MOBILE APPLICATION DEVELOPMENT USING ANDROID

Topics

- ✓ What is Android?
- ✓ Evolution of Android
- ✓ Why Android?
- ✓ Features
- ✓ Architecture

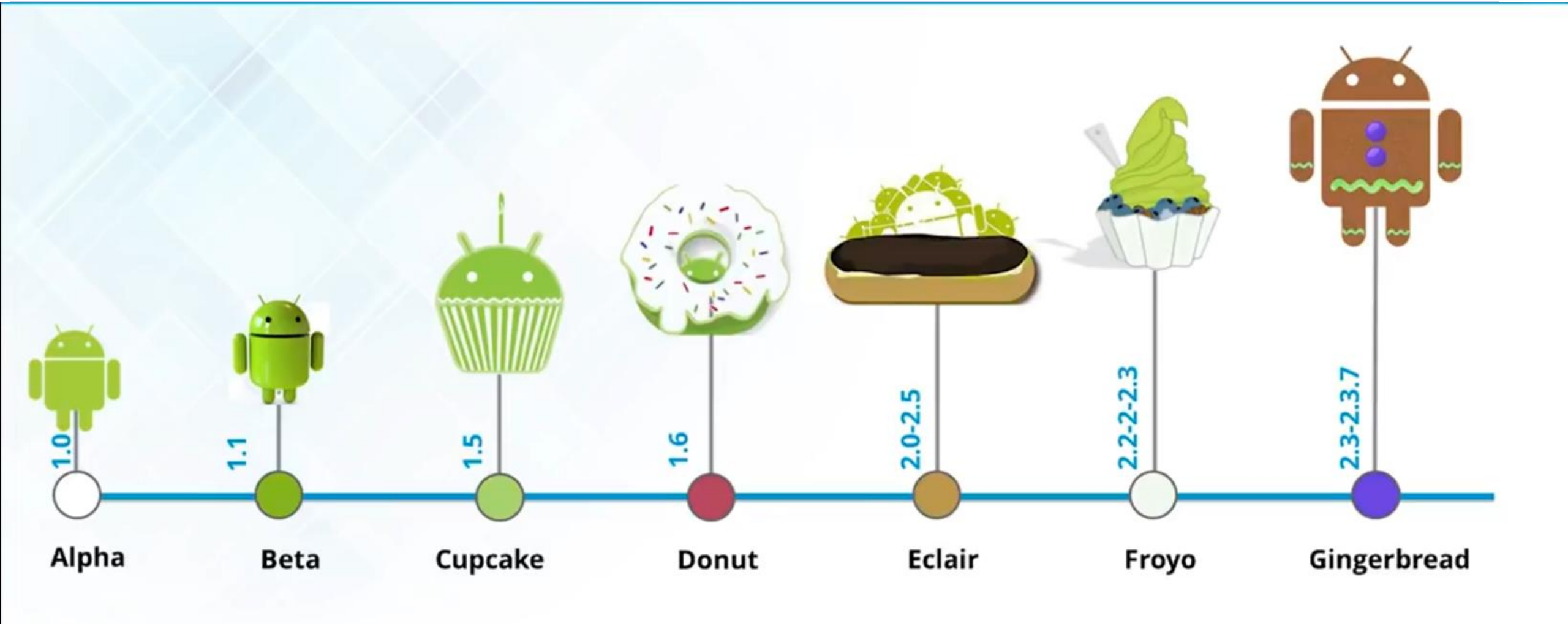
What is Android ?

- ✓ It is a open source software platform and operating system for mobile devices
- ✓ Based on the Linux kernel
- ✓ Developed by Google and later the Open Handset Alliance (OHA)
- ✓ Allows writing managed code in the Java language
- ✓ Android has its own virtual machine i.e. DVM(Dalvik Virtual Machine),which is used for executing the android application.
- ✓ Google purchased the initial developer of the software , android incorporated in 2005.

OHA INCLUDED COMPANIES

SAMSUNG
MOTOROLA
vodafone
htc smart mobility
LG
DELL
ASUS
ALCATEL mobile phones
acer
docomo
NEC
HUAWEI
KYOCERA
T-Mobile
SHARP
Sony Ericsson
TOSHIBA
ebay
Google
intel
NVIDIA
QUALCOMM
Texas Instruments

Android Versions



Android Versions

Why Android ?

Variety of apps can be developed

Customizable operating system

Reduces the overall complexity

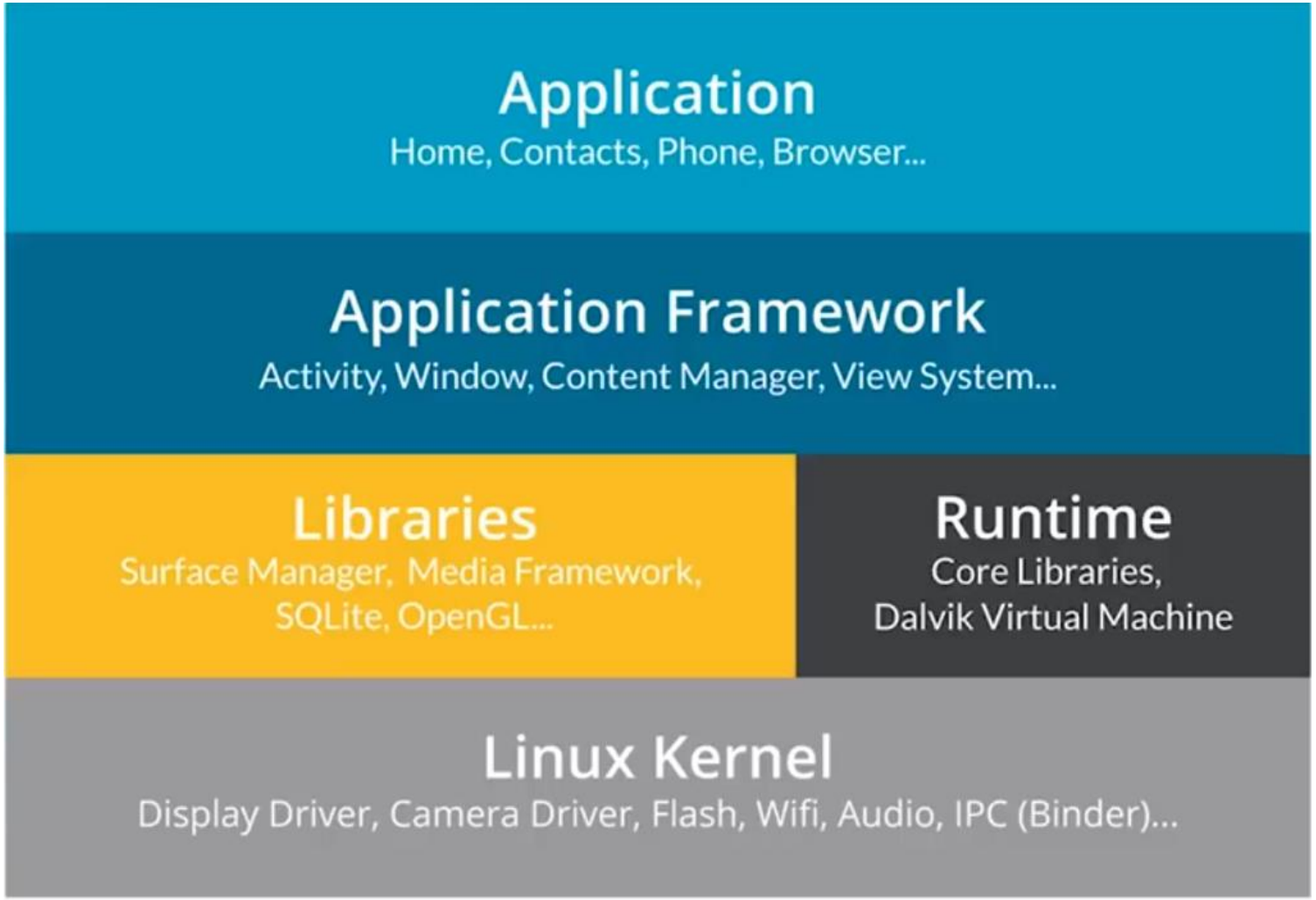
Powerful open source

Ready made, low cost for high-tech devices

Android - Features

- ✓ Open Source
- ✓ GUI
- ✓ Accessibility
- ✓ Storage
- ✓ Messaging
- ✓ Media Support
- ✓ Multitasking
- ✓ Resizable widgets
- ✓ Multi Languages
- ✓ Voice based features

Android Architecture



Applications –

- Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc. and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only. It runs within the Android run time with the help of the classes and services provided by the application framework.

Application framework –

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation. It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

Application runtime –

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

Platform libraries –

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

•**Media** library provides support to play and record an audio and video formats.

•**Surface manager** responsible for managing access to the display subsystem.

•**SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.

•**SQLite** provides database support and **FreeType** provides font support.

•**Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.

•**SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.

Linux Kernel –

Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

Linux Kernel Features–

•**Security:** The Linux kernel handles the security between the application and the system.

•**Memory Management:** It efficiently handles the memory management thereby providing the freedom to develop our apps.

•**Process Management:** It manages the process well, allocates resources to processes whenever they need them.

•**Network Stack:** It effectively handles the network communication.

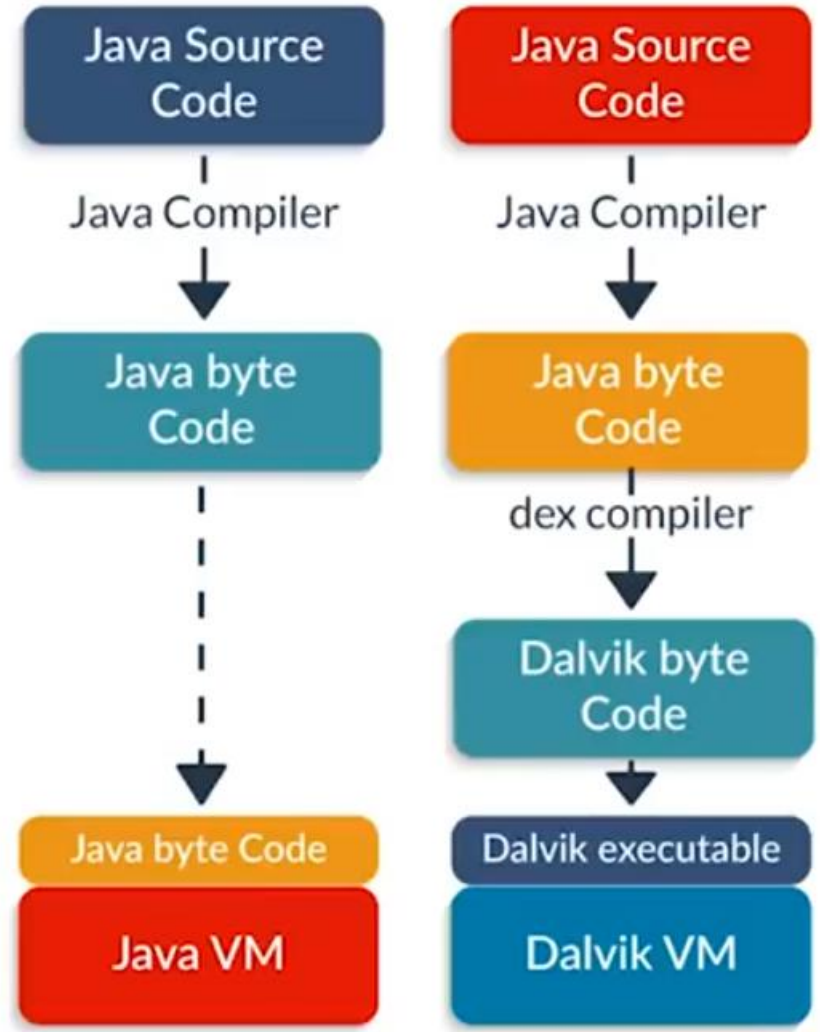
•**Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

Android Runtime

What is runtime?

In a simplest term it is a system used by operating system which takes care of converting the code that you write in a high level language like Java to machine code and understand by CPU/Processor.

