Subject Name: Mobile Computing and Wireless Communication
Subject Code: 3170710

Faculty: Alpa Rupala, Ms. Kinjal Parmar

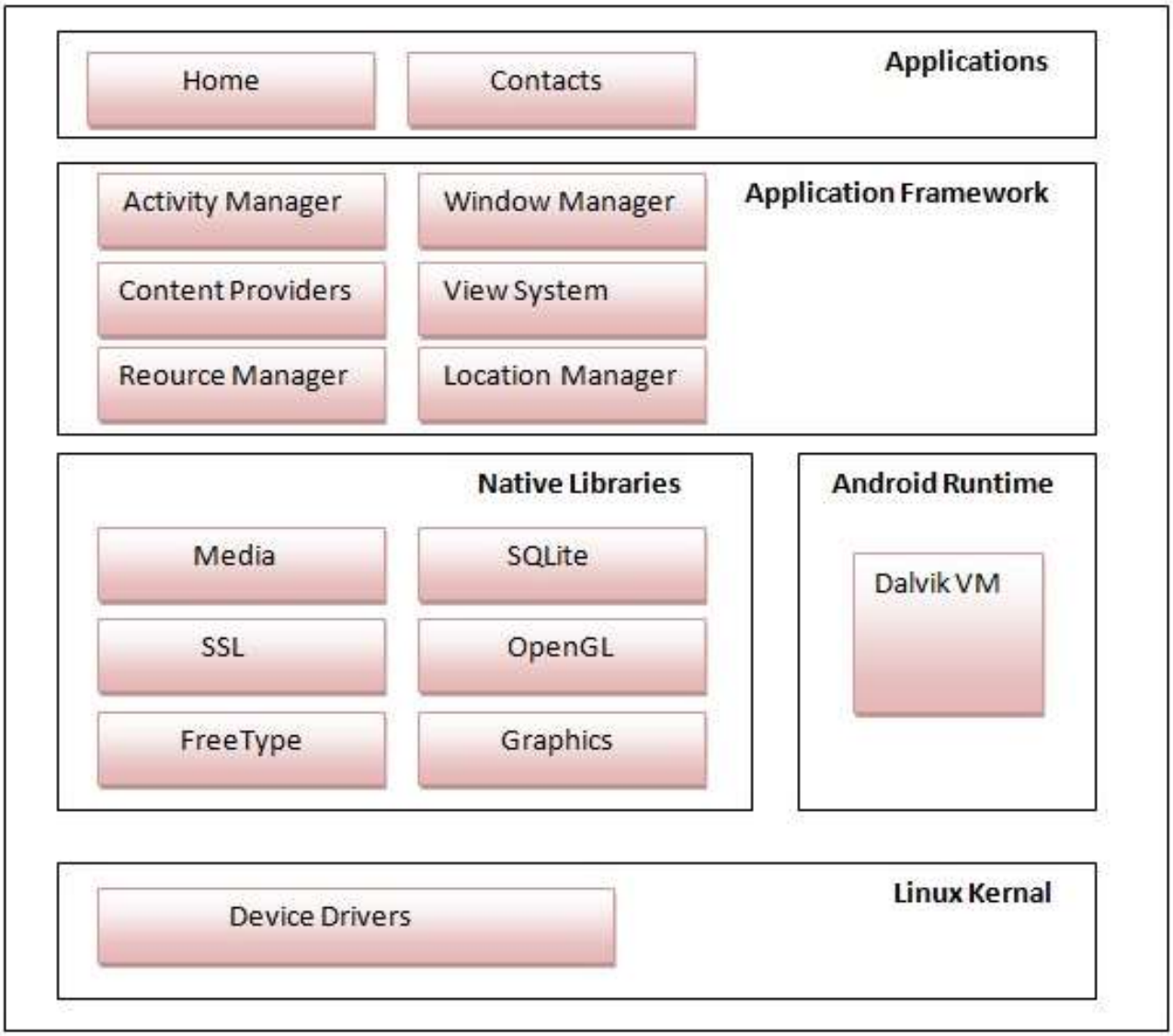
CHAPTER NO -6: Android:

DESCRIPTIVE QUESTIONS

1. Draw Android Architecture. Also explain Android Application Framework in brief. **(Nov-2017) [L.J.I.E.T]** 7
Explain the Android Architecture with the neat diagram. **(May-2018) [L.J.I.E.T]** ,
4

android architecture or **Android software stack** is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications



1) Linux kernel
It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries
On the top of linux kernel, their are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.
The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

- Provides access to the application model and is the cornerstone of all Android applications.
- **android.content** − Facilitates content access, publishing and messaging between applications and application components.
- **android.database** − Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** − Used to render and manipulate text on a device display.
- **android.view** − The fundamental building blocks of application user interfaces.
- **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

3) Android Runtime
In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework
On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

The Android framework includes the following key services −

- **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** − Allows applications to publish and share data with other applications.
- **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** − Allows applications to display alerts and notifications to the user.
- **View System** − An extensible set of views used to create application user interfaces.

5) Applications
On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

2.	Explain Android application framework with their components. **(May-2017) [L.J.I.E.T]**	7

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application −

Sr.No	Components & Description
1	**Activities** They dictate the UI and handle the user interaction to the smart phone screen.
2	**Services** They handle background processing associated with an application.
3	**Broadcast Receivers** They handle communication between Android OS and applications.
4	**Content Providers** They handle data and database management issues.

- **Activities**

  An activity represents a single screen with a user interface,in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

  An activity is implemented as a subclass of **Activity** class as follows −

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

- **Services**

  A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application,

or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of **Service** class as follows −

```
public class MyService extends Service {   }
```

- **Broadcast Receivers**
  Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

  A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

```
public class MyReceiver  extends  BroadcastReceiver {
  public void onReceive(context,intent){}}
```

