UNIT - III

ANDROID ACTIVITIES & GUI DESIGN CONCEPTS

3.1 Design Criteria for Android

- 3.1.1 Hardware-imposed Design Consideration

- Small and portable, mobile devices offer exciting opportunities for software development. Their limited screen size and reduced memory, storage, and processor power are far less compare to current Desktop computers, and instead present some unique challenges:

  - **Be Efficient**
  - **Expect Limited Capacity**
  - **Design for Small Screens**
  - **Expect Low Speeds, High Latency**
  - **Low Cost**

3.1 Design Criteria for Android (Cont.)

- 3.1.2 Design Demand for Android Application

- The Android design philosophy demands that applications should be designed for:

  - **Performance**
  - **Responsive**
  - **Security**
  - **Seamless User Experience**

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.1 Activity

- An android application is usually consists of one or more activities, similar to a forms in window application, that are loosely bound to each other.

- In Android, an activity is made up of a UI component for example, activity_main.xml and a class component for example, MainActivity.java.

- Each activity is given a window in which to draw its user interface. Activities use Views like TextView, Button etc. to form graphical user interfaces that display information and interact with user actions.

- Typically, one activity in an application is specified as the "main" activity, which is presented to the user when the application is launched. Each activity can then start another activity in order to perform different actions. You require to create a new Activity for every screen you want to display.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.1 Activity

- An Activity can be created by extending Activity class. Within this new class you must define the user interface and implement your functionality. The basic skeleton code for a new Activity is shown in below code.

```
public class MyActivity extends Activity
{
    /** Called when the activity is first created. */

     @Override
    public void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
    }
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.1 Activity

- Activity begins its execution, by making call to its **onCreate()** method. To assign a user interface to an Activity, it is required to call **setContentView()** method within the onCreate() method of your Activity. There are two ways to build user interface for the activity:
  - Using xml file
  - Using program code

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **Creating User Interface Using xml file :**
- In this approach user interface of the activity is defined, in the XML file reside in your “res/layout” folder. In the following example, it is defined in the xml file named **activity_main.xml** as shown in below code:
- **activity_main.xml**

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</LinearLayout>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **Creating User Interface Using xml file (Cont.) :**

- In activity_main.xml file, there is single **TextView** is defined in the layout as UI. To assign this a user interface to an Activity, it is required to call **setContentView()** method by passing the resource ID for a layout defined in activity_main.xml file as argument shown in below code:

```
public class MyActivity extends Activity
{
   /** Called when the activity is first created. */

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

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **Creating User Interface Using xml file (Cont.) :**

- When your run the application, above activity looks as shown in below figure:

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **Creating User Interface Using program code :**

- You can also create the same user interface as defined in xml file, at run time using program code in the activity class as shown in below code:

```
public class MyActivity extends Activity
{
    /** Called when the activity is first created. */

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **Creating User Interface Using program code (Cont.) :**
-

```
        LinearLayout ll = new LinearLayout(this);
        ll.setOrientation(LinearLayout.VERTICAL);

        TextView textView1 = new TextView(this);
        textView1.setText("Hello world!");

        int lHeight = LinearLayout.LayoutParams.MATCH_PARENT;
        int lWidth = LinearLayout.LayoutParams.MATCH_PARENT;

        ll.addView(textView1, new LinearLayout.LayoutParams(lHeight, lWidth));

        setContentView(ll);
    }
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **Creating User Interface Using program code (Cont.) :**

- When your run the application, above activity looks as shown in below figure :

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.1 Activity (Cont.)

- Every activity you have in your application **must be declared** in your **AndroidManifest.xml** file, as shown below:

```
<?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
              package="com.example.app1"
               android:versionCode="1"
              android:versionName="1.0" >

                         <uses-sdk  android:minSdkVersion="8"
                                  android:targetSdkVersion="19" />
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.1 Activity (Cont.)

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

        <activity
        android:name="com.example.app1.MyActivity"
        android:label="@string/app_name" >
                <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>

        </activity>

 </application>
</manifest>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.1 Activity (Cont.)

- Each String constant you have in your application **must be declared** in **strings.xml** file, and reference these strings using the **@string** identifier as android:text attribute is specified in TextView.
- . The strings.xml file located in the res/values folder.

```
<?xml version="1.0" encoding="utf-8"?>
 <resources>
        <string name="app_name">App1</string>
        <string name="action_settings">Settings</string>
        <string name="hello_world">Hello world!</string>
</resources>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle

- From the moment an activity appears on the screen to the moment it is hidden, it goes through a number of states, known as an activity's life cycle. An activity can be in different states which are described by the following Table:

State	Description
Running	Activity is visible and interacts with the user.
Paused	Activity is still visible but partially hidden.
Stopped	Activity is not visible; instance is running but might be killed by the system.
Killed	Activity has been terminated by the system.