Android runtime= DVM + Core Java Libraries

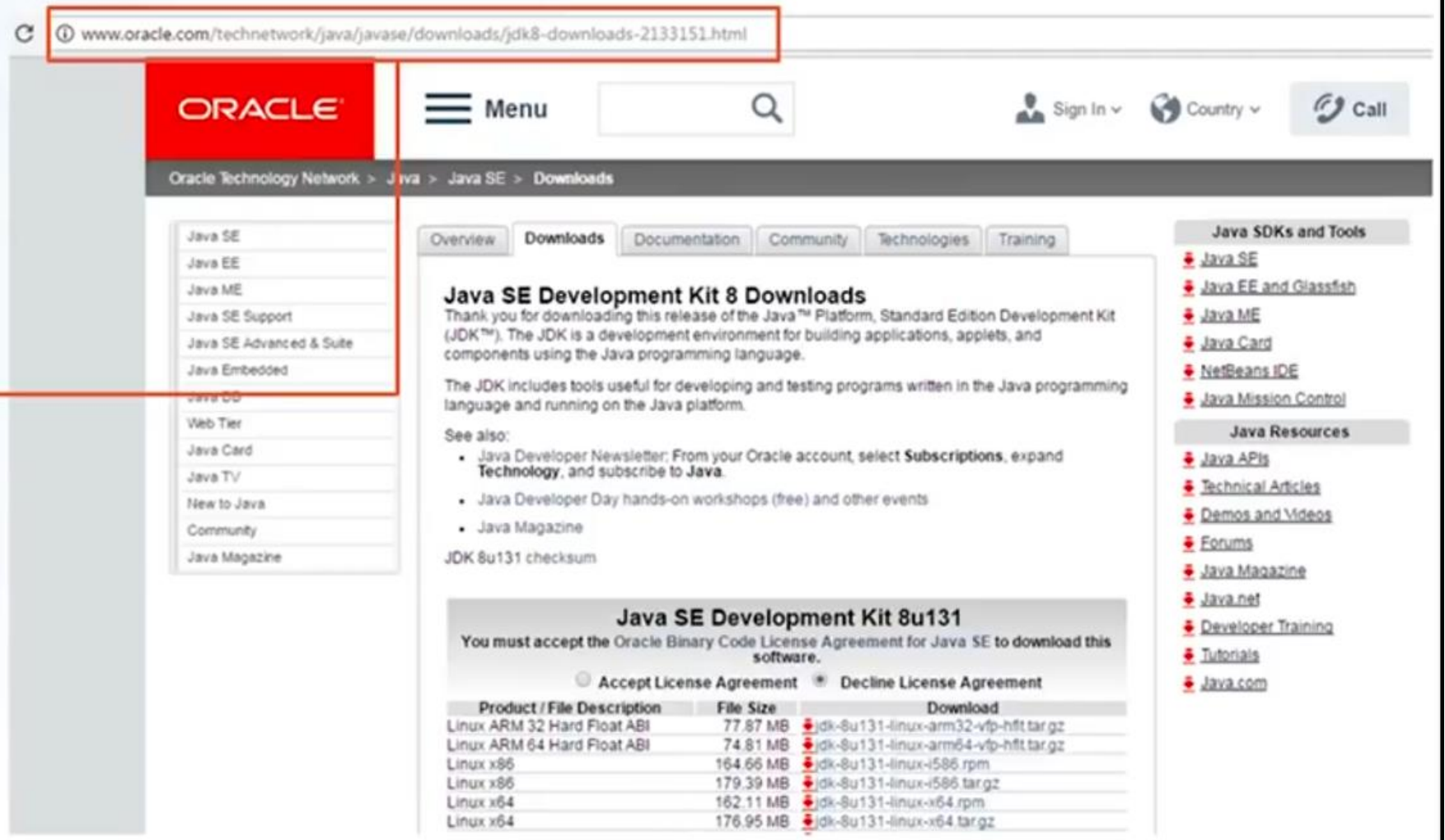

Java Source Code
Java Compiler
Java byte Code
Java byte Code
Java VM
Java Source Code
Java Compiler
Java byte Code
dex compiler
Dalvik byte Code
Dalvik executable
Dalvik VM

JVM	DVM
Support multiple OS	Support only Android OS
Compiles Java Byte code	Compiles Dalvik Byte Codes
Forms .class files	Forms .dex file
Runs on more memory	Runs on less memory
Stack Based	Register Based
Executable format is JAR	Executable format is APK
Runs .class byte codes directly	The .dex files are optimized to .odex format before executing in DVM

Configuration of Android Environment

JDK-Installation

1. Go to the link : http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
And download JDK.



Android Environment

The Android SDK (software development kit) is a set of development tools used to develop applications for the Android platform

The Android SDK includes the following:

•Required libraries.

•Debugger.

•An emulator. (AVD)

•Relevant documentation for the Android application program

interfaces (APIs).

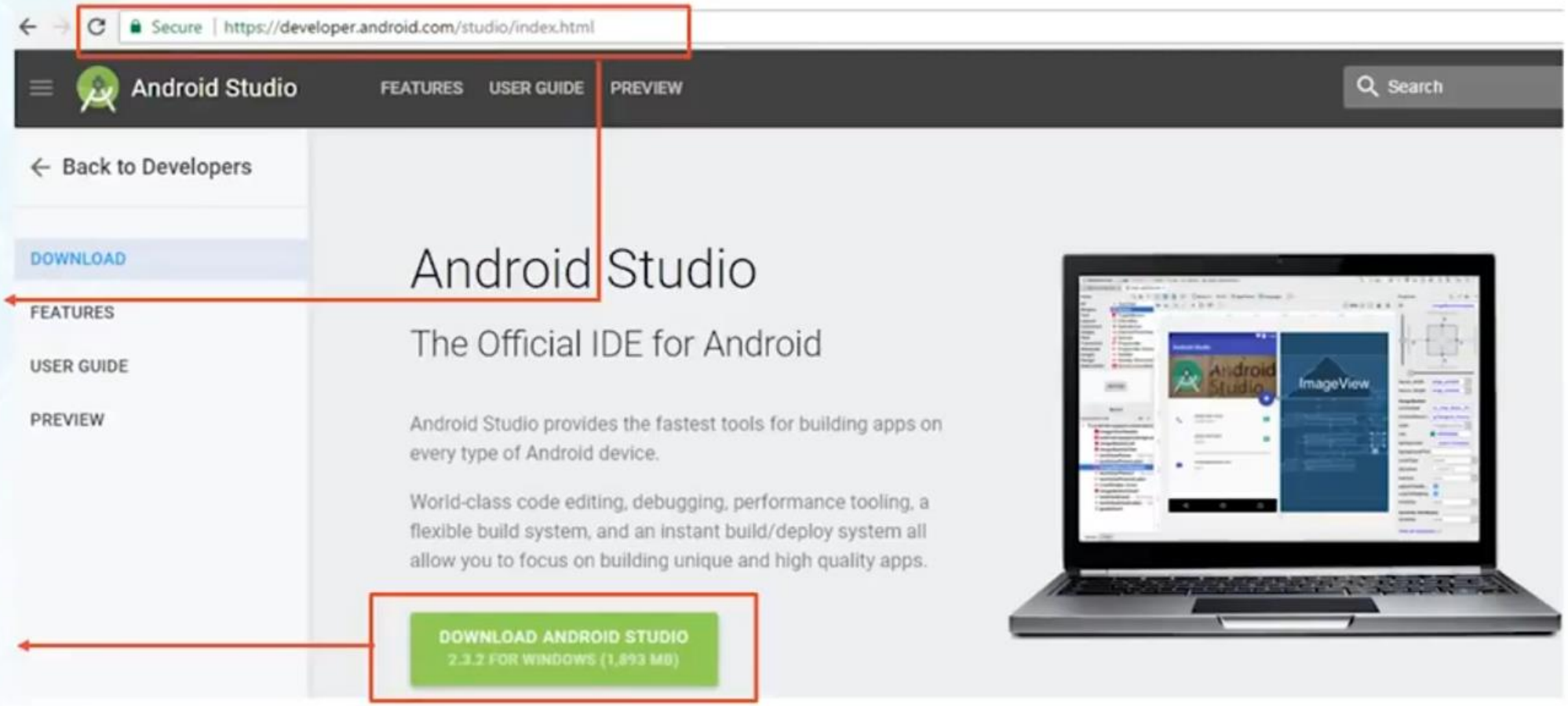
•Sample source code.

•Tutorials for the Android OS.

Android Studio-Installation

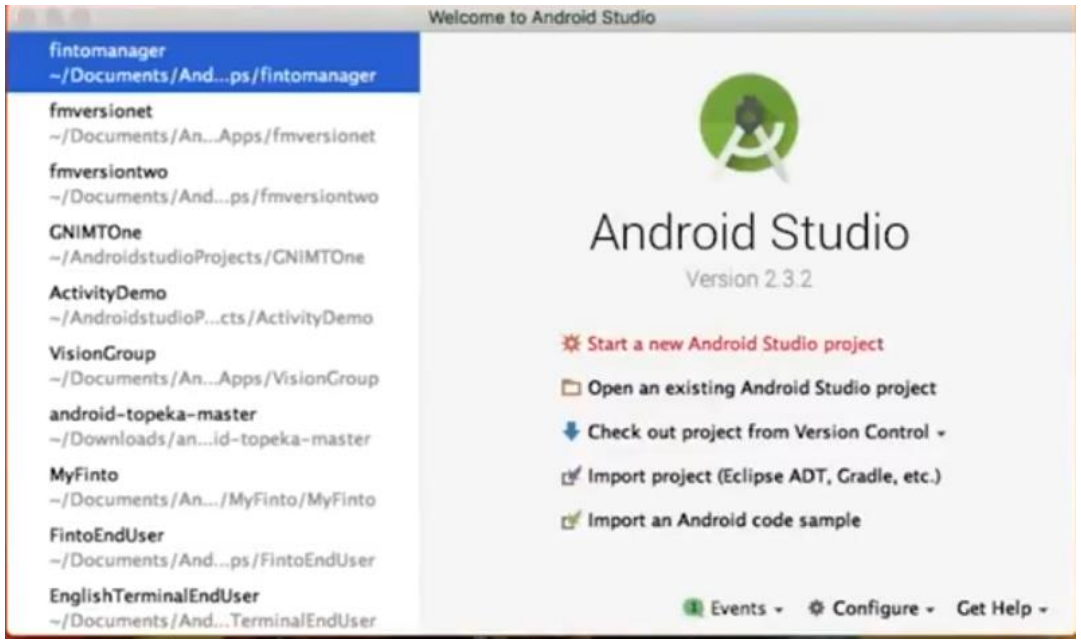
1 Go to the link : https://developer.android.com/studio/index.html