- **Content Providers**
  A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

  A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends  ContentProvider {
  public void onCreate(){} }
```

Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are −

S.No	Components & Description
1	**Fragments** Represents a portion of user interface in an Activity.
2	**Views** UI elements that are drawn on-screen including buttons, lists forms etc.
3	**Layouts** View hierarchies that control screen format and appearance of the views.
4	**Intents** Messages wiring components together.
5	**Resources** External elements, such as strings, constants and drawable pictures.

6	**Manifest** Configuration file for the application.

- **View**
  A view is the UI element such as button, label, text field etc. Anything that you see is a view.

- **Intent**
  Intent is used to invoke components. It is mainly used to:
  - Start the service
  - Launch an activity
  - Display a web page
  - Display a list of contacts
  - Broadcast a message
  - Dial a phone call etc.

- **Fragment**
  Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

- **AndroidManifest.xml**
  It contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

- **Android Virtual Device (AVD)**
  It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices

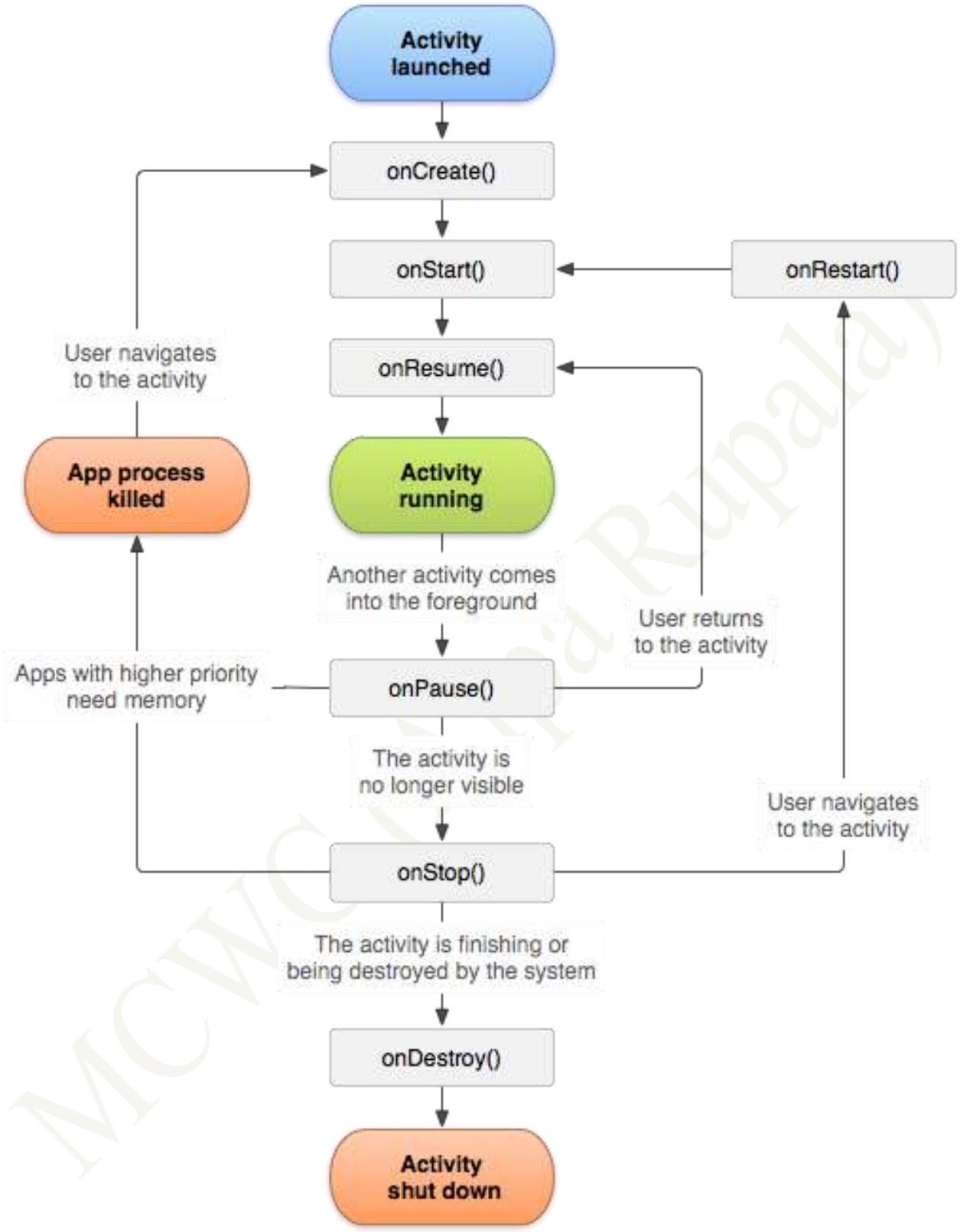
3. Explain Lifecycle of android API. **[L.J.I.E.T]** 7

- **Android Activity Lifecycle** is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.
- An activity is the single screen in android. It is like window or frame of Java.
- By the help of activity, you can place all your UI components or widgets in a single screen.
- The 7 lifecycle method of Activity describes how activity will behave at different states.

- **Android Activity Lifecycle methods**

Let's see the 7 lifecycle methods of android activity.

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.



onCreate():

Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

onRestart():

Called after your activity has been stopped, prior to it being started again. Always followed by onStart()

onStart():

Called when the activity is becoming visible to the user. Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.

onResume():

Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by onPause().

onPause ():

Called as part of the activity lifecycle when an activity is going into the background, but has not (yet) been killed. The counterpart to onResume(). When activity B is launched in front of activity A, this callback will be invoked on A. B will not be created until A's onPause() returns, so be sure to not do anything lengthy here.

onStop():

Called when you are no longer visible to the user. You will next receive either onRestart(), onDestroy(), or nothing, depending on later user activity.

Note that this method may never be called, in low memory situations where the system does not have enough memory to keep your activity's process running after its onPause() method is called.

onDestroy():

