Unit-2: Introduction to Android Programming

Syllabus:

Unit -2

Introduction to Android Programming [4 HRS]

Android Platform, History of Android, Environment Setup, Creating an android project, Laying out the user interface (The view hierarchy, widget attributes, creating string resources, previewing the layout), Creating a new class, Setting up the project, Running on the Emulator.

Android Platform:

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.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU (*General Public License*) version 2.

Android is an open development platform. However, it is not open in the sense that everyone can contribute while a version is under development. This is all done behind closed-doors at Google. Rather, the openness of Android starts when its source code is released to the public after it is finalized. This means once it is released anyone interested can take the code and alter it as they see fit.

To create an application for the platform, a developer requires the Android SDK, which includes tools and APIs. To shorten development time, Android developers typically integrate the SDK into graphical user IDEs (Integrated Development Environments). Beginners can also make use of the App Inventor, an application for creating Android apps that can be accessed online.

Lecturer: Teksan Gharti Magar

Why Android?

Features of Android:

Android is a powerful operating system competing with Apple iOS and supports great features. Few of them are listed below

SN	Feature & Description
1	**Beautiful UI:** Android OS basic screen provides a beautiful and intuitive (*easy to understand*) user interface.
2	**Connectivity:** GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
3	**Storage:** SQLite, a lightweight relational database, is used for data storage purposes.

4	**Media support:** H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
5	**Messaging:** SMS and MMS
6	**Web browser:** Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
7	**Multi-touch:** Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
8	**Multi-tasking:** User can jump from one task to another and same time various applications can run simultaneously.
9	**Resizable widgets:** Widgets are resizable, so users can expand them to show more content or shrink them to save space.
10	**Multi-Language:** Supports single direction and bi-directional text.
11	**GCM:** Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices.
12	**Wi-Fi Direct:** A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
13	**Android Beam:** A popular NFC (*Near Field Communications*) based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Android Applications:

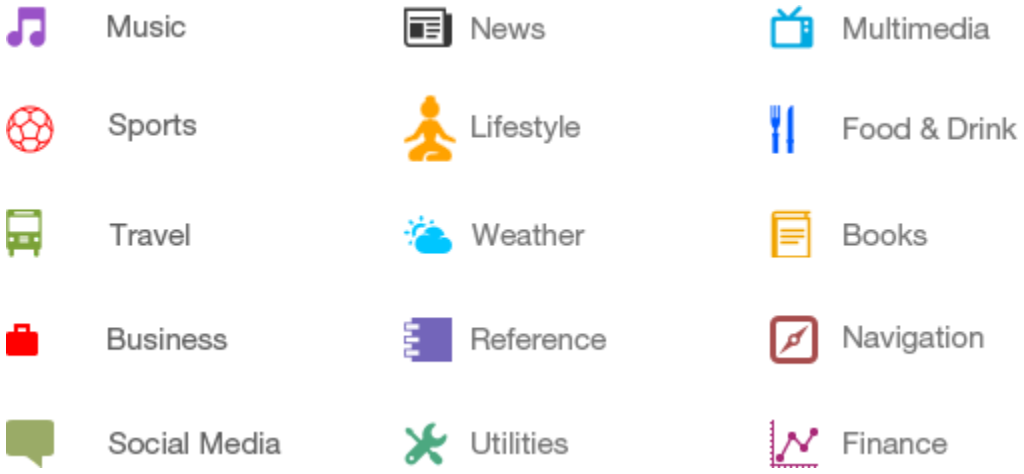
Android applications are usually developed in the Java language using the Android Software Development Kit.

Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid and the Amazon Appstore.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

Categories of Android applications:

There are many android applications in the market. The top categories are −



What is an API?

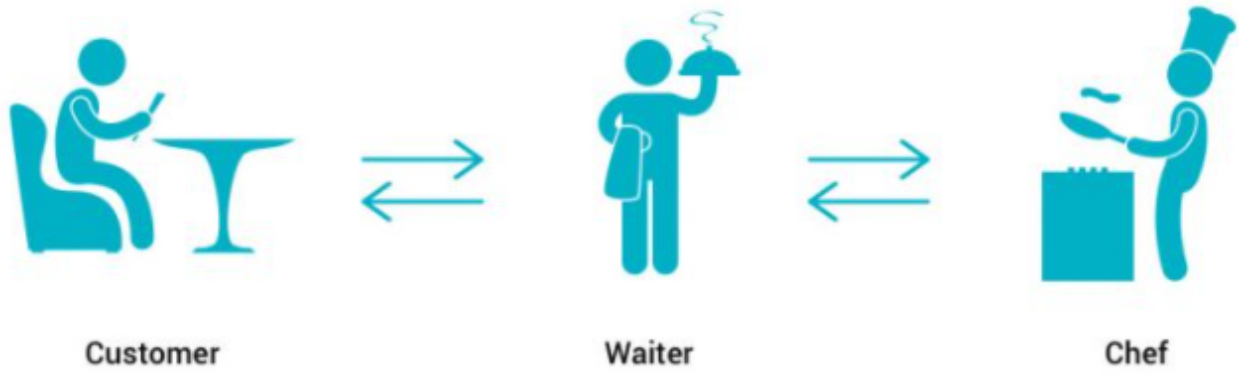
An Application Programming Interface (API) is a particular set of rules ('code') and specifications that programs can follow to communicate with each other. APIs are growing exponentially every year.

The world of software moves fast. In earlier days data entered and processed in the same system, but now the origin of the data and processing place is entirely different. We should be able to access the data from anywhere at any time, that's why we store this data in cloud storage.

For sending and receiving data from/to the server, we want a middle man who is platform independent. That middleman handles the requests and serves the response to the user that middleman is API. The below diagram illustrates this better than words.

The End user sends a request; API executes the instruction then gets the data from the server and respond to the user.

Let's make it easy to understand, think about it this way: You are at a restaurant and you want to eat something, place your order to the waiter he conveys the order to the kitchen and serves it to you once it ready. Here the waiter is act as an API. Customer may change but the waiter serves them in the same manner. The following diagram explains it simply.



History of Android:

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, Marshmallow, Nougat, Oreo, and Pie. Let's understand the android history in a sequence.

- Initially Andy Rubin founded Android Incorporation in Palo Alto, California, US, in October, 2003.
- In 17th August, 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.
- The key employees of Android Incorporation are ***Andy Rubin, Rich Miner, Chris White,*** and ***Nick Sears.***
- Android OS is originally intended for camera but shifted to smart phones later because of low market for camera only.
- Android is the nick name of Andy Rubin given by his co-workers because of his love to robots.
- In 2007, Google announced the development of Android OS.
- In 2008, HTC launched the first Android Mobile.

Table below lists codenames, their corresponding version numbers, associated API levels and NDK (*Native Development Kit*) releases.

Name	**Internal codename**	**Version number(s)**	**API level NDK release**	**Initial stable release date**
Android 1.0	N/A	1.0	1	September 23, 2008
Android 1.1	Petit Four	1.1	2	February 9, 2009
Android Cupcake	Cupcake	1.5	3, NDK 1	April 27, 2009
Android Donut	Donut	1.6	4, NDK 2	September 15, 2009
Android Eclair	Eclair	2.0	5	October 27, 2009
		2.0.1	6	December 3, 2009
		2.1	7, NDK 3	January 11, 2010
Android Froyo	Froyo	2.2 – 2.2.3	8, NDK 4	May 20, 2010
Android Gingerbread	Gingerbread	2.3 – 2.3.2	9, NDK 5	December 6, 2010
		2.3.3 – 2.3.7	10	February 9, 2011
Android Honeycomb	Honeycomb	3.0	11	February 22, 2011
		3.1	12, NDK 6	May 10, 2011
		3.2 – 3.2.6	13	July 15, 2011
Android Ice Cream Sandwich	Ice Cream Sandwich	4.0 – 4.0.2	14, NKD 7	October 18, 2011
		4.0.3 – 4.0.4	15, NDK 8	December 16, 2011
Android Jelly Bean	Jelly Bean	4.1 – 4.1.2	16	July 9, 2012
		4.2 – 4.2.2	17	November 13, 2012

		4.3 – 4.3.1	18	July 24, 2013
Android KitKat	Key Lime Pie	4.4 – 4.4.4	19	October 31, 2013
		4.4W – 4.4W.2	20	June 25, 2014
Android Lollipop	Lemon Meringue Pie	5.0 – 5.0.2	21	November 4, 2014
		5.1 – 5.1.1	22	March 2, 2015
Android Marshmallow	Macadamia Nut Cookie	6.0 – 6.0.1	23	October 2, 2015
Android Nougat	New York Cheesecake	7.0	24	August 22, 2016
		7.1 – 7.1.2	25	October 4, 2016
Android Oreo	Oatmeal Cookie	8.0	26	August 21, 2017
		8.1	27	December 5, 2017
Android Pie	Pistachio Ice Cream	9	28	August 6, 2018
Android 10	Quince Tart	10	29	September 3, 2019
Android 11	Red Velvet Cake	11	30	September 8, 2020
Android 12	Snow Cone	12	31	October 4, 2021
Android 12L	Snow Cone v2	TBA	32	Q1 2022

Lecturer: Teksan Gharti Magar

Environment Setup:

Whether you're building an Android application from scratch, you must set up the Android Software Development Kit (SDK) before you can build and run any code on your Android device.