2 Download latest Android Studio 2.3.2

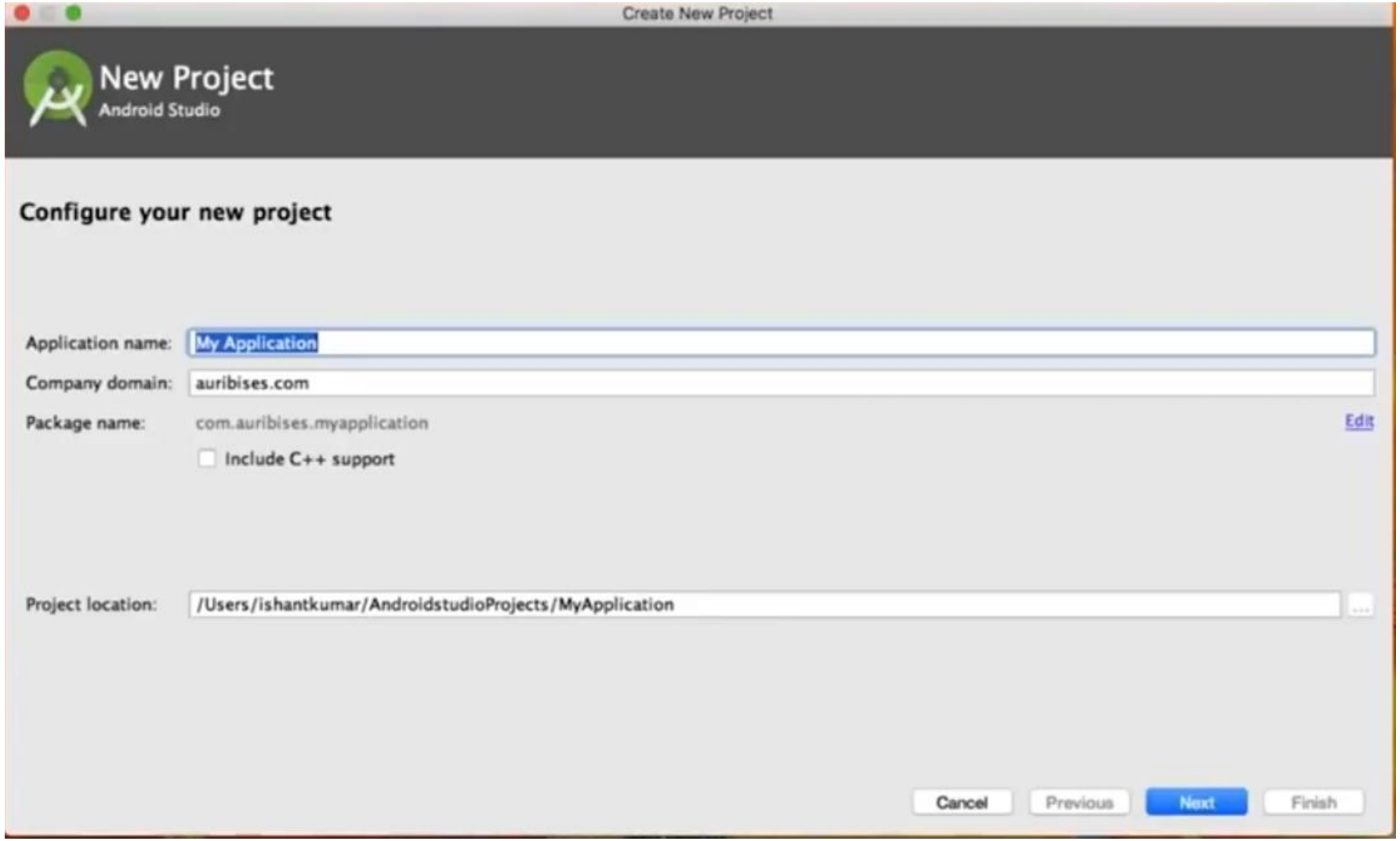


ANDROID APP DEVELOPMENT

Creating Android Application

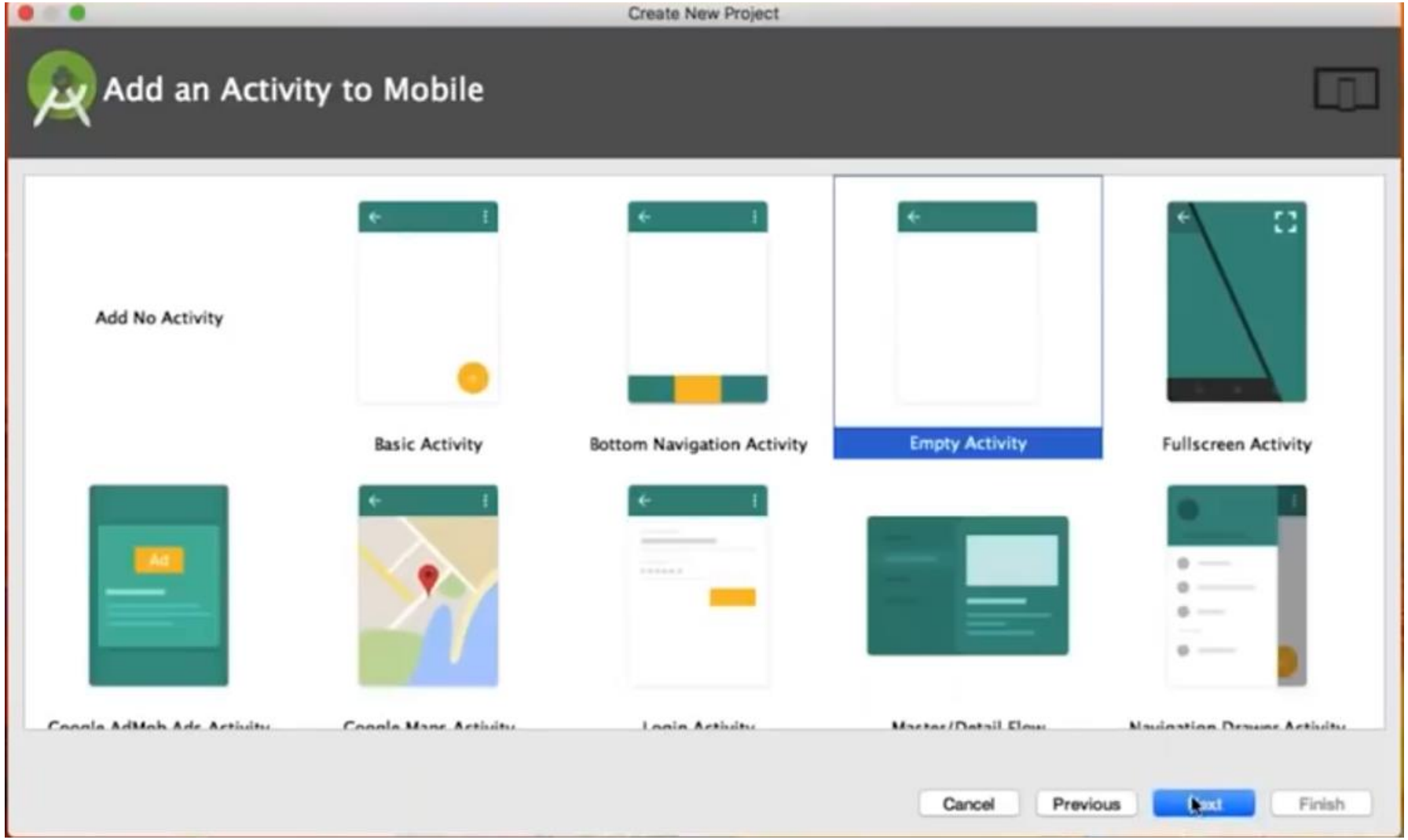


Creating Android Application

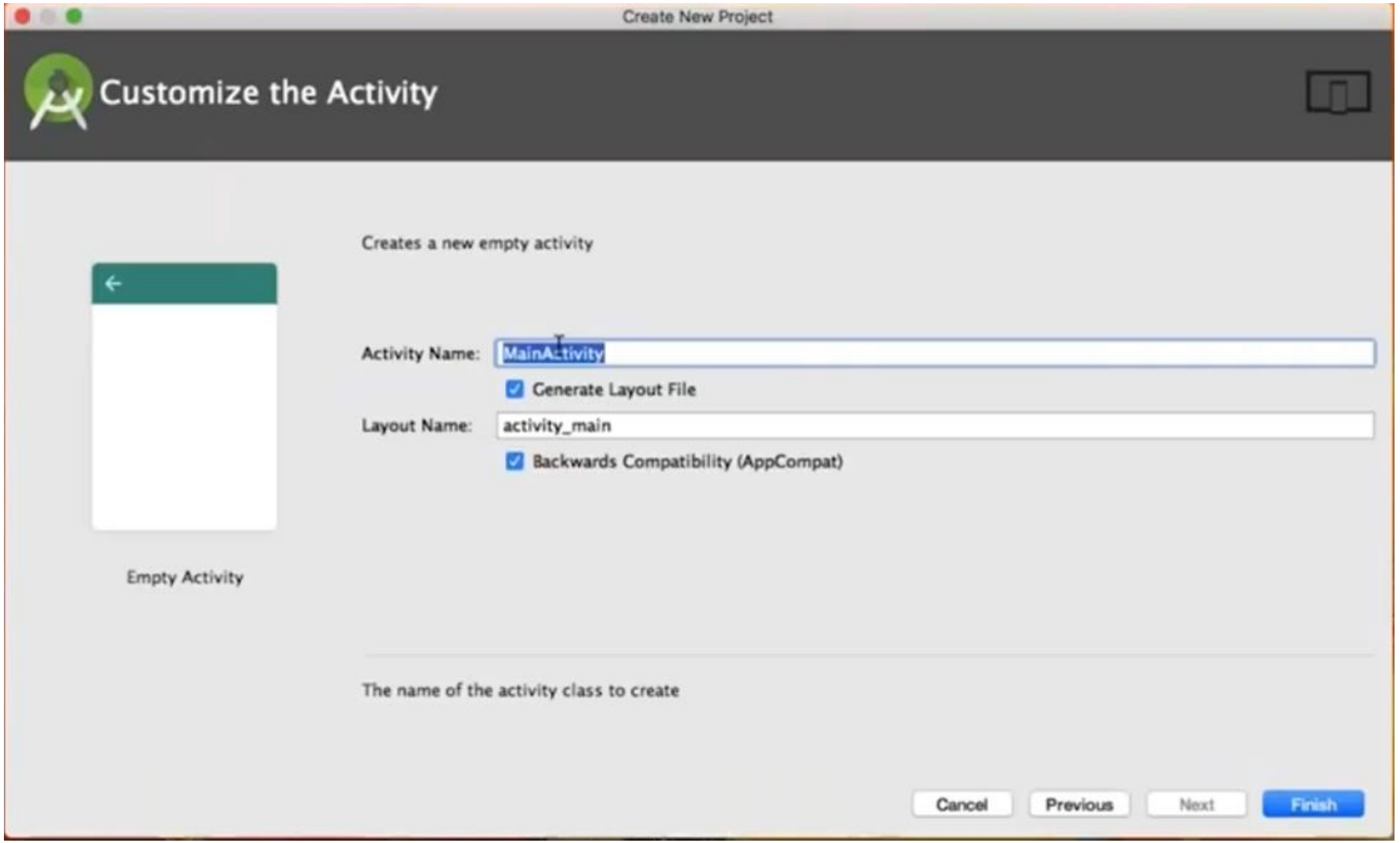


Creating Android Application

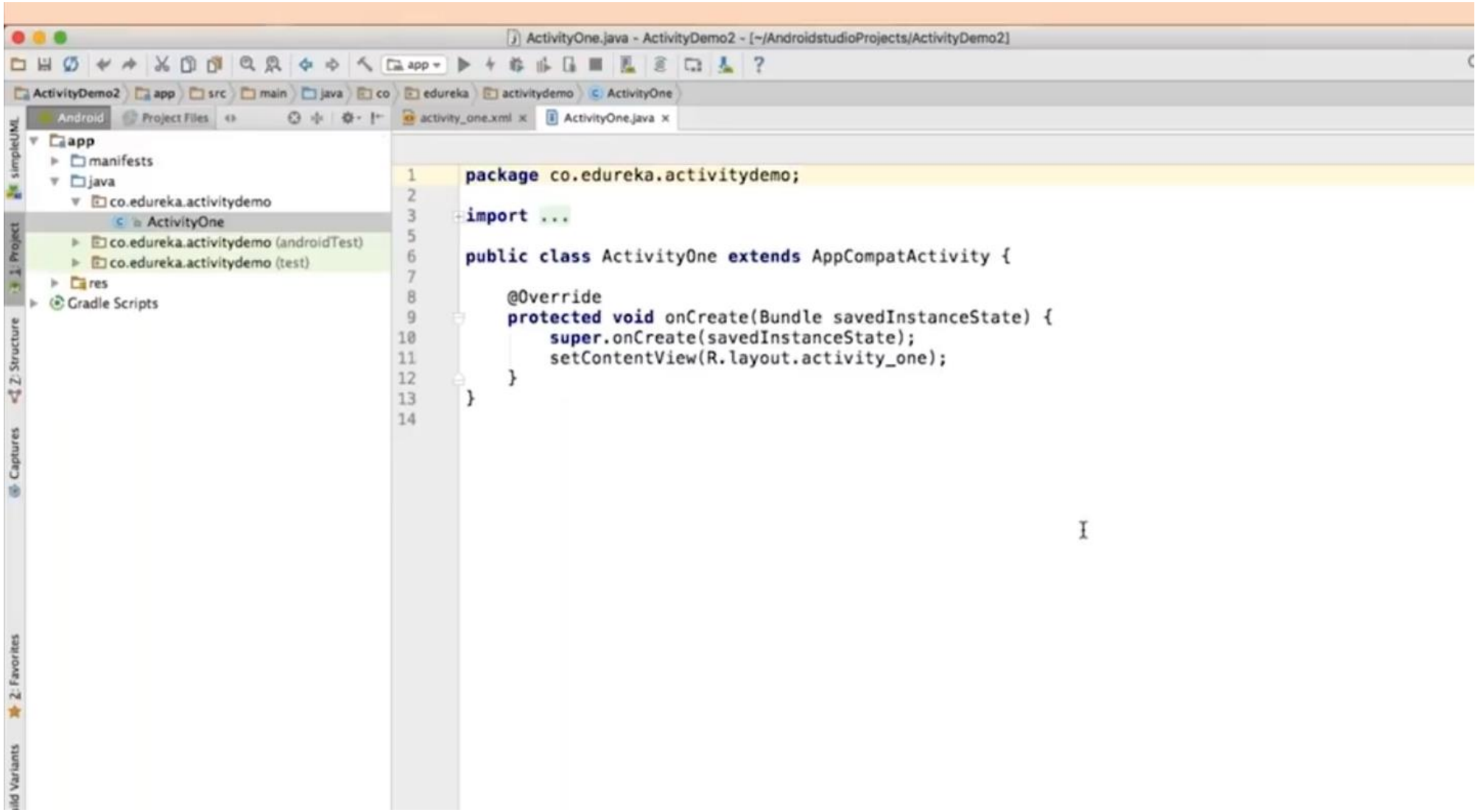
Creating Android Application



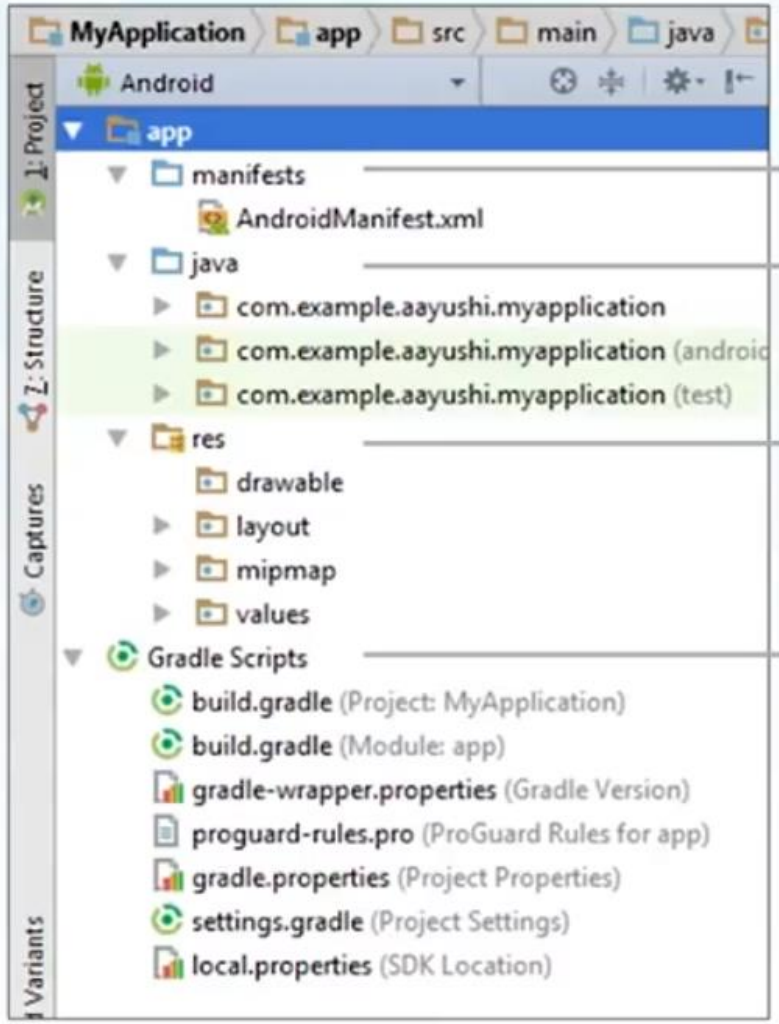
Creating Android Application



Steps for Creating Android Application



Structure of Android Application



It describes the fundamental characteristics of the app and defines each of its components.