The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called finish() on it, or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the isFinishing() method.

When the Activity **first time loads** the events are called as below:

onCreate()
onStart()
onResume()

When you **click on Phone button** the Activity goes to the background and the below events are called:

onPause()
onStop()

Exit the phone dialer and the below events will be called:

onRestart()
onStart()
onResume()

When you click the **back button** OR try to **finish()** the activity the events are called as below:

onPause()
onStop()
onDestroy()

4. Define Android layout. Explain various Android layouts. **[New](Nov-2016) (May-2018) [L.J.I.E.T]** 7, 7

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

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

At third level we have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup**objects or you can declare your layout using simple XML file **main_layout.xml** which is located in the res/layout folder of your project.

Android Layout Types

There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

Sr.No	Layout & Description
1	**Linear Layout** LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
2	**Relative Layout** RelativeLayout is a view group that displays child views in relative positions.
3	**Table Layout** TableLayout is a view that groups views into rows and columns.
4	**Absolute Layout** AbsoluteLayout enables you to specify the exact location of its children.
5	**Frame Layout** The FrameLayout is a placeholder on screen that you can use to display a single view.
6	**List View** ListView is a view group that displays a list of scrollable items.
7	**Grid View** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

Layout Attributes

Each layout 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:

Sr.No	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:layout_width** This is the width of the layout.
13	**android:layout_width** This is the width of the layout.
14	**android:paddingLeft** This is the left padding filled for the layout.
15	**android:paddingRight** This is the right padding filled for the layout.
16	**android:paddingTop** This is the top padding filled for the layout.

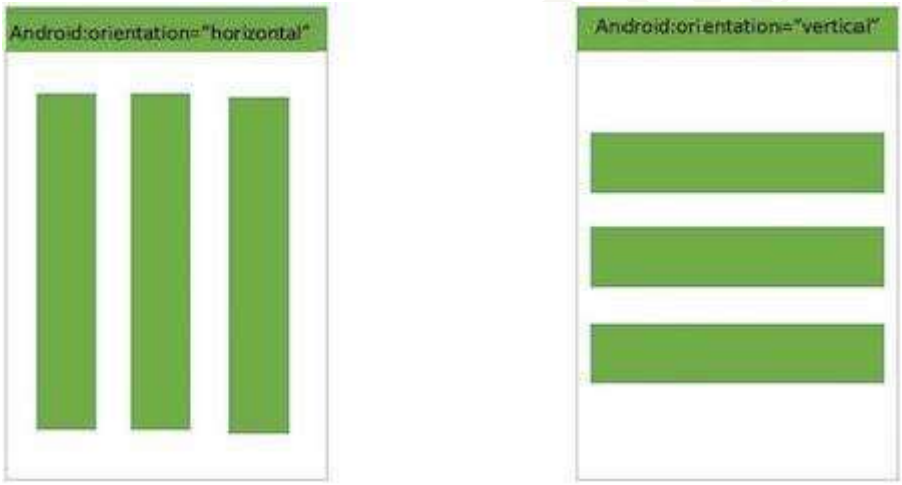
17	**android:paddingBottom** This is the bottom padding filled for the layout.

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 finally 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=fill_parent** tells your view to become as big as its parent view.

Linear Layout



A Layout that arranges its children in a single column or a single row.

android:gravity

- This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.

android:orientation

- This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal.

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

    <Button android:id="@+id/btnA"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Button A"/>

        <Button android:id="@+id/btnB"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Button B"/>
    </LinearLayout>
```