First, to start develop your Android application you must have PC with either of the following operating systems

- Microsoft Windows XP or Later Version
- Mac OS X 10.5.8 or Later Version with Intel chip
- Linux including GNU C Library 2.7 or Later Version

Second, some software tools are required which are freely available and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

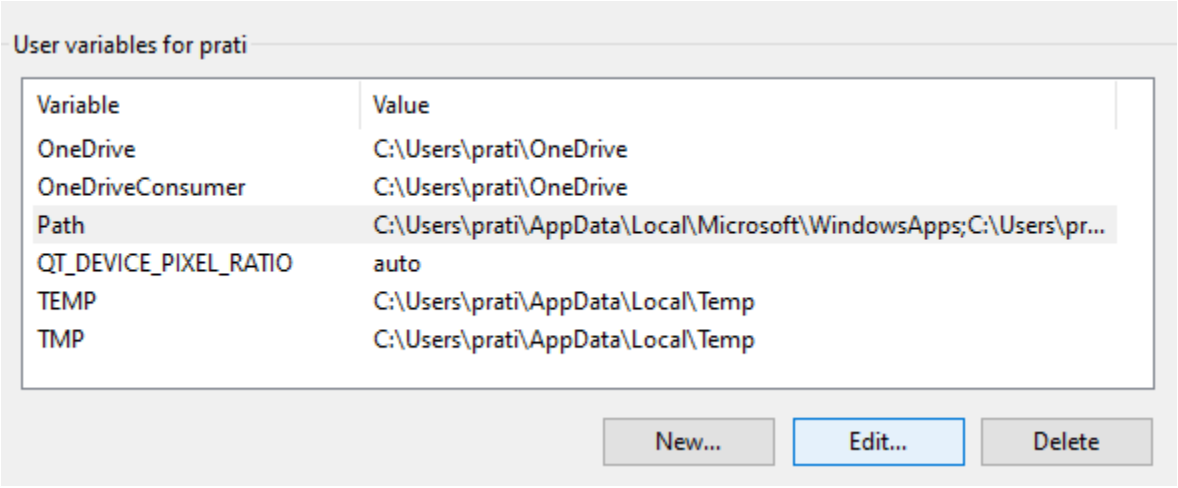
- Java JDK5 or later version
- Android Studio

Following are the procedure for installation of required software tools for Android Programming.

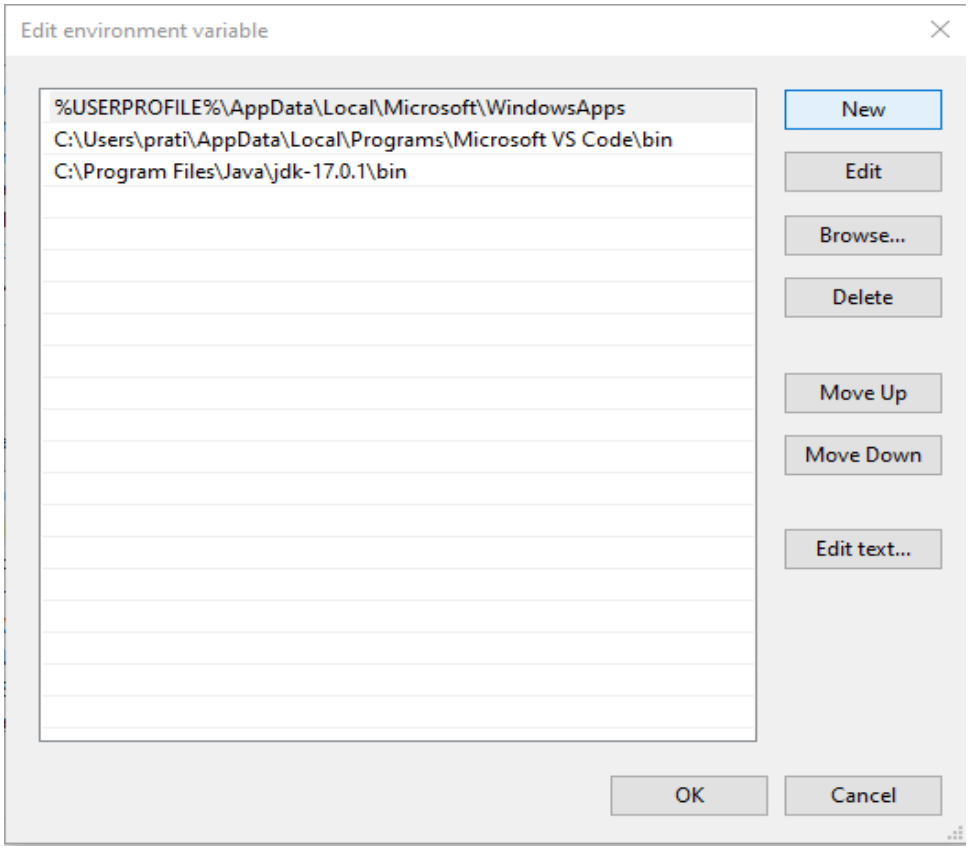
Step 1: First of all download and install Java Development Kit (JDK) from the link https://www.oracle.com/java/technologies/downloads/#jdk17-windows

Step 2: After installing JDK, setup the environment variables in your PC , to do this Right Click on This PC (My Computer) -> Properties -> Advance System Settings -> Environment Variables, setup path and new environment variable as follows

In the User Variable, select Path and click on Edit button

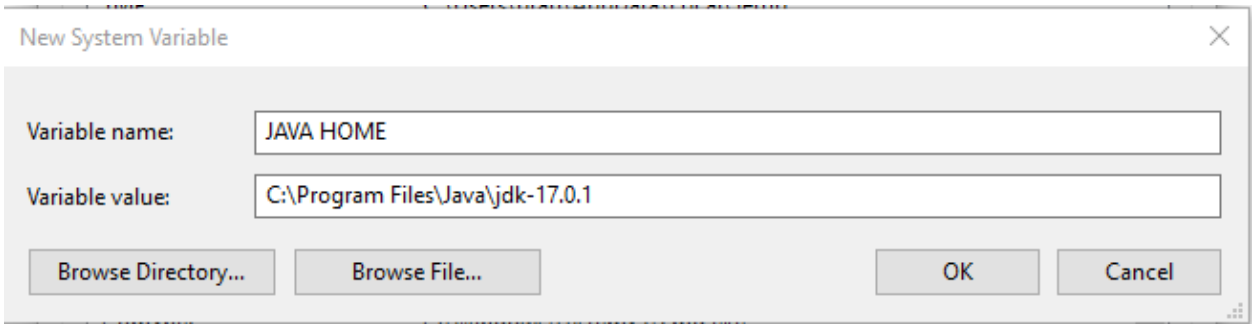


After clicking in Edit button following window will open



Here click on new button and Copy the path of JDK bin folder and paste here, and click Ok button.

After setting up Use Variable, setup the System Variable, to do this Click on New button of System Variable, follow window will open, and setup as following, then Click on Ok button.



Congratulations!! You completed installation and environment variable setup of JDK.

