Unit-IV Designing User Interface

Course Outcome:

Use User Interface components for android application development..

Unit Outcomes:

4a. Develop rich user Interfaces for the given Android application.
4b. Develop Android application using the given view.
4c. Explain the significance of the given display Alert.
4d. Develop the given application using time and date picker.

Contents:

4.1 Text View, Edit Text; Button, Image Button; Toggle Button, Radio Button And Radio Group; Checkbox; Progress Bar
4.2 List View; Grid View; Image View; Scroll View; Custom Toast Alert
4.3 Time And Date Picker

Fundamentals of UI Design

With View

The "views" are the building blocks of a U.I design and composes of almost every basic U.I element like TextViews, EditTexts, ImageViews etc. This '***view***' however comes along with a few properties bundled to them. Some of the important and are often used to build up a complete meaningful screen design.

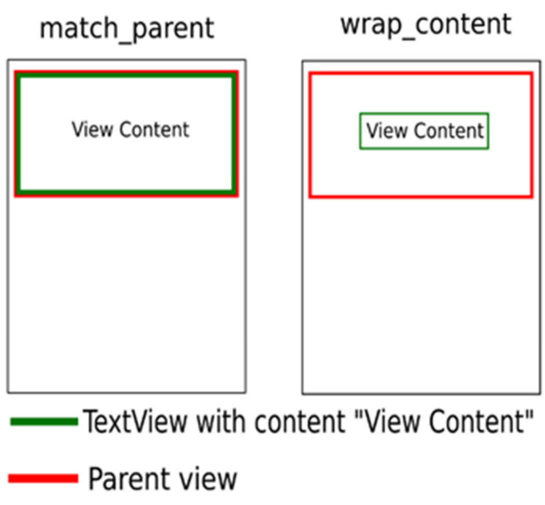
1. ***"id"***
2. ***"width"***
3. ***"height"***
4. ***"margin"***
5. ***"padding"***

"id"

This is basically the name of the particular view and will be used to refer that particular view through out the project. It has to be unique(*multiple views referencing to same id will confuse the compiler*). Common ethic to name an id of a view is **"descriptionOfView_viewType" (**For ex. if there is a textview that denotes an email, its id will be **""email_textview")**

"width" and "height"

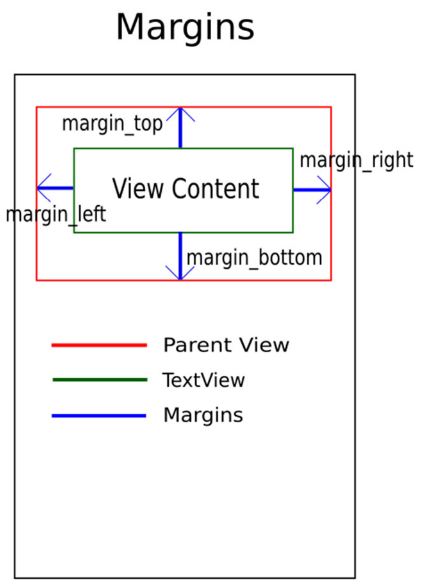
As the name of these properties suggest, these are the dimensions of that particular view. Now these dimensions can be set using hard-coded values and it will adopt to them in most layouts, but its not a very good design as the content inside them might get cropped or will have unwanted spaces. Android provides two pre-defined options to set these dimensions — ***"match_parent"*** and ***"wrap_content".***



Setting the dimensions to "**match_parent**" will make them equal to those of its parent's dimensions. If there is no parent to that particular view, it merely sets to the screen's dimensions (parent of a view is the U.I element in which it is housed in). And setting it to "**wrap_content**" will force the view to adopt its dimensions according to its content.

"margin"

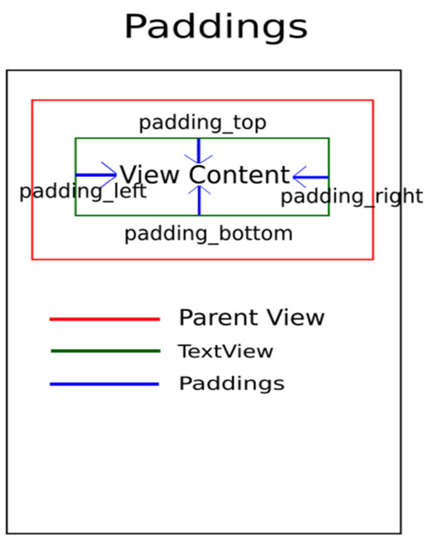
This is the minimum distance that a view has to maintain from its neighbouring views. That's it. Since there are four sides to any view, the four margins corresponding to them are **"margin_left", "margin_right", "margin_top"** and **"margin_bottom"**. If the same margin is needed to be set on all sides, it can be set directly through **"margin"** property.



Margins

"padding"

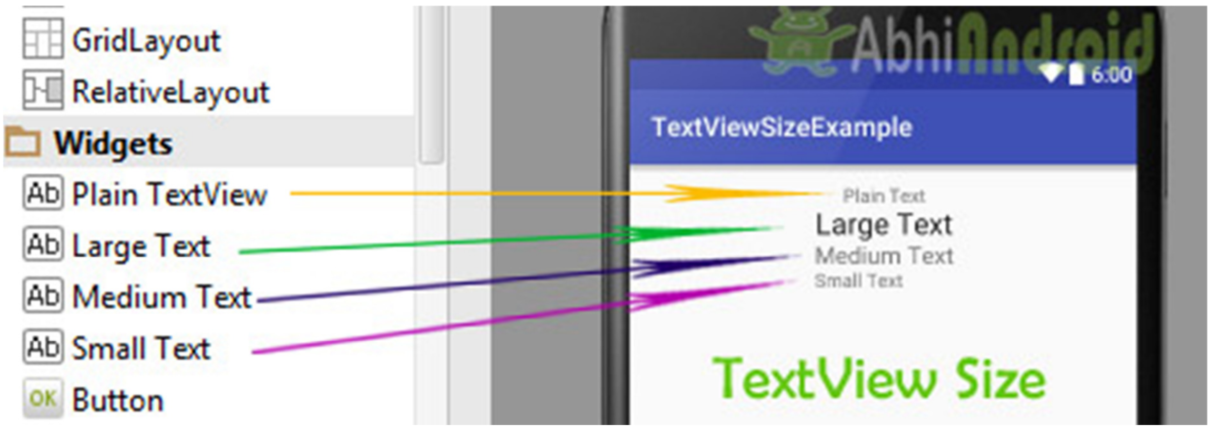
The distance between the view's outline and its content. Again similar to the "margin" property, "padding" too has **"padding_left", "padding_right", "padding_top", "padding_bottom"** and the common padding to all sides can be set through **"padding"** property.



Padding

TextView In Android Studio

In Android, TextView displays text to the user and optionally allows them to edit it programmatically. TextView is a complete text editor, however basic class is configured to not allow editing but we can edit it.



View is the parent class of TextView. Being a subclass of view the text view component can be used in your app's GUI inside a ViewGroup, or as the content view of an activity. We can create a TextView instance by declaring it inside a layout(XML file) or by instantiating it programmatically(Java Class).

TextView code in XML:

```
<TextView android:id="@+id/simpleTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="AbhiAndroid" />
```

TextView code in JAVA:

```
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText("AbhiAndroid"); //set text for text view
```

Attributes of TextView:
Now let's we discuss about the attributes that helps us to configure a TextView in your xml file.

1**. id:** id is an attribute used to uniquely identify a text view. Below is the example code in which we set the id of a text view.

```
<TextView
android:id="@+id/simpleTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

2. **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc.

Below is the example code with explanation included in which we set the center_horizontal gravity for text of a TextView.

```
<TextView
android:id="@+id/simpleTextView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="AbhiAndroid"
android:textSize="20sp"
android:gravity="center_horizontal"/> <!--center horizontal gravity-->
```

3. **text:** text attribute is used to set the text in a text view. We can set the text in xml as well as in the java class.
Below is the example code with explanation included in which we set the text “AbhiAndroid” in a text view.

```
<TextView
android:id="@+id/simpleTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:textSize="25sp"
android:text="AbhiAndroid"/><!--Display Text as AbhiAndroid-->
```

In Java class:
Below is the example code in which we set the text in a textview programmatically means in java class.

```
TextView textView = (TextView)findViewById(R.id.textView);
textView.setText("AbhiAndroid"); //set text for text view
```

4. **textColor:** textColor attribute is used to set the text color of a text view. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Below is the example code with explanation included in which we set the red color for the displayed text.

```
<TextView
android:id="@+id/simpleTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="AbhiAndroid"
android:layout_centerInParent="true"
android:textSize="25sp"
android:textColor="#f00"/><!--red color for text view-->
```

In Java class:
Below is the example code in which we set the text color of a text view programmatically means in java class.

```
TextView textView = (TextView)findViewById(R.id.textView);
textView.setTextColor(Color.RED); //set red color for text view
```

5. **textSize:** textSize attribute is used to set the size of text of a text view. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below is the example code in which we set the 20sp size for the text of a text view.

```
<TextView
   android:id="@+id/simpleTextView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="AbhiAndroid"
   android:layout_centerInParent="true"
   android:textSize="40sp" /><!--Set size-->
```

In Java class:
Below is the example code in which we set the text size of a text view programmatically means in java class.

```
TextView textView = (TextView)findViewById(R.id.textView);
textView.setTextSize(20); //set 20sp size of text
```

6. **textStyle:** textStyle attribute is used to set the text style of a text view. The possible text styles are bold, italic and normal. If we need to use two or more styles for a text view then “|” operator is used for that.

Below is the example code with explanation included in which we set the bold and italic text styles for text.

```
<TextView
   android:id="@+id/simpleTextView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="AbhiAndroid"
   android:layout_centerInParent="true"
   android:textSize="40sp"
   android:textStyle="bold|italic"/><!--bold and italic text style of text-->
```

7. **background:** background attribute is used to set the background of a text view. We can set a color or a drawable in the background of a text view.

8. **padding:** padding attribute is used to set the padding from left, right, top or bottom. In above example code of background we also set the 10dp padding from all the side's of text view.

Below is the example code with explanation included in which we set the black color for the background, white color for the displayed text and set 10dp padding from all the side's for text view.

```
<TextView
android:id="@+id/simpleTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="AbhiAndroid"
android:layout_centerInParent="true"
android:textSize="40sp"
android:padding="10dp"
android:textColor="#fff"
android:background="#000"/> <!--red color for background of text view-->
```

In Java class:
Below is the example code in which we set the background color of a text view programmatically means in java class.

```
TextView textView = (TextView)findViewById(R.id.textView);
textView.setBackgroundColor(Color.BLACK);//set background color
```

Example of TextView:
Below is the example of TextView in which we display a text view and set the text in xml file and then change the text on button click event programmatically. Below is the final output and code:

Step 1: Create a new project and name it textViewExample.
Select File -> New -> New Project. Fill the forms and click "Finish" button.

Step 2: Open res -> layout -> xml (or) activity_main.xml and add following code. Here we will create a button and a textview in Relative Layout.

```
<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"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <TextView
    android:id="@+id/simpleTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="Before Clicking"
    android:textColor="#f00"
    android:textSize="25sp"
    android:textStyle="bold|italic"
    android:layout_marginTop="50dp"/>

  <Button
    android:id="@+id/btnChangeText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:background="#f00"
    android:padding="10dp"
    android:text="Change Text"
    android:textColor="#fff"
    android:textStyle="bold" />
</RelativeLayout>
```

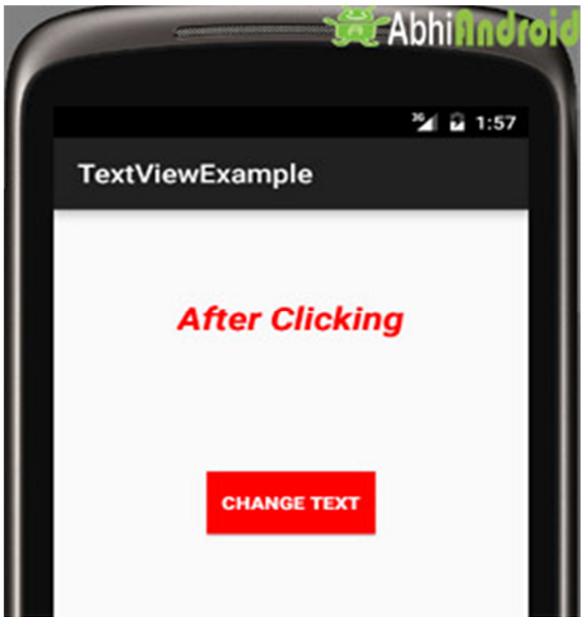
Step 3: Open app -> java -> package and open MainActivity.java and add the following code. Here we will change the text of TextView after the user click on Button.

```
package example.abhiandriod.textviewexample;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); //set the layout
        final TextView simpleTextView = (TextView) findViewById(R.id.simpleTextView);
//get the id for TextView
        Button changeText = (Button) findViewById(R.id.btnChangeText); //get the id for
button
        changeText.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                simpleTextView.setText("After Clicking"); //set the text after clicking button
            }
        });
    }


}
```

Output:

Now run the app in Emulator and click on the button. You will see text will change "After Clicking".



TextView Example Output

EditText In Android Studio: Input Field

In Android, EditText is a standard entry widget in android apps. It is an overlay over TextView that configures itself to be editable. EditText is a subclass of TextView with text editing operations. We often use EditText in our applications in order to provide an input or text field, especially in forms. The most simple example of EditText is Login or Sign-in form.

EditText Input Field Example in Android Text Fields in Android Studio are basically EditText:
Important Note: An EditText is simply a thin extension of a TextView. An EditText inherits all the properties of a TextView.

EditText Code:
We can create a EditText instance by declaring it inside a layout(XML file) or by instantiating it programmatically (i.e. in Java Class).

EditText code in XML:

```
<EditText
android:id="@+id/simpleEditText"
android:layout_height="wrap_content"
android:layout_width="match_parent"/>
```

Retrieving / Getting the Value From EditText In Java Class:

Below is the example code of EditText in which we retrieve the value from a EditText in Java class. We have used this code in the example you will find at the end of this post.

```
EditText simpleEditText = (EditText) findViewById(R.id.simpleEditText);
String editTextValue = simpleEditText.getText().toString();
```

Attributes of EditText:
Now let's we discuss few attributes that helps us to configure a EditText in your xml.

1. id: id is an attribute used to uniquely identify a text EditText. Below is the example code in which we set the id of a edit text.

```
<EditText
android:id="@+id/simpleEditText"
android:layout_height="wrap_content"
android:layout_width="match_parent"/>
```

2. gravity: The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc.

Below is the example code with explanation included in which we set the right gravity for text of a EditText.