- **Relative Layout**
  - RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or center).
  - **android:layout_alignParentTop**
    If "true", makes the top edge of this view match the top edge of the parent.
  - **android:layout_centerVertical**
    If "true", centers this child vertically within its parent.
  - **android:layout_below**
    Positions the top edge of this view below the view specified with a resource ID
  - **android:layout_toRightOf**
    Positions the left edge of this view to the right of the view specified with a resource ID.

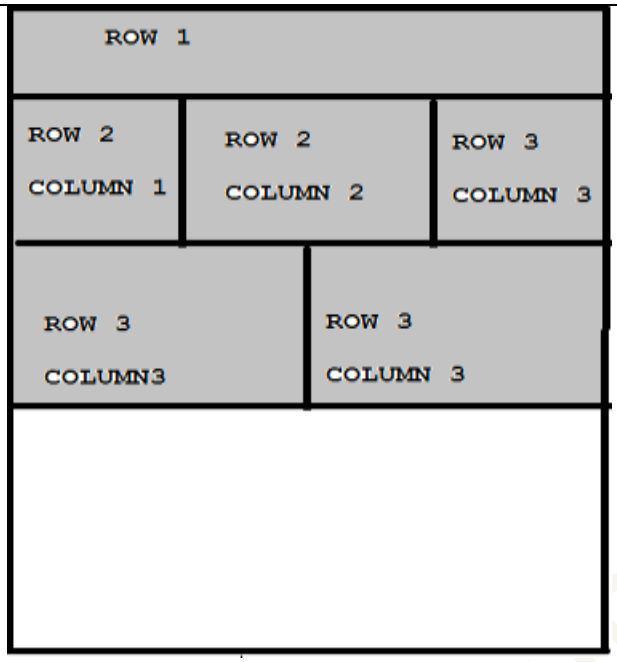
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent”>

    <Spinner android:id="@id/SpA" android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true" />

    <Button android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/SpA"
```

```
                android:layout_alignParentRight="true"/>
        </RelativeLayout>
```

- **ScrollView**
  - **ScrollView** is a special kind of layout, designed to hold view larger than its actual size. When the Views size goes beyond the ScrollView size, it automatically adds scroll bars and can be scrolled vertically.
  - ScrollView can hold only one direct child. This means that, if you have complex layout with more view controls, you must enclosethem inside anotherstandard layout like LinearLayout, TableLayout or RelativeLayout.
  - You can specify layout_height and layout_width to adjust height and width of screen.
  - Scrollview is ideal for screens where scrolling is required, but it is an overhead when scroll view is used to render a larger list of data. Android provides specialized adapter views like ListView, GridView are recommended for long lists.
  - You should never use a ScrollView with a ListView or GridView, because they both takes care of their own vertical scrolling.
  - ScrollView only supports vertical scrolling. Use HorizontalScrollView for horizontal scrolling.

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

                <ImageView android:id="@+id/imageView"
                    android:layout_width="wrap_content"
                    android:layout_height="200dp"
                    android:scaleType="centerCrop"
                    android:src="@drawable/image" />

                <TextView android:id="@+id/textView"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/description"/>
            </LinearLayout>
        </ScrollView>
```

- **Table Layout**
  - Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

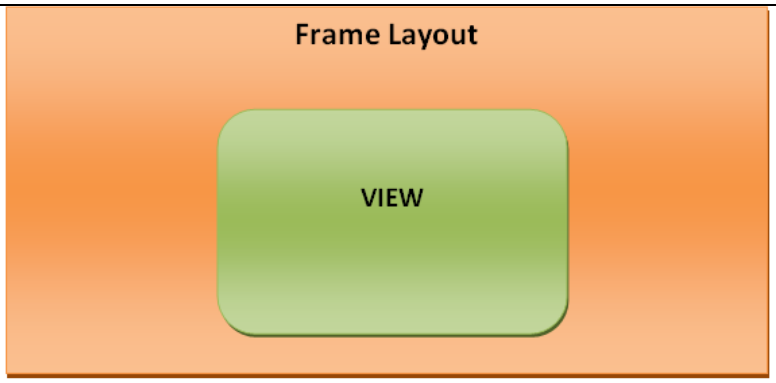


- **android:collapseColumns**
  This specifies the zero-based index of the columns to collapse.
- **android:collapseColumns**
  The zero-based index of the columns to shrink.
- **android:stretchColumns**
  The zero-based index of the columns to stretch.

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

 <!-- Row 2 with 3 columns -->
   <TableRow android:id="@+id/tableRow2"
   android:layout_height="wrap_content"
   android:layout_width="match_parent">

   <TextView android:id="@+id/TextView01" android:text="Row 2 column
   1" android:layout_weight="1"/>

   <TextView android:id="@+id/TextView02" android:text="Row 2 column
   2" android:layout_weight="1" />

   <TextView android:id="@+id/TextView03" android:text="Row 2 column
   3" android:layout_weight="1" />