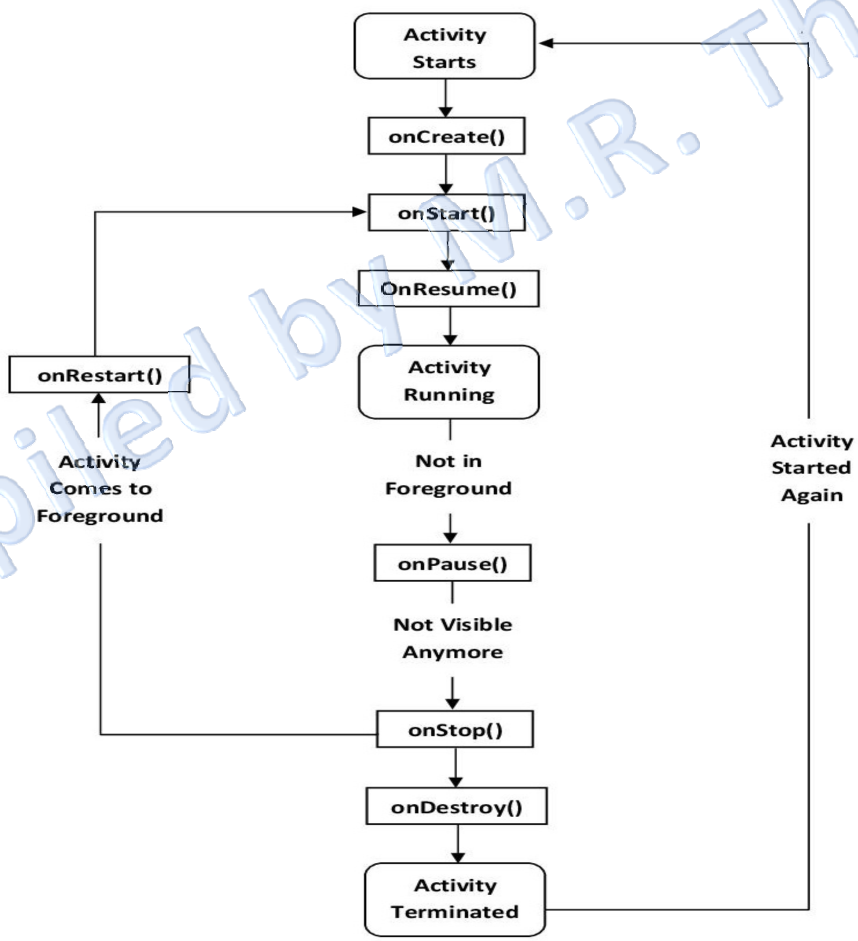
3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)
- The Activity base class defines a series of events via predefined lifecycle methods that govern the life cycle of an activity. The Activity class defines the methods as shown in following Table :

Method	Purpose
onCreate()	Called when the activity is created. Used to initialize the activity, for example create the user interface.
onStart()	Called when the activity becomes visible to the user
onResume()	Called if the activity gets visible again and the user starts interacting with the activity again. Used to initialize fields, register listeners, bind to services, etc.
onPause()	Called once another activity gets into the foreground. Always called before the activity is not visible anymore. Used to release resources or save application data. For example you unregister listeners, intent receivers, unbind from services or remove system service listeners.
onStop()	Called once the activity is no longer visible. Time or CPU intensive shut-down operations, such as writing information to a database should be down in the onStop() method.
onDestroy()	Called before the activity is destroyed by the system, either manually or by the system to conserve memory.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)
- Following figure shows the life cycle of an activity and the various stages it goes through from when the activity is started until it ends.



3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

Example

- In the following example, all the methods which are called on particular event during activity lifecycle are implemented.

activity_main.xml

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

    <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="32dp"
    android:text="@string/txt1_text"/>

</RelativeLayout>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

strings.xml

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

<resources>
    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
    <string name="txt1_text">Activity Lifecycle..</string>
</resources>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- **MainActivity.java**

```
package com.example.app1;

import android.support.v7.app.ActionBarActivity;
Import android.widget.TextView;
Import android.os.Bundle;

public class MainActivity extends ActionBarActivity
{
      TextView  textView1;
      String text;
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- **<u>MainActivity.java (onCreate)</u>**

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

    textView1 = (TextView) findViewById(R.id.textView1);
    text = textView1.getText().toString();
    text = text + "\n Inside onCreate Method.";
    textView1.setText(text);
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- **MainActivity.java (onStart)**

```
Public void onStart()
{
      super.onStart();
      text = textView1.getText().toString();
      text = text + "\n Inside onStart Method.";
      textView1.setText(text);

}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- **MainActivity.java (onResume)**

```
Public void onResume()
{
    super.onResume();
    text = textView1.getText().toString();
    text = text + "\n Inside onResume Method.";
    textView1.setText(text);
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- **MainActivity.java (onPause)**

```
Public void onPause()
{
    super.onPause();
    text = textView1.getText().toString();
    text = text + "\n Inside onPause Method.";
    textView1.setText(text);
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- **<u>MainActivity.java (onStop)</u>**

```
Public void onStop()
{
    super.onStop();
    text = textView1.getText().toString();
    text = text + "\n Inside onStop Method.";
    textView1.setText(text);
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- **MainActivity.java (onRestart)**

```
Public void onRestart()
{
        super.onRestart();
        text = textView1.getText().toString();
        text = text + "\n Inside onRestart Method.";
        textView1.setText(text);
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