```
<EditText
   android:id="@+id/simpleEditText"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Enter Email"
   android:gravity="right"/><!--gravity of a edit text-->
```

3. text: text attribute is used to set the text in a EditText. We can set the text in xml as well as in the java class.
Below is the example code in which we set the text "Username" in a edit text.

```
<EditText
   android:id="@+id/simpleEditText"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:text="Username"/><!--set text in edit text-->
```

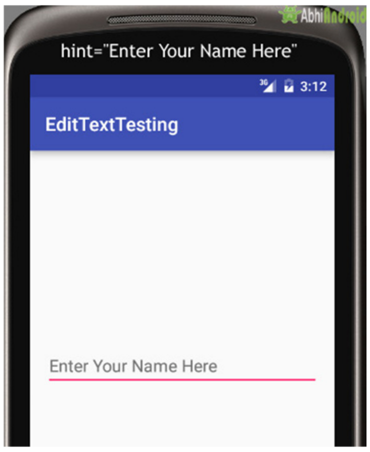
Below is the example code in which we set the text in a text view programmatically means in java class.

```
EditText editText = (EditText)findViewById(R.id.simpleEditText);
editText.setText("Username");//set the text in edit text
```

4. hint: hint is an attribute used to set the hint i.e. what you want user to enter in this edit text. Whenever user start to type in edit text the hint will automatically disappear.

Below is the example code with explanation in which we set the hint of a edit text.

```
<EditText
   android:id="@+id/simpleEditText"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:hint="Enter Your Name Here" /><!--display the hint-->
```



In java class

Below is the example code in which we set the text in a text view programmatically means in java class.

```
EditText editText = (EditText)findViewById(R.id.simpleEditText);
editText.setHint("Enter Your Name Here");//display the hint
```

5. textColor: textColor attribute is used to set the text color of a text edit text. Color value is in the form of "#argb", "#rgb", "#rrggbb", or "#aarrggbb".

Below is the example code with explanation included in which we set the red color for the displayed text of an edit text.

```
<EditText
  android:id="@+id/simpleEditText"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_centerInParent="true"
  android:text="Password"
  android:textColor="#f00"/><!--set the red text color-->
```

In Java class:
Below is the example code in which we set the text color of a edit text programmatically means in java class.

```
EditText simpleEditText=(EditText)findViewById(R.id.simpleEditText);
simpleEditText.setTextColor(Color.RED);//set the red text color
```

6. textColorHint: textColorHint is an attribute used to set the color of displayed hint.

Below is the example code with explanation included in which we set the green color for displayed hint of a edit text.

```
<EditText
   android:id="@+id/simpleEditText"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:hint="Enter Your Name Here"
   android:textColorHint="#0f0"/><!--set the hint color green-->
```

textColorHint in EditText AndroidSetting textColorHint in EditText In Java class:
Below is the example code in which we set the hint color of a edit text programmatically means in java class.

```
EditText simpleEditText=(EditText)findViewById(R.id.simpleEditText);
simpleEditText.setHintTextColor(Color.green(0));//set the green hint color
```

7. textSize: textSize attribute is used to set the size of text of a edit text. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below is the example code in which we set the 25sp size for the text of a edit text.

```
<EditText
   android:id="@+id/simpleEditText"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:text="AbhiAndroid"
   android:textSize="25sp" /><!--set 25sp text size-->
```

Setting textSize in EditText AndroidSetting textSize in EditText in Java class:
Below is the example code in which we set the text size of a edit text programmatically means in java class.

```
EditText simpleEditText=(EditText)findViewById(R.id.simpleEditText);
simpleEditText.setTextSize(25);//set size of text
```

8. textStyle: textStyle attribute is used to set the text style of a edit text. The possible text styles are bold, italic and normal. If we need to use two or more styles for a edit text then “|” operator is used for that.

Below is the example code with explanation included, in which we set the bold and italic text styles for text.

```
<EditText
   android:id="@+id/simpleEditText"
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="Email"
    android:textSize="25sp"
    android:textStyle="bold|italic"/><!--set bold and italic text style-->
```

9. background: background attribute is used to set the background of a edit text. We can set a color or a drawable in the background of a edit text.
Below is the example code with explanation included in which we set the black color for the background, white color for the displayed hint and set 10dp padding from all the side's for edit text.

```
<EditText
    android:id="@+id/simpleEditText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:hint="Enter Your Name Here"
    android:padding="15dp"
    android:textColorHint="#fff"
    android:textStyle="bold|italic"
    android:background="#000"/><!--set background color black-->
```

Background in EditText AndroidSetting Background in EditText In Java class:
Below is the example code in which we set the background color of a edit text programmatically means in java class.

```
EditText simpleEditText=(EditText)findViewById(R.id.simpleEditText);
simpleEditText.setBackgroundColor(Color.BLACK);//set black background color
```

10. padding: padding attribute is used to set the padding from left, right, top or bottom. In above example code of background we also set the 10dp padding from all the side's of edit text.

Example – EditText in Android Studio
Below is the example of edit text in which we get the value from multiple edittexts and on button click event the Toast will show the data defined in Edittext.

EditText Example In Android Studio
Step 1: Create a new project in Android Studio and name it EditTextExample.
Step 2: Now Open res -> layout -> xml (or) activity_main.xml and add following code. In this code we have added multiple edittext and a button with onclick functionality.

```
<?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:id="@+id/activity_main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="com.example.edittextexample.MainActivity">

  <EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="50dp"
    android:layout_marginStart="50dp"
    android:layout_marginTop="24dp"
    android:ems="10"
    android:hint="@string/name"
    android:inputType="textPersonName"
    android:selectAllOnFocus="true" />

  <EditText
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText1"
    android:layout_alignStart="@+id/editText1"
    android:layout_below="@+id/editText1"
    android:layout_marginTop="19dp"
    android:ems="10"
    android:hint="@string/password_0_9"
    android:inputType="numberPassword" />

  <EditText
    android:id="@+id/editText3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText2"
    android:layout_alignStart="@+id/editText2"
    android:layout_below="@+id/editText2"
    android:layout_marginTop="12dp"
    android:ems="10"
    android:hint="@string/e_mail"
    android:inputType="textEmailAddress" />

  <EditText
    android:id="@+id/editText4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText3"
    android:layout_alignStart="@+id/editText3"
    android:layout_below="@+id/editText3"
    android:layout_marginTop="18dp"
    android:ems="10"
    android:hint="@string/date"
    android:inputType="date" />

  <EditText
    android:id="@+id/editText5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText4"
    android:layout_alignStart="@+id/editText4"
    android:layout_below="@+id/editText4"
    android:layout_marginTop="18dp"
    android:ems="10"
    android:hint="@string/contact_number"
    android:inputType="phone" />

  <Button
    android:id="@+id/button"
    style="@android:style/Widget.Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/editText5"
    android:layout_marginTop="62dp"
    android:text="@string/submit"
    android:textSize="16sp"
    android:textStyle="normal|bold" />
```

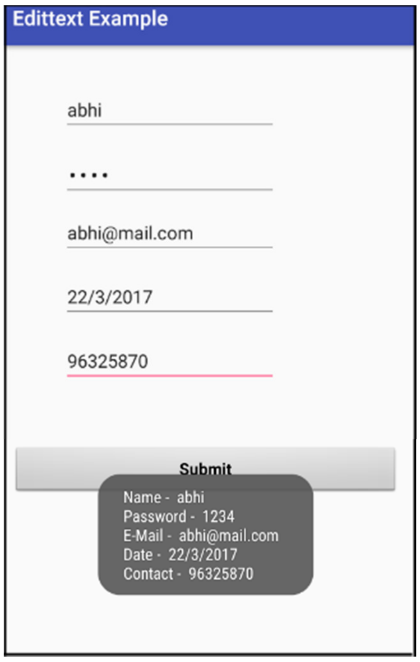
```
</RelativeLayout>
```

Step 3: Now open app -> java -> package -> MainActivity.java and add the below code. In this we just fetch the text from the edittext, further with the button click event a toast will show the text fetched before.

```
package com.example.edittextexample;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

   Button submit;
   EditText name, password, email, contact, date;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      name = (EditText) findViewById(R.id.editText1);
      password = (EditText) findViewById(R.id.editText2);
      email = (EditText) findViewById(R.id.editText3);
      date = (EditText) findViewById(R.id.editText4);
      contact = (EditText) findViewById(R.id.editText5);
      submit = (Button) findViewById(R.id.button);

      submit.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            if (name.getText().toString().isEmpty() ||
password.getText().toString().isEmpty() || email.getText().toString().isEmpty() ||
date.getText().toString().isEmpty()
                  || contact.getText().toString().isEmpty()) {
               Toast.makeText(getApplicationContext(), "Enter the Data",
Toast.LENGTH_SHORT).show();
            } else {
               Toast.makeText(getApplicationContext(), "Name -  " +
name.getText().toString() + " \n" + "Password -  " + password.getText().toString()
```

```
                        + " \n" + "E-Mail -  " + email.getText().toString() + " \n" + "Date -  " +
date.getText().toString()
                        + " \n" + "Contact -  " + contact.getText().toString(),
Toast.LENGTH_SHORT).show();
              }
          }
      });
  }
}
```

Output:



abhi

••••

abhi@mail.com

22/3/2017

96325870

Name - abhi
Password - 1234
E-Mail - abhi@mail.com
Date - 22/3/2017
Contact - 96325870

Now start the AVD in Emulator and run the App. You will see screen asking you to fill the data in required fields like name, password(numeric), email, date, contact number. Enter data and click on button. You will see the data entered will be displayed as Toast on screen.

Button In Android Studio

In Android, Button represents a push button. A Push buttons can be clicked, or pressed by the user to perform an action. There are different types of buttons used in android such as CompoundButton, ToggleButton, RadioButton.

Button Example

AndroidButton is a subclass of TextView class and compound button is the subclass of Button class. On a button we can perform different actions or events like click event, pressed event, touch event etc.

Android buttons are GUI components which are sensible to taps (clicks) by the user. When the user taps/clicks on button in an Android app, the app can respond to the click/tap. These buttons can be divided into two categories: the first is Buttons with text on, and second is buttons with an image on. A button with images on can contain both an image and a text. Android buttons with images on are also called ImageButton.

Button code in XML:

The below code will create Button and write “Abhi Android” text on it.

```
<Button
android:id="@+id/simpleButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Abhi Android"/>
```

Attributes of Button in Android:
Now let’s we discuss some important attributes that helps us to configure a Button in your xml file (layout).

1. id: id is an attribute used to uniquely identify a text Button. Below is the example code in which we set the id of a Button.

```
<Button
android:id="@+id/simpleButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Abhi Android"/>
```

2. gravity: The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc.

Below is the example code with explanation included in which we set the right and center vertical gravity for text of a Button.

```
<Button
android:id="@+id/simpleButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Abhi Android"
android:layout_centerInParent="true"
android:gravity="right|center_vertical"/><!--set the gravity of button-->
```

Button Gravity in Android

3. text: text attribute is used to set the text in a Button. We can set the text in xml as well as in the java class.

Below is the example code with explanation included in which we set the text “Learning Android @ AbhiAndroid” in a Button.

```
<Button
   android:id="@+id/simpleButton"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:text="Learn Android @ AbhiAndroid"/><!--display text on button-->
```

Setting Text on Button in AndroidSetting Text Using Java class:

Below is the example code in which we set the text on Button programmatically means in java class. The output will be same as the above.

```
Button button = (Button) findViewById(R.id.simpleButton);
button.setText("Learn Android @ AbhiAndroid");//set the text on button
```

4.textColor: textColor attribute is used to set the text color of a Button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Below is the example code with explanation included in which we set the red color for the displayed text of a Button.

```
<Button
android:id="@+id/simpleButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:text="AbhiAndroid"
android:textColor="#f00"/><!--red color for the text-->
```

Setting Text Color on Button in AndroidSetting Text Color On Button Inside Java class:

Below is the example code in which we set the text color of a Button programmatically means in java class.

```
Button simpleButton=(Button) findViewById(R.id.simpleButton);
simpleButton.setTextColor(Color.RED);//set the red color for the text
```

5. textSize: textSize attribute is used to set the size of the text on Button. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below is the example code in which we set the 25sp size for the text of a Button.

```
<Button
android:id="@+id/simpleButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:text="AbhiAndroid"
android:textSize="25sp" /><!--25sp text size-->
```

Setting TextSize on Button in AndroidSetting textSize In Java class:
Below is the example code in which we set the text size of a Button programmatically means in java class.

```
Button simpleButton=(Button)findViewById(R.id.simpleButton);
simpleButton.setTextSize(25);//set the text size of button
```

6. textStyle: textStyle attribute is used to set the text style of a Button. The possible text styles are bold, italic and normal. If we need to use two or more styles for a Button then "|" operator is used for that.

Below is the example code with explanation included, in which we set the bold and italic text styles for text of a button.

```
<Button
  android:id="@+id/simpleButton"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_centerInParent="true"
  android:text="AbhiAndroid"
  android:textSize="20sp"
  android:textStyle="bold|italic"/><!--bold and italic text style-->
```

7. background: background attribute is used to set the background of a Button. We can set a color or a drawable in the background of a Button.
Below is the example code in which we set the gren color for the background, Black color for the displayed text and set 15dp padding from all the side's for Button.

```
<Button
android:id="@+id/simpleButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:text="Download"
android:textSize="20sp"
android:padding="15dp"
```

```
android:textStyle="bold|italic"
android:background="#147D03" /><!--Background green color-->
```

setting background in Button AndroidSetting background in Button In Java class:
Below is the example code in which we set the background color of a Button programmatically means in java class.

```
Button simpleButton=(Button)findViewById(R.id.simpleButton);
simpleButton.setBackgroundColor(Color.BLACK);//set the black color of button
background
```

8. padding: padding attribute is used to set the padding from left, right, top or bottom. In above example code of background we also set the 10dp padding from all the side's of button.

9. drawableBottom: drawableBottom is the drawable to be drawn to the below of the text.

Below is the example code in which we set the icon to the below of the text.

Make sure you have image saved in your drawable folder name ic_launcher.