            </TableRow>

   </TableLayout>
```

- **FrameLayout**

  Frame Layout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other.



- **android:foreground**
  This defines the drawable to draw over the content and possible values may be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
- **android:foregroundGravity**
  Defines the gravity to apply to the foreground drawable. The gravity defaults to fill. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
- **android:measureAllChildren**
  Determines whether to measure all children or just those in the VISIBLE or INVISIBLE state when measuring. Defaults to false.

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

        <ImageView android:src="@drawable/ic_launcher"
            android:scaleType="fitCenter"
            android:layout_height="250px"
            android:layout_width="250px"/>

         <TextView
            android:text="Frame Demo"
            android:layout_height="wrap_parent"
            android:layout_width="wrap_parent"
            android:gravity="center"/>
        </FrameLayout>
```

5. Explain Android EditText and TextView control with an example.**[New](Nov-2016)[L.J.I.E.T]** 4

➢ **Textview**

A **TextView** displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing.

TextView Attributes

Following are the important attributes related to TextView control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Sr.No.	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the control.
2	**android:capitalize** If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types. • Don't automatically capitalize anything - 0 • Capitalize the first word of each sentence - 1 • Capitalize the first letter of every word - 2 • Capitalize every character – 3
3	**android:cursorVisible** Makes the cursor visible (the default) or invisible. Default is false.
4	**android:editable** If set to true, specifies that this TextView has an input method.
5	**android:fontFamily** Font family (named by string) for the text.
6	**android:gravity** Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view.
7	**android:hint** Hint text to display when the text is empty.
8	**android:inputType** The type of data being placed in a text field. Phone, Date, Time, Number, Password etc.
9	**android:maxHeight** Makes the TextView be at most this many pixels tall.
10	**android:maxWidth** Makes the TextView be at most this many pixels wide.
11	**android:minHeight** Makes the TextView be at least this many pixels tall.

12	**android:minWidth** Makes the TextView be at least this many pixels wide.
13	**android:password** Whether the characters of the field are displayed as password dots instead of themselves. Possible value either "true" or "false".
14	**android:phoneNumber** If set, specifies that this TextView has a phone number input method. Possible value either "true" or "false".
15	**android:text** Text to display.
16	**android:textAllCaps** Present the text in ALL CAPS. Possible value either "true" or "false".
17	**android:textColor** Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
18	**android:textColorHighlight** Color of the text selection highlight.
19	**android:textColorHint** Color of the hint text. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
20	**android:textIsSelectable** Indicates that the content of a non-editable text can be selected. Possible value either "true" or "false".
21	**android:textSize** Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).
22	**android:textStyle** Style (bold, italic, bolditalic) for the text. You can use or more of the following values separated by '|'. • normal - 0 • bold - 1

	• italic – 2
23	**android:typeface** Typeface (normal, sans, serif, monospace) for the text. You can use or more of the following values separated by '|'. • normal - 0 • sans - 1 • serif - 2 • monospace – 3