This contains the **.java** source files for your project. By default, it includes an *MainActivity.java* source file.

This is a directory for files that define your app's user interface.

This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

Structure of Android Application

1. AndroidManifest.xml: Every project in Android includes a manifest file, which is AndroidManifest.xml, stored in the root directory of its project hierarchy. The manifest file is an important part of our app because it defines the structure and metadata of our application, its components, and its requirements. This file includes nodes for each of the Activities, Services, Content Providers and Broadcast Receiver that make the application and using Intent Filters and Permissions, determines how they co-ordinate with each other and other applications.

Structure of Android Application

2. Java: The Java folder contains the Java source code files. These files are used as a controller for controlled UI (Layout file). It gets the data from the Layout file and after processing that data output will be shown in the UI layout. It works on the backend of an Android application.

3. drawable: A Drawable folder contains resource type file (something that can be drawn). Drawable may take a variety of file like Bitmap (PNG, JPEG), Nine Patch, Vector (XML), Shape, Layers, States, Levels, and Scale.

Structure of Android Application

4. layout: A layout defines the visual structure for a user interface, such as the UI for an Android application. This folder stores Layout files that are written in XML language. You can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout file.

5. mipmap: Mipmap folder contains the Image Asset file that can be used in Android Studio application. You can generate the following icon types like Launcher icons, Action bar and tab icons, and Notification icons.

Structure of Android Application

6. values: This is the directory for other various XML files that contain collection of resources such as Strings and Color definitions

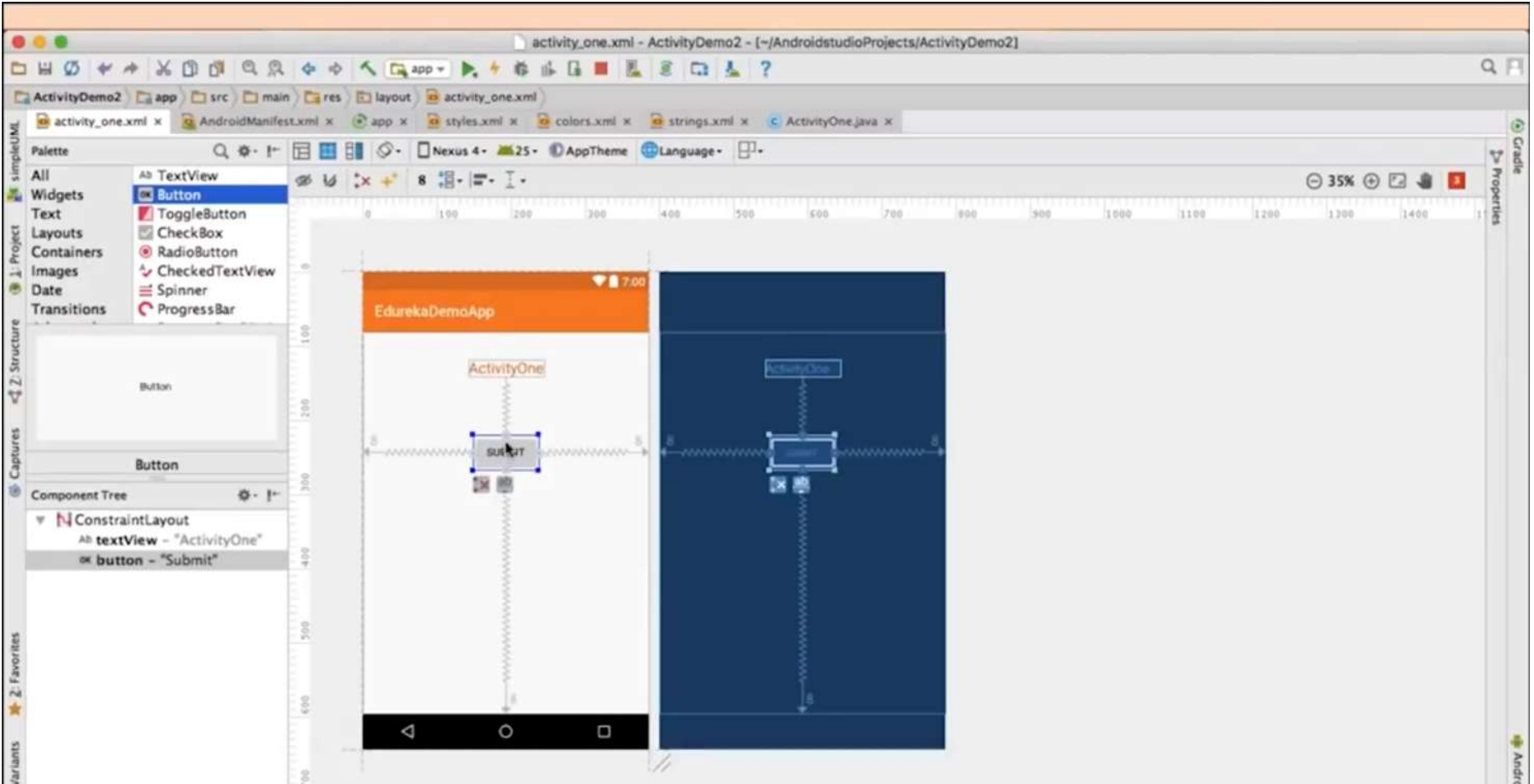
colors.xml: colors.xml file contains color resources of the Android application. Different color values are identified by a unique name that can be used in the Android application program.

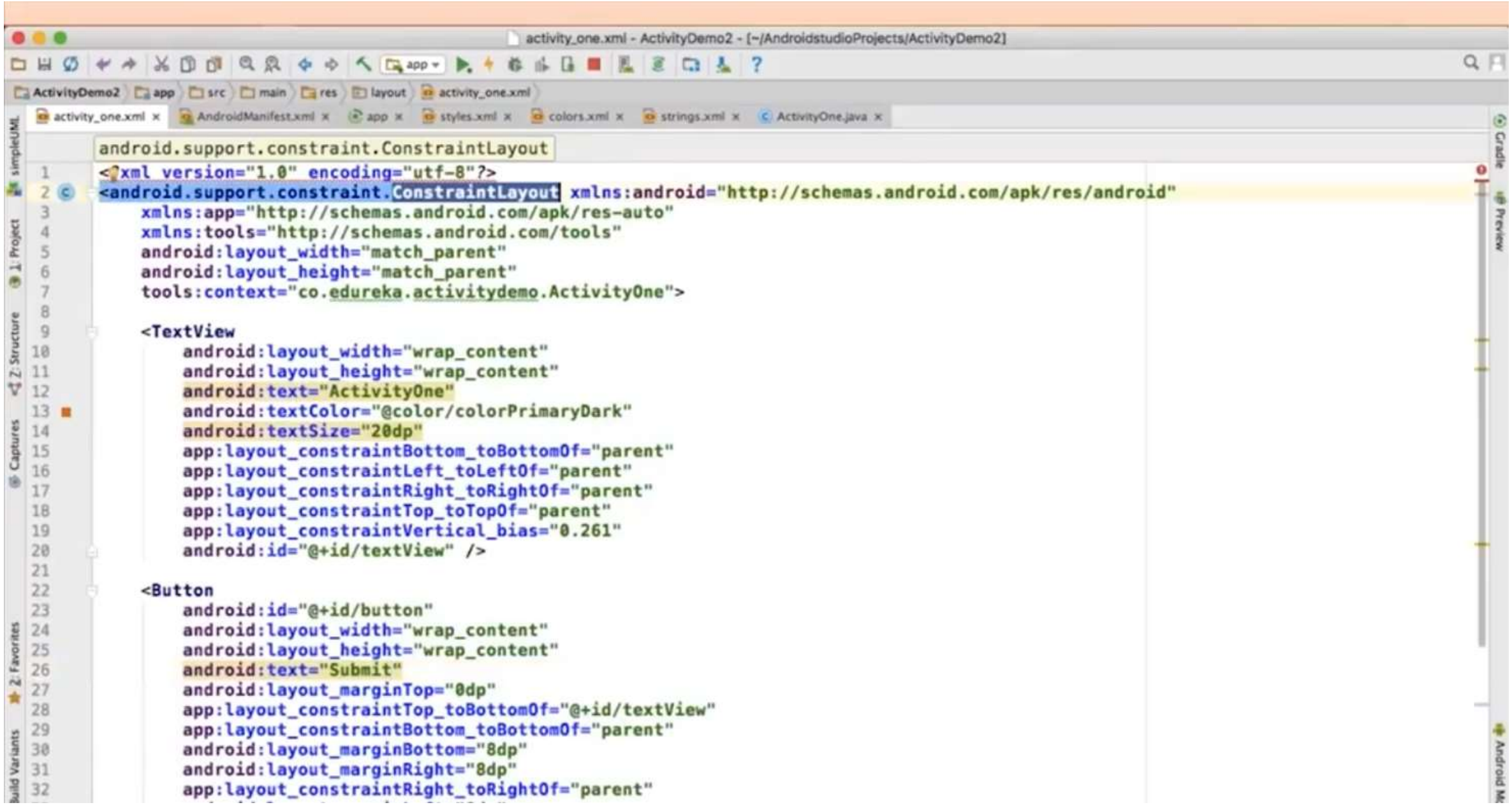
strings.xml: The strings.xml file contains string resources of the Android application. The different string value is identified by a unique name that can be used in the Android application program. This file also stores string array by using XML language.

ANDROID - LAYOUTS

What is Android Layouts?

An Android layout is a class that handles arranging the way its children appear on the screen. Anything that is a View (or inherits from **View**) can be a child of a layout. All of the layouts inherit from **ViewGroup** (which inherits from **View**) so you can nest layouts. You could also create your own custom layout by making a class that inherits from **ViewGroup**.

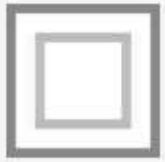

activity_one.xml - ActivityDemo2 - [~/AndroidstudioProjects/ActivityDemo2]
ActivityDemo2
app
src
main
res
layout
activity_one.xml
activity_one.xml
AndroidManifest.xml
app
styles.xml
colors.xml
strings.xml
ActivityOne.java
Palette
All
Widgets
Text
Layouts
Containers
Images
Date
Transitions
TextView
Button
ToggleButton
CheckBox
RadioButton
CheckedTextView
Spinner
ProgressBar
Nexus 4
25
AppTheme
Language
35%
EdurekaDemoApp
ActivityOne
SUBMIT
Button
Component Tree
ConstraintLayout
textView - "ActivityOne"
button - "Submit"
Gradle
Properties


activity_one.xml - ActivityDemo2 - [~/AndroidstudioProjects/ActivityDemo2]
ActivityDemo2 app src main res layout activity_one.xml
activity_one.xml AndroidManifest.xml app styles.xml colors.xml strings.xml ActivityOne.java
android.support.constraint.ConstraintLayout
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="co.edureka.activitydemo.ActivityOne">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="ActivityOne"
android:textColor="@color/colorPrimaryDark"
android:textSize="20dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.261"
android:id="@+id/textView" />
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Submit"
android:layout_marginTop="0dp"
app:layout_constraintTop_toBottomOf="@+id/textView"
app:layout_constraintBottom_toBottomOf="parent"
android:layout_marginBottom="8dp"
android:layout_marginRight="8dp"
app:layout_constraintRight_toRightOf="parent"

Android Standard Layouts?

Layout	Description
LinearLayout  (Horizontal)  (Vertical)	**LinearLayout** is a view group that aligns all children in a single direction, vertically or horizontally.
RelativeLayout 	**RelativeLayout** is a view group that displays child views in relative positions.
TableLayout 	**TableLayout** is a view that groups views into rows and columns.

Android Standard Layouts?

GridLayout 	**GridLayout** uses a grid of infinitely-thin lines to separate its drawing area into: rows, columns, and cells. It supports both row and column spanning, which together allow a widget to occupy a rectangular range of cells that are next to each other.
FrameLayout 	The **FrameLayout** is a placeholder on screen that you can use to display a single view.
AbsoluteLayout	**AbsoluteLayout** enables you to specify the exact location of its children. Arrange the children views according coordinates x, y.

Layouts

Linear Layout- A Linear Layout aligns each of the child View in either a vertical or a horizontal line. A vertical layout has a column of Views, whereas in a horizontal layout there is a row of Views. It supports a weight attribute for each child View that can control the relative size of each child View within the available space.