```
<Button
   android:id="@+id/simpleButton"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:background="#147D03"
   android:text="Download Code"
   android:textSize="20sp"
   android:padding="15dp"
   android:textStyle="bold|italic"
   android:drawableBottom="@drawable/ic_launcher"/><!--image drawable on button-
->
```

drawableBottom in Button in Android10. drawableTop, drawableRight And drawableLeft: Just like the above attribute we can draw drawable to the left, right or top of text.

In the Below example we set the icon to the right of the text. In the same way you can do for other two attribute by your own:

```
<Button
   android:id="@+id/simpleButton"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
```

```
    android:layout_centerInParent="true"
    android:background="#147D03"
    android:text="Download Code"
    android:textSize="20sp"
    android:padding="15dp"
    android:textStyle="bold|italic"
    android:drawableRight="@drawable/ic_launcher"/>drawableRight of Text on Button
in Android
```

Button Example In Android Studio:
Below is the example of button in which we display two buttons with different background and whenever a user click on the button the text of the button will be displayed in a toast.

Step 1: Create a new project in Android Studio and name it ButtonExample.
Select File -> New -> New Project and Fill the forms and click "Finish" button.
Step 2: Now open res -> layout -> xml (or) activity_main.xml and add following code.
Here we are designing the UI of two button in Relative Layout.

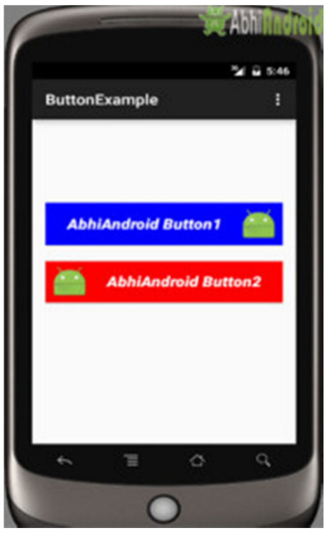
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/simpleButton1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="100dp"
        android:background="#00f"
        android:drawableRight="@drawable/ic_launcher"
        android:hint="AbhiAndroid Button1"
        android:padding="5dp"
        android:textColorHint="#fff"
        android:textSize="20sp"
        android:textStyle="bold|italic" />

    <Button
        android:id="@+id/simpleButton2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:background="#f00"
        android:drawableLeft="@drawable/ic_launcher"
        android:hint="AbhiAndroid Button2"
        android:padding="5dp"
        android:textColorHint="#fff"
        android:textSize="20sp"
        android:textStyle="bold|italic" />
</RelativeLayout>
```

Step 3: Now Open app -> package -> MainActivity.java and the following code. Here using setOnClickListener() method on button and using Toast we will display which button is clicked by user.

```
package example.abhiandriod.buttonexample;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

  Button simpleButton1, simpleButton2;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    simpleButton1 = (Button) findViewById(R.id.simpleButton1);//get id of button 1
    simpleButton2 = (Button) findViewById(R.id.simpleButton2);//get id of button 2

    simpleButton1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Toast.makeText(getApplicationContext(), "Simple Button 1",
Toast.LENGTH_LONG).show();//display the text of button1
      }
    });
    simpleButton2.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Toast.makeText(getApplicationContext(), "Simple Button 2",
Toast.LENGTH_LONG).show();//display the text of button2
      }
    });
  }
}
```

Output:

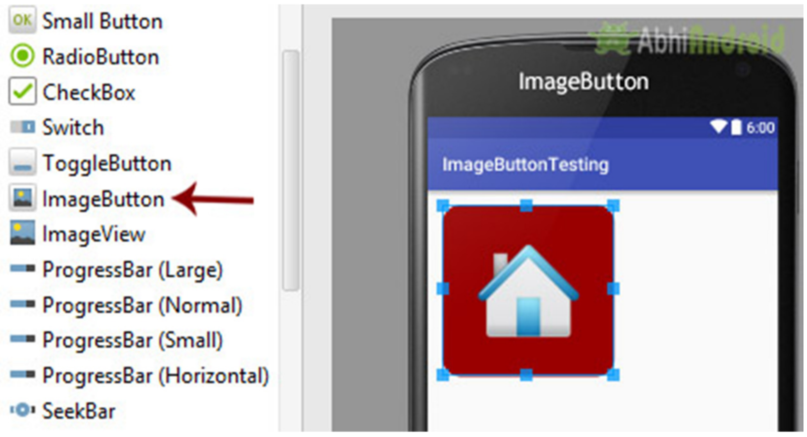
Now start the AVD in Emulator and run the App. You will see two button. Click on any button and you will see the message on screen which button is clicked.



button example output android

ImageButton In Android Studio

In Android, ImageButton is used to display a normal button with a custom image in a button. In simple words we can say, ImageButton is a button with an image that can be pressed or clicked by the users. By default it looks like a normal button with the standard button background that changes the color during different button states.



An image on the surface of a button is defined within a xml (i.e. layout) by using src attribute or within java class by using setImageResource() method. We can also set an image or custom drawable in the background of the image button.

Important Note: Standard button background image is displayed in the background of button whenever you create an image button. To remove that image, you can define your own background image in xml by using background attribute or in java class by using setBackground() method.

Below is the code and image which shows how custom imagebutton looks in Android:

Important Note: ImageButton has all the properties of a normal button so you can easily perform any event like click or any other event which you can perform on a normal button.

ImageButton code in XML:

```
<!--Make Sure To Add Image Name home in Drawable Folder-->

<ImageButton

android:id="@+id/simpleImageButton"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:src="@drawable/home" />
```

Attributes of ImageButton:

Now let’s we discuss some important attributes that helps us to configure a image button in your xml file (layout).

1. id: id is an attribute used to uniquely identify a image button. Below is the example code in which we set the id of a image button.

```
<ImageButton
android:id="@+id/simpleImageButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

2. src: src is an attribute used to set a source file of image or you can say image in your image button to make your layout look attractive.

Below is the example code in which we set the source of an image button. Make sure you have saved an image in drawable folder name home before using below code.

```
<ImageButton
  android:id="@+id/simpleImageButton"
  android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"

    android:src="@drawable/home"/> <!--src(source)file from drawable folder which display an imagebutton-->
```

Setting Image Source In ImageButton Using Java class:

We can also set the source image at run time programmatically in java class. For that we use setImageResource() function as shown in below example code.

```
/*Add in Oncreate() funtion after setContentView()*/

ImageButton simpleImageButton = (ImageButton)findViewById(R.id.simpleImageButton);

simpleImageButton.setImageResource(R.drawable.home); //set the image programmatically
```

3. background: background attribute is used to set the background of an image button. We can set a color or a drawable in the background of a Button.

Below is the example code in which we set the black color for the background and an home image as the source of the image button.

```
<ImageButton

    android:id="@+id/simpleImageButton"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:src="@drawable/home"

    android:background="#000"/><!-- black background color for image button-->
```

Setting Background In ImageButton Using Java class:
Below is the example code in which we set the black background color of a image button programmatically means in java class.

```
/*Add in Oncreate() funtion after setContentView()*/

ImageButton simpleImageButton =findViewById(R.id.simpleImageButton);

simpleImageButton.setBackgroundColor(Color.BLACK);
```

4. padding: padding attribute is used to set the padding from left, right, top or bottom of the ImageButton.

- **paddingRight :** set the padding from the right side of the image button**.**
- **paddingLeft :** set the padding from the left side of the image button**.**
- **paddingTop :** set the padding from the top side of the image button**.**
- **paddingBottom :** set the padding from the bottom side of the image button**.**
- **padding :** set the padding from the all side's of the image button**.**

Below is the example code of padding attribute in which we set the 20dp padding from all the side’s of a image button.

```
<ImageButton
   android:id="@+id/simpleImageButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:background="#000"
   android:src="@drawable/home"
   android:padding="30dp"/>
```

Example of ImageButton In Android Studio:

In the below example of ImageButton we display two custom image buttons with source and background. One is simple image button with simple background and other one is a round corner image button and whenever you click on an button, the name of the button will be displayed in a toast. Below is the code and final output:

Step 1: Create a new project and name it ImageButtonExample
In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

```
Select File -> New -> New Project and Fill the forms and click "Finish" button.
```

Step 3: Right click on **drawable** -> New -> Drawable resource file and create new xml file name **custom_image-buttton.xml** and add following code

In this Step we create drawable xml in which we used solid and corner properties, solid is used to set the background color for the image button and corner is used to set the radius for button corners.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="#900" /><!-- background color for imagebutton-->
   <corners android:radius="20dp" /><!-- round corners for imagebutton-->
</shape>
```

Step 3: Open app -> layout -> **activity_main.xml (or) main.xml** and add following code:

In this step, we open an xml file and add the code which display two custom image buttons by using src, background, gravity and other attributes in Relative Layout.

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

<!--Make sure to save two images home and youtube in drawable folder-->

    <ImageButton
        android:id="@+id/simpleImageButtonHome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:background="@drawable/custom_image_button"
        android:padding="20dp"
        android:src="@drawable/home" />

    <ImageButton
        android:id="@+id/simpleImageButtonYouTube"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/simpleImageButtonHome"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:background="#005"
        android:padding="20dp"
        android:src="@drawable/youtube" />

</RelativeLayout>
```

Step 4: Open app -> package -> **MainActivity.java**

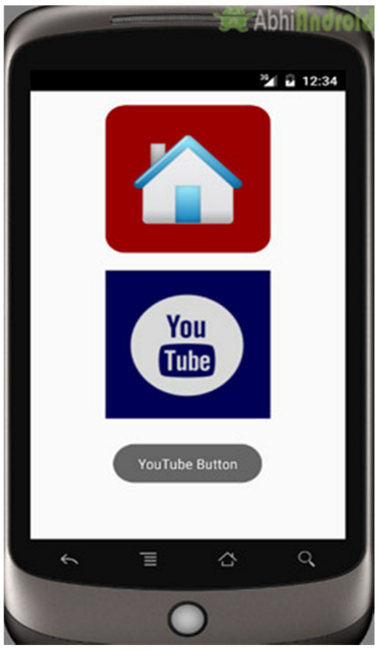
In this step, we add the code to initiate the image button’s and then perform click event on them and display the text for selected item using a toast.

```
package example.abhiandriod.imagebuttonexample;


import android.app.Activity;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import android.widget.Toast;



public class MainActivity extends Activity {


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
// initiate view's
    ImageButton simpleImageButtonHome = (ImageButton)findViewById(R.id.simpleI
mageButtonHome);
    ImageButton simpleImageButtonYouTube = (ImageButton)findViewById(R.id.simpl
eImageButtonYouTube);


// perform click event on button's
    simpleImageButtonHome.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Toast.makeText(getApplicationContext(),"Home  Button",Toast.LENGTH_LONG)
.show();// display the toast on home button click
      }
```

```
        });
        simpleImageButtonYouTube.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(getApplicationContext(),"YouTube  Button",Toast.LENGTH_LO
NG).show();// display the toast on you tube button click
            }
        });
    }
}
```

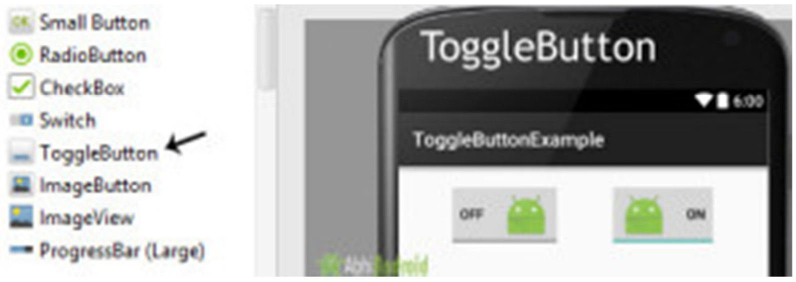
Output:

Now start the AVD in Emulator and run the App. You will see two ImageButton out of which top one is round corner. Click on any image and its name will be displayed on screen.

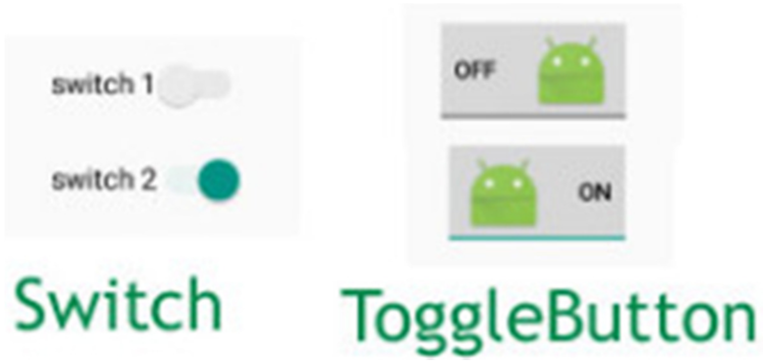


ToggleButton (On/Off) In Android

In Android, ToggleButton is used to display checked and unchecked state of a button. ToggleButton basically an off/on button with a light indicator which indicate the current state of toggle button. The most simple example of ToggleButton is doing on/off in sound, Bluetooth, wifi, hotspot etc. It is a subclass of compoundButton.



ToggleButton Vs Switch In Android:
ToggleButton allow the users to change the setting between two states like turn on/off your wifi, Bluetooth etc from your phone's setting menu. Since, Android 4.0 version (API level 14) there is an another kind of ToggleButton called Switch which provide the user slider control. You can learn more about it reading Switch tutorial.



Important Note: Android Switch and ToggleButton both are the subclasses of CompoundButton class.
ToggleButton code in XML:

```
<ToggleButton

android:id="@+id/simpleToggleButton"

android:layout_width="wrap_content"

android:layout_height="wrap_content"/>
```

How To Check Current State Of ToggleButton:

To check current state of a toggle button programmatically we use isChecked() method. This method returns a Boolean value either true or false. If a toggle button is checked then it returns true otherwise it returns false. Below is the code which checks the current state of a toggle button.

```
/*Add in Oncreate() funtion after setContentView()*/

ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleBu
tton); // initiate a toggle button

Boolean ToggleButtonState = simpleToggleButton.isChecked();
```

Attributes of ToggleButton:

Now let's we discuss important attributes that helps us to configure a Toggle Button in XML file (layout).

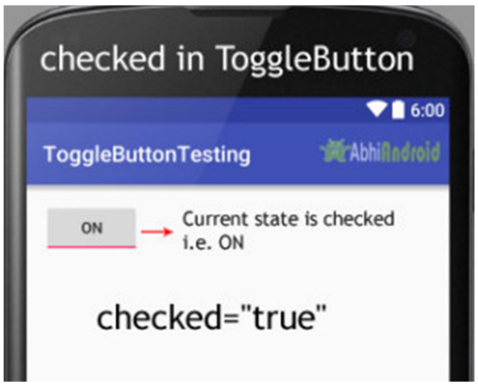
1. id: id is an attribute used to uniquely identify a toggle button.

```
<ToggleButton
android:id="@+id/simpleToggleButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

2. checked: checked is an attribute of toggle button used to set the current state of a toggle button. The value should be true or false where true shows the checked state and false shows unchecked state of a toggle button. The default value of checked attribute is false. We can also set the current state programmatically.