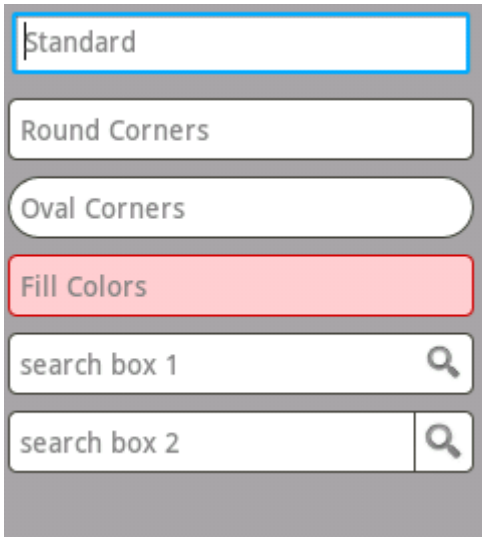
Example of TextView

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<TextView android:id="@+id/text_id" android:layout_width="300dp"
android:layout_height="200dp" android:text="hello_world"
android:textColor="@android:color/holo_blue_dark" android:textSize="50dp"/>
</RelativeLayout
```

❖ EditText

Definition

An EditText is an overlay over TextView that configures itself to be editable. It is the predefined subclass of TextView that includes rich editing capabilities.



STYLES OF EDIT TEXT

EditText Attributes

Following are the important attributes related to EditText control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Inherited from **android.widget.TextView** Class −

Sr.No	Attribute & Description
1	**android:autoText** If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
2	**android:drawableBottom** This s the drawable to be drawn below the text.
3	**android:drawableRight** This is the drawable to be drawn to the right of the text.
4	**android:editable** If set, specifies that this TextView has an input method.
5	**android:text** This is the Text to display.

Inherited from **android.view.View** Class −

Sr.No	Attribute & Description
1	**android:background** This is a drawable to use as the background.
2	**android:contentDescription** This defines text that briefly describes content of the view.
3	**android:id** This supplies an identifier name for this view.
4	**android:onClick** This is the name of the method in this View's context to invoke when the view is clicked.

5	**android:visibility** This controls the initial visibility of the view.

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<EditText android:id="@+id/edittext"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/button"
android:layout_below="@+id/textView1
" android:layout_marginTop="61dp"
android:ems="10"
android:text="@string/enter_text"android:inputType="text"/>
</RelativeLayout>
```

❖ Button View

A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.

Button Attributes

Following are the important attributes related to Button control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Inherited from **android.widget.TextView** Class −

Sr.No	Attribute & Description
1	**android:autoText** If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
2	**android:drawableBottom** This is the drawable to be drawn below the text.
3	**android:drawableRight** This is the drawable to be drawn to the right of the text.
4	**android:editable** If set, specifies that this TextView has an input method.
5	**android:text** This is the Text to display.

Inherited from **android.view.View** Class −

Attribute	Description
1	**android:background** This is a drawable to use as the background.
2	**android:contentDescription** This defines text that briefly describes content of the view.
3	**android:id** This supplies an identifier name for this view.
4	**android:onClick** This is the name of the method in this View's context to invoke when the view is clicked.

5	**android:visibility** This controls the initial visibility of the view.

Example

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android
" xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Button"
android:id="@+id/button"
android:layout_alignTop="@+id/editText"
android:layout_alignLeft="@+id/textView1"
android:layout_alignStart="@+id/textView1"
android:layout_alignRight="@+id/editText"
android:layout_alignEnd="@+id/editText" />
</RelativeLayout>
```

❖ Radio Button

RadioButton is a two states button which is either checked or unchecked. If a single radio button is unchecked, we can click it to make checked radio button. Once a radio button is checked, it cannot be marked as unchecked by user.

RadioButton is generally used with *RadioGroup*. RadioGroup contains several radio buttons, marking one radio button as checked makes all other radio buttons as unchecked.

RadioGroup Attributes

Following are the important attributes related to RadioGroup control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Attribute	Description
android:checkedButton	This is the id of child radio button that should be checked by default within this radio group.

Inherited from **android.view.View** Class −

Sr.No.	Attribute & Description
1	**android:background** This is a drawable to use as the background.
2	**android:contentDescription** This defines text that briefly describes content of the view.
3	**android:id** This supplies an identifier name for this view
4	**android:onClick** This is the name of the method in this View's context to invoke when the view is clicked.
5	**android:visibility** This controls the initial visibility of the view.

Example of Radio Button

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

<RadioGroup android:layout_width="fill_parent" android:layout_height="90dp"
android:layout_below="@+id/imageView" android:layout_marginTop="58dp"
android:weightSum="1" android:id="@+id/radioGroup">

<RadioButton android:layout_width="wrap_content" android:layout_height="55dp"
android:text="Male" android:id="@+id/radioButton"
android:layout_gravity="center_horizontal" android:checked="false"
android:textSize="25dp"/>

<RadioButton android:layout_width="wrap_content" android:layout_height="wrap_content"
```

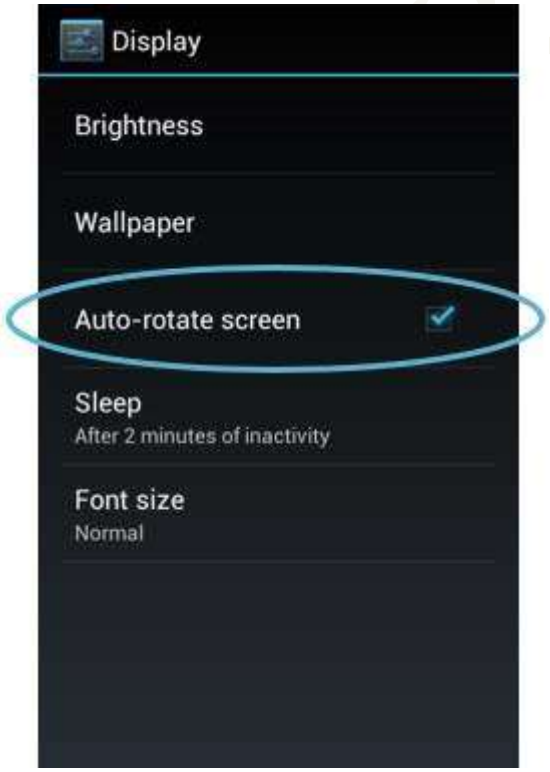
```
android:text="Female" android:id="@+id/radioButton2"
android:layout_gravity="center_horizontal" android:checked="false" android:textSize="25dp"
android:layout_weight="0.13"/>
</RadioGroup>
</RelativeLayout>
```

❖ CheckBox

Android CheckBox is a type of two state button either checked or unchecked.

There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

Android CheckBox class is the subclass of CompoundButton class.



CheckBox Attributes

Following are the important attributes related to CheckBox control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Inherited from **android.widget.TextView** Class −

Sr.No	Attribute & Description

1	**android:autoText** If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
2	**android:drawableBottom** This is the drawable to be drawn below the text.
3	**android:drawableRight** This is the drawable to be drawn to the right of the text.
4	**android:editable** If set, specifies that this TextView has an input method.
5	**android:text** This is the Text to display.

Inherited from **android.view.View** Class −

Sr.No	Attribute & Description
1	**android:background** This is a drawable to use as the background.
2	**android:contentDescription** This defines text that briefly describes content of the view.
3	**android:id** This supplies an identifier name for this view.
4	**android:onClick** This is the name of the method in this View's context to invoke when the view is clicked.
5	**android:visibility** This controls the initial visibility of the view.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
android:layout_height="match_parent" tools:context=".MainActivity">
<CheckBox android:id="@+id/checkBox1" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Do you like Tutorials Point"
```

```
android:layout_above="@+id/button"
android:layout_centerHorizontal="true"/>

<CheckBox android:id="@+id/checkBox2" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Do you like android "
android:checked="false" android:layout_above="@+id/checkBox1"
android:layout_alignLeft="@+id/checkBox1"
android:layout_alignStart="@+id/checkBox1"/>
</RelativeLayout>
```

❖ Image View

Definition

- An ImageButton is an AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.

ANDROID BUTTON STYLE SET

ImageButton Attributes

Following are the important attributes related to ImageButton control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Inherited from **android.widget.ImageView** Class –

Sr.No	Attribute & Description
1	**android:adjustViewBounds** Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
2	**android:baseline** This is the offset of the baseline within this view.
3	**android:baselineAlignBottom** If true, the image view will be baseline aligned with based on its bottom edge.
4	**android:cropToPadding** If true, the image will be cropped to fit within its padding.
5	**android:src** This sets a drawable as the content of this ImageView.

Inherited from **android.view.View** Class −

Sr.No	Attribute & Description
1	**android:background** This is a drawable to use as the background.
2	**android:contentDescription** This defines text that briefly describes content of the view.
3	**android:id** This supplies an identifier name for this view
4	**android:onClick** This is the name of the method in this View's context to invoke when the view is clicked.
5	**android:visibility** This controls the initial visibility of the view.

Example

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

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"android:layout_width="match_parent"
android:layout_height="match_parent"
```

```
tools:context=".MainActivity">
<ImageButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageButton"
android:layout_centerVertical="true"
android:layout_centerHorizontal="true"
android:src="@drawable/abc"/>
</RelativeLayout>
```