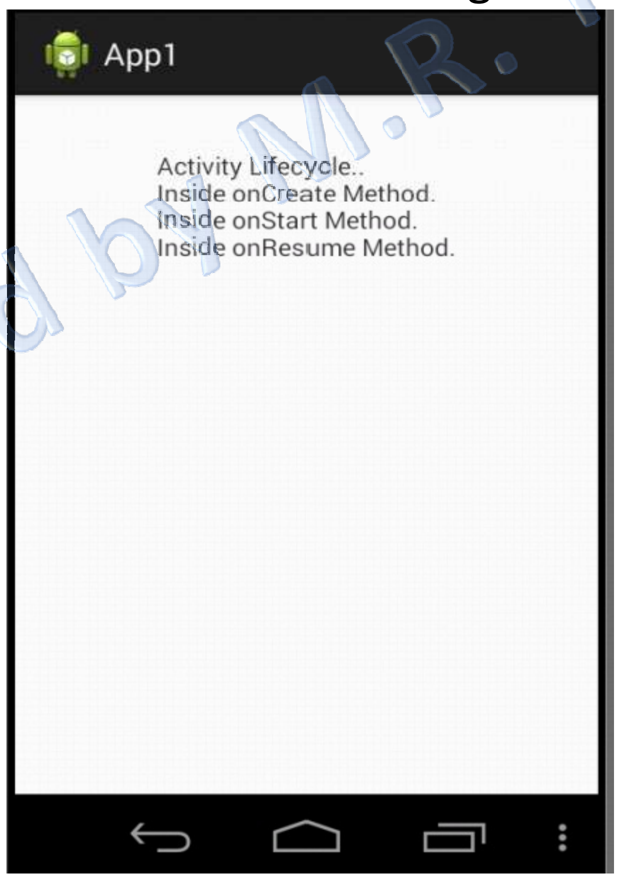
- **MainActivity.java (onDestroy)**

```
Public void onDestroy()
{
      super.onDestroy();
      text = textView1.getText().toString();
      text = text + "\n Inside onDestroy Method.";
      textView1.setText(text);
}

}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)

- When you run the application, first the **onCreate**() method is called and then **onStart**(), **onResume**() methods are called sequentially and the Text of the TextView look like as shown in below figure:



3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.2 Activity Lifecycle (Cont.)
- Then press home button on the handset, and the activity will not be on foreground any more, and the **onPause()** method is called. When activity is not visible, it calls **onStop()** method. Again when the activity is visible, it calls **onRestart()**, **onStart()** and **onResume()** methods sequentially as shown in following figure :

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent

- An Android application can contain zero or more activities. When your application has more than one activity, you often need to navigate from one to another. In Android, you navigate between activities through an intent.

- The intent can be used to:
  - Link Activities
  - Call Built-in Applications
  - Pass Data between Activities

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Link Activities using Intent:**

  The most common use of Intents is to navigate between activities.

Example:

- In the following application we have two activities, named **MainActivity** and **SecondActivity**.
- MainActivity consist of the **activity_main.xml** as UI component and **MainActivity.java** as class component. Another activity in the same application consist of **activity_second.xml** as UI component and **SecondActivity.java** as class component.
- To navigate from the MainActivity to SecondActivity, you need to call the **startActivity()** method in MainActivity class, passing an Intent object as argument:
- startActivity(new Intent(this, SecondActivity.class));

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

- **Link Activities using Intent (Cont.):**

activity_main.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="@string/txt1_text" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="59dp"
        android:onClick="showSecondActivity"
        android:text="@string/btn1_text" />

</RelativeLayout>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

- **Link Activities using Intent (Cont.):**

activity_second.xml

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

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerInParent="true"
        android:text="@string/txt2_text" />

</RelativeLayout>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

- **Link Activities using Intent (Cont.):**

strings.xml

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

<resources>
    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_second">SecondActivity</string>
    <string name="txt1_text">This is MainActivity</string>
    <string name="txt2_text">This is SecondActivity</string>
    <string name="btn1_text">Click for SecondActivity</string>
</resources>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

- **Link Activities using Intent (Cont.):**

MainActivity:

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

- **<u>Link Activities using Intent (Cont.):</u>**

<u>SecondActivity:</u>

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

- **<u>Link Activities using Intent (Cont.):</u>**

<u>MainActivity.java</u>

```
public class MainActivity extends ActionBarActivity
{
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   public void showSecondActivity(View v)
   {
      Intent i= new Intent(this, SecondActivity.class));
      startActivity(i);

    //startActivity(new Intent(this, SecondActivity.class));
   }
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Link Activities using Intent (Cont.):**

SecondActivity.java

```
public class SecondActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

    }
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Call Built-in Application:**

- Your application can call the built-in applications that are included with an Android device.

- For example, if your application needs to load a web page, you can use the Intent object to invoke the built-in web browser to display the web page, instead of building your own web browser for this purpose.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Call Built-in Application:**

Example:

activity_main.xml

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

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="31dp"
      android:onClick="openGoogle"
      android:text="@string/btn1_text" />

</RelativeLayout>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Call Built-in Application:**

strings.xml

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

<resources>
    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
    <string name="btn1_text">Click to open Google</string>
</resources>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Call Built-in Application:**

MainActivity.java

```
public class MainActivity extends ActionBarActivity
{
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
   }

   public void openGoogle(View v)
   {
     Intent i = new Intent(android.content.Intent.ACTION_VIEW,
                                             Uri.parse("http://www.google.co.in"));
     startActivity(i);
   }
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **<u>Call Built-in Application:</u>**

- In Android, intents usually come in pairs: action and data.

- The action describes what is to be performed, such as editing an item, viewing the content of an item, and so on.

- The data specifies what is affected, such as a URL address. The data is specified as an Uri object.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Call Built-in Application:**

- Some examples of action are as follows:

  ACTION_VIEW

  ACTION_DIAL

  ACTION_PICK

- Some examples of data include the following:

  www.google.co.in

  tel:+651234567

  geo:37.827500,-122.481670

  content://contacts

- Collectively, the action and data pair describes the operation to be performed. For example, to open Google site, you would use the pair ACTION_VIEW,http://www.google.co.in.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **<u>Call Built-in Application:</u>**

- When you run an application the MainActivity starts as shown in below figure :

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Call Built-in Application:**