Below we set true value for checked attribute sets the current state to checked.

```
<ToggleButton
   android:id="@+id/simpleToggleButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="true" />
```

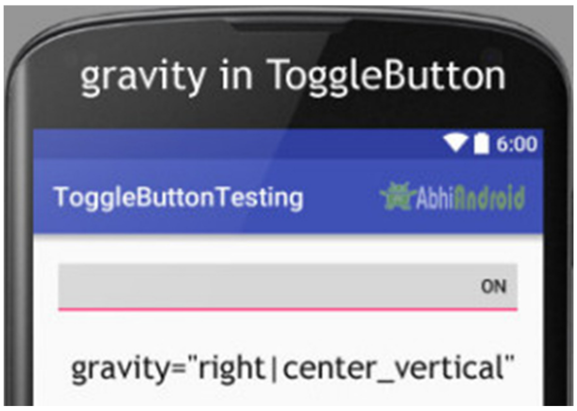


Setting Checked Of TóogleButton Current State In Java Class:
Below we set the current state of toggle button to checked:

```
/*Add in Oncreate() funtion after setContentView()*/
ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleButton); // initiate a toggle button
simpleToggleButton.setChecked(true); // set the current state of a toggle button
```

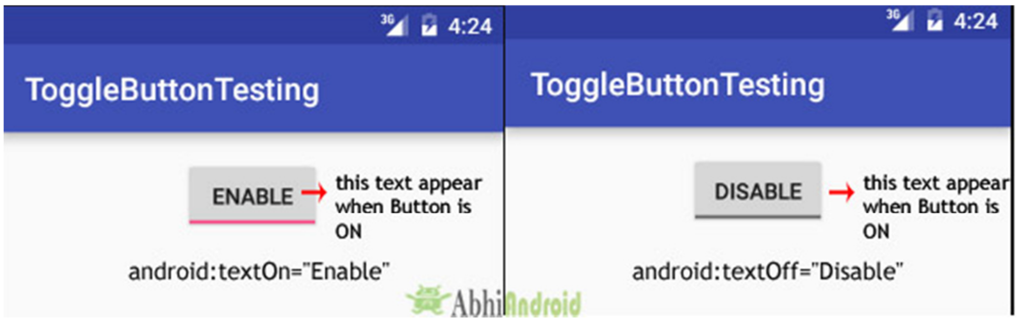
3. gravity: The gravity attribute is an optional attribute which is used to control the alignment of the text in ToogleButton like left, right, center, top, bottom, center_vertical, center_horizontal etc.

```
<ToggleButton
   android:id="@+id/simpleToggleButton"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerHorizontal="true"
   android:checked="true"
   android:gravity="right|center_vertical"/>
```



4. textOn And textOff: textOn attribute is used to set the text when toggle button is in checked/on state. We can set the textOn in XML as well as in the java class.
Below is the example code with explanation included in which we set the textOn "Enabble Attribute Of Toggle button" for a toggle button.

```
<ToggleButton
   android:id="@+id/simpleToggleButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="true"
   android:layout_centerHorizontal="true"
   android:textOff="Disable"
   android:textOn="Enable"/>
```


ToggleButtonTesting
ENABLE
this text appear when Button is ON
android:textOn="Enable"
ToggleButtonTesting
DISABLE
this text appear when Button is ON
android:textOff="Disable"
AbhiAndroid

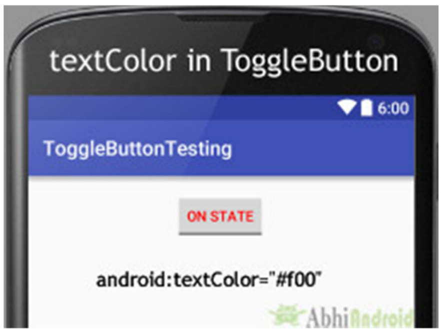
Setting textOn and textOff Of ToggleButton In Java class:

```
/*Add in Oncreate() funtion after setContentView()*/
// initiate toggle button
ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleBu
tton);
// displayed text of the toggle button whenever it is in checked or on state
simpleToggleButton.setTextOn("TextOn Attribute Of Toggle b3`utton");
// displayed text of the toggle button whenever it is in unchecked or off state
simpleToggleButton.setTextOff("TextOff Attribute Of Toggle b3`utton");
```

5. textColor: textColor attribute is used to set the text color of a toggle button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Below we set the red color for the displayed text of a Toggle button.

```
<ToggleButton
   android:id="@+id/simpleToggleButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="false"
   android:textOff="On State"
   android:textOn="Off State"
   android:layout_centerHorizontal="true"
   android:textColor="#f00" />
```



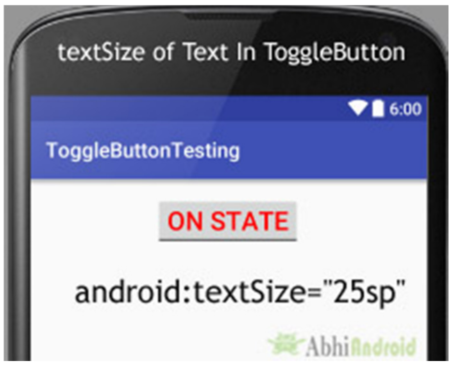
Setting textColor Of ToggleButton In Java class:

```
/*Add in Oncreate() funtion after setContentView()*/
ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleButton);// initiate toggle button
simpleToggleButton.setTextColor(Color.RED); //red color for displayed text of toggle button
```

6. textSize: textSize attribute set the size of the text of a toggle button. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below we set the 25sp size for the text of a toggle button.

```
<ToggleButton
    android:id="@+id/simpleToggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="false"
    android:textOff="On State"
    android:textOn="Off State"
    android:layout_centerHorizontal="true"
    android:textColor="#f00"
    android:textSize="25sp"/>
```



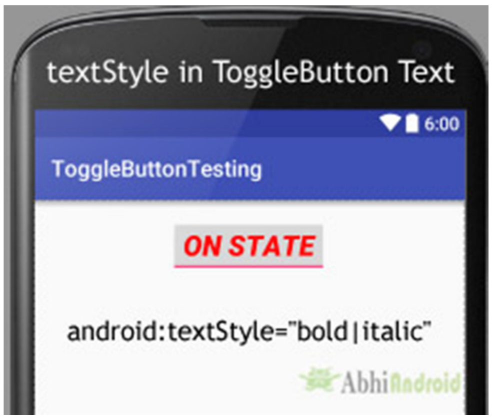
Setting textSize Of ToggleButton Text In Java class:

```
/*Add in Oncreate() funtion after setContentView()*/
ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleButton); // initiate toggle button
simpleToggleButton.setTextSize(25); // set 25sp displayed text size of toggle button
```

7. textStyle: textStyle attribute is used to set the text style of the text of a Toggle button. You can set bold, italic and normal. If you need to use two or more styles for a text view then "|" operator is used for that.

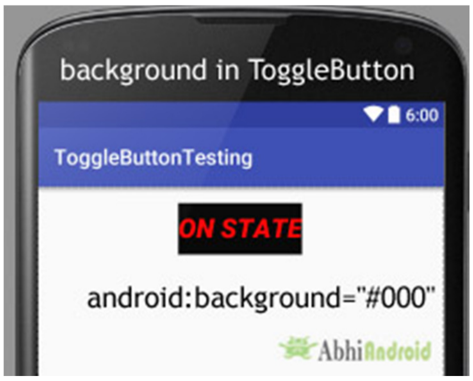
Below we set the bold and italic text styles for text of a toggle button.

```
<ToggleButton
  android:id="@+id/simpleToggleButton"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:checked="true"
  android:textOff="Off State"
  android:textOn="On State"
  android:textSize="25sp"
  android:layout_centerHorizontal="true"
  android:textColor="#f00"
  android:textStyle="bold|italic"/>
```


textStyle in ToggleButton Text
6:00
ToggleButtonTesting
ON STATE
android:textStyle="bold|italic"
AbhiAndroid

8. background: background attribute is used to set the background of a toggle button. We can set a color or a drawable in the background of a toggle button.
Below we set the black color for the background and red color for the displayed text of a ToggleButton.

```
<ToggleButton
  android:id="@+id/simpleToggleButton"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:checked="true"
  android:textOff="Off State"
  android:textOn="On State"
  android:textSize="25sp"
  android:layout_centerHorizontal="true"
  android:textStyle="bold|italic"
  android:textColor="#f00"
  android:background="#000"/>
```



Setting Background Of ToggleButton Text In Java class:

```
/*Add in Oncreate() funtion after setContentView()*/
ToggleButton simpleToggleButton = (ToggleButton) findViewById(R.id.simpleToggleButton);
simpleToggleButton.setBackgroundColor(Color.BLACK);
```

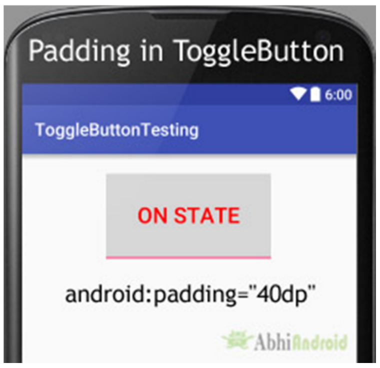
9. padding: padding attribute is used to set the padding from left, right, top or bottom.

- **paddingRight :**set the padding from the right side of the toggle button**.**
- **paddingLeft :**set the padding from the left side of the toggle button**.**
- **paddingTop :**set the padding from the top side of the toggle button**.**
- **paddingBottom :**set the padding from the bottom side of the toggle button**.**
- **Padding :**set the padding from the all side's of the toggle button**.**

Below we set the 40dp padding from all the side's of the toggle button.

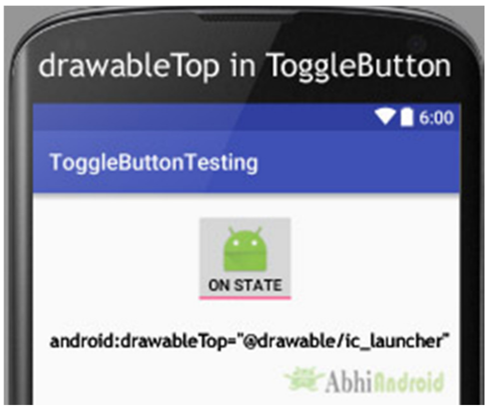
```
<ToggleButton
  android:id="@+id/simpleToggleButton"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:checked="true"
  android:textOff="Off State"
  android:textOn="On State"
  android:textSize="25sp"
  android:layout_centerHorizontal="true"
```

```
   android:textColor="#f00"
   android:padding="40dp"/>
```



10. drawableBottom, drawableTop, drawableRight And drawableLeft: These attribute draw the drawable below, top, right and left of the text of ToggleButton. Below we set the icon to the Top of the text of a ToggleButton. In the similar way you can try for other three attribute yourself.

```
<!--Make sure to add ic_launcher image in drawable folder-->
<ToggleButton
   android:id="@+id/simpleToggleButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="true"
   android:textOff="Off State"
   android:textOn="On State"
   android:layout_centerHorizontal="true"
   android:textColor="#000"
   android:drawableTop="@drawable/ic_launcher" />
```


drawableTop in ToggleButton
6:00
ToggleButtonTesting
ON STATE
android:drawableTop="@drawable/ic_launcher"
AbhiAndroid

ToggleButton Example In Android Studio:

Below is the example of ToggleButton in Android Studio. In this example we display two toggle button with background and one "submit" button using attributes discussed earlier in this post. Whenever user click on the submit button, the current state of both toggle button's is displayed in a Toast. Below is the final output, download code and step by step explanation:

Step 1: Create a new project and name it ToggleButtonExample
In this step we create a new project for ToggleButton in Android Studio by filling all the necessary details of the app like app name, package name, api versions etc.

```
Select File -> New -> New Project -> Fill the forms and click "Finish" button.
```

Step 2: Open res -> layout -> activity_main.xml (or) main.xml and add following code:

In this step we open an xml file and add the code for displaying two toggle button and one normal Button.

```
<LinearLayout 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"
  android:orientation="vertical"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:orientation="horizontal">

    <ToggleButton
      android:id="@+id/simpleToggleButton1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
```

```
        android:checked="false"
        android:drawablePadding="20dp"
        android:drawableRight="@drawable/ic_launcher"
        android:textColor="#000" />

      <ToggleButton
        android:id="@+id/simpleToggleButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="50dp"
        android:checked="true"
        android:drawableLeft="@drawable/ic_launcher"
        android:drawablePadding="20dp"
        android:textColor="#000" />
    </LinearLayout>

    <Button
      android:id="@+id/submitButton"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:layout_marginTop="50dp"
      android:background="#0f0"
      android:padding="10dp"
      android:text="Submit"
      android:textColor="#fff"
      android:textSize="20sp"
      android:textStyle="bold" />
</LinearLayout>
```

Step 3: Open app -> java -> package -> **MainActivity.java**

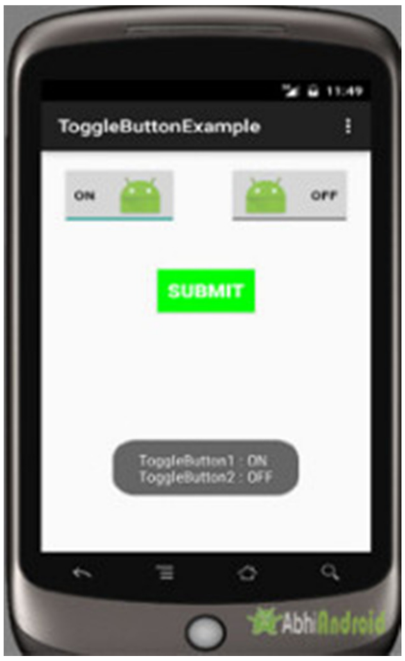
In this step we open MainActivity where we add the code to initiate the Toggle Buttons and normal Button. After initiating we perform click event on button and display the text of current state of ToggleButton using a Toast.

```
package example.abhiandriod.togglebuttonexample;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.ToggleButton;


public class MainActivity extends AppCompatActivity {


   ToggleButton simpleToggleButton1, simpleToggleButton2;
   Button submit;


   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // initiate toggle button's
      simpleToggleButton1 = (ToggleButton) findViewById(R.id.simpleToggleButton1);
      simpleToggleButton2 = (ToggleButton) findViewById(R.id.simpleToggleButton2);
      submit = (Button) findViewById(R.id.submitButton);
      submit.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            String status = "ToggleButton1 : " + simpleToggleButton1.getText() + "\n" + "To
ggleButton2 : " + simpleToggleButton2.getText();
            Toast.makeText(getApplicationContext(), status, Toast.LENGTH_SHORT).show(
); // display the current state of toggle button's
```

```
        }
      });
   }



}
```

Output:

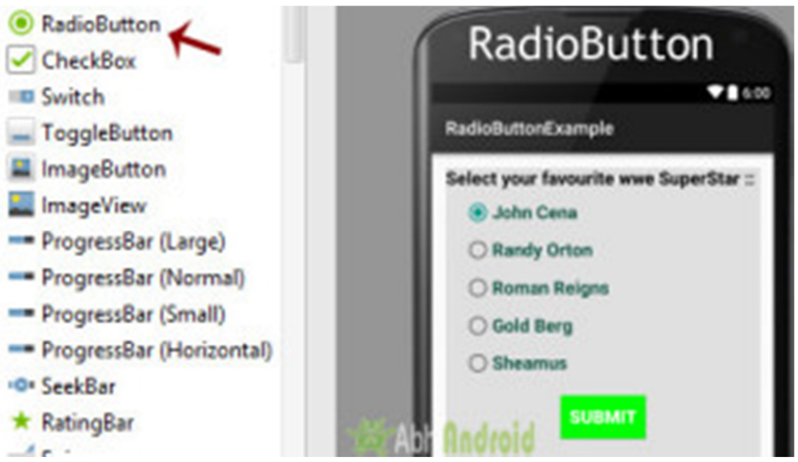
Now start the AVD in Emulator and run the App. You will see two ToggleButton and submit Button. Click on the submit button which will display the state of ToggleButton.