Relative Layout- It is flexible than other native layouts as it lets us to define the position of each child View relative to the other views and the dimensions of the screen.

Layouts

Table Layout -The Table Layout groups views into rows and columns

Grid Layout- It was introduced in Android 4.0 (API level 14), the Grid Layout used a rectangular grid of infinitely thin lines to lay out Views in a series of rows and columns. The Grid Layout is incredibly flexible and can be used to greatly simplify layouts and reduce or eliminate the complex nesting often required to construct UIs using the layouts described before.

Layouts

Frame Layout- It is the simplest of the Layout Managers that pins each child view within its frame. By default the position is the top-left corner, though the gravity attribute can be used to alter its locations. You can add multiple children stacks each new child on top of the one before, with each new View potentially obscuring the previous ones.

Layouts

The Absolute Layout enables you to specify the exact location of its children.

DESIGNING USER INTERFACE WITH VIEWS

VIEWS

View is the basic building block of UI in Android. View refer to **Android.view.View** class, which is the super class of all the GUI components like Text View, Button, Image Button etc. Views also know as Widgets.

Views Category

1. Basic Views
2. Picker Views
3. List Views

Syntax for creating views

<View Name

Attribute1 =value

Attribute2=value

…….

Attribute N=value

/>

Attribute that will define how that view will look on the screen of the application along with a value for the attribute

TextView

In android, **TextView** is a user interface control that is used to set and display the text to the user based on our requirements. The TextView control will act as like label control and it optionally allow them to edit text.

In android, we can create a TextView control in two ways either in XML layout file or create it in Activity file programmatically

Create a TextView in Layout File

```
<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:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:text="Welcome"
        android:textColor="#86AD33"
        android:textSize="20dp"
        android:textStyle="bold" />
</LinearLayout>
```

Create a TextView in Activity File

```
public class MainActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      LinearLayout linearLayout = (LinearLayout) findViewById(R.id.linearlayout);
      TextView textView = new TextView(this);
      textView.setText("Welcome to Tutlane");
      linearLayout.addView(textView);
   }
}
```

Nested classes

enum	**TextView.BufferType** **Type** of the text buffer that defines the characteristics of the text such as static, styleable, or editable.
interface	**TextView.OnEditorActionListener** **Interface** definition for a callback to be invoked when an action is performed on the editor.
class	**TextView.SavedState** **User** interface state that is stored by TextView for implementing View#onSaveInstanceState.

Attribute	Description
android: id	It is used to uniquely identify the control
android:autoLink	It will automatically found and convert URLs and email addresses as clickable links.
android: ems	It is used to make the textview be exactly this many ems wide.
android:hint	It is used to display the hint text when text is empty
android:width	It makes the TextView be exactly this many pixels wide.
android:height	It makes the TextView be exactly this many pixels tall.
android:text	It is used to display the text.
android:textColor	It is used to change the color of the text.
android:gravity	It is used to specify how to align the text by the view's x and y-axis.
android:maxWidth	It is used to make the TextView be at most this many pixels wide.
android:minWidth	It is used to make the TextView be at least this many pixels wide.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:textAllCaps	It is used to present the text in all CAPS
android:typeface	It is used to specify the Typeface (normal, sans, serif, monospace) for the text.
android:textColor	It is used to change the color of the text.
android:textColorHighlight	It is used to change the color of text selection highlight.
android:textColorLink	It is used to change the text color of links.
android:inputType	It is used to specify the type of text being placed in text fields.
android:fontFamily	It is used to specify the fontFamily for the text.
android:editable	If we set, it specifies that this TextView has an input method.

Button

In android, **Button** is a user interface control that is used to perform an action whenever the user clicks or tap on it.

Generally, Buttons in android will contain a text or an icon or both and perform an action when the user touches it.

In android, we have a different type of buttons available to use based on our requirements, those are **Toggle Button**, **Radio Button, Image Button**.

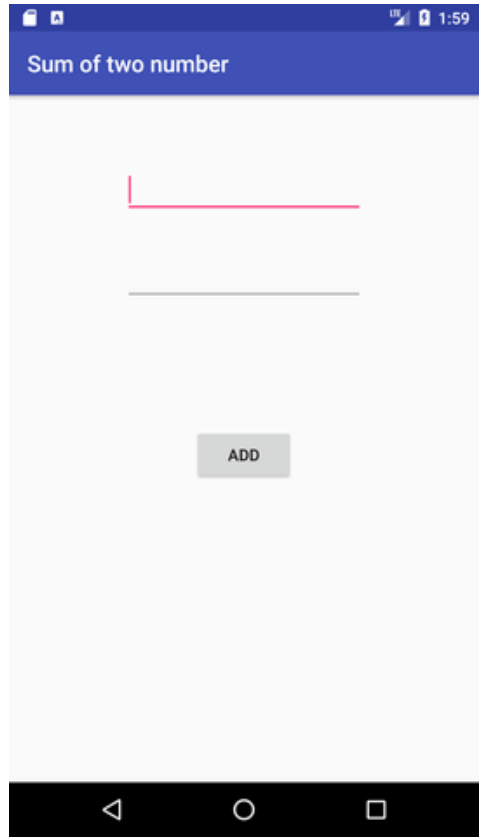
In android, we can create a Button control in two ways either in the XML layout file or create it in the Activity file programmatically.

Create Button in XML Layout File

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/addBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ADD" />
</LinearLayout>
```

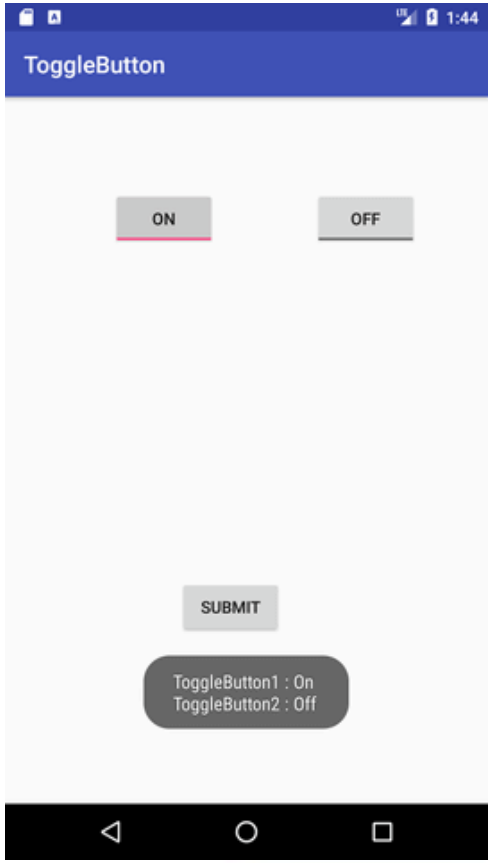
Create Button Control in Activity File

```
LinearLayout layout = (LinearLayout)findViewById(R.id.l_layout);

Button btn = new Button(this);

btn.setText("ADD");

layout.addView(btn);
```



Attribute	Description
android:id	It is used to uniquely identify the control
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for button control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:drawableBottom	It’s drawable to be drawn to the below of text.
android:drawableRight	It’s drawable to be drawn to the right of text.
android:drawableLeft	It’s drawable to be drawn to the left of the text.

ToggleButton



Android Toggle Button can be used to display checked/unchecked (On/Off) state on the button.

It is beneficial if user have to change the setting between two states. It can be used to On/Off Sound, Wifi, Bluetooth etc.

Since Android 4.0, there is another type of toggle button called *switch* that provides slider control.

Create ToggleButton in XML Layout File

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent" android:layout_height="match_parent">
   <ToggleButton
      android:id="@+id/toggle1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="100dp"
      android:layout_marginTop="120dp"
      android:checked="true"
      android:textOff="OFF"
      android:textOn="ON"/>

</RelativeLayout>
```

Create ToggleButton Control in Activity File

```
RelativeLayout layout = (RelativeLayout)findViewById(R.id.r_layout);
ToggleButton tb = new ToggleButton(this);
tb.setTextOff("OFF");
tb.setTextOn("ON");
tb.setChecked(true);
layout.addView(tb);
```

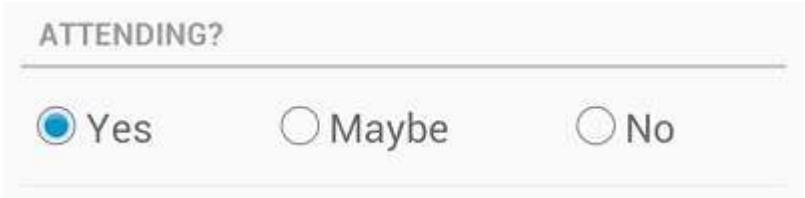
Attribute	Description
android:id	It is used to uniquely identify the control
android:checked	It is used to specify the current state of toggle button
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text.
android:textOn	It is used to set the text when the toggle button is in the ON / Checked state.
android:textOff	It is used to set the text when the toggle button is in the OFF / Unchecked state.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for toggle button control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:drawableBottom	It’s drawable to be drawn to the below text.
android:drawableRight	It’s drawable to be drawn to the right of the text.
android:drawableLeft	It’s drawable to be drawn to the left of text.

RadioButton

In android, **Radio Button** is a two-states button that can be either checked or unchecked and it's the same as CheckBox control, except that it will allow only one option to select from the group of options.

The user can press or click on the radio button to make it select. In android, CheckBox control allow users to change the state of control either Checked or Unchecked but the radio button cannot be unchecked once it is checked.

Generally, we can use **RadioButton** controls in an android application to allow users to select only one option from the set of values.



In android, we use radio buttons with in a **RadioGroup** to combine multiple radio buttons into one group and it will make sure that users can select only one option from the group of multiple options.

By default, the android **RadioButton** will be in **OFF** (**Unchecked**) state. We can change the default state of **RadioButton** by using **android:checked** attribute.

In case, if we want to change the state of **RadioButton** to **ON** (**Checked**), then we need to set **android:checked = "true"** in our XML layout file.

In android, we can create **RadioButton** control in two ways either in the XML layout file or create it in the programmatically.

```
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
   <RadioButton android:id="@+id/radio_pirates"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=“Male"
      android:onClick="onRadioButtonClicked"/>
   <RadioButton android:id="@+id/radio_ninjas"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=“Female"
      android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```

Attribute	Description
android:id	It is used to uniquely identify the control
android:checked	It is used to specify the current state of radio button
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text for the radio button.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for radio button control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:onClick	It’s the name of the method to invoke when the radio button clicked.
android:visibility	It is used to control the visibility of control.

ImageView and ImageButton in Android

ImageViewis used to show any picture on the user interface.

Attribute	Description
android:maxHeight	Used to specify a maximum height for this view.
android:maxWidth	Used to specify a maximum width for this view.
android:src	Sets a drawable as the content for this ImageView.
android:scaleType	Controls how the image should be resized or moved to match the size of the ImageView.
android:tint	Tints the color of the image in the ImageView.

```
<ImageButton
    android:id="@+id/imgButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scaleType="fitCenter"
    android:src="@drawable/img_nature"/>
```

ImageButton has the same property as ImageView. Only one feature is extra, which is, images set through ImageButton are clickable, and actions can be attached with them upon clicking.

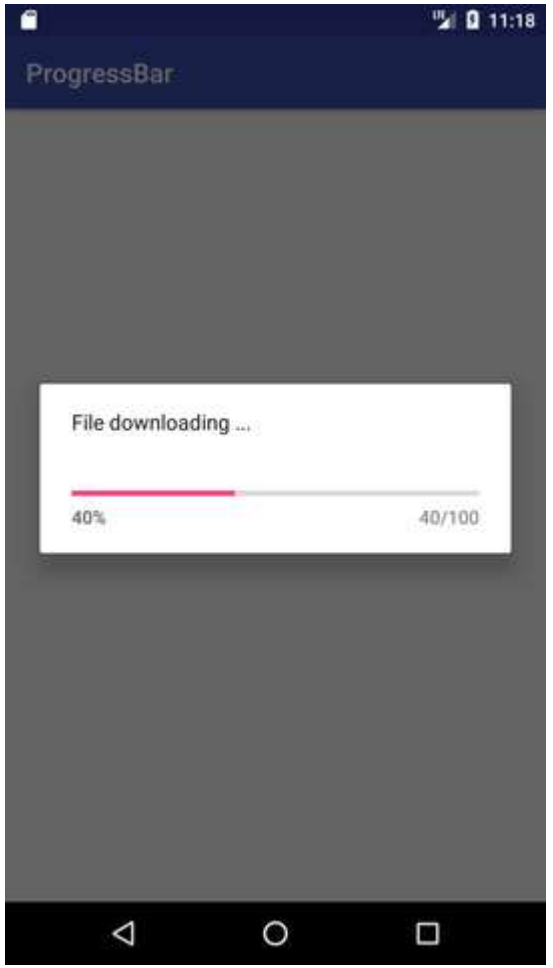
Here is how you define an ImageButton in the layout XML file:

Attribute	Description
android:id	It is used to uniquely identify the control
android:src	It is used to specify the source file of an image
android:background	It is used to set the background color for an image button control.
android:padding	It is used to set the padding from left, right, top and bottom of the image button.
android:baseline	It is used to set the offset of the baseline within the view.

DESIGNING USER INTERFACE WITH VIEWS

Progress Bar

In android, **ProgressBar** is a user interface control that is used to indicate the progress of an operation. For example, downloading a file, uploading a file.



By default the **ProgressBar** will be displayed as a spinning wheel, in case if we want to show it like a horizontal bar then we need to change the style property to horizontal

like style="?android:attr/progressBarStyleHorizontal".

Progress Bar

In android, the **ProgressBar** supports two types of modes to show the progress, those are **Determinate** and **Indeterminate**.

Generally, we use the **Determinate** progress mode in progress bar when we want to show the quantity of progress has occurred. For example, the percentage of file downloaded, number of records inserted into a database, etc.

To use Determinate progress, we need to set the style of the progress bar to **Widget_ProgressBar_Horizontal** or **progressBarStyleHorizontal** and set the amount of progress using **android:progress** attribute.