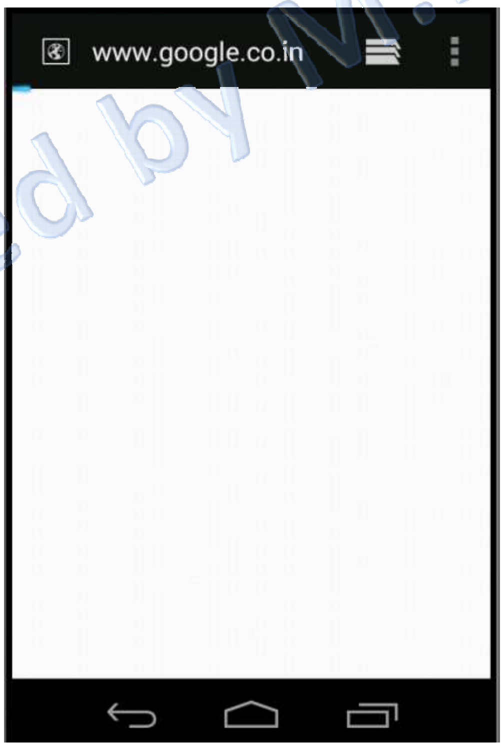
- When the button is pressed, you create an Intent object and then pass two arguments to its constructor, the action and the data.

```
Intent i = new Intent(android.content.Intent.ACTION_VIEW,
                      Uri.parse("http://www.google.co.in"));
startActivity(i);
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Call Built-in Application:**

- The startActivity() method starts browser application and open the url(www.google.co.in) as shown in below figure :



3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **<u>Pass Data between Activities</u>**

When you call an activity, it is common to pass data to an activity. You can use the Intent object to pass the data to the target activity.

- With an intent object, use the **putExtra()** method to add a name/value pair.
- On the target activity, to obtain the data sent using the Intent object, you first obtain the Intent object using the **getIntent()** method.
- Then, call **get<type>Extra()** methods to extract additional information stored using putExtra() method.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**

- **activity_main.xml (Layout)**

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

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="34dp"
        android:ems="10"
        android:hint="@string/edttxt1_hint" >

        <requestFocus />
    </EditText>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**

- **activity_main.xml (Layout)**

```
<Button
   android:id="@+id/button1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignLeft="@+id/editText1"
   android:layout_below="@+id/editText1"
   android:layout_marginTop="31dp"
   android:onClick="showSecondActivity"
   android:text="@string/btn1_text" />

</RelativeLayout>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

<u>MainActivity:</u>

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**
- **activity_second.xml (Layout)**

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

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerInParent="true"
        android:text="@string/txt2_text" />

</RelativeLayout>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)

SecondActivity:

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**
- **strings.xml (String Constants)**

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

<resources>

    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_second">SecondActivity</string>
    <string name="edttxt1_hint">Enter your name</string>
    <string name="txt2_text">This is SecondActivity</string>
    <string name="btn1_text">Click for SecondActivity</string>

</resources>
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**
- **MainActivity.java (program code)**

```
public class MainActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**
- **MainActivity.java (program code)**

```
public void showSecondActivity(View v)
{

    EditText editText1 = (EditText) findViewById(R.id.editText1);
    String name = editText1.getText().toString();

    Intent i = new Intent(this, SecondActivity.class);

    i.putExtra("name", name );
    startActivity(i);
}
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**
- **SecondActivity.java (program code)**

```
public class SecondActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_second);

      Intent  i = getIntent();
      String name = i.getStringExtra("name");

      TextView textView2 = (TextView) findViewById(R.id.textView2);
      textView2.setText(name);
    }
}
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **<u>Example (Pass Data between Activities)</u>**

When you run an application the MainActivity starts as shown below:

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **<u>Example (Pass Data between Activities)</u>**

Enter your name into EditText and press the Button as shown in below figure :

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**

When you enter the name in EditText and press the Button, it makes call to showSecondActivity() method.

The showSecondActivity() method, get the name entered in the EditText using getText() method and pass this name as argument in the **putExtra()** method with an instance of Intent.

```
String name = editText1.getText().toString();

Intent i = new Intent(this, SecondActivity.class);
i.putExtra("name", name );
startActivity(i);
```

At last it call startActivity() method, which starts SecondActivity.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**

- When SecondActivity get started, it first receive the intent object using getIntent() method.

```
Intent  i = getIntent();
```

- Then extract the name using getStringExtra() method by passing key as argument.

```
String name = i.getStringExtra("name");
```

- The name extracted is displayed in the TextView using the setText() method.

```
textView2.setText(name);
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- 3.2.3 Intent (Cont.)
- **Example (Pass Data between Activities)**

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **3.2.4 AndroidManifest**

- Each Android project includes a manifest file, the manifest lets you define the structure and metadata of your application, its components, and its requirements.

- Whatever component you develop as a part of your application, you must declare all its components in a manifest file called AndroidManifest.xml which resides at the root of the application project directory.

- This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS.

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **3.2.4 AndroidManifest**

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.app1"
         android:versionCode="1"
         android:versionName="1.0" >

     <uses-sdk
          android:minSdkVersion="8"
         android:targetSdkVersion="19" />
```

3.2 Activity, Activity Lifecycle, Intent and Manifest

- **3.2.4 AndroidManifest**

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

  <activity
      android:name="com.example.app1.MainActivity"
      android:label="@string/app_name" >

    <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>

  </activity>

</application>

</manifest>
```

3.3 Creating Application

- 3.3.1 Categories of Application

- Most of the applications you create in Android will fall into one of the following categories:

  - **Foreground Application**
  - **Background Service Application**
  - **Intermittent Application**
  - **Widget**

3.3 Creating Application

- 3.3.2 Components of Application

- The following six components provide the building blocks for your applications:

  - **Activities**
  - **Services**
  - **Content Providers**
  - **Intents**
  - **Widgets**
  - **Notifications**

3.4 Introduction to Android User Interface Design

- Android application user interface is everything that the user can see and interact with.

- All user interface elements in an Android application are built using:
  - ViewGroup (layout)
  - View (GUI controls such as TextView, EditText, Button etc)

3.4 Introduction to Android User Interface Design

- 3.4.1 Introducing Layouts

- A layout defines the visual structure for an Android user interface.

- Layout can be created either declaring your layout using simple **XML file** activity_main.xml which is located in the **res/layout** folder of your project or **at run time by program code**.

3.4 Introduction to Android User Interface Design

- 3.4.2 Layout Types

- The list of the layouts :

  - LinearLayout
  - RelativeLayout
  - TableLayout
  - AbsoluteLayout
  - FrameLayout

LinearLayout

- LinearLayout is a view group that aligns all child view in a single direction, vertically or horizontally.

