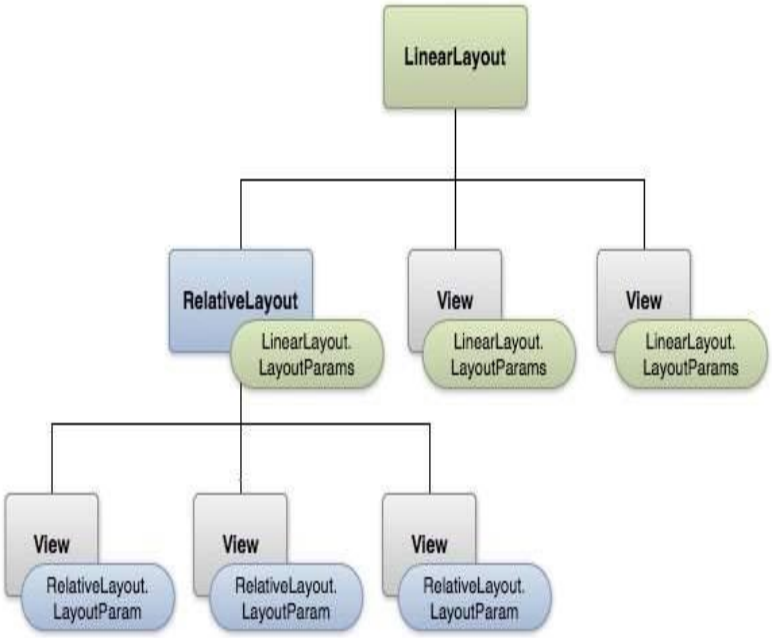

Figure 11

Following is an example of XML file having LinearLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a TextView" />

  <Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a Button" />

  <!-- More GUI components go here  -->

</LinearLayout>
```

Once the layout has been created, the layout resource can be loaded from the application code, in *Activity.onCreate()* callback implementation the onCreate() implementation is given below.

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
}
```

Android Layout Types

- LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
- RelativeLayout is a view group that displays child views in relative positions.
- TableLayout is a view that groups views into rows and columns.
- AbsoluteLayout supports specifying exact location of its children.
- The FrameLayout is a placeholder on screen that can be used to display a single view.
- ListView is a view group that displays a list of scrollable items.
- GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

Exercise questions

1. Write a program to create a simple user profile which includes user's professional details.
2. Write a program to create a registration form :
   a. Form includes email id, mobile number and password as its field.
   b. Perform validation for email id and mobile number.
   c. Display customized toast message based on user input.

Additional question

1. Write a program to display the following table.