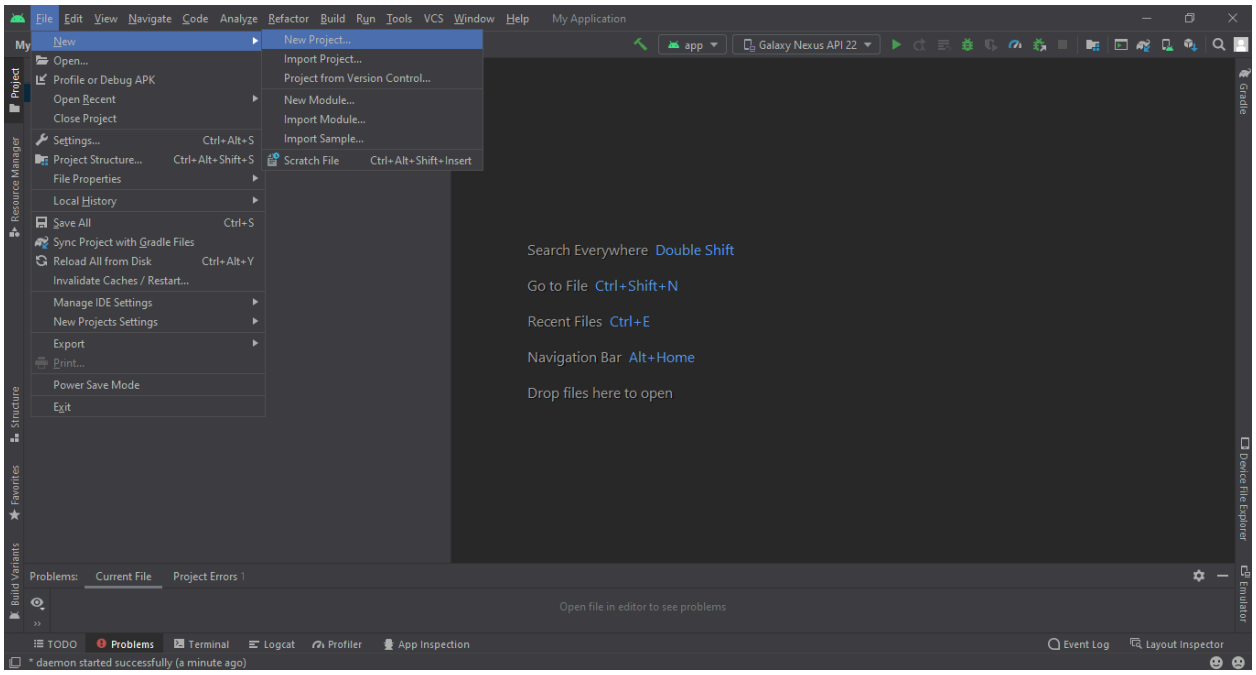
Step 3: Download and Install latest version of Android Studio from the link https://developer.android.com

Installation process is same as other software.

Creating an Android Project:

To create an android project, first of all start up the android studio and choose

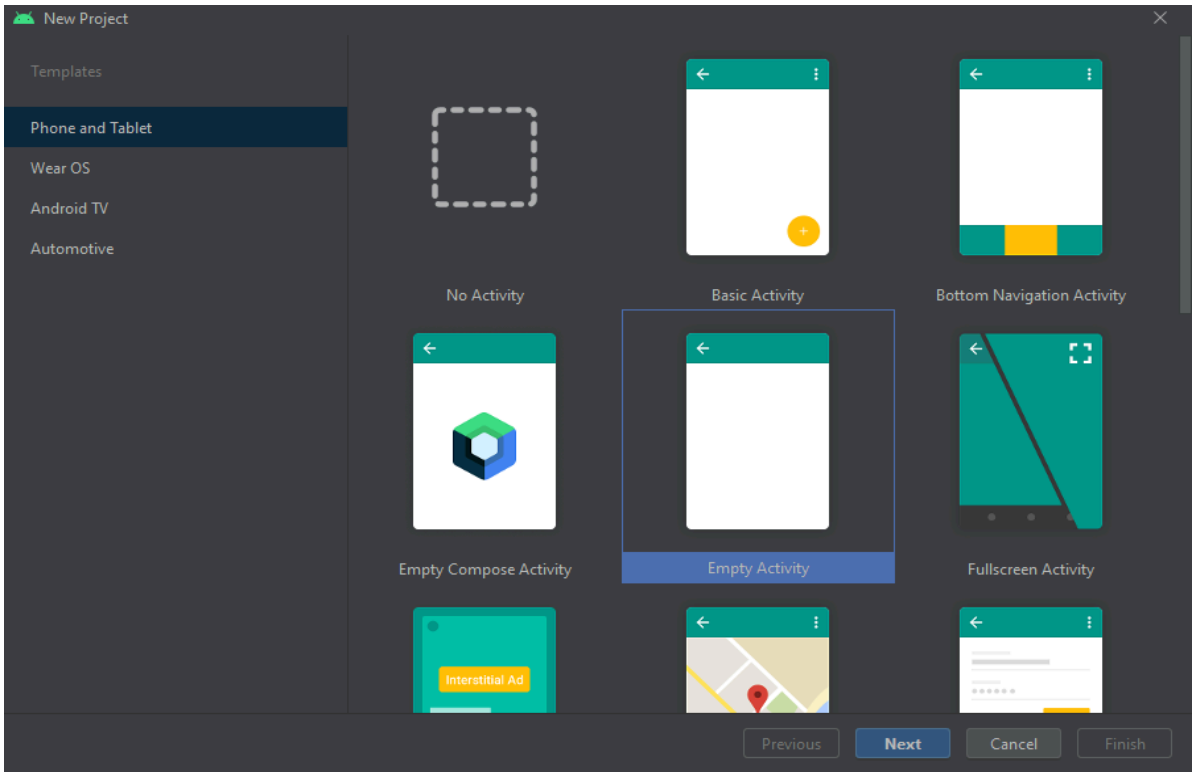
File -> New -> New Project



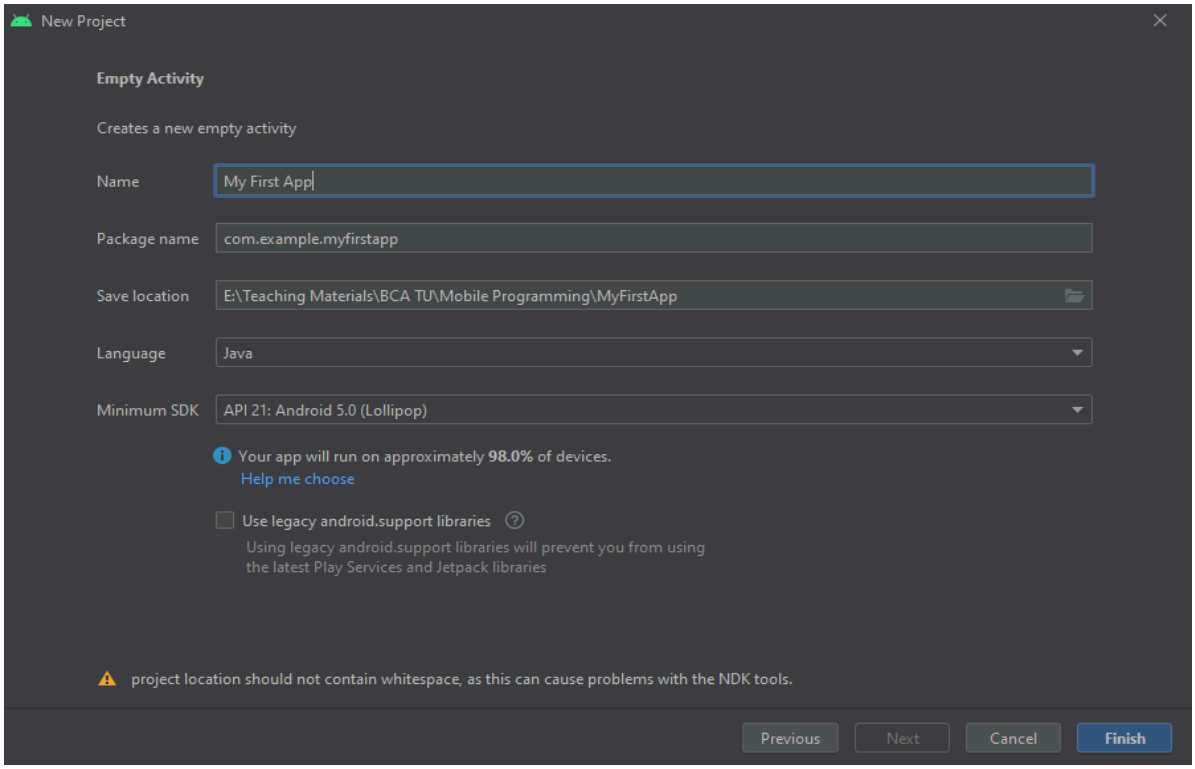
After choosing New Project, following window will open.

Inside the windows select Phone and Tablet. There will be many templates; every single template has their own application criteria.

Select any one of the template, generally select the Empty Activity template and click on Next button.



After clicking on Next button following window will open

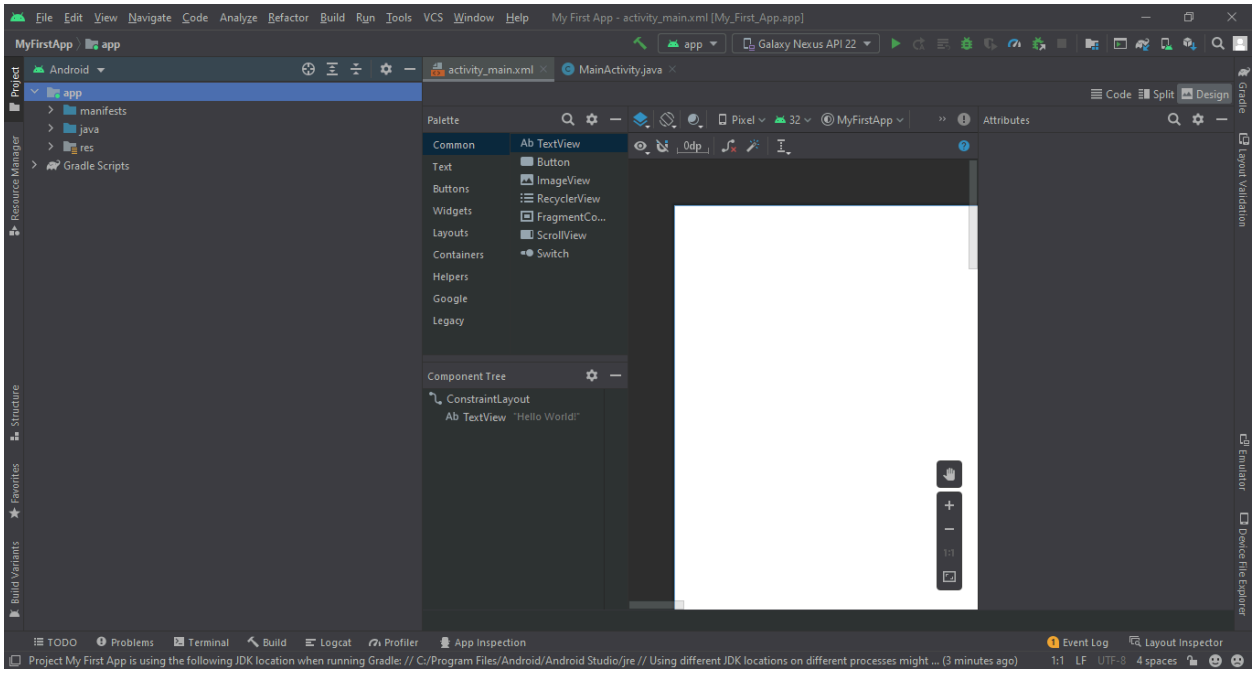


Lecturer: Teksan Gharti Magar

In the Name field enter your application name. Leave Package Name as default. Select the location to save your project. Select a programming language as Java or Kotlin. Select device with API level and Android Version.

Note: Select minimum API level to target maximum devices. Here in our application we select API level 21 which can run on 98% of android deveies.

After doing all, click on Finish button.



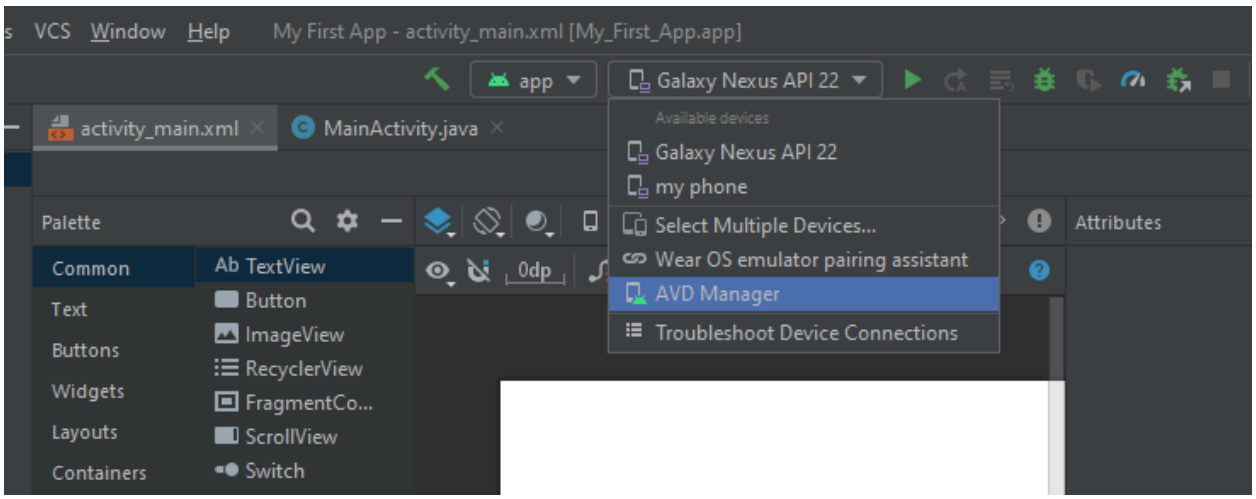
Congratulations!! You successfully created your first Android Application.

Now, make your best design for your application and write necessary logical code.

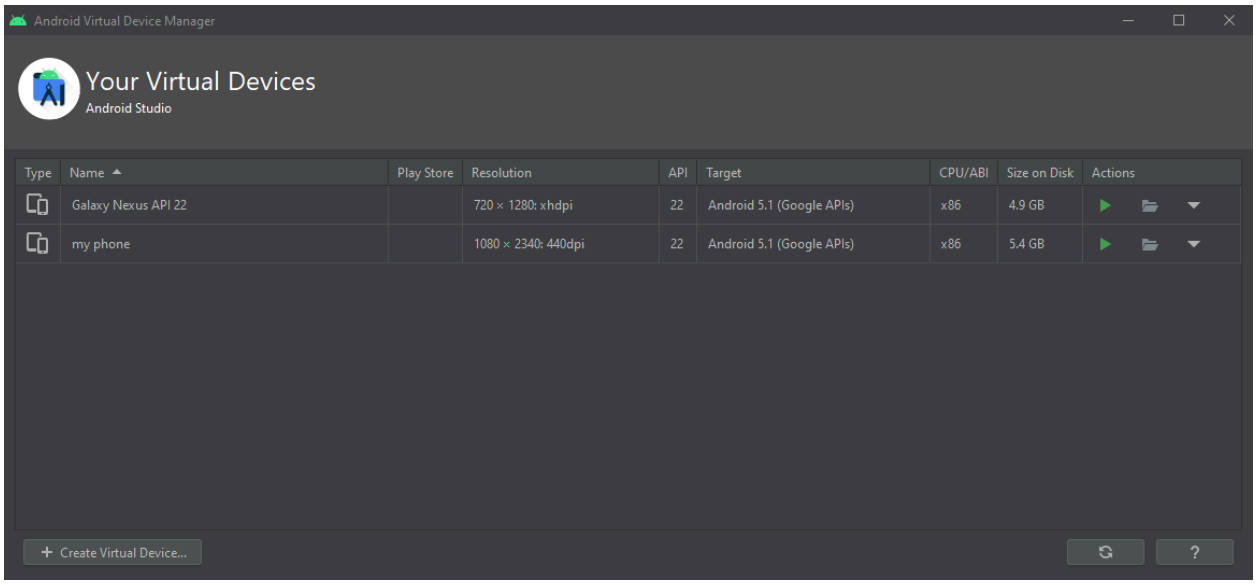
Creating a Virtual Device (Emulator):

Virtual device acts like physical mobile device to test the android application during development.