- Attributes:

- android:id - uniquely identifies the layout.
- android:orientation - specifies the direction of arrangmet and you will use **"horizontal"** for a row, **"vertical"** for a column.
- android:gravity - specifies how an object should position its content, Possible values are **top, bottom, left, right, center, center_vertical, center_horizontal** etc.

LinearLayout

- Example:

activity_main.xml

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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/btn1_text" />
```

LinearLayout

```
<Button
      android:id="@+id/button2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:text="@string/btn2_text" />

  <Button
      android:id="@+id/button3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:text="@string/btn3_text" />

</LinearLayout>
```

LinearLayout

LinearLayout

- Now let's change the orientation of Layout as **android:orientation**="**horizontal**“

RelativeLayout

- RelativeLayout enables you to specify how child views are positioned **relative to each other.** The position of each view can be specified as **relative to sibling elements or relative to the parent.**

- Attributes:

  - **android:id** - uniquely identifies the layout.
  - **android:layout_alignParentTop** - If true, makes the top edge of this view match the top edge of the parent.
  - **android:layout_alignParentBottom** - If true, makes the bottom edge of this view match the bottom edge of the parent.
  - **android:layout_alignParentLeft** - If true, makes the left edge of this view match the left edge of the parent.
  - **android:layout_alignParentRight** - If true, makes the right edge of this view match the right edge of the parent.

RelativeLayout

- Attributes:

- **android:layout_above** - positions the view above the given view ID
- **android:layout_below** - positions the view below the given view ID
- **android:layout_alignLeft** - positions the view left of the given view ID
- **android:layout_alignRight** - positions the view right of the given view ID

RelativeLayout

- Attributes:

- **android:layout_centerHorizontal** - If true, centers this child horizontally within its parent.
- **android:layout_centerVertical** - If true, centers this child vertically within its parent.
- **android:layout_centerInParent** - If true, centers this child horizontally and vertically within its parent.

RelativeLayout

- Example:

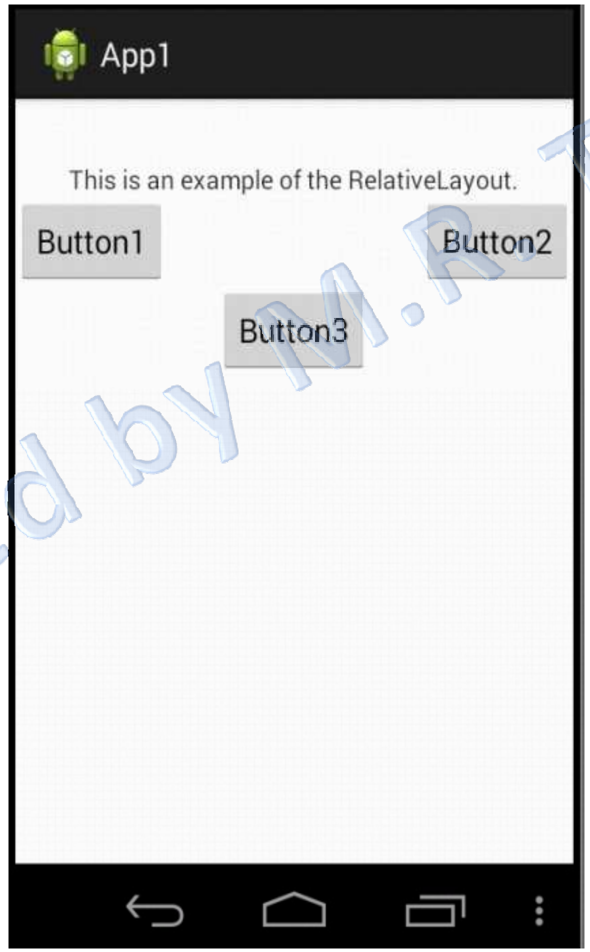
activity_main.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="33dp"
        android:text="@string/txt1_text" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:text="@string/btn1_text" />
```

RelativeLayout

```
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/textView1"
    android:text="@string/btn2_text" />

<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button2"
    android:layout_centerHorizontal="true"
    android:text="@string/btn3_text" />

</RelativeLayout>
```

RelativeLayout



AbsoluteLayout

- An Absolute Layout lets you specify exact locations (x/y coordinates) of its child view.

- Attributes:

  - **android:id** - uniquely identifies the layout.
  - android:layout_x - specifies the x-coordinate of the view.
  - android:layout_y - specifies the y-coordinate of the view.

AbsoluteLayout

- Example:

activity_main.xml

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

  <Button
     android:id="@+id/button1"
     android:layout_width="100dp"
     android:layout_height="wrap_content"
     android:layout_x="50px"
     android:layout_y="200px"
     android:text="@string/btn1_text" />

  <Button
     android:id="@+id/button2"
     android:layout_width="100dp"
     android:layout_height="wrap_content"
     android:layout_x="225px"
     android:layout_y="200px"
     android:text="@string/btn2_text" />

</AbsoluteLayout>
```

AbsoluteLayout

TableLayout

- TableLayout groups 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.

- Attributes:

  - **android:id** - uniquely identifies the layout.
  - **android:shrinkColumns** - specifies the columns to shrink.
  - **android:stretchColumns** - specifies the columns to stretch.
  - **android:collapseColumns** – specifies the columns to collapse.

TableLayout

- Example:

activity_main.xml

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

   <TableRow
      android:id="@+id/tableRow1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >

      <TextView
         android:id="@+id/textView1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/txt1_text" />

      <EditText
         android:id="@+id/editText1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content“
         android:hint="@string/et1_text" >

      </EditText>

   </TableRow>
```

TableLayout

```
<TableRow
    android:id="@+id/tableRow2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/txt2_text" />

    <EditText
      android:id="@+id/editText2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:ems="10"
      android:hint="@string/et2_text" />

  </TableRow>
```

TableLayout

```
<TableRow
      android:id="@+id/tableRow3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >

      <Button
         android:id="@+id/button1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/btn1_text" />

   </TableRow>

</TableLayout>
```

TableLayout

FrameLayout

- The FrameLayout is a placeholder on screen that you can use to display a single view.