```
<ProgressBar
    android:id="@+id/pBar"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:max="100"
    android:progress="50" />
```

By using **setProgress(int)** method, we can update the percentage of progress displayed in app or by calling **incrementProgressBy(int)** method, we can increase the value of current progress completed based on our requirements.

Generally, when the progress value reaches **100** then the progress bar is full. By using **android:max** attribute we can adjust this default value.

Generally, we use the **Indeterminate** progress mode in progress bar when we don't know how long an operation will take or how much work has done.

In indeterminate mode the actual progress will not be shown, only the cyclic animation will be shown to indicate that some progress is happing like as shown in the above progress bar loading images.

By using progressBar.**setIndeterminate(true)** in activity file programmatically or using **android:indeterminate = "true"** attribute in XML layout file, we can enable **Indeterminate** progress mode.

```
<ProgressBar
    android:id="@+id/progressBar1"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true"/>
```

Attribute	Description
android:id	It is used to uniquely identify the control
android:max	It is used to specify the maximum value of the progress can take
android:progress	It is used to specify default progress value.
android:background	It is used to set the background color for a progress bar.
android:indeterminate	It is used to enable the indeterminate progress mode.
android:padding	It is used to set the padding for left, right, top or bottom of a progress bar.

AutoCompleteTextView

In Android, AutoCompleteTextView is a view i.e similar to EditText, except that it displays a list of completion suggestions automatically while the user is typing. A list of suggestions is displayed in drop down menu from which user can choose an item which actually replace the content of Editbox with that.

Generally, the dropdown list of suggestions can be obtained from the data adaptor and those suggestions will be appeared only after giving the number characters defined in the **Threshold** limit.

The **Threshold** property of AutoCompleteTextView is used to define the minimum number of characters the user must type to see the list of suggestions.

The dropdown list of suggestions can be closed at any time in case if no item is selected from the list or by pressing the **back** or **enter** key.

In android, we can create an **AutoCompleteTextView** control in two ways either in the XML layout file or create it in the Activity file programmatically.

```
<?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:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:orientation="vertical"
    android:id="@+id/linear_Layout">
    <AutoCompleteTextView
        android:id="@+id/ac_Country"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:hint="Enter Country Name"/>
</LinearLayout>
```

Attribute	Description
android:id	It is used to uniquely identify the control
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text.
android:hint	It is used to display the hint text when text is empty
android:textColor	It is used to change the color of the text.
android:textColorHint	It is used to change the text color of hint text.
android:textSize	It is used to specify the size of text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for autocomplete textview control
android:ems	It is used to make the textview be exactly this many ems wide.
android:width	It makes the TextView be exactly this many pixels wide.
android:height	It makes the TextView be exactly this many pixels tall.
android:textColorHighlight	It is used to change the color of the text selection highlight.
android:fontFamily	It is used to specify the fontFamily for the text.

To display the Array content in an autocompletetextview we need to implement Adapter. In AutoCompleteTextView we mainly display text values so we use Array Adapter for that.

ArrayAdapter In Android:

ArrayAdapter is used when we need list of single type of items which is backed by an Array. For example, list of phone contacts, countries or names.

Syntax:

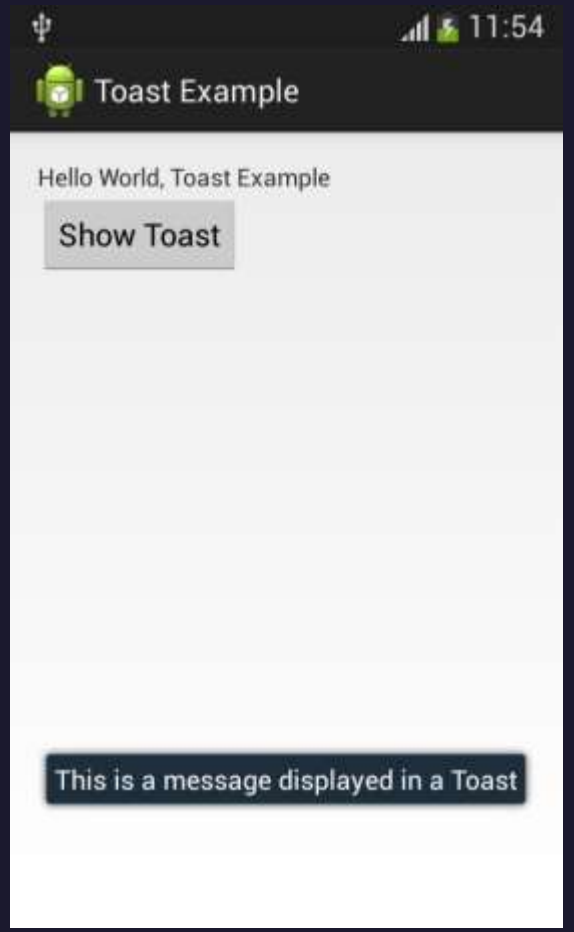
ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)

```
AutoCompleteTextView simpleAutoCompleteTextView = (AutoCompleteTextView)
findViewById(R.id.simpleAutoCompleteTextView);
```

Eg: ArrayAdapter adapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, countryNameList);


This is a Toast message

An Android **Toast** is a small message displayed on the screen, similar to a tool tip or other similar popup notification. A Toast is displayed on top of the main content of an activity, and only remains visible for a short time period. This screenshot shows how a Toast looks like on the screen:



Create a Toast in Android

In android, we can create a Toast by instantiating an **android.widget.Toast** object using **makeText()** method. The **makeText()** method will take three parameters: application context, text message and the duration for the toast. We can display the **Toast notification** by using **show()** method.

Following is the syntax of creating a **Toast** in android applications.

```
Toast.makeText(context, "message", duration).show();
```

Example

```
Toast.makeText(MainActivity.this, "Details Saved Successfully.", Toast.LENGTH_SHORT).show();
```

Parameter	Description
context	**It’s our application context.**
message	**It’s our custom message which we want to show in Toast notification.**
duration	**It is used to define the duration for notification to display on the screen.**

We have two ways to define the Toast **duration**, either in **LENGTH_SHORT** or **LENGTH_LONG** to display the toast notification for a short or longer period of time.

Constant	Description
public static final int LENGTH_LONG	displays view for the long duration of time.
public static final int LENGTH_SHORT	displays view for the short duration of time.

Android Toast Notification Example

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/btnShow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Toast"
        android:layout_marginTop="200dp"
android:layout_marginLeft="140dp"/>
</LinearLayout>
```

```
package com.tutlane.toastexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = (Button)findViewById(R.id.btnShow);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, "You Clicked on Button..",
Toast.LENGTH_SHORT).show();
            } } } }
```

Change the Position of Android Toast Notification

By default, the android Toast notification will always appear near the bottom of the screen, centered horizontally like as shown in the above image.

In case if we want to change the position of Toast notification, we can do it by using the **setGravity(int, int, int)** method. The **setGravity()** method will accept three parameters: a **Gravity** constant, an **x-position** offset, and a **y-position** offset.

Example

```
Toast toast =  Toast.makeText(MainActivity.this, "You Clicked on Button..", Toast.LENGTH_SHORT);
toast.setGravity(Gravity.TOP|Gravity.RIGHT, 100, 250);
toast.show();
```

ANDROID-ACTIVITY

In android, **Activity** represents a single screen with a user interface (UI) of an application and it will acts an entry point for users to interact with an app.

Generally, the android apps will contain multiple screens and each screen of our application will be an extension of Activity class. By using activities, we can place all our android application UI components in a single screen.

From the multiple activities in android app, one activity can be marked as a **main activity** and that is the first screen to appear when we launch the application. In android app each activity can start another activity to perform different actions based on our requirements.

To use activities in our application we need to define an activities with required attributes in manifest file

(**AndroidMainfest.xml**) like as shown below

```
<?xml version="1.0" encoding="utf-8"?>
<manifest …..>
   <application …..>
      <activity android:name=".MainActivity" >
        …….

        …….

      </activity>

    …….

</application>
</manifest>
```

In android application, activities can be implemented as a subclass of **Activity** class like as shown below.

```
public class MainActivity extends Activity
{
        // code
}
```

There are two methods almost all subclasses of Activity will implement:

1) onCreate(Bundle)

2) onPause()

onCreate(Bundle) is where you initialize your activity. Most importantly, here you will usually call **setContentView(int)** with a layout resource defining your UI, and using **findViewById(int)** to retrieve the widgets in that UI that you need to interact with programmatically

onPause() is where you deal with the user pausing active interaction with the activity. Any changes made by the user should at this point be committed (usually to the ContentProvider holding the data). In this state the activity is still visible on screen.

States of Activity

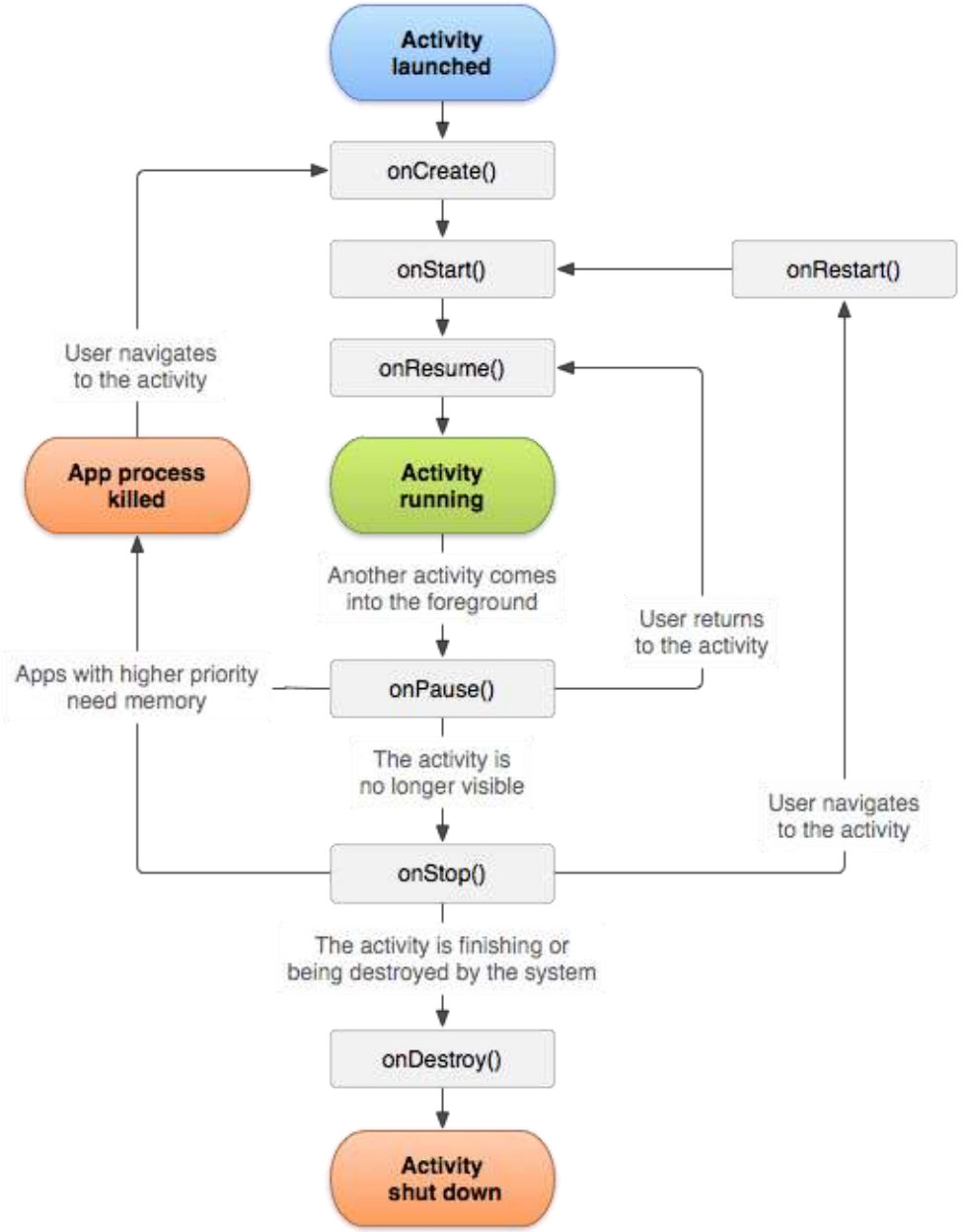
As a user navigates throughout an app, Android maintains the visited activities in a stack, with the currently visible activity always placed at the top of the stack.

At any point in time a particular activity can be in one of the following 4 states:

Activity State	Description
Running	Activity is visible and interacting with the user
Paused	Activity is still visible, but no longer interacting with the user
Stopped	Activity is no longer visible
Killed	Activity has been killed by the system (low memory) or its finish () method has been called

Android Activity Life Cycle

Any Android activity goes through a certain life cycle during its life inside the Android app. The following diagram shows the whole Activity lifecycle:



Generally, the activities in our android application will go through a different stages in their life cycle. In android, Activity class have 7 callback methods like **onCreate(), onStart(), onPause(), onRestart(), onResume(), onStop()** and **onDestroy()** to describe how the activity will behave at different stages.

Lifecycle Method	Description	Common Uses
onCreate()	The activity is starting (but not visible to the user)	Most of the activity initialization code goes here. This is where you setContentView() for the activity, initialize views, set up any adapters, etc.
onStart()	The activity is now visible (but not ready for user interaction)	This lifecycle method isn't used much, but can come in handy to register a BroadcastReceiver to monitor for changes that impact the UI (since the UI is now visible to the user).
onResume()	The activity is now in the foreground and ready for user interaction	This is a good place to start animations, open exclusive-access devices like the camera, etc.
onPause()	Counterpart to onResume(). The activity is about to go into the background and has stopped interacting with the user. This can happen when another activity is launched in front of the current activity.	It's common to undo anything that was done in onResume() and to save any global state (such as writing to a file).

onStop()	**Counterpart to onStart(). The activity is no longer visible to the user.**	**It's common to undo anything that was done in onStart().**
onDestroy()	Counterpart to onCreate(...). This can be triggered because finish() was called on the activity or the system needed to free up some memory.	It's common to do any cleanup here. For example, if the activity has a thread running in the background to download data from the network, it may create that thread in onCreate() and then stop the thread here in onDestroy()
onRestart()	Called when the activity has been stopped, before it is started again	It isn't very common to need to implement this callback.

INTENT

What is Intent in Android?

In Android, it is quite usual for users to witness a jump from one application to another as a part of the whole process, for example, searching for a location on the browser and witnessing a direct jump into Google Maps or receiving payment links in Messages Application (SMS) and on clicking jumping to PayPal or G-Pay (Google Pay). This process of taking users from one application to another is achieved by passing the **Intent** to the system. **Intents**, in general, are used for navigating among various activities within the same application, but note, is not limited to one single application, i.e., they can be utilized from moving from one application to another as well.

Android uses Intent for communicating between the components of an Application and also from one application to another application.

Intent are the objects which is used in android for passing the information among Activities in an Application and from one app to another also. Intent are used for communicating between the Application components and it also provides the connectivity between two apps.

In android, Intents are the objects of **android.content.Intent** types and intents are mainly useful to perform the following things.

Component	Description
Starting an Activity	By sending an **Intent** object to startActivity() method we can start a new Activity or existing Activity to perform required things.
Starting a Service	By sending an **Intent** object to startService() method we can start a new Service or send required instructions to an existing Service.
Delivering a Broadcast	By sending an **Intent** object to sendBroadcast() method we can deliver our messages to other app broadcast receivers.

Android Intent Structure

The primary pieces of information in an intent are:

action — The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.

data — The data to operate on, such as a person record in the contacts database, expressed as a URI (Uniform Resource Identifier).

category - Generally, the android category is optional for intents and it specifies the additional information about the type of component that should handle an intent. We can specify a category for intent by using addCategory().

The above properties will represent the characteristics of an intent.

Android Intent Types

Types of Android Intents

There are two types of intents in android:

1.Implicit and

2.Explicit.

Implicit Intent

In implicit intent target component name is not passed in the intent at the time of creating it.

The Android system decided itself which component of which application should receive this intent.

```
Intent i=new Intet(Intent.ACTION_VIEW);

i.setData(Uri.parse(https://amazon.in));

startActivity(i);
```

Explicit Intent

In explicit intent target component name is directly passed in the intent at the time of creating it.

```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("Value1", "This is ActivityTwo ");
i.putExtra("Value2", "This value two for ActivityTwo");
startActivity(i);
```

Intent Filter

An Intent filter is an expression in an app's manifest file that specifies the types of intents that the component would like to receive.

Syntax:

```
<intent-filter        android:icon="drawable resource"

                      android:label="String resource"

                      android:priority="integer"
</intent-filter>
```

Attributes:

1) android:icon

An icon that represents the parent activity, service, or broadcast receiver when that component is presented to the user as having the capability described by the filter.

This attribute must be set as a reference to a drawable resource containing the image definition.

The default value is the icon set by the parent component's icon attribute.

If the parent does not specify an icon, the default is the icon set by the <application> element.

2) android:label

A user-readable label for the parent component. This label, rather than the one set by the parent component, is used when the component is presented to the user as having the capability described by the filter.

The label should be set as a reference to a string resource, so that it can be localized like other strings in the user interface.

However, as a convenience while you're developing the application, it can also be set as a raw string.

The default value is the label set by the parent component.

If the parent does not specify a label, the default is the label set by the <application> element's label attribute.

3) android:priority

The priority that should be given to the parent component with regard to handling intents of the type described by the filter.

This attribute has meaning for both activities and broadcast receivers:

It provides information about how able an activity is to respond to an intent that matches the filter, relative to other activities that could also respond to the intent. When an intent could be handled by multiple activities with different priorities, Android will consider only those with higher priority values as potential targets for the intent.

It controls the order in which broadcast receivers are executed to receive broadcast messages. Those with higher priority values are called before those with lower values. (The order applies only to synchronous messages; it's ignored for asynchronous messages.)

Use this attribute only if you really need to impose a specific order in which the broadcasts are received, or want to force Android to prefer one activity over others.

The value must be an integer, such as "100". Higher numbers have a higher priority. The default value is 0.

In certain circumstances the requested priority is ignored and the value is capped to 0. This occurs when:

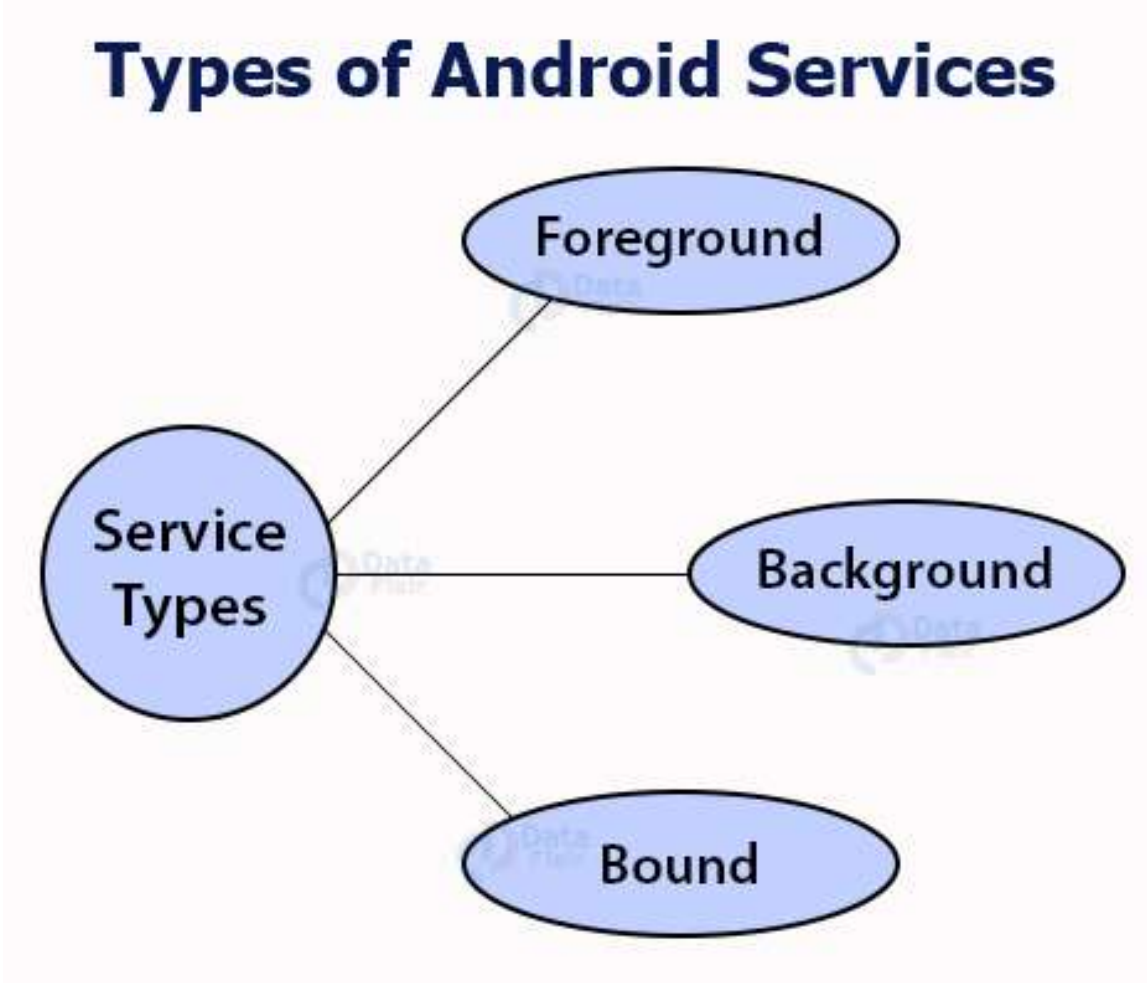
A non-privileged application requests any priority >0

A privileged application requests a priority >0 for ACTION_VIEW, ACTION_SEND, ACTION_SENDTO or ACTION_SEND_MULTIPLE

Android-Services

What are Android Services?

Service is basically a process. *Android service* is a component that runs in the background in order to perform long-running operations without interacting with the user and it works even if the application is destroyed. Another application component can start a service and it continues to run in the background even if you switch to another application.


Types of Android Services
Foreground
Service Types
Background
Bound

1. Foreground Services

Foreground services are those services that are visible to the users. The users can interact with them at ease and track what's happening. These services continue to run even when users are using other applications.

The perfect example of this is Music Player and Downloading.

2. Background Services

These services run in the background, such that the user can't see or access them. These are the tasks that don't need the user to know them.

Syncing and Storing data can be the best example.

3. Bound Services

Bound service runs as long as some other ***application component*** is bound to it. Many components can bind to one service at a time, but once they all unbind, the service will destroy.

To bind an application component to the service, **bindService()** is used.

Android- Services Life cycle

Android Services life cycle can have two forms of services. The lifecycle of a service follows two different paths, namely:

1.Started

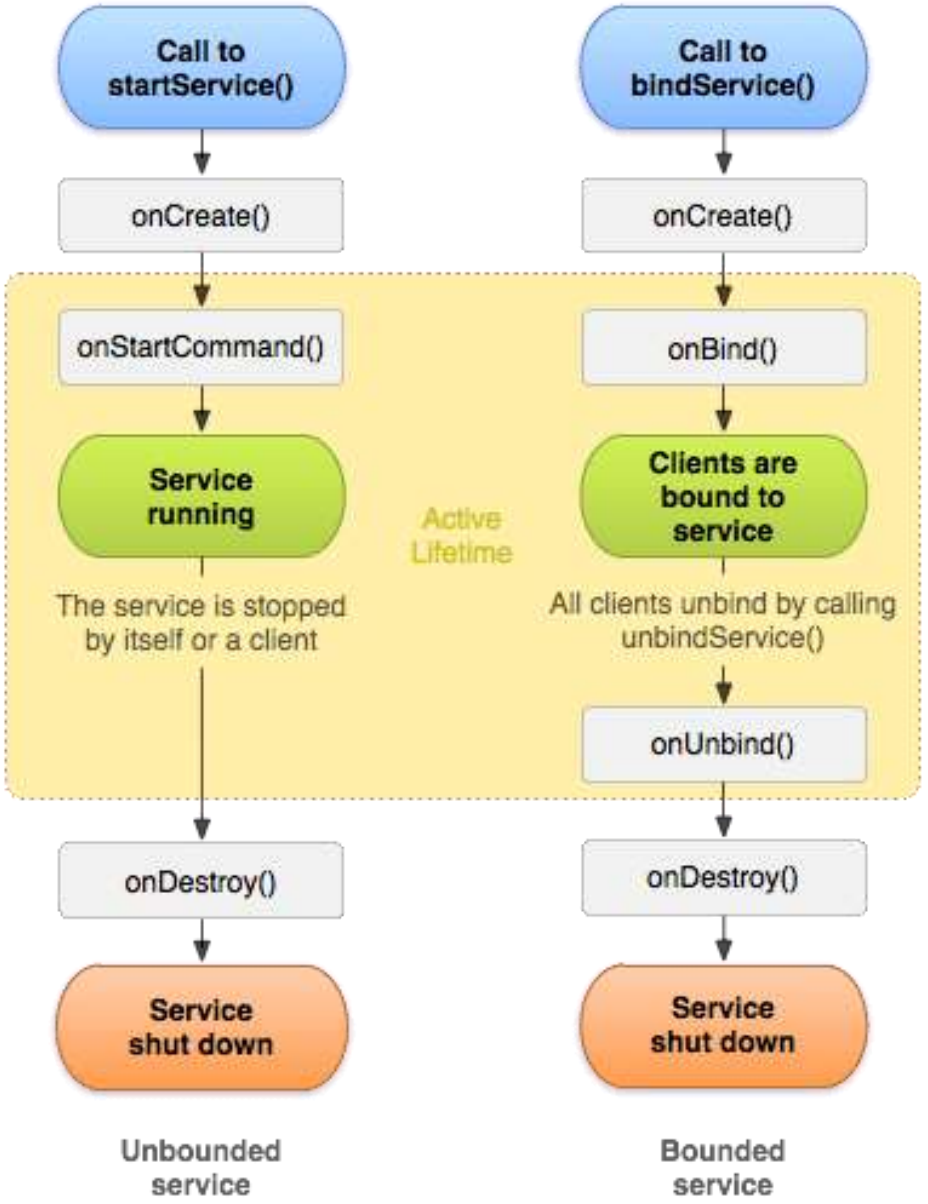
2.Bounded

Started Service

A service is started when an application component calls ***startService()*** method. Once started, a service can run in the background indefinitely, even if the component which is responsible for the start is destroyed. It is stopped by using the ***stopService()*** method. The service can also stop itself by calling the ***stopSelf()*** method.