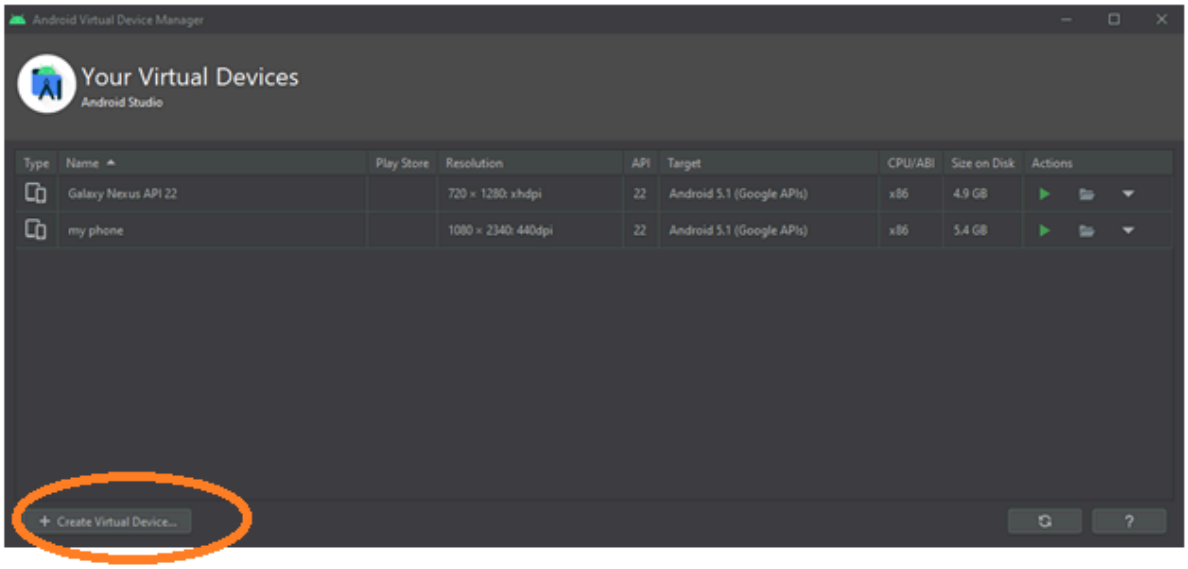
Following are the steps to create virtual device

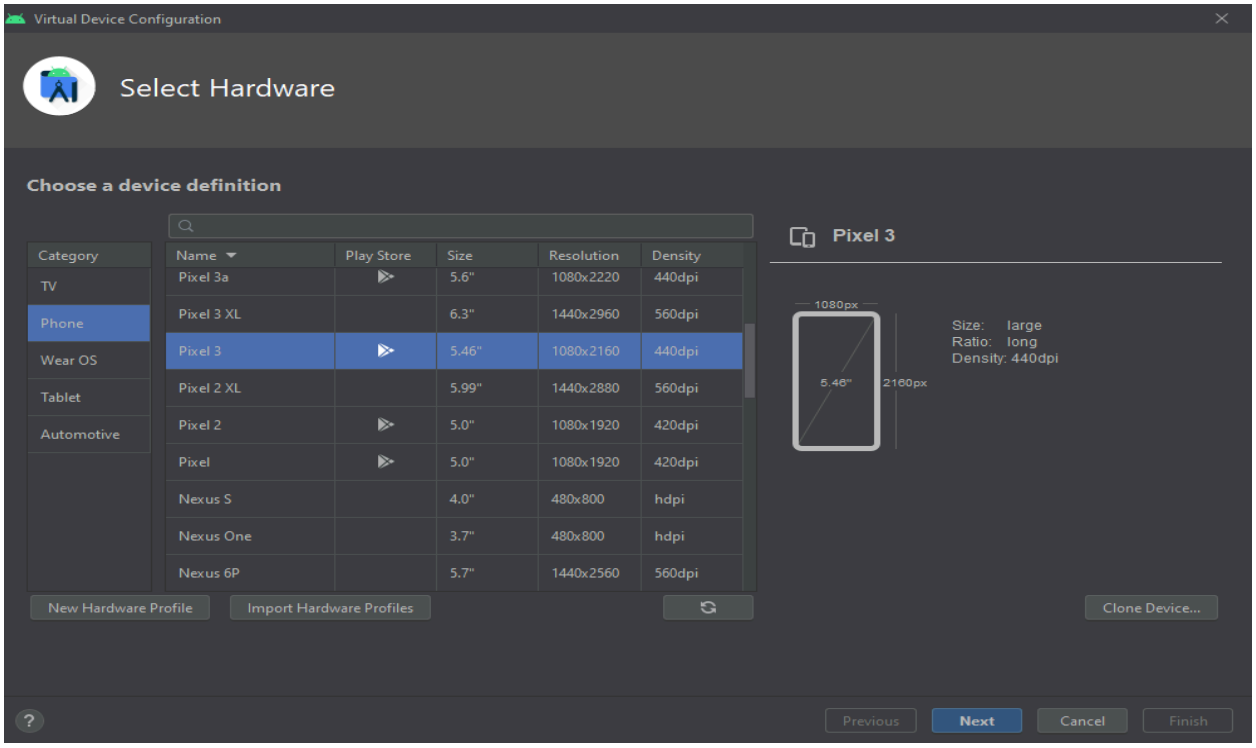


Select AVD Manager, following window will open

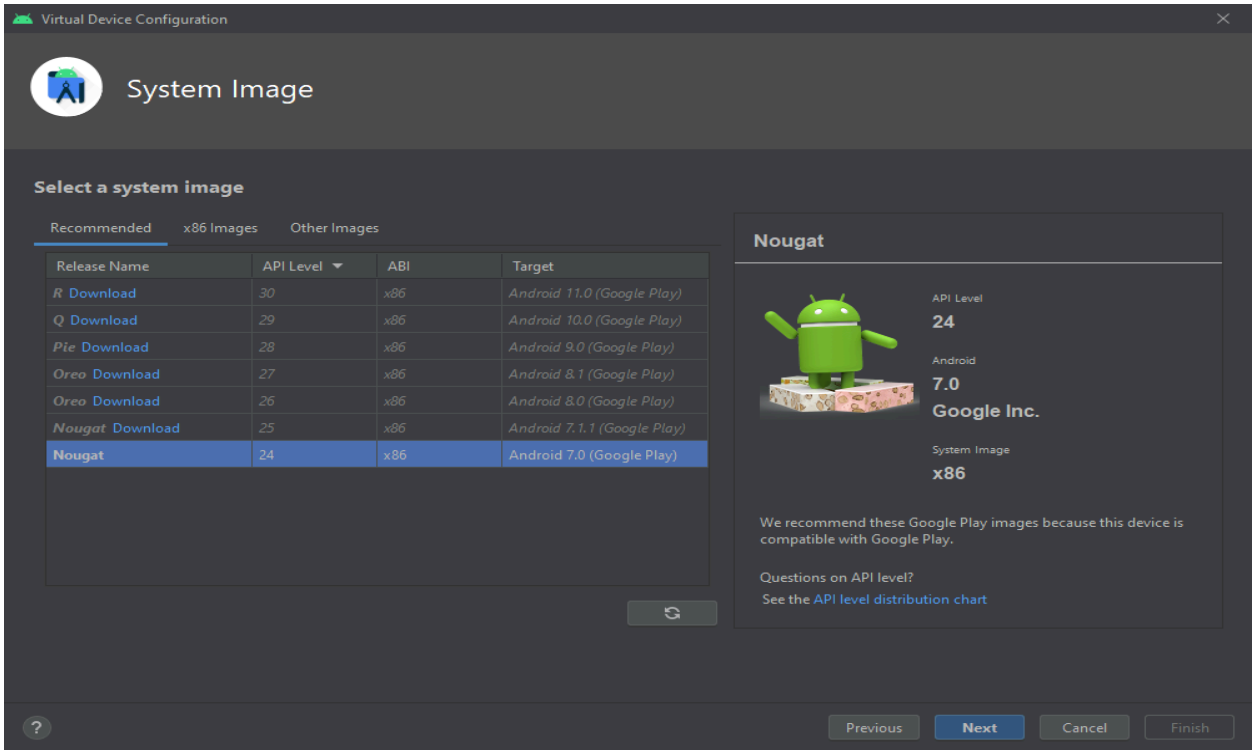


Click on Create Virtual Device button

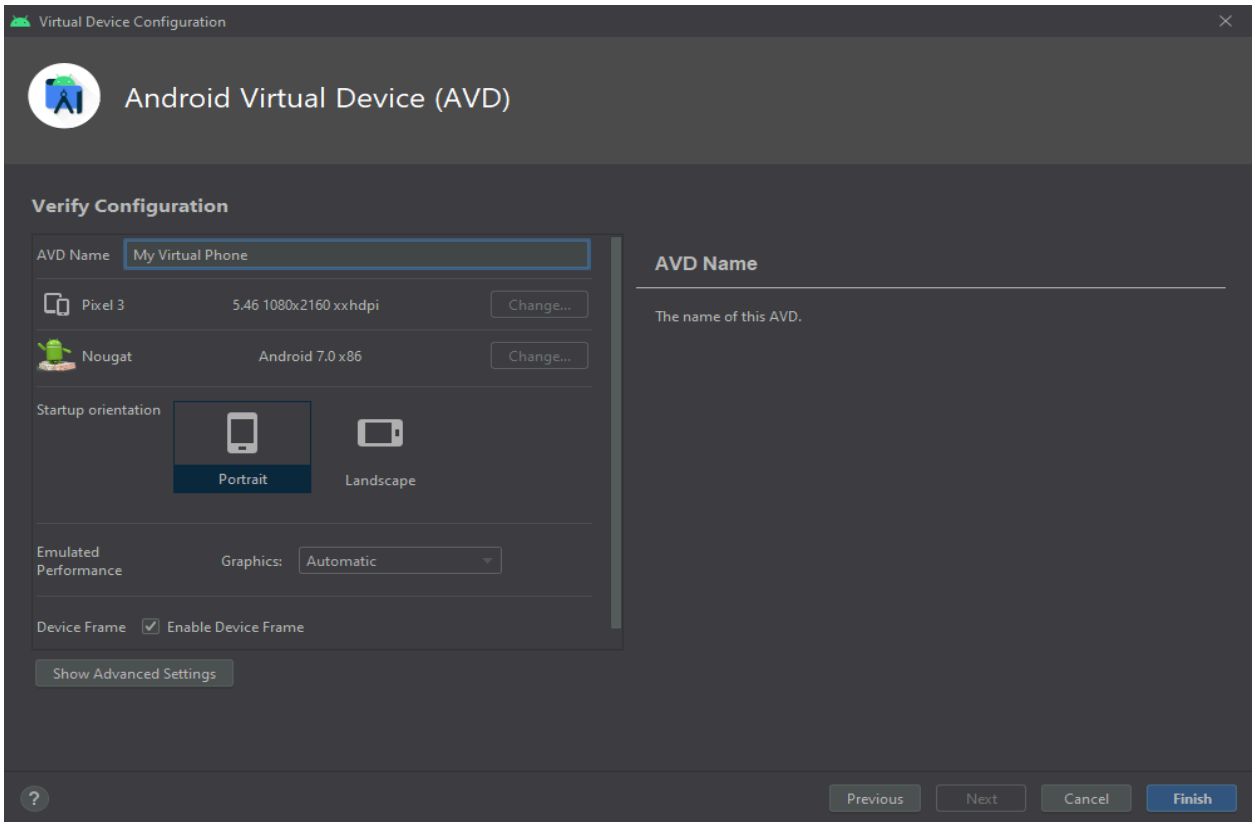




Select Phone Size and click on Next button. In the figure below select android OS version and API level. If required OS is not found download it.

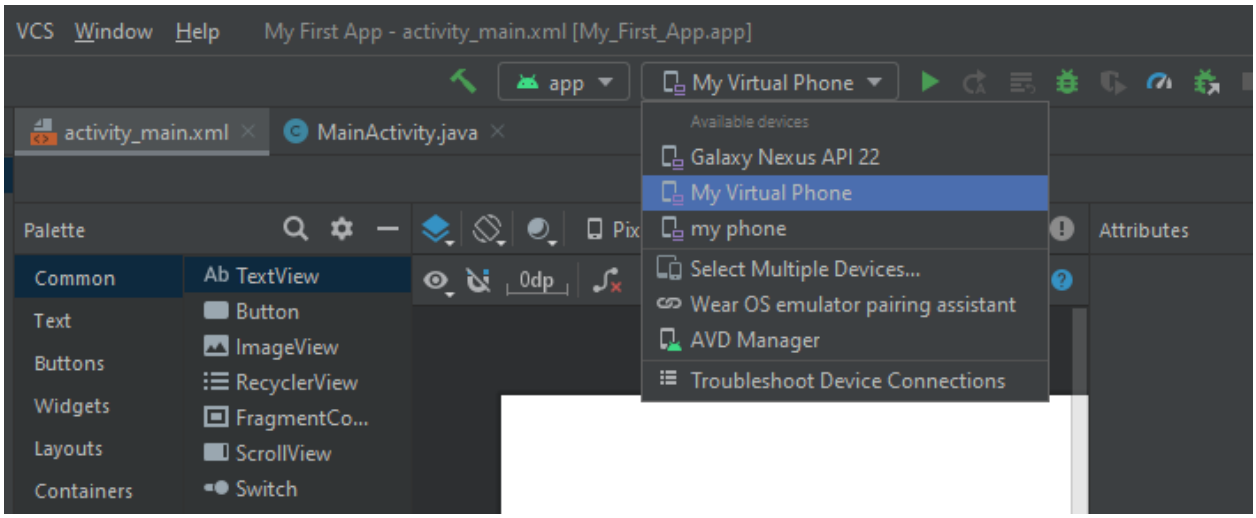


Enter your virtual device name and click on Finish button.



Lecturer: Teksan Gharti Magar

Now, select your virtual device while run the application.

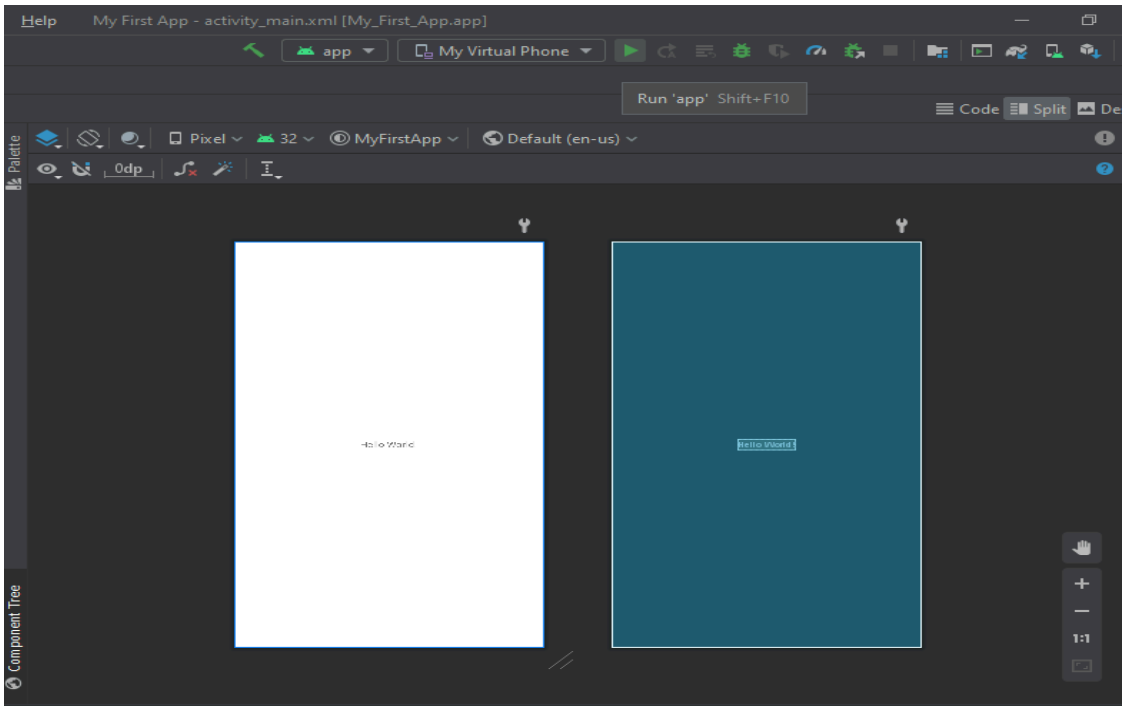


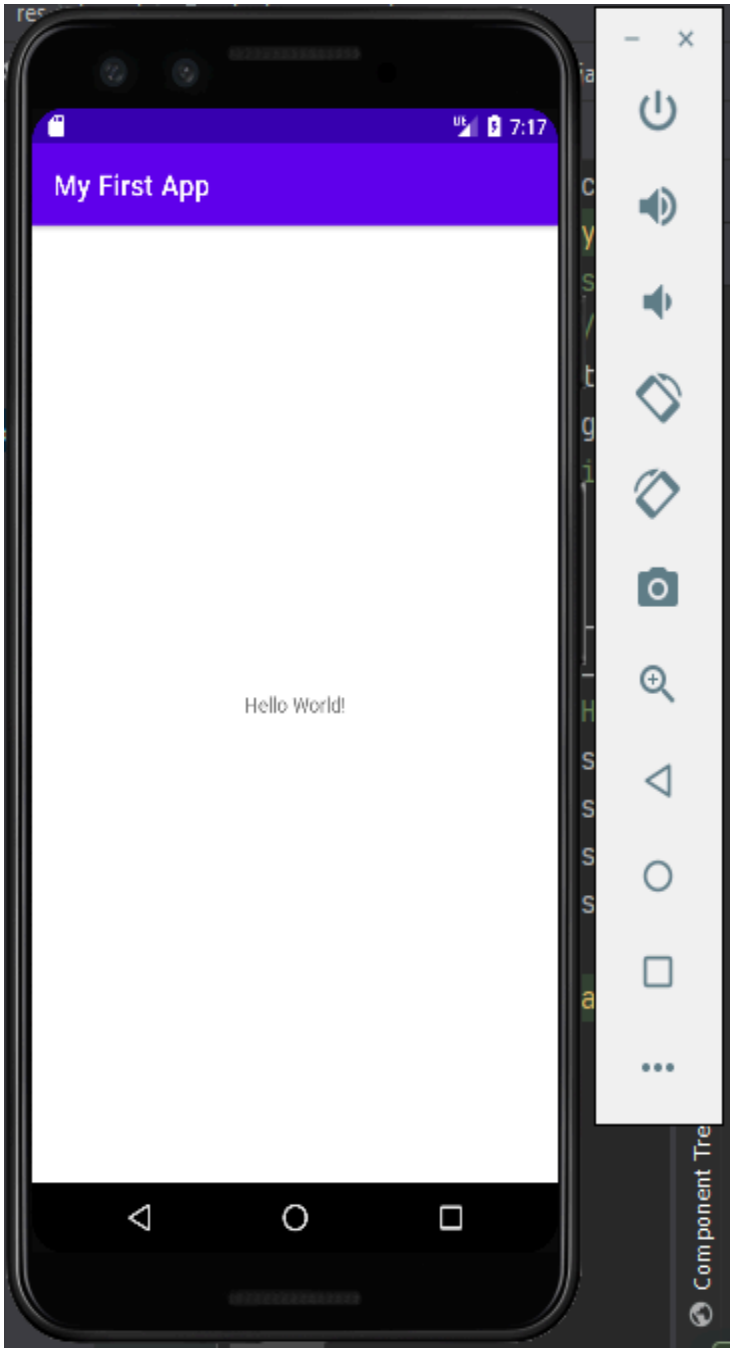
Display "Hello World!" in the virtual device:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Select your target virtual device and run by pressing **Shift+F10** keys or click on run icon. It will take some time to build Gradle and start emulator.





Congratulations!! Your first mobile app named My First App is lunched successfully in your virtual device showing text “Hello World!”.

Now we can customize this app according to our requirements.

Lecturer: Teksan Gharti Magar

Laying Out the User Interface:

After creating a new project, firstly we design user interface for our project. In android project we can design UI using xml file. In android studio all xml files, images, audio, videos, animations, colors etc. that are used for designing purpose are kept inside the **res** (*Resources*) directory.

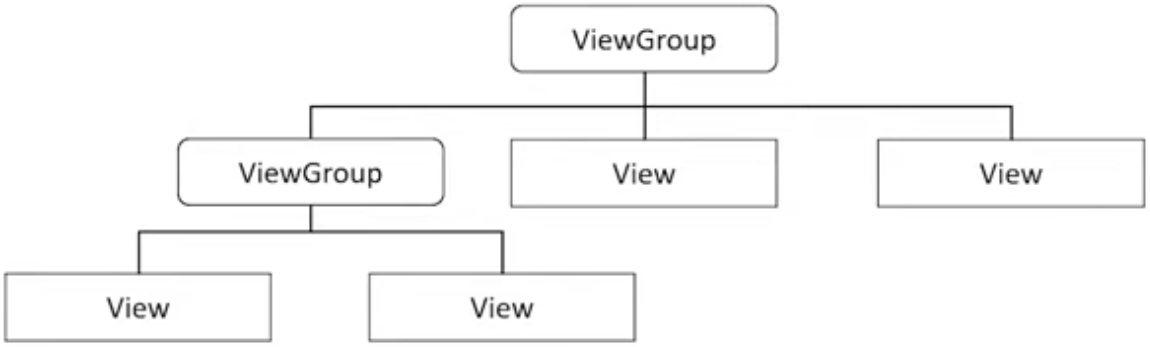
For better arrangement of project create a sub-directory named **layout** inside **res** directory to store all the layout related files. If **layout** directory is already exists then no need to create it.

View Hierarchy:

A layout defines the structure for a user interface in your app. Or in another words the layouts are ViewGroups objects, containers that control how their child views are positioned on the screen. All elements in the layout are built using a hierarchy of View and ViewGroup objects.

A ***View*** usually draws something the user can see and interact with so it is called as building block of application frontend that is UI.

A ViewGroup is a type of view which can contain other views. It is an invisible container that defines the layout structure for ***View*** and other ***ViewGroup*** objects, as shown in figure bellow.



View Hierarchy

View is the basic building block for user interface components. A View represents a rectangular area on the screen and is responsible for drawing and event handling. View is the superclass for all GUI components in Android, including layouts (which are a type of ViewGroup), buttons, text fields, and other widgets.

A widget is a term commonly used to refer to UI elements that are displayed on the screen. Widgets can range from simple components like buttons and text fields to complex containers that hold other widgets. Widgets are essentially views (View objects) that the user interacts with.

Hence, "***widgets***" are the ***View*** objects such as ***Button*** or ***TextView etc*** and "***layouts***" are usually The ***ViewGroup*** objects. They can be one of many types that provide a different layout structure, such as ***LinearLayout*** or ***ConstraintLayout***.

We can declare a layout in two ways:

- **Declare UI elements in XML:** Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
  You can also use Android Studio's Layout Editor to build your XML layout using a drag-and-drop interface.
- **Instantiate layout elements at runtime**: Your app can create View and ViewGroup objects (and manipulate their properties) programmatically.

Declaring your UI in XML allows you to separate the presentation of your app from the code that controls its behavior. Using XML files also makes it easy to provide different layouts for different screen sizes and orientations.

Widget Attributes:

Each layout or widget has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and there are other attributes which are specific to that layout.

Following are common attributes and will be applied to all the layouts:

SN	***Attribute & Description***
1	***android:id*** This is the ID which uniquely identifies the view.
2	***android:layout_width*** This is the width of the layout.
3	***android:layout_height*** This is the height of the layout
4	***android:layout_marginTop*** This is the extra space on the top side of the layout.

5	***android:layout_marginBottom*** This is the extra space on the bottom side of the layout.
6	***android:layout_marginLeft*** This is the extra space on the left side of the layout.
7	***android:layout_marginRight*** This is the extra space on the right side of the layout.
8	***android:layout_gravity*** This specifies how child Views are positioned.
9	***android:layout_weight*** This specifies how much of the extra space in the layout should be allocated to the View.
10	***android:layout_x*** This specifies the x-coordinate of the layout.
11	***android:layout_y*** This specifies the y-coordinate of the layout.
12	***android:paddingLeft*** This is the left padding filled for the layout.
13	***android:paddingRight*** This is the right padding filled for the layout.
14	***android:paddingTop*** This is the top padding filled for the layout.

15	***android:paddingBottom*** This is the bottom padding filled for the layout.

Here width and height are the dimension of the layout/view which can be specified in terms of **dp** (*Density-independent Pixels*), **sp** (*Scale-independent Pixels*), **pt** (*Points which is 1/72 of an inch*), **px**(*Pixels*), **mm** (*Millimeters*) and **in** (*inches*).

You can specify width and height with exact measurements but more often, you will use one of these constants to set the width or height

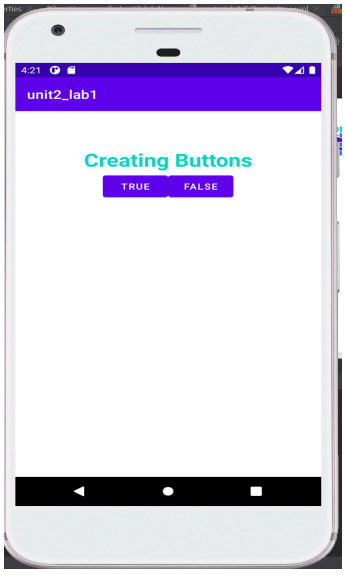
- **android:layout_width = "wrap_content "** tells your view to size itself to the dimensions required by its content.
- **android:layout_width = "match_parent "** tells your view to become as big as its parent view.

Exammple:
Activity_main.xml

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

    <TextView
        android:id="@+id/tvid"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Creating Buttons"

        android:textSize="30sp"
        android:textStyle="bold"
        android:textColor="#FF03DAC5"
        android:gravity="center"
        android:layout_marginTop="60dp"

        />

    <LinearLayout
```

```
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        >

        <Button
            android:id="@+id/btn1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="true"
            />
        <Button
            android:id="@+id/btn2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="false"
            />
</LinearLayout>
</LinearLayout>
```

Output:



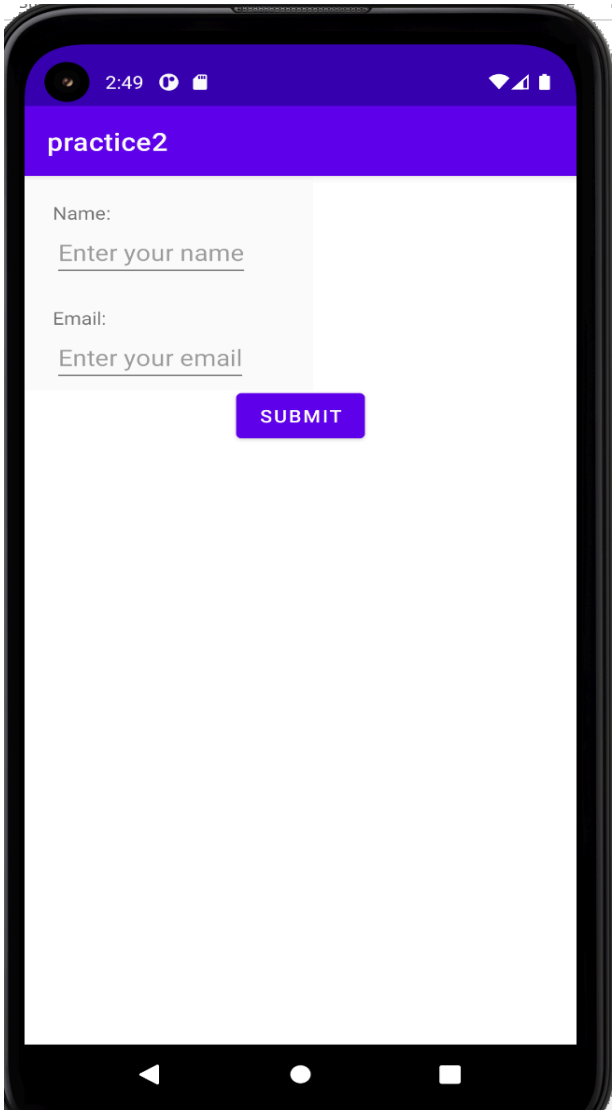
Program-2:Simple User interface

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/a
ndroid"

xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name:"
        android:layout_marginTop="20dp"
        android:layout_marginStart="20dp"

        />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:hint="Enter your name"


        />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email:"
        android:layout_marginTop="20dp"
        android:layout_marginStart="20dp"

        />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:hint="Enter your email"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="submit"
        android:layout_gravity="center"

        />


</LinearLayout>
```

Output:



Lecturer: Teksan Gharti Magar

Creating String Resources:
String resource is the XML file that contains the string values used in Layouts or widgets. Every project includes a default string file named ***strings.xml*** in the directory ***res/values/string.xml***.

Here is the some strings written in ***string.xml*** file and used in ***activity_main.xml*** file.

String.xml

```
<resources>
   <string name=”app_name”>Android Practice</string>
   <string name =”txt1”> This is first string, Thank you..</string>
</resources>
```

activity_main.xml

```
<?xml version=”1.0” encoding=”utf-8”?>
<ndroid.constraintlayout.widget.ConstraintLayout
xmlns:android=”http://schemas.android.com/apk/res/android”
   xmlns:app=”http://schemas.android.com/apk/res-auto”
   xmlns:tools=”http://schemas.android.com/tools”
   android:layout_width=”match_parent”
   android:layout_height=”match_parent”
   tools:context=”.MainActivity”>

   <TextView
      android:layout_width=”wrap_content”
      android:layout_height=”wrap_content”
      android:text=”@string/txt1”
      app:layout_constraintBottom_toBottomOf=”parent”
      app:layout_constraintLeft_toLeftOf=”parent”
      app:layout_constraintRight_toRightOf=”parent”
      app:layout_constraintTop_toTopOf=”parent” />

</ndroid.constraintlayout.widget.ConstraintLayout>
```

Create a new layout by adding buttons:
activity_main.xml

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

<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Namaste"
   android:gravity="center"
   android:textSize="30sp"
   android:textStyle="bold"
   android:layout_marginTop="60dp"
   />
<LinearLayout
   android:orientation="horizontal"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center">
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="True"
      />
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="False"
      />
</LinearLayout>

</LinearLayout>
```

Creating a new class or java file:
We have to create a java file which is also called the activity to lunch the XML file. XML file contains all the design related stuffs and java file handles the logic and coding section of related XML file.

To create an java file we have to give name of java file same as xml file which want to link.

Here if we want to link ***activity_main.xml*** file to java file, then we have to give the java file name as ***MainActivity.java***

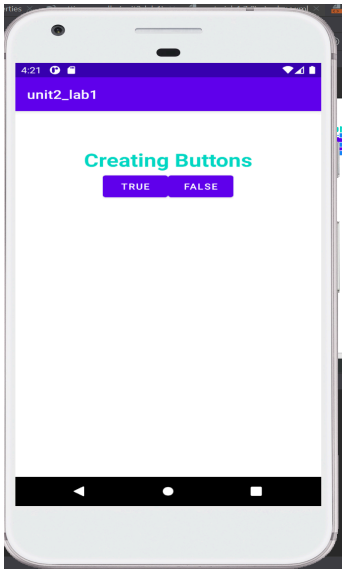
MainActivity.java

```
package com.example.androidpractice;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

here our activity_main.xml file is link to MainActivity.java class, as can see in ***setContentView(R.layout.activity_main);***

Running the project in Emulator:

After setting up all the project files, then click on run button. Our first app will be display in our virtual device.



Congratulations!! You are successfully created and run your first android application.

Lecturer: Teksan Gharti Magar

Assignment:

1. What is android platform? Explain about history of android platform in detail.
2. List and explain different versions of android in detail.
3. How can set up environment for android studio? Explain.
4. What do you mean by view hierarchy? Explain with help of suitable example.
5. Define widget. Explain different widget attributes with example.
6. How can we create string resources? Explain with the help of suitable example.
7. Write a set of code for creating activity name MainActivity.java. Also link xml file activity_main.xml to that activity.
8. Explain about android manifest file. How can we add activity in manifest?
9. What is the role of intent-filter for any activity?
10. Explain the purpose of creating an AVD. Also write steps for running application on Emulator.

==========End of Unit-2==========

Lecturer: Teksan Gharti Magar