- Attributes:

- **android:id** - uniquely identifies the layout.
- **android:foreground** - defines the drawable to draw over the content and possible values may be a color value.
- **android:foregroundGravity** - defines the gravity to apply to the foreground drawable. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.

FrameLayout

- Example:

activity_main.xml

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

  <ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:contentDescription="@string/img1_text"
    android:src="@drawable/ic_launcher" />

</FrameLayout>
```

FrameLayout

3.5 Introduction to GUI objects

- The list of UI controls provided by Android that allow you to build the graphical user interface are:

  - TextView
  - Button
  - EditText
  - ToggleButton
  - ImageButton
  - CheckBox
  - RadioButton

TextView

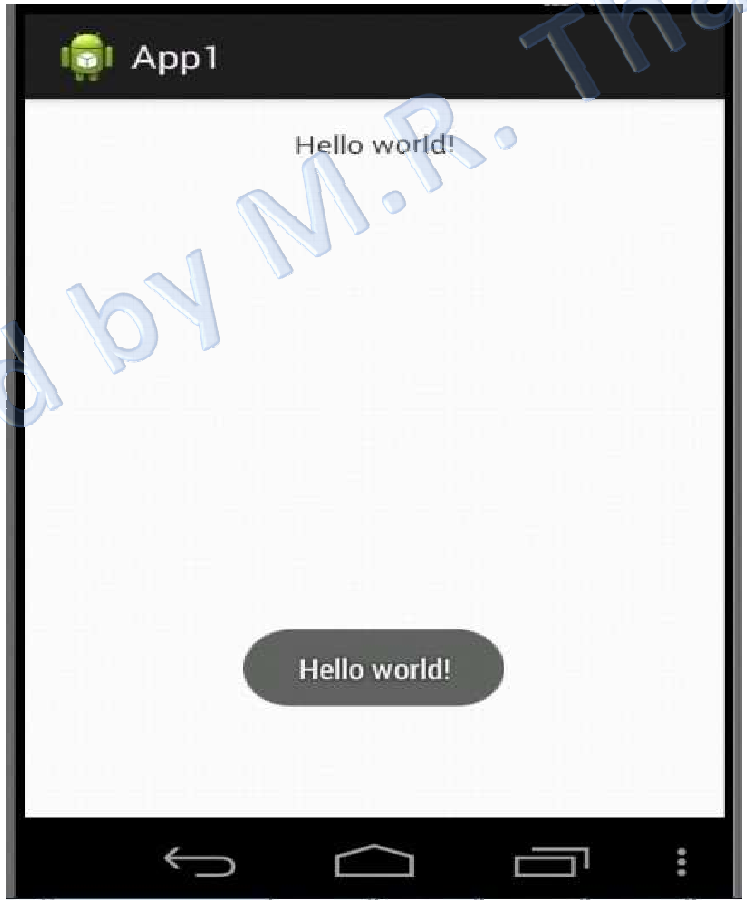
- TextView view is used to display text to the user.

- <u>Attributes:</u>

  - **android:id** - uniquely identifies the TextView.
  - **android:text** – text to display.
  - **android:textColor** - text color, specified in **#rgb** format.
  - **android:textSize** - size of the text.
  - **android:textStyle** – Style for the text. **normal - 0 ,bold - 1 ,italic – 2**

TextView

- Example:



TextView

- Example:

activity_main.xml

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

      <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content“
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:text="@string/hello_world" />

</RelativeLayout>
```

TextView

- Example:

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
</resources>
```

TextView

- Example:

MainActivity.java

```
public class MainActivity extends ActionBarActivity
{
 @Override
        protected void onCreate(Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            TextView textView1 = (TextView) findViewById(R.id.textView1);
            String  text = textView1.getText().toString();

            Toast.makeText(this, text, Toast.LENGTH_LONG).show();
        }
}
```

Button

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

- <u>Attributes:</u>

  - **android:id** - uniquely identifies the Button.
  - **android:text** – text to display.
  - **android:onClick** – specifies the name of the method in this View's context to invoke when the view is clicked.
  - **android:visibility** - controls the visibility of the view.

Button

- <u>Example:</u>

Button

- Example:

activity_main.xml

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

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="30dp"
      android:text="@string/hello_world" />
```

Button

- <u>Example:</u>

<u>activity_main.xml</u>

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="30dp"
    android:onClick="showText"
    android:text="@string/btn_label" />

</RelativeLayout>
```

Button

- <u>Example:</u>

strings.xml

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

<resources>
    <string name="app_name">App1</string>
   <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="btn_label">Display Text</string>
</resources>
```

Button

- <u>Example:</u>

<u>MainActivity.java</u>

```
public class MainActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void showText(View v)
    {
        TextView textView1 = (TextView) findViewById(R.id.textView1);
        String  text = textView1.getText().toString();

        Toast.makeText(this, text, Toast.LENGTH_LONG).show();
    }
}
```

EditText

- EditText is a subclass of the TextView class that is similar to the TextView but it allows users to edit its text content.

- <u>Attributes:</u>

  - **android:id** - uniquely identifies the EditText.
  - **android:text** – text to display.
  - **android:hint** - display hint text, when the text is empty.
  - **android:password** - Whether the characters of the field are displayed as password dots instead of themselves. Possible value either "true" or "false".