ToggleButton1 : ON
ToggleButton2 : OFF

RadioButton & RadioGroup In Android Studio

In Android, RadioButton are mainly used together in a RadioGroup. In RadioGroup checking the one radio button out of several radio button added in it will automatically unchecked all the others. It means at one time we can checked only one radio button from a group of radio buttons which belong to same radio group. The most common use of radio button is in Quiz Android App code.



RadioButon is a two state button that can be checked or unchecked. If a radio button is unchecked then a user can check it by simply clicking on it. Once a RadiaButton is checked by user it can't be unchecked by simply pressing on the same button. It will automatically unchecked when you press any other RadioButton within same RadioGroup.

Important Note: RadioGroup is a widget used in Android for the grouping of radio buttons and provide the feature of selecting only one radio button from the set. When a user try to select any other radio button within same radio group the previously selected radio button will be automatically unchecked.

RadioGroup And RadioButton code in XML:

```
<RadioGroup
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
    <RadioButton
    android:id="@+id/simpleRadioButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
    <RadioButton
    android:id="@+id/simpleRadioButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</RadioGroup>
```

Checking Current State Of Radio Button:

You can check the current state of a radio button programmatically by using isChecked() method. This method returns a Boolean value either true or false. if it is checked then returns true otherwise returns false. Below is an example code with explanation in which we checked the current state of a radio button.

```
/*Add in Oncreate() funtion after setContentView()*/

RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.simpleRadioButton); // initiate a radio button


Boolean RadioButtonState = simpleRadioButton.isChecked();
```

Attributes of RadioButton In Android:

Now let's we discuss important attributes that helps us to create a beautiful radio button in xml file (layout).

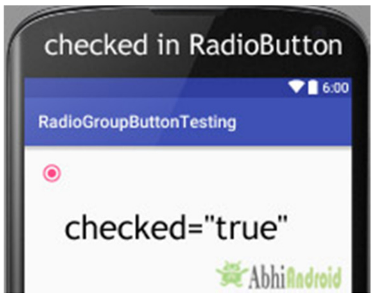
1. id: id is an attribute used to uniquely identify a radio button.

```
<RadioButton
   android:id="@+id/simpleRadioButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
```

2. checked: checked attribute in radio button is used to set the current state of a radio button. We can set it either true or false where true shows the checked state and false shows unchecked state of a radio button. As usual default value of checked attribute is false. We can also set the current state in JAVA.

Below we set true value for checked attribute which sets the current state to checked of a Button

```
<RadioButton
   android:id="@+id/simpleRadioButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="true"/>
```



Setting checked State Of RadioButton In Java Class:
Below code set the current state of RadioButton to checked programmatically.

```
/*Add in Oncreate() funtion after setContentView()*/
// initiate a radio button
```

```
RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.simpleRadioButto
n);


// set the current state of a radio button
simpleRadioButton.setChecked(true);
```

3. text: text attribute is used to set the text in a radio button. We can set the text both ways either in XML or in JAVA class.

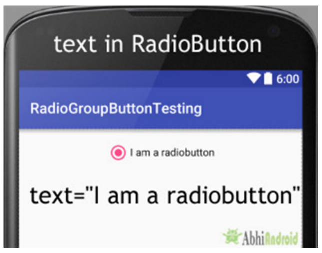
Below is the example code with explanation included in which we set the text "I am a radiobutton" of a radio button.

```
<RadioButton
  android:id="@+id/simpleRadioButton"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:checked="true"
  android:layout_centerHorizontal="true"
  android:text="I am a radiobutton" /> <!-- displayed text of radio button-->
```

Setting text of RadioButton In Java class:

Below we set the text of a radio button programmatically:

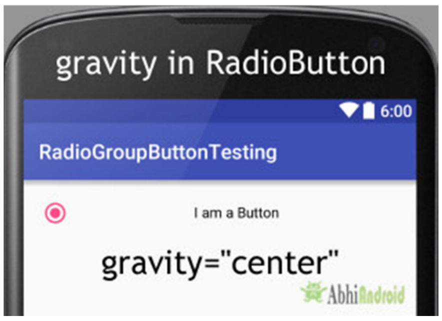
```
/*Add in Oncreate() funtion after setContentView()*/
RadioButton simpleRadioButton=(RadioButton) findViewById(R.id.simpleRadioButton)
;
simpleRadioButton.setText("I am a radiobutton");
```



4. gravity: The gravity attribute is an optional attribute which is used to control the alignment of text like left, right, center, top, bottom, center_vertical, center_horizontal etc.
Below is the example code with explanation included in which we set the center gravity for the text of a radio button.

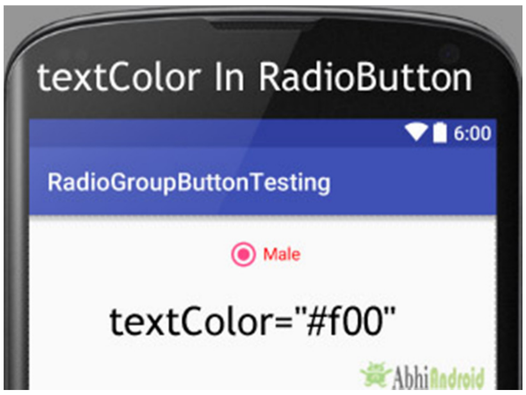
```
<RadioButton
  android:id="@+id/simpleRadioButton"
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="I am a Button"
    android:gravity="center"/>
```



5. textColor: textColor attribute is used to set the text color of a radio button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.
Below we set the red color for the displayed text of a radio button.

```
<RadioButton
    android:id="@+id/simpleRadioButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:layout_centerHorizontal="true"
    android:text="Male"
    android:textColor="#f00" />
```



Setting textColor of RadioButton text In Java class:
Below we set the text color of a radio button programmatically.

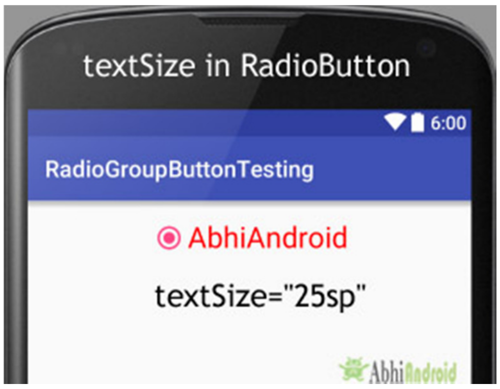
```
/*Add in Oncreate() funtion after setContentView()*/
```

```
RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.simpleRadioButton);// initiate radio button


simpleRadioButton.setTextColor(Color.RED); //red color for displayed text of radio button
```

6. textSize: textSize attribute is used to set the size of the text of a radio button. We can set the text size in sp(scale independent pixel) or dp(density pixel).

Below we set the 25sp size for the text of a radio button.

```
<RadioButton
    android:id="@+id/simpleRadioButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:layout_centerHorizontal="true"
    android:text="AbhiAndroid"
    android:textColor="#f00"
    android:textSize="25sp"/>
```



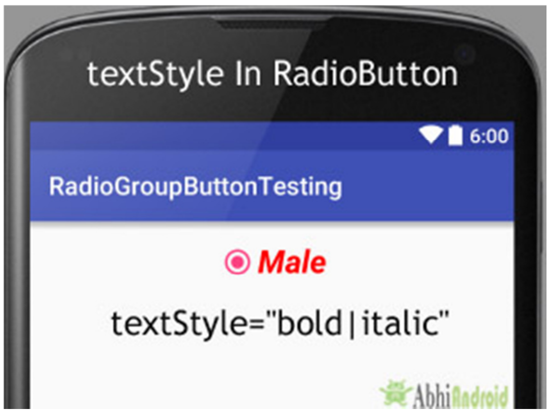
Setting textSize Of RadioButton Text In Java class:
Below we set the text size of a radio button programmatically:

```
/*Add in Oncreate() funtion after setContentView()*/

RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.simpleRadioButton); // initiate radio button


simpleRadioButton.setTextSize(25); // set 25sp displayed text size of radio button
```

7. textStyle: textStyle attribute is used to set the text style of the text of a radio button. The possible text styles are bold, italic and normal. If we need to use two or more styles for a text view then "|" operator is used for that.

Below is the example code with explanation included in which we set the bold and italic text styles for text of a radio button.

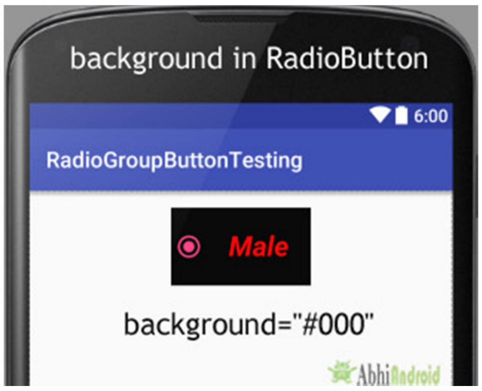
```
<RadioButton
   android:id="@+id/simpleRadioButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="true"
   android:textSize="25sp"
   android:layout_centerHorizontal="true"
   android:text="Male"
   android:textColor="#f00"
   android:textStyle="bold|italic"/>
```



8. background: background attribute is used to set the background of a radio button. We can set a color or a drawable in the background of a radio button.
Below we set the black color for the background and red color for the displayed text of a radio button.

```
<RadioButton
   android:id="@+id/simpleRadioButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="true"
   android:textSize="25sp"
   android:textStyle="bold|italic"
   android:padding="20dp"
```

```
    android:layout_centerHorizontal="true"
    android:text="Male"
    android:textColor="#f00"
    android:background="#000"/>
```



Setting Background Of RadioButton In Java class:
Below we set the background color of a radio button programmatically.

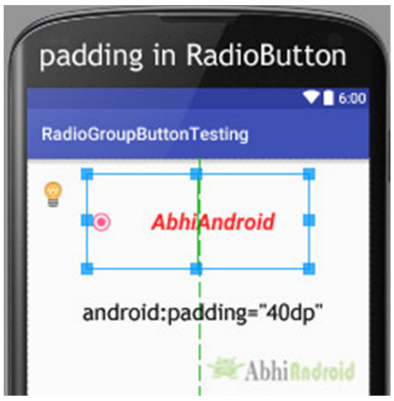
```
/*Add in Oncreate() funtion after setContentView()*/
RadioButton simpleRadioButton = (RadioButton) findViewById(R.id.simpleRadioButton);


simpleRadioButton.setBackgroundColor(Color.BLACK);
```

9. padding: padding attribute is used to set the padding from left, right, top or bottom.

- **paddingRight:** set the padding from the right side of the radio button.
- **paddingLeft :** set the padding from the left side of the radio button.
- **paddingTop :** set the padding from the top side of the radio button.
- **paddingBottom:** set the padding from the bottom side of the radio button.
- **Padding:** set the padding from the all side's of the radio button.

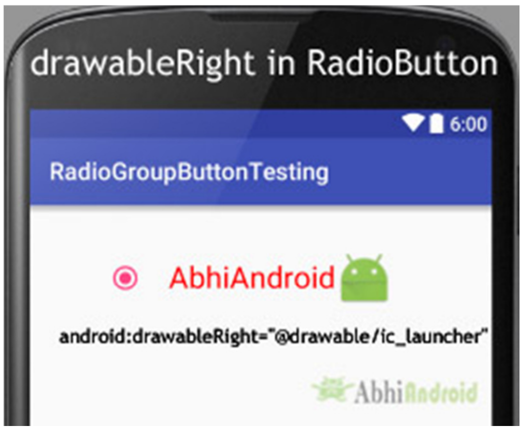
Below we set padding attribute of 20dp padding from all the side's of the radio button.

```
<RadioButton
    android:id="@+id/simpleRadioButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:textSize="25sp"
    android:textStyle="bold|italic"
    android:layout_centerHorizontal="true"
    android:text="AbhiAndroid"
    android:textColor="#f00"
    android:padding="40dp"/>
```



10. drawableBottom, drawableTop, drawableLeft And drawableRight: These attribute draw the drawable to the below of the text of RadioButton.
Below we set the icon to the right of the text of a RadioButton.

```
<RadioButton
    android:id="@+id/simpleRadioButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:textSize="25sp"
    android:padding="20dp"
    android:layout_centerHorizontal="true"
    android:text="AbhiAndroid"
    android:textColor="#f00"
    android:drawableRight="@drawable/ic_launcher" />
```


drawableRight in RadioButton
6:00
RadioGroupButtonTesting
AbhiAndroid
android:drawableRight="@drawable/ic_launcher"
AbhiAndroid

Example Of RadioButton And RadioGroup in Android Studio:

Below is the example of Radiobutton in Android where we display five radio buttons with background and other attributes. The radio buttons are used to select your favorite WWE superstar with one "submit" button. Below is the final output, download code and step by step explanation of tutorial:

Step 1: Create a new project and name it RadioButtonExample
Select File -> New -> New Project and Fill the forms and click "Finish" button.

Step 2: Open res -> layout -> activity_main.xml (or) main.xml and add following code:

In this step we open an xml file and add the code for displaying 5 RadioButton and one normal button.

```
<LinearLayout 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"
  android:orientation="vertical"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#e0e0e0"
    android:orientation="vertical">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Select your favourite wwe SuperStar :: "
    android:textColor="#000"
    android:textSize="20sp"
    android:textStyle="bold" />

<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <RadioButton
        android:id="@+id/johnCena"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="10dp"
        android:checked="true"
        android:text="@string/johnCena"
        android:textColor="#154"
        android:textSize="20sp"
        android:textStyle="bold" />

    <RadioButton
        android:id="@+id/randyOrton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="10dp"
        android:checked="false"
        android:text="@string/randyOrton"
        android:textColor="#154"
        android:textSize="20sp"
        android:textStyle="bold" />

    <RadioButton
        android:id="@+id/romanReigns"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="10dp"
        android:checked="false"
        android:text="@string/romanReigns"
        android:textColor="#154"
        android:textSize="20sp"
        android:textStyle="bold" />

    <RadioButton
        android:id="@+id/goldBerg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="10dp"
        android:checked="false"
        android:text="@string/goldBerg"
        android:textColor="#154"
        android:textSize="20sp"
        android:textStyle="bold" />



    <RadioButton
        android:id="@+id/sheamus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
```

```
            android:layout_marginTop="10dp"
            android:checked="false"
            android:text="@string/sheamus"
            android:textColor="#154"
            android:textSize="20sp"
            android:textStyle="bold" />

        </RadioGroup>

        <Button
          android:id="@+id/submitButton"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_gravity="center"
          android:layout_margin="20dp"
          android:background="#0f0"
          android:padding="10dp"
          android:text="Submit"
          android:textColor="#fff"
          android:textSize="20sp"
          android:textStyle="bold" />
    </LinearLayout>



</LinearLayout>
```

Step 3: Open src -> package -> **MainActivity.java**

In this step we open MainActivity and add the code to initiate the RadioButton and normal button. We also perform click event on button and display the selected superstar's name by using a Toast.

```
package example.gb.radiobuttonexample;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    RadioButton johnCena, randyOrton, goldBerg, romanReigns, sheamus;
    String selectedSuperStar;
    Button submit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        johnCena = (RadioButton) findViewById(R.id.johnCena);
        randyOrton = (RadioButton) findViewById(R.id.randyOrton);
        goldBerg = (RadioButton) findViewById(R.id.goldBerg);
        romanReigns = (RadioButton) findViewById(R.id.romanReigns);
        sheamus = (RadioButton) findViewById(R.id.sheamus);
        submit = (Button) findViewById(R.id.submitButton);
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (randyOrton.isChecked()) {
                    selectedSuperStar = randyOrton.getText().toString();
                } else if (sheamus.isChecked()) {
                    selectedSuperStar = sheamus.getText().toString();
                } else if (johnCena.isChecked()) {
                    selectedSuperStar = johnCena.getText().toString();
                } else if (romanReigns.isChecked()) {
                    selectedSuperStar = romanReigns.getText().toString();
                } else if (goldBerg.isChecked()) {
```

```
                selectedSuperStar = goldBerg.getText().toString();
            }
            Toast.makeText(getApplicationContext(), selectedSuperStar, Toast.LENGTH_LO
NG).show(); // print the value of selected super star
        }
    });
  }
}
```

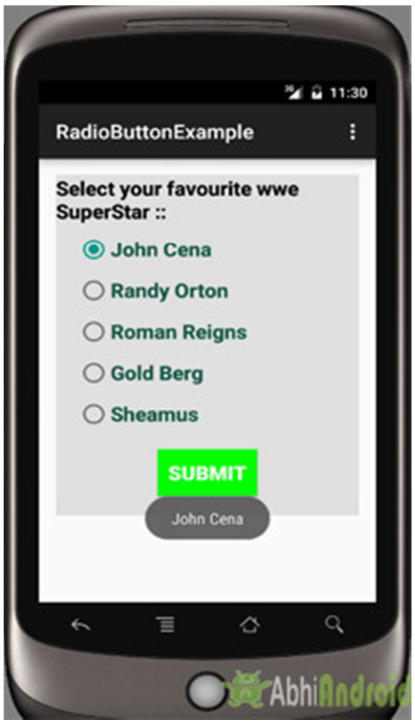
Step 4: Open res -> values -> **strings.xml**

In this step we open String file which is used to store string data of the app.

```
<resources>
  <string name="app_name">RadioButtonExample</string>
  <string name="hello_world">Hello world!</string>
  <string name="action_settings">Settings</string>
  <string name="randyOrton">Randy Orton</string>
  <string name="johnCena">John Cena</string>
  <string name="romanReigns">Roman Reigns</string>
  <string name="goldBerg">Gold Berg</string>
  <string name="sheamus">Sheamus</string>
</resources>
```

Run The App:

Now run the App in Emulator and you will see 5 RadioButton in RadioGroup listing WWE superstar name. Now choose your favorite one and click on Submit Button. The name will be displayed on Screen.



ProgressBar In Android Studio

In Android, ProgressBar is used to display the status of work being done like analyzing status of work or downloading a file etc. In Android, by default a progress bar will be displayed as a spinning wheel but If we want it to be displayed as a horizontal bar then we need to use style attribute as horizontal. It mainly use the **"android.widget.ProgressBar"** class.

Important Note: A progress bar can also be made indeterminate. In this mode a progress bar shows a cyclic animation without an indication of progress. This mode is used in application when we don't know the amount of work to be done.
To add a progress bar to a layout (xml) file, you can use the <ProgressBar> element. By default, a progress bar is a spinning wheel (an indeterminate indicator). To change to a horizontal progress bar, apply the progress bar's horizontal style.

ProgressBar code:

```
<ProgressBar
android:id="@+id/simpleProgressBar"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

Horizontal ProgressBar code:

```
<ProgressBar
android:id="@+id/simpleProgressBar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

```
style="@style/Widget.AppCompat.ProgressBar.Horizontal"/>
```

Important Methods Used In ProgressBar:

1. getMax() – returns the maximum value of progress bar

We can get the maximum value of the progress bar in java class. This method returns a integer value. Below is the code to get the maximum value from a Progress bar.

```
ProgressBar simpleProgressBar=(ProgressBar) findViewById(R.id.simpleProgressBar); // initiate the progress bar

int maxValue=simpleProgressBar.getMax(); // get maximum value of the progress bar
```

2. getProgress() – returns current progress value

We can get the current progress value from a progress bar in java class. This method also returns a integer value. Below is the code to get current progress value from a Progress bar.

```
ProgressBar simpleProgressBar=(ProgressBar)findViewById(R.id.simpleProgressBar); // initiate the progress bar

int progressValue=simpleProgressBar.getProgress(); // get progress value from the progress bar
```

Attributes of ProgressBar In Android:

Now let's discuss important attributes that helps us to configure a Progress bar in xml file (layout).

1. id: id is an attribute used to uniquely identify a Progress bar.

```
<ProgressBar

android:id="@+id/simpleProgressBar"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

style="@style/Widget.AppCompat.ProgressBar.Horizontal"/>
```

2. max: max is an attribute used in android to define maximum value of the progress can take. It must be an integer value like 100, 200 etc.

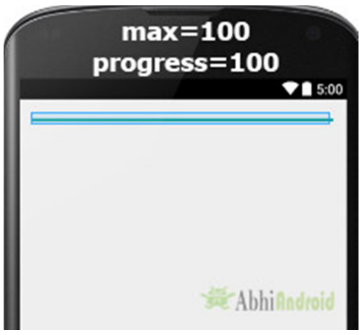
Below we set 100 maximum value for a progress bar.

```
<ProgressBar

   android:id="@+id/simpleProgressBar"

   android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:max="100" /><!--set 100 maximum value for the progress bar-->
```

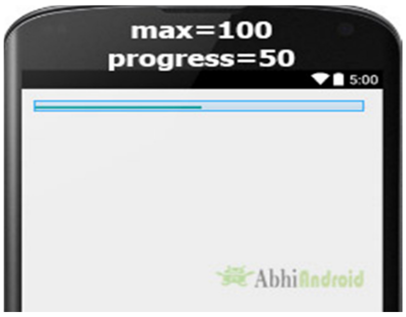
Set Max Value of ProgressBar In Java Class :

```
ProgressBar  simpleProgressBar=(ProgressBar)  findViewById(R.id.simpleProgressBar); // initiate the progress bar
simpleProgressBar.setMax(100);
```



3. progress: progress is an attribute used in android to define the default progress value between 0 and max. It must be an integer value.
Below we set the 100 max value and then set 50 default progress.

```
<ProgressBar
    android:id="@+id/simpleProgressBar"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:max="100"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:progress="50"/>
```



Setting Progress Value of ProgressBar In Java Class :

```
ProgressBar simpleProgressBar=(ProgressBar)findViewById(R.id.simpleProgressBar); // initiate the progress bar
simpleProgressBar.setMax(100); // 100 maximum value for the progress value
simpleProgressBar.setProgress(50); // 50 default progress value for the progress bar
```

4. progressDrawable: progress drawable is an attribute used in Android to set the custom drawable for the progress mode.

Below we set a custom gradient drawable for the progress mode of a progress bar. Before you try below code make sure to download a progress icon from the web and add in your drawable folder.

Step 1: Add this code in activity_main.xml or main.xml inside layout.

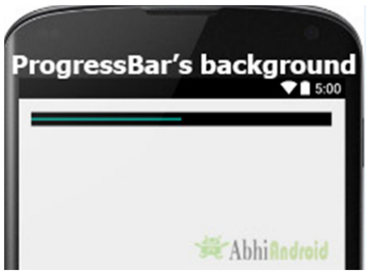
```
<ProgressBar
android:id="@+id/simpleProgressBar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:max="100"
android:progress="60"
android:layout_marginTop="100dp"
style="@style/Widget.AppCompat.ProgressBar.Horizontal"
android:progressDrawable="@drawable/custom_progress"/><!--custom progress dra
wable for progress mode-->
```

Step 2: Create a new drawable resource xml in drawable folder and name it custom_progress. Here add the below code which creates gradient effect in progressbar.

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
  <item>
    <shape>
      <gradient
        android:endColor="#fff"
        android:startColor="#1f1"
        android:useLevel="true" />
    </shape>
  </item>
</layer-list>
```

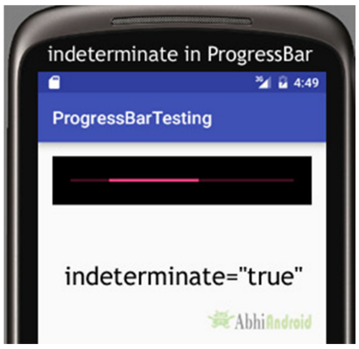
5. background: background attribute is used to set the background of a Progress bar.
We can set a color or a drawable in the background of a Progress bar.
Below we set the black color for the background of a Progress bar.

```
<ProgressBar
   android:id="@+id/simpleProgressBar"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:max="100"
   android:progress="50"
   style="@style/Widget.AppCompat.ProgressBar.Horizontal"
   android:background="#000"/>
```



6. indeterminate: indeterminate attribute is used in Android to enable the indeterminate mode. In this mode a progress bar shows a cyclic animation without an indication of progress. This mode is used in application when we don't know the amount of work to be done. In this mode the actual working will not be shown.
In below code we set the indeterminate to true.

```
<ProgressBar
   android:id="@+id/simpleProgressBar"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:max="100"
   android:progress="50"
   android:background="#000"
   android:padding="20dp" style="@style/Widget.AppCompat.ProgressBar.Horizontal"
   android:indeterminate="true"/>
```



Setting indeterminate of ProgressBar In Java class:

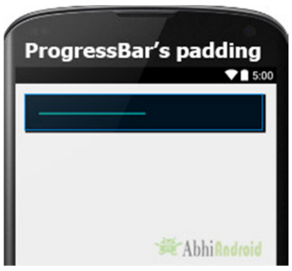
```
ProgressBar simpleProgressBar=(ProgressBar)findViewById(R.id.simpleProgressBar); // initiate the progress bar

simpleProgressBar.setBackgroundColor(Color.BLACK); // black background color for the progress bar
```

7. padding: padding attribute is used to set the padding from left, right, top or bottom of ProgressBar.

- **paddingRight:** set the padding from the right side of the Progress bar**.**
- **paddingLeft:** set the padding from the left side of the Progress bar**.**
- **paddingTop:** set the padding from the top side of the Progress bar**.**
- **paddingBottom:** set the padding from the bottom side of the Progress bar**.**
- **Padding:** set the padding from the all side's of the Progress bar**.**

Below we set the 20dp padding from all the side's of the Progress bar.

```
<ProgressBar
  android:id="@+id/simpleProgressBar"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:max="100"
  android:progress="50"
  android:background="#000"
  style="@style/Widget.AppCompat.ProgressBar.Horizontal"
  android:padding="20dp"/>
```



ProgressBar Example In Android Studio:

In the first example of ProgressBar we displayed a default spinning wheel progress bar and a start button whenever a user click on the button the progress bar is displayed. Below is the final output, download code and step by step explanation:

Step 1: Create a new project and name it ProgressBarExample.

```
Select File -> New -> New Project… then Fill the forms and click "Finish" button.
```

Step 2: Open res -> layout -> **activity_main.xml (or) main.xml** and add following code:

In this step we open an xml file and add the code for displaying a progress bar and set its visibility to invisible and one start button.

```
<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"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <ProgressBar
    android:id="@+id/simpleProgressBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:visibility="invisible"
    android:layout_centerHorizontal="true"/>

  <Button
    android:id="@+id/startButton"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:text="Start"
    android:textSize="20sp"
    android:textStyle="bold"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="100dp"
    android:padding="10dp"
    android:background="#0f0"
    android:textColor="#fff"/>
</RelativeLayout>
```

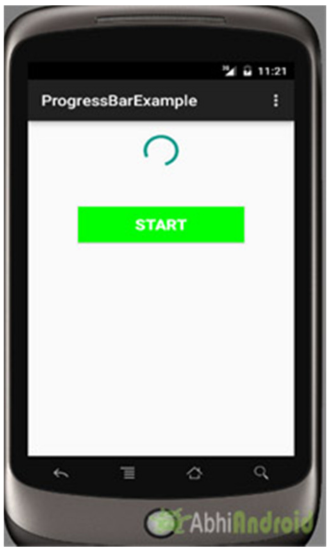
Step 3: Now Open src -> package -> **MainActivity.java**

In this step we open MainActivity where we add the code to initiate the progress bar & button and then perform click event on button which display the progress bar.

```
package example.gb.progressbarexample;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // initiate progress bar and start button
    final ProgressBar simpleProgressBar = (ProgressBar) findViewById(R.id.simplePro
gressBar);
    Button startButton = (Button) findViewById(R.id.startButton);
    // perform click event on button
    startButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        // visible the progress bar
        simpleProgressBar.setVisibility(View.VISIBLE);
      }
    });
  }
}
```

Output:

Now start the AVD in Emulator and run the App. Click on the start button and Progress Bar will be displayed on screen.



ImageView In Android

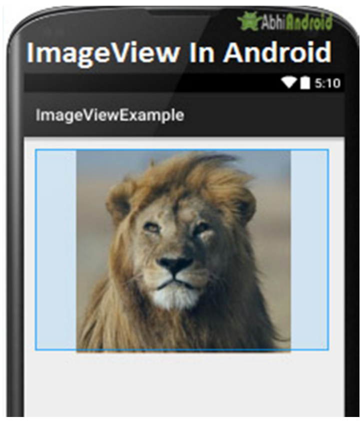
In Android, ImageView class is used to display an image file in application. Image file is easy to use but hard to master in Android, because of the various screen sizes in Android devices. An android is enriched with some of the best UI design widgets that allows us to build good looking and attractive UI based application.

Important Note: ImageView comes with different configuration options to support different scale types. Scale type options are used for scaling the bounds of an image to the bounds of the imageview. Some of them scaleTypes configuration properties are center, center_crop, fit_xy, fitStart etc. You can read our ScaleType tutorial to learn all details on it.

Below is an ImageView code in XML:

Make sure to save lion image in drawable folder

```
<ImageView
android:id="@+id/simpleImageView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:src="@drawable/lion" />
```



Attributes of ImageView:

Now let's we discuss some important attributes that helps us to configure a ImageView in your xml file.

1. id: id is an attribute used to uniquely identify a image view in android. Below is the example code in which we set the id of a image view.

```
<ImageView
android:id="@+id/simpleImageView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
```

2. src: src is an attribute used to set a source file or you can say image in your imageview to make your layout attractive.

Below is the example code in which we set the source of a imageview lion which is saved in drawable folder.

```
<ImageView
android:id="@+id/simpleImageView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:src="@drawable/lion" /><!--set the source of an image view-->
```

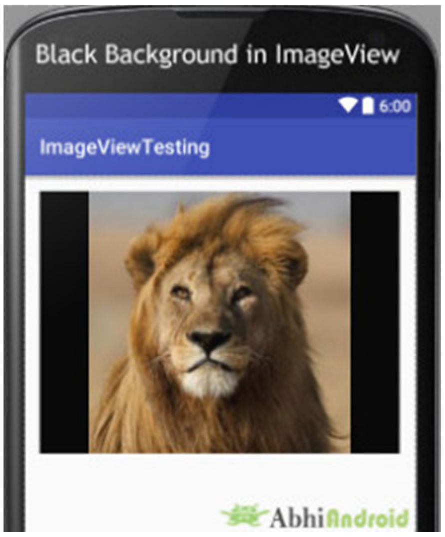
In Java:

We can also set the source image at run time programmatically in java class. For that we use setImageResource() method as shown in below example code.

```
/*Add in Oncreate() funtion after setContentView()*/
ImageView simpleImageView=(ImageView) findViewById(R.id.simpleImageView);
simpleImageView.setImageResource(R.drawable.lion);
```

3. background: background attribute is used to set the background of a ImageView. We can set a color or a drawable in the background of a ImageView. Below is the example code in which we set the black color in the background and an image in the src attribute of image view.

```
<ImageView
  android:id="@+id/simpleImageView"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:src="@drawable/lion"
  android:background="#000"/>
```



In Java:
We can also set the background at run time programmatically in java class. In below example code we set the black color in the background of a image view.

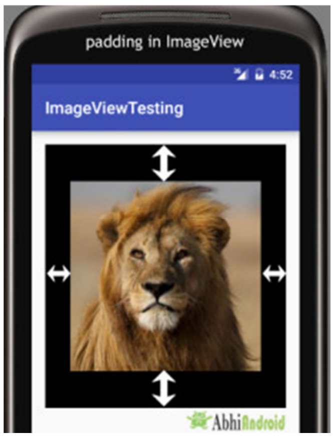
```
/*Add in Oncreate() funtion after setContentView()*/
ImageView simpleImageView=(ImageView) findViewById(R.id.simpleImageView);
simpleImageView.setBackgroundColor(Color.BLACK);//set black color in background of a image view in j
ava class
```

4. padding: padding attribute is used to set the padding from left, right, top or bottom of the Imageview.

- **paddingRight:** set the padding from the right side of the image view.
- **paddingLeft:** set the padding from the left side of the image view.
- **paddingTop:** set the padding from the top side of the image view.
- **paddingBottom:** set the padding from the bottom side of the image view.
- **padding:** set the padding from the all side's of the image view.

Below is the example code of padding attribute in which we set the 30dp padding from all the side's of a image view.

```
<ImageView
  android:id="@+id/simpleImageView"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:background="#000"
  android:src="@drawable/lion"
  android:padding="30dp"/>
```

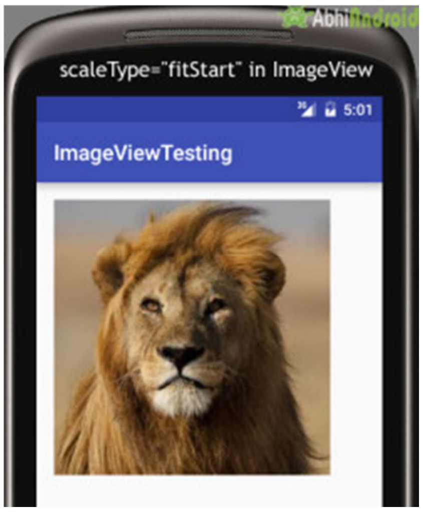


5. scaleType: scaleType is an attribute used to control how the image should be re-sized or moved to match the size of this image view. **The value for scale type attribute can be fit_xy, center_crop, fitStart etc.**
Below is the example code of scale type in which we set the scale type of image view to fit_xy.

```
<ImageView
android:id="@+id/simpleImageView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:src="@drawable/lion"
android:scaleType="fitXY"/>
```

Let's we take an another example of scale type to understand the actual working of scale type in a image view.
In below example code we set the value for scale type "fitStart" which is used to fit the image in the start of the image view as shown below:

```
<ImageView
  android:id="@+id/simpleImageView"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:src="@drawable/lion"
  android:scaleType="fitStart"/>
```



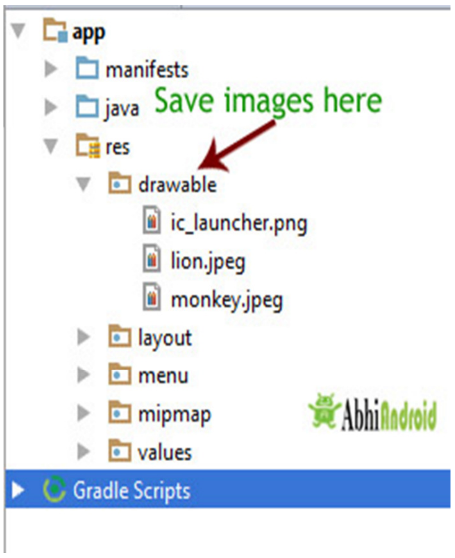
Example of ImageView:

Below is the example of imageview in which we display two animal images of Lion and Monkey. And whenever user click on an image Animal name is displayed as toast on screen. Below is the final output and code:

Step 1: Create a new project and name it ImageViewExample.
In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

```
Select File -> New -> New Project and Fill the forms and click "Finish" button.
```

Step 2: Download two images lion and monkey from the web. Now save those images in the drawable folder of your project.



Step 3: Now open res -> layout -> activity_main.xml (or) main.xml and add following code:
In this step we add the code for displaying an image view on the screen in a relative layout. **Here make sure you have already saved two images name lion and monkey in your drawable folder.**

```
<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"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">



  <ImageView
    android:id="@+id/simpleImageViewLion"
    android:layout_width="fill_parent"
    android:layout_height="200dp"
    android:scaleType="fitXY"
    android:src="@drawable/lion" />

  <ImageView
  android:id="@+id/simpleImageViewMonkey"
  android:layout_width="fill_parent"
  android:layout_height="200dp"
  android:layout_below="@+id/simpleImageViewLion"
  android:layout_marginTop="10dp"
  android:scaleType="fitXY"
  android:src="@drawable/monkey" />


</RelativeLayout>
```

Step 4: Now open app -> java -> package -> MainActivity.java and add the following code:

In this step we add the code to initiate the image view's and then perform click event on them.

```
package example.abhiandriod.imageviewexample;


import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ImageView simpleImageViewLion = (ImageView) findViewById(R.id.simpleImageViewLion);//get th
e id of first image view
    ImageView simpleImageViewMonkey = (ImageView) findViewById(R.id.simpleImageViewMonkey);/
/get the id of second image view
    simpleImageViewLion.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Toast.makeText(getApplicationContext(), "Lion", Toast.LENGTH_LONG).show();//display the text
on image click event
      }
    });
    simpleImageViewMonkey.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Toast.makeText(getApplicationContext(), "Monkey", Toast.LENGTH_LONG).show();//display the
text on image click event
      }
    });
  }
}
```

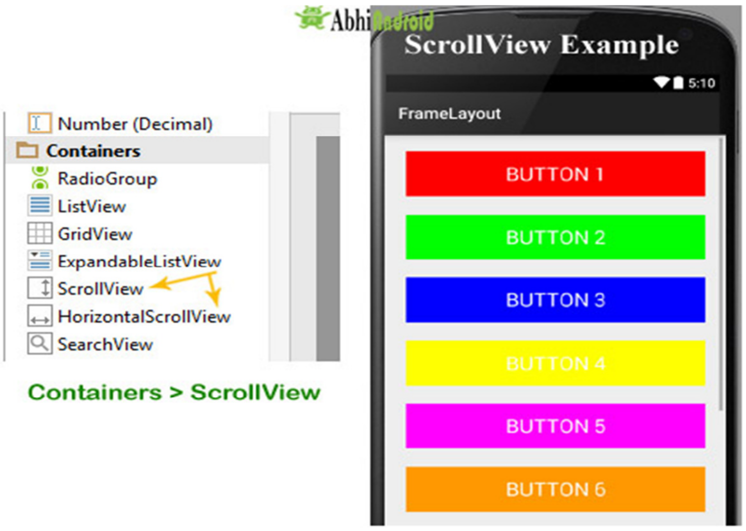
Output:

Now start AVD in Emulator and run the App. You will see the images of Lion and Monkey displayed on screen. Click on any Animal image and his name will appear on Screen. We clicked on Lion.

ScrollView And Horizontal ScrollView In Android

In android ScrollView can hold only one direct child. This means that, if you have complex layout with more views(Buttons, TextViews or any other view) then you must enclose them inside another standard layout like Table Layout, Relative Layout or Linear Layout. You can specify layout_width and layout_height to adjust width and height of screen. You can specify height and width in dp(density pixel) or px(pixel). Then after enclosing them in a standard layout, enclose the whole layout in ScrollView to make all the element or views scrollable.

ScrollView in Android Studio Design: It is present inside Containers >> ScrollView or HorizontalScrollView



Important Note 1: We never use a Scroll View with a ListView because List View is default scrollable(i.e. vertical scrollable). More importantly, doing this affects all of the important optimizations in a List View for dealing with large lists(list items). Just because it effectively forces the List View to display its entire list of items to fill up the infinite container supplied by a ScrollView so we don't use it with List View.

Important Note 2: In android default ScrollView is used to scroll the items in vertical direction and if you want to scroll the items horizontally then you need to implement horizontal ScrollView.

ScrollView Syntax:

```
<ScrollView
android:id="@+id/scrollView"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<!-- add child view's here -->
</ScrollView>
```

Horizontal ScrollView:

In android, You can scroll the elements or views in both vertical and horizontal directions. To scroll in Vertical we simply use ScrollView as we shown in the previous code of this article and to scroll in horizontal direction we need to use HorizontalScrollview.

Below is HorizontalScrollView syntax in Android is:

```
<HorizontalScrollView
android:id="@+id/horizontalscrollView"
android:layout_width="fill_parent"
android:layout_height="fill_parent">


<-- add child view's here -->


</HorizontalScrollView >
```

Attributes Of Scroll View:

ScrollView and HorizontalScrollView has same attributes, the only difference is scrollView scroll the child items in vertical direction while horizontal scroll view scroll the child items in horizontal direction.

Now let's we discuss about the attributes that helps us to configure a ScrollView and its child controls. Some of the most important attributes you will use with ScrollView include:

1. id: In android, id attribute is used to uniquely identify a ScrollView.

Below is id attribute's example code with explanation included.

```
<ScrollView
android:id="@+id/scrollView"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
/>
```

2. scrollbars: In android, scrollbars attribute is used to show the scrollbars in horizontal or vertical direction. The possible Value of scrollbars is vertical, horizontal or none. By default scrollbars is shown in vertical direction in scrollView and in horizontal direction in HorizontalScrollView.

Below is scrollbars attribute's example code in which we set the scrollbars in vertical direction.

```
< HorizontalScrollView
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:scrollbars="vertical"/><!--scrollbars in vertical direction-->
```

Example of ScrollView In Android Studio:

Example 1: In this example we will use 10 button and scroll them using ScrollView in vertical direction. Below is the code and final Output we will create:

Step 1: Create a new project in Android Studio and name it scrollviewExample.

Select File -> New -> New Project -> Android Application Project (or) Android Project. Fill the forms and click "Finish" button.

Step 2: Open res -> layout -> activity_main.xml (or) main.xml and add below code. Here we are creating a Relative Layout having 10 buttons which are nested in Linear Layout and then in ScrollView.

Important Note: Remember ScrollView can hold only one direct child. So we have to jointly put 10 buttons inside Linear Layout to make it one child. And then we put it inside ScrollView.

```
<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">
```

```
<ScrollView
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:scrollbars="vertical">

<LinearLayout
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_margin="20dp"
android:orientation="vertical">

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:background="#f00"
android:text="Button 1"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#0f0"
android:text="Button 2"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#00f"
```

```
android:text="Button 3"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#ff0"
android:text="Button 4"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#f0f"
android:text="Button 5"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#f90"
android:text="Button 6"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

```
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#f00"
android:text="Button 7"
android:textColor="#ff9"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#444"
android:text="Button 8"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#ff002211"
android:text="Button 9"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="20dp"
android:background="#0f0"
android:text="Button 10"
android:textColor="#fff"
android:textSize="20sp" />
```

```
</LinearLayout>


</ScrollView>


</RelativeLayout>
```

Step 3: Now Open src -> package -> MainActivity.java and paste the below code

```
package com.example.gourav.scrollviewExample;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;


public class MainActivity extends AppCompatActivity {


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

Step 4: Now open manifest.xml and paste the below code

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


<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".MainActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
```

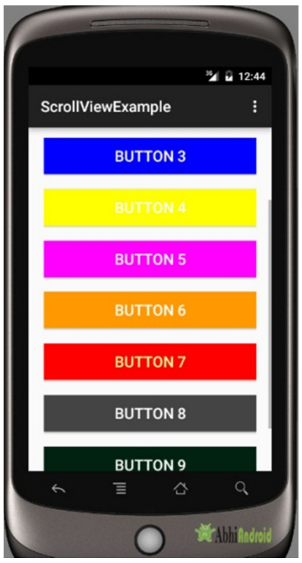
```
</intent-filter>
</activity>
</application>

</manifest>
```

Step 5: Lastly open res ->values -> strings.xml and paste the below code

```
<resources>
<string name="app_name">ScrollViewExample</string>
<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
</resources>
```

Output:

Now run the App in Emulator / AVD or in real device. You will see the 10 buttons which can be scrollable in vertical direction.



Horizontal ScrollView

Example 2: In this example we will scroll the buttons in horizontal direction. Below is the complete code and final output:
Step 1: Create a new project and name it horizontalscrollviewExample.

Select File -> New -> New Project and Fill the forms and click "Finish" button.

Step 2: Now open res -> layout -> activity_mail.xml (or) main.xml and add below code. Here we are creating same buttons in HorizontalScrollView.

```
<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">

<HorizontalScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:scrollbars="horizontal">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="horizontal">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#f00"
            android:padding="10dp"
            android:text="Button 1"
            android:textColor="#fff"
            android:textSize="20sp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#0f0"
            android:padding="10dp"
            android:text="Button 2"
            android:textColor="#fff"
            android:textSize="20sp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#00f"
            android:padding="10dp"
            android:text="Button 3"
            android:textColor="#fff"
            android:textSize="20sp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#ff0"
            android:padding="10dp"
            android:text="Button 4"
            android:textColor="#fff"
            android:textSize="20sp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#f0f"
            android:padding="10dp"
            android:text="Button 5"
            android:textColor="#fff"
            android:textSize="20sp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#f90"
            android:padding="10dp"
            android:text="Button 6"
            android:textColor="#fff"
            android:textSize="20sp" />

          <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#f00"
            android:padding="10dp"
            android:text="Button 7"
            android:textColor="#ff9"
            android:textSize="20sp" />

          <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#444"
            android:padding="10dp"
            android:text="Button 8"
            android:textColor="#fff"
            android:textSize="20sp" />

          <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#ff002211"
            android:padding="10dp"
            android:text="Button 9"
            android:textColor="#fff"
```

```
            android:textSize="20sp" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:background="#0f0"
            android:padding="10dp"
            android:text="Button 10"
            android:textColor="#fff"
            android:textSize="20sp" />

    </LinearLayout>

  </HorizontalScrollView>

</RelativeLayout>
```

Step 3: Now open app -> java -> MainActivity.java in package and paste the below code:

```
package com.example.gourav.horizontalscrollviewExample;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Step 5: Now open AndroidManifest.xml inside manifests and paste the below code:

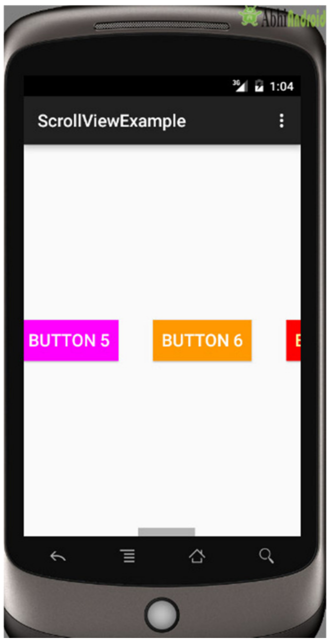
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.gourav.horizontalscrollviewExample" >

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name="com.example.gourav.horizontalscrollviewExample.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>
```

Step 6: Open res ->values -> strings.xml and paste the below code:

```
<resources>
  <string name="app_name">ScrollViewExample</string>
  <string name="hello_world">Hello world!</string>
  <string name="action_settings">Settings</string>
</resources>
```

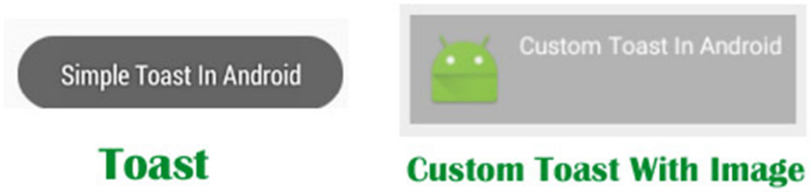
Output:

Now run the App in Emulator /AVD or real device and you will see the 10 buttons can now be scrollable in Horizontal direction.



Toast & Custom Toast In Android Studio

In Android, Toast is used to display information for a period of time. It contains a message to be displayed quickly and disappears after specified period of time. It does not block the user interaction. Toast is a subclass of Object class. In this we use two constants for setting the duration for the Toast. Toast notification in android always appears near the bottom of the screen. We can also create our custom toast by using custom layout(xml file).



Special Note: In Android, Toast is used when we required to notify user about an operation without expecting any user input. It displays a small popup for message and automatically fades out after timeout.

Important Methods Of Toast:

Let's we discuss some important methods of Toast that may be called in order to manage the Toast.

1. makeText(Context context, CharSequence text, int duration): This method is used to initiate the Toast. This method take three parameters First is for the application Context, Second is text message and last one is duration for the Toast.

Constants of Toast: Below is the constants of Toast that are used for setting the duration for the Toast.

1. LENGTH_LONG: It is used to display the Toast for a long period of time. When we set this duration the Toast will be displayed for a long duration.

2. LENGTH_SHORT: It is used to display the Toast for short period of time. When we set this duration the Toast will be displayed for short duration.

Below we show the use of makeText() method of Toast in which we set application context, a text message and duration for the Toast.

```
Toast toast = Toast.makeText(getApplicationContext(), "Simple Toast", Toast.LENGTH_LONG); // initiate
the Toast with context, message and duration for the Toast
```

2. show(): This method is used to display the Toast on the screen. This method is display the text which we create using makeText() method of Toast.

Below we Firstly initiate the Toast and then display it using show() method.

```
Toast toast = Toast.makeText(getApplicationContext(), "Simple Toast In Android", Toast.LENGTH_LONG);
// initiate the Toast with context, message and duration for the Toast

toast.show(); // display the Toast
```

3. setGravity(int,int,int): This method is used to set the gravity for the Toast. This method accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

Below we Firstly initiate the Toast, set top and left gravity and then display it using show() method.

```
Toast toast = Toast.makeText(getApplicationContext(), "Simple Toast In Android", Toast.LENGTH_LONG);
// initiate the Toast with context, message and duration for the Toast

toast.setGravity(Gravity.TOP | Gravity.LEFT, 0, 0);     // set gravity for the Toast.

toast.show(); // display the Toast
```

4. setText(CharSequence s): This method is used to set the text for the Toast. If we use makeText() method and then we want to change the text value for the Toast then we use this method.

Below we firstly create a new Toast using makeText() method and then set the text for the Toast.

```
Toast toast = Toast.makeText(getApplicationContext(), "Simple Toast In Android", Toast.LENGTH_LONG);
// initiate the Toast with context, message and duration for the Toast

toast.setGravity(Gravity.TOP | Gravity.LEFT, 0, 0);     // set gravity for the Toast.

toast.setText("Changed Toast Text"); // set the text for the Toast

toast.show(); // display the Toast
```

5. setDuration(int duration): This method is used to set the duration for the Toast. If we use makeText() method and then we want to change the duration for the Toast then we use this method.

Below we firstly create a new Toast using makeText() method and then set the duration for the Toast.

```
Toast toast = Toast.makeText(getApplicationContext(), "Simple Toast In Android", Toast.LENGTH_LONG);
// initiate the Toast with context, message and duration for the Toast

toast.setGravity(Gravity.TOP | Gravity.LEFT, 0, 0);     // set gravity for the Toast.

toast.setDuration(Toast.LENGTH_SHORT); // set the duration for the Toast.

toast.show(); // display the Toast
```

6. inflate(int, ViewGroup): This method is used to inflate the layout from the xml. In this method first parameter is the layout resource ID and the second is the root View.

Below we retrieve the Layout Inflater and then inflate the layout from the xml file.

```
// Retrieve the Layout Inflater and inflate the layout from xml

LayoutInflater inflater = getLayoutInflater();

View layout = inflater.inflate(R.layout.custom_toast_layout,

(ViewGroup) findViewById(R.id.toast_layout_root));
```

7. setView(View): This method is used to set the view for the Toast. In this method we pass the inflated layout which we inflate using inflate() method.

Below we firstly retrieve the layout inflater and then inflate the layout and finally create a new Toast and pass the inflated layout in the setView() method.

```
// Retrieve the Layout Inflater and inflate the layout from xml

LayoutInflater inflater = getLayoutInflater();

View layout = inflater.inflate(R.layout.custom_toast_layout,

(ViewGroup) findViewById(R.id.toast_layout_root));


// create a new Toast using context

Toast toast = new Toast(getApplicationContext());

toast.setDuration(Toast.LENGTH_LONG); // set the duration for the Toast

toast.setView(layout); // set the inflated layout

toast.show(); // display the custom Toast
```

Custom Toast in Android:

In Android, Sometimes simple Toast may not be satisfactory, and then we can go for customizing a Toast. For creating a custom layout, define a View layout, in XML and pass the root View object to the setView(View) method.

Steps for Implementation of Custom Toast In Android:

Step1: Firstly Retrieve the Layout Inflater with getLayoutInflater ()
(or getSystemService ()) and then inflate the layout from XML using inflate (int, ViewGroup). In inflate method first parameter is the layout resource ID and the second is the root View.

Step 2: Create a new Toast with Toast(Context) and set some properties of the Toast, such as the duration and gravity.

Step 3: Call setView(View) and pass the inflated layout in this method.

Step 4: Display the Toast on the screen using show() method of Toast.

In the below example we have shown the functioning of Toast and custom Toast both.

Toast And Custom Toast Example In Android Studio:

Below is the example of Toast and Custom Toast in Android. In this example we display two Button's one for Simple Toast and other for Custom Toast and perform click event on them. Whenever a user click on simple Toast Button a Toast with message "Simple Toast In Android" displayed on the screen and when a user clicks on custom toast Button a message "Custom Toast In Android" with a image displayed on the screen. For Creating a custom toast we firstly retrieve the layout inflater and then inflate the custom toast layout from the xml file. After that we get the reference of TextView and ImageView from the inflated layout and set the text and image in the TextView and ImageView. Finally we create a new Toast and pass the inflated layout in the setView() method and then display the Toast by using show() method of Toast.

Step 1: Create a new project and name it ToastExample

Step 2: Open res -> layout ->activity_main.xml (or) main.xml and add following code:

```
<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"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
```

```
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">
<!-- Button's for simple and custom Toast -->
<Button
android:id="@+id/simpleToast"
android:layout_width="200dp"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="150dp"
android:background="#f00"
android:text="Simple Toast"
android:textColor="#fff"
android:textSize="20sp" />

<Button
android:id="@+id/customToast"
android:layout_width="200dp"
android:layout_height="wrap_content"
android:layout_below="@+id/simpleToast"
android:layout_centerHorizontal="true"
android:layout_margin="50dp"
android:background="#0f0"
android:text="Custom Toast"
android:textColor="#fff"
android:textSize="20sp" />

</RelativeLayout>
```

Step 3: Now create a xml layouts by right clicking on res/layout -> New -> Layout Resource File and name it custom_toast_layout.xml

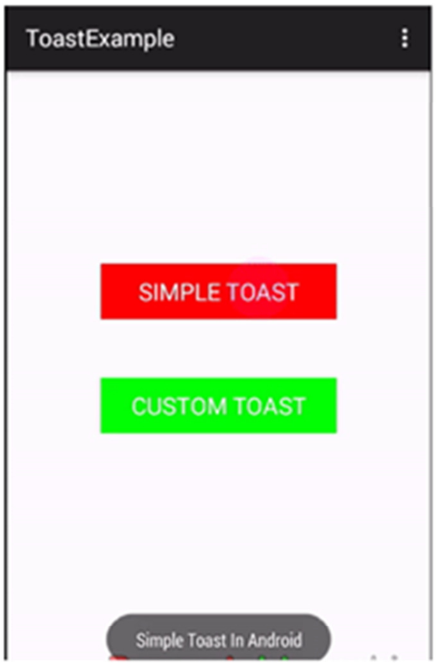
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/toast_layout_root"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#DAAA"
android:orientation="horizontal"
android:padding="8dp">
```

```
<!-- ImageVView and TextView for custom Toast -->
<ImageView
android:id="@+id/toastImageView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginRight="8dp" />

<TextView
android:id="@+id/toastTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textColor="#FFF" />
</LinearLayout>
```

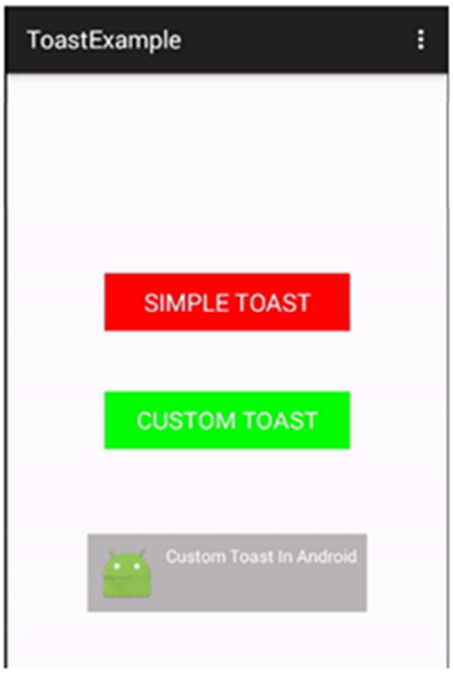
Step 4: Open src -> package -> MainActivity.java

In this step we open MainActivity and add the code for initiate the Button's and perform click event on Button's. Whenever a user click on simple Toast Button a Toast with message "Simple Toast In Android" displayed on the screen and when a user clicks on custom toast Button a message "Custom Toast In Android" with a image displayed on the screen. For Creating a custom toast we firstly retrieve the layout inflater and then inflate the custom toast layout from the xml file. After that we get the reference of TextView and ImageView from the inflated layout and set the text and image in the TextView and ImageView. Finally we create a new Toast and pass the inflated layout in the setView() method and then display the Toast by using show() method of Toast.

```
package com.abhiandroid.toastexample;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Button;
import android.view.ViewGroup;


public class MainActivity extends AppCompatActivity {
    Button simpleToast, customToast;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // get the reference of Button's
        simpleToast = (Button) findViewById(R.id.simpleToast);
        customToast = (Button) findViewById(R.id.customToast);
        // perform setOnClickListener event on simple Toast Button
        simpleToast.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // initiate a Toast with message and duration
                Toast toast = Toast.makeText(getApplicationContext(), "Simple Toast In Android", Toast.LENGTH
_LONG); // initiate the Toast with context, message and duration for the Toast
                toast.setGravity(Gravity.BOTTOM | Gravity.CENTER_HORIZONTAL, 0, 0);      // set gravity for the
Toast.
                toast.show(); // display the Toast


            }
        });
        // perform setOnClickListener event on custom Toast Button
        customToast.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Retrieve the Layout Inflater and inflate the layout from xml
                LayoutInflater inflater = getLayoutInflater();
                View layout = inflater.inflate(R.layout.custom_toast_layout,
                        (ViewGroup) findViewById(R.id.toast_layout_root));
                // get the reference of TextView and ImageVIew from inflated layout
                TextView toastTextView = (TextView) layout.findViewById(R.id.toastTextView);
                ImageView toastImageView = (ImageView) layout.findViewById(R.id.toastImageView);
                // set the text in the TextView
                