Bound Service

A service is bound when an application component binds to it by calling ***bindService()*.** Bound service offers a client-server interface that allows components to interact with the service, send requests and, get results. It processes across inter-process communication (IPC). The client can unbind the service by calling the ***unbindService()*** method.


Call to startService()
onCreate()
onStartCommand()
Service running
The service is stopped by itself or a client
onDestroy()
Service shut down
Unbounded service
Call to bindService()
onCreate()
onBind()
Clients are bound to service
All clients unbind by calling unbindService()
onUnbind()
onDestroy()
Service shut down
Bounded service
Active Lifetime

Methods of Android Services

Method	Description
onStartCommand()	This method is called when any other component, like say an activity, requests the service to be started, by calling startService(). It is your responsibility to stop the service when the corresponding work is done by using stopSelf() or stopService() methods.
onBind()	Calls this method when another component wants to bind with the service by calling bindService(). To implement this, you must provide an interface that clients use in order to communicate with the service. It returns an IBinder object. If you don't want to allow binding, then return null.
onUnbind()	The system calls this method when all clients are disconnected from a particular interface published by the service.
onRebind()	Calls this method when new clients are connected to the service after it had previously been notified that all are disconnected in onUnbind(Intent).
onCreate()	The system calls this method when the service is created first using onStartCommand() or onBind(). It is required to perform a one-time set-up.
onDestroy()	This method is called when the service is no longer used and is being destroyed. Your service should implement this in order to clean up any resources such as threads, registered listeners, receivers, etc.

IntentService()

There's an additional service class, that extends Service class, **IntentService** Class. It is a base class for services to handle asynchronous requests. It enables running an operation on a single background. It executes long-running programs without affecting any user's interface interaction. Intent services run and execute in the background and terminate themself as soon as they are executed completely.

Certain important features of Intent are :

•It queues up the upcoming request and executes them one by one.

•Once the queue is empty it stops itself, without the user's intervention in its lifecycle.

•It does proper thread management by handling the requests on a separate thread.

TELEPHONEY & MESSAGING

Telephony is the field of technology involving the development, application, and deployment of telecommunication services for the purpose of electronic transmission of voice, fax, or data, between distant parties.

Android devices can send and receive messages to or from any other phone that supports Short Message Service (SMS).

Android offers the Messenger app that can send and receive SMS messages.

Sending SMS

In android, we can send SMS from our android application in two ways either by using SMSManager API or

Intents based on our requirements.

If we use SMSManager API, it will directly send SMS from our application.

In case if we use Intent with proper action (ACTION_VIEW), it will invoke a built-in SMS app to send SMS from

our application.

Android Send SMS using SMSManager API

In android, to send SMS using SMSManager API we need to write the code like as shown below.

```
SmsManager smgr = SmsManager.getDefault();

smgr.sendTextMessage(MobileNumber,null,Message,null,null);
```

SMSManager API required SEND_SMS permission in our android manifest to send SMS.

Following is the code snippet to set SEND_SMS permissions in manifest file.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Android Send SMS using Intent

In android, Intent is a messaging object which is used to request an action from another app component such as activities, services, broadcast receivers, and content providers. To know more about an Intent object in android check this Android Intents with Examples.

To send SMS using the Intent object, we need to write the code like as shown below.

```
Intent sInt = new Intent(Intent.ACTION_VIEW);
sInt.putExtra("address", new String[]{txtMobile.getText().toString()});
sInt.putExtra("sms_body",txtMessage.getText().toString());
sInt.setType("android-dir/mms-sms");
```

Even for Intent, it required a SEND_SMS permission in our android manifest to send SMS.
Following is the code snippet to set SEND_SMS permissions in manifest file.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Receiving SMS

To receive SMS messages, use the onReceive() method of the BroadcastReceiver class.

The Android framework sends out system broadcasts of events such as receiving an SMS message,

containing intents that are meant to be received using a BroadcastReceiver.

You need to add the RECEIVE_SMS permission to your app's AndroidManifest.xml file

Android Wi-Fi Activities

In android, **Wi-Fi** is a wireless network protocol that allows devices to connect to the internet or connect wirelessly with other devices to exchange the data.

Generally, in android applications by using Wi-Fi API's we can manage all aspects of WI-FI connectivity, such as a scan or search for available networks, add/save/delete Wi-Fi connections and managing the data transfer between devices within the range.

By using android Wi-Fi API's in android applications, we can perform following functionalities.

- Scan for the available Wi-Fi networks within the range
- Allow devices to connect to the internet
- Connect to other devices through service discovery
- Manage list of configured networks.
- Manage multiple connections

Android Set Wi-Fi Permissions

To use Wi-Fi features in our android applications, we must need to add multiple permissions, such as CHANGE_WIFI_STATE, ACCESS_WIFI_STATE and INTERNET in our manifest file.

Following is the example of defining the Bluetooth permissions in android manifest file.

```
<manifest ... >
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
...

</manifest>
```

Android Wi-fi Manager Class

In android, we can perform Wi-Fi related activities by using WifiManager class in our applications. This class will provide required API’s to manage all aspects of Wi-Fi connectivity.

By using WifiManager class, we can perform the operations that are related to network connectivity. We can instantiate this class by using Context.getSystemService(Class) with the argument WifiManager.class or Context.getSystemService(String) with the argument Context.WIFI_SERVICE.

Following is the code snippet to initialize WifiManager class using Context.getSystemService(String) with the argument Context.WIFI_SERVICE.

```
WifiManager wmgr = (WifiManager)Context.getSystemService(Context.WIFI_SERVICE);
```

If you observe above code snippet, we used getSystemService() method to instantiate a WifiManager class.

In case if getSystemService() method returns NULL, then the device does not support Wi-Fi and we can disable all Wi-Fi features.

JSON AND XML

What is XML…???

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. XML tags are not predefined in XML. We must define our own Tags. Xml as itself is well readable both by human and machine. Also, it is scalable and simple to develop. In Android we use xml for designing our layouts because xml is lightweight language so it doesn’t make our layout heavy

Example

```
<employees>
  <employee>
    <firstName>Amal</firstName> <lastName>Thomas</lastName>
  </employee>
  <employee>
    <firstName>Jithin</firstName> <lastName>Saju</lastName>
  </employee>
  <employee>
    <firstName>Anto</firstName> <lastName>Shaji</lastName>
  </employee>
</employees>
```

What is JSON?

JSON (JavaScript Object Notation) is a text based file format that is used for sharing data between applications in a standard format supported by most programming languages and platforms. In addition to being programming language and platform independent JSON is also human readable

A JSON file is made up of a JSON object enclosed in curly brackets “{” and “}” and consists of attribute name and value pairs. Attribute names are Strings enclosed in double quotation marks and are associated to a value with a colon “:” in between. Values can be of different formats such as a number, String, Boolean, array, a nested JSON object or null. Attribute and value pairs are comma “,” separated

Example

```
{
"users": [
{
"name": “Aamir Khan",
"designation": "Team Leader",
"location": “Kochi"
},
{
"name": "Rohini Prakash",
"designation": "Agricultural Officer",
"location": “Kollam"
}
]
}
```

Generally, the JSON nodes will start with a **square bracket ([)** or with a **curly bracket ({)**. The difference between square bracket and curly bracket is, the **square bracket ([)** represents the starting of a **JSONArray** node, whereas **curly bracket ({)** represents a **JSONObject** so we need to call appropriate method to get the data.

JSON is Like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XML HttpRequest

JSON is Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

The biggest difference is:

XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

Why JSON is Better Than XML ?

XML is much more difficult to parse than JSON.

JSON is parsed into a ready-to-use JavaScript object.

JSON Data Types

Number
double- precision floating-point format in JavaScript

String
double-quoted Unicode with backslash escaping

Boolean
true or false

Array
an ordered sequence of values

Object
an unordered collection of key: value pairs

null
empty

JSON - Objects

•It is an unordered set of name/value pairs.

•Objects are enclosed in curly braces that is, it starts with '{' and ends with '}'.

•Each name is followed by ':'(colon) and the key/value pairs are separated by , (comma).

•The keys must be strings and should be different from each other.

•Objects should be used when the key names are arbitrary strings.

Creating Objects

1. var JSONObj = {};

2. var JSONObj = new Object();

3. var JSONObj = { "bookname “ : "VB BLACK BOOK", "price":500 };

Accessing Object Values

1. By using dot (.) notation

2. By using bracket ([]) notation

Example

1. str= JSONObj.bookname;

2. str=JSONObj[“bookname”];

JSON Arrays

The JSON Arrays is similar to JavaScript Arrays.

Syntax of Arrays in JSON Objects:

JSON Arrays Syntax

```
{
    "name":"Aamir Khan",
    "heroName": "Spiderman",
    "friends" : ["Deadpool", "Hulk", "Wolverine"]
}
```

Accessing Array Values:

The Array values can be accessed using the index of each element in an Array.

```
var x = JSONArrayname.friends[0];

    document.getElementById("paraId").innerHTML = x;
```

JSON	XML
JSON object has a type	XML data is typeless
JSON types: string, number, array, Boolean	All XML data should be string
Data is readily accessible as JSON objects	XML data needs to be parsed.
JSON is supported by most browsers.	Cross-browser XML parsing can be tricky
JSON has no display capabilities.	XML offers the capability to display data because it is a markup language.

JSON supports only text and number data type.	XML support various data types such as number, text, images, charts, graphs, etc. It also provides options for transferring the structure or format of the data with actual data.
Retrieving value is easy	Retrieving value is difficult
Supported by many Ajax toolkit	Not fully supported by Ajax toolkit
A fully automated way of deserializing/serializing JavaScript.	Developers have to write JavaScript code to serialize/de-serialize from XML
Native support for object.	The object has to be express by conventions - mostly missed use of attributes and elements.
It supports only UTF-8 encoding.	It supports various encoding.
It doesn't support comments.	It supports comments.
JSON files are easy to read as compared to XML.	XML documents are relatively more difficult to read and interpret.
It does not provide any support for namespaces.	It supports namespaces.
It is less secured.	It is more secure than JSON.

Functions for Parsing

getInt(int index): This function is used to get the value that is of integer type at a particular index.
If not present then you will get JSONException.

getDouble(int index): This function is used to get the value that is of double type at a particular index.
If not present then you will get JSONException.

getBoolean(int index): This function is used to get the value that is of boolean type at a particular index.
If not present then you will get JSONException.

getLong(int index): This function is used to get the value that is of long type at a particular index.
If not present then you will get JSONException.

getString(int index): This function is used to get the value that is of string type at a particular index.
If not present then you will get JSONException.

getJSONArray(int index): This function is used to get the value that is of array type at a particular index i.e. we can have a JSON array in JSONObject.
If the array is not present then you will get JSONException.

getJSONObject(int index): This function is used to get the value that is of object type at a particular index i.e.
we can have a JSON object in JSONObject. If the object is not present then you will get JSONException.

length(): This function is used to find the total number of values present in the JSON file.

Example:

Parse the following JSON file

```
{
        "FirstObject":
                    { "attr1":"one value" ,"attr2":"two value",

                        "sub": { "sub1":[
{"sub1_attr":"sub1_attr_value" },{"sub1_attr":"sub2_attr_value" }]}
                }
}
```

Step – 1:

create a JSONObject with the received response string:

```
JSONObject   jsonObject = new JSONObject(strJSONResponse);
```

Step – 2:

Get the main object from the created json object by using getJSONObject() method:

```
JSONObject object = jsonObject.getJSONObject("FirstObject");
```

Step – 3:
Now this FirstObject contains 2 strings namely “attr1”,”attr2” and a object namely “sub”.
So get 2 strings by using getString() method.

```
String attr1 = object.getString("attr1");

String attr2 = object.getString("attr2");

JSONObject subObject = object.getJSONObject("sub");
```

Step – 4:

Now this “sub” sub-object contains 1 array namely “sub1”. So we can get this JSON array by using getJSONArray() method:

```
JSONArray subArray = subObject.getJSONArray("sub1");



for(int i=0; i<subArray.length(); i++)
{

strParsedValue+= subArray.getJSONObject(i).getString("sub1_attr");
}
```