EditText

- <u>Example:</u>

EditText

- <u>Example:</u>

<u>activity_main.xml</u>

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

<EditText
       android:id="@+id/editText1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_alignParentTop="true"
       android:layout_centerHorizontal="true"
       android:layout_marginTop="22dp“
       android:hint="@string/hint1" />
```

EditText

- Example:

activity_main.xml

```
<EditText
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content“
    android:layout_below="@+id/editText1“
   android:hint="@string/hint2" />

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editText2"
    android:layout_centerHorizontal="true“
    android:onClick="calculateSum"
    android:text="@string/btn_label" />

</RelativeLayout>
```

EditText

- <u>Example:</u>

<u>strings.xml</u>

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

<resources>
    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
    <string name="btn_label">SUM</string>
    <string name="hint1">value1</string>
    <string name="hint2">value2</string>
</resources>
```

EditText

- <u>Example:</u>

<u>MainActivity.java</u>

```
public class MainActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

EditText

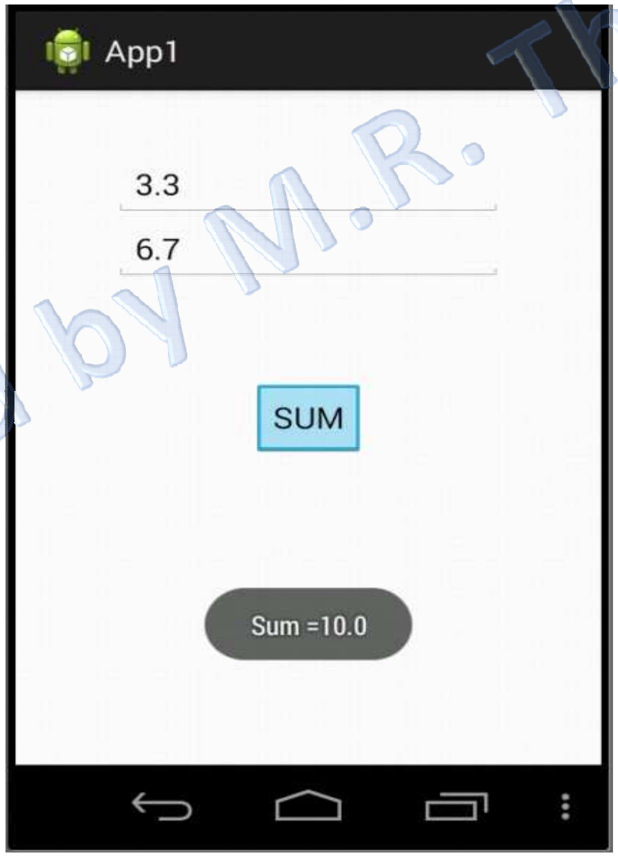
- Example:

MainActivity.java

```
public void calculateSum(View v)
{
        EditText editText1 = (EditText) findViewById(R.id.editText1);
        EditText editText2 = (EditText) findViewById(R.id.editText2);

        String value1 = editText1.getText().toString();
        String value2 = editText2.getText().toString();

        double result = Double.parseDouble(value1) + Double.parseDouble(value2);

        String sum = String.valueOf(result);

        Toast.makeText(this, "Sum =" + sum, Toast.LENGTH_LONG).show();
}

}
```

EditText

- <u>Example:</u>



ToggleButton

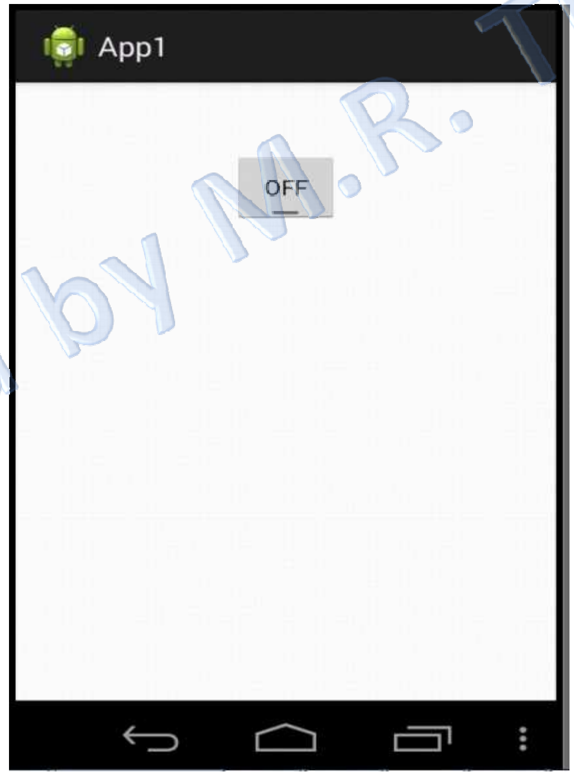
- It is basically an on/off button with a light indicator.

- <u>Attributes:</u>

- **android:id** - uniquely identifies the ToggleButton.
- **android:textOn** – text for the button when it is checked.
- **android:textOff** - text for the button when it is not checked.
- **android:onClick** - the name of the method in this View's context to invoke when the view is clicked.

ToggleButton

- Example:



ToggleButton

- Example:

activity_main.xml

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

   <ToggleButton
      android:id="@+id/toggleButton1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="30dp"
      android:onClick="showText" />

</RelativeLayout>
```

ToggleButton

- Example:

strings.xml

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

<resources>
    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
 </resources>
```

ToggleButton

- Example:

MainActivity.java

```
public class MainActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
    }

    public void showText(View v)
    {
      ToggleButton tb = (ToggleButton) findViewById(R.id.toggleButton1);
      String text = tb.getText().toString();

      Toast.makeText(this,"TobbleButton is " + text, Toast.LENGTH_LONG).show();
    }
}
```

ToggleButton

Example:

ImageButton

- A ImageButton is shows a button with an image, instead of text that can be pressed or clicked by the user.

- <u>Attributes:</u>

  - **android:id** - uniquely identifies the ImageButton.
  - **android:src** - sets a drawable(image) as the content of this ImageView.
  - **android:onClick** – specifies the name of the method in this View's context to invoke when the view is clicked.
  - **android:visibility** - controls the visibility of the view.

ImageButton

- Example:

ImageButton

- Example:

activity_main.xml

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

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="38dp"
      android:text="@string/hello_world" />
```

ImageButton

- Example:

activity_main.xml

```
<ImageButton
    android:id="@+id/imageButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="34dp"
    android:contentDescription="@string/image_button"
    android:onClick="showText"
    android:src="@drawable/ic_launcher" />

</RelativeLayout>
```

ImageButton

- Example:

strings.xml

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

<resources>
<string name="app_name">App1</string>
<string name="action_settings">Settings</string>
<string name="image_button">Image Button</string>
<string name="hello_world">Hello world!</string>
</resources>
```

ImageButton

- Example:

MainActivity.java

```
public class MainActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void showText(View v)
    {
        TextView textView1 = (TextView) findViewById(R.id.textView1);
        String  text = textView1.getText().toString();

        Toast.makeText(this,text, Toast.LENGTH_LONG).show();
    }
}
```

CheckBox

- A CheckBox is an on/off switch that can be toggled by the user.
- You should use check-boxes when presenting users with a group of selectable options that are not mutually exclusive.

- <u>Attributes:</u>
  - **android:id** - uniquely identifies the CheckBox.
  - **android:text** – text to display.
  - **android:checked** - Set this to **true** if you want the CheckBox to set checked by default.

CheckBox

- Example:

CheckBox

- Example:

activity_main.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="38dp"
        android:text="@string/txt_language" />
```

CheckBox

- Example:

activity_main.xml

```
<CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content“
   android:layout_below="@+id/textView1“
  android:text="@string/language1" />

  <CheckBox
    android:id="@+id/checkBox2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content“
   android:layout_below="@+id/checkBox1"
    android:text="@string/language2" />
```

CheckBox

- Example:

activity_main.xml

```
<CheckBox
    android:id="@+id/checkBox3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content“
   android:layout_below="@+id/checkBox2"
    android:text="@string/language3" />

  <Button
    android:id="@+id/Submit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content“
     android:layout_below="@+id/checkBox3“
   android:onClick="Submit_click"
    android:text="@string/btn_submit" />

</RelativeLayout>
```

CheckBox

- Example:

strings.xml

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

<resources>

<string name="app_name">App1</string>
<string name="action_settings">Settings</string>
<string name="txt_language">Which languages do you know?</string>
<string name="language1">Gujarati</string>
<string name="language2">Hindi</string>
<string name="language3">English</string>
<string name="btn_submit">SUBMIT</string>

</resources>
```

CheckBox

- Example:

MainActivity.java

```
public class MainActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
    }
```

CheckBox

- Example:

MainActivity.java

```
public void Submit_click(View v)
{
      checkBox cb1 = (CheckBox) findViewById(R.id.checkBox1);
      CheckBox cb2 = (CheckBox) findViewById(R.id.checkBox2);
      CheckBox cb3 = (CheckBox) findViewById(R.id.checkBox3);

      String result="You have Selected: \n";

      if (cb1.isChecked())
      {
            result =result + cb1.getText().toString() + "\n";
      }
      if (cb2.isChecked())
      {
            result =result + cb2.getText().toString() + "\n";
      }
      if (cb3.isChecked())
      {
            result =result + cb3.getText().toString();
      }

        Toast.makeText(this,result, Toast.LENGTH_LONG).show();
  }
}
```

CheckBox

- Example:

RadioButton

- A RadioButton has two states: either checked or unchecked.
- It allows the user to select one option from a set. If we check one radio button that belongs to a radio group, it automatically uncheck any previously checked radio button within the same group.
- A **RadioGroup** class is used to form set of radio buttons.

- <u>Attributes:</u>

○ **android:id** - uniquely identifies the RadioButton.

○ **android:text** – text to display.

○ **android:checked** - Set this to **true** if you want the RadioButton to set checked by default.

RadioButton

- Example:

RadioButton

- Example:

activity_main.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="@string/txt_fruit" />
```

RadioButton

- Example:

activity_main.xml

```
<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/textView1"
    android:layout_centerHorizontal="true" >

    <RadioButton
       android:id="@+id/radio1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content“
       android:checked="true"
       android:text="@string/fruit1" />

    <RadioButton
       android:id="@+id/radio2"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/fruit2" />
```

RadioButton

- Example:

activity_main.xml

```
<RadioButton
        android:id="@+id/radio3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/fruit3" />

</RadioGroup>

 <Button
      android:id="@+id/Submit"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/radioGroup1"
      android:layout_centerHorizontal="true“
      android:onClick="Submit_click"
      android:text="@string/btn_submit" />

</RelativeLayout>
```

RadioButton

- Example:

strings.xml

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

<resources>
    <string name="app_name">App1</string>
    <string name="action_settings">Settings</string>
    <string name="txt_fruit">Which is your favourate fruit?</string>
    <string name="fruit1">Apple</string>
    <string name="fruit2">Mango</string>
    <string name="fruit3">Citrus</string>
    <string name="btn_submit">SUBMIT</string>

</resources>
```

RadioButton

- **Example:**

MainActivity.java

```
public class MainActivity extends ActionBarActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
    }
```

RadioButton

- Example:

MainActivity.java

```
public void Submit_click(View v)
{
    RadioGroup rg = (RadioGroup) findViewById(R.id.radioGroup1);

    int selectedRadioButtonId = rg.getCheckedRadioButtonId();

    RadioButton selectedRadioButton =(RadioButton)
                                findViewById(selectedRadioButtonId);

    String selectedFruit = selectedRadioButton.getText().toString();

    Toast.makeText(this,"Your favourate fruite is: " + selectedFruit,
                                Toast.LENGTH_LONG).show();
}

}
```

RadioButton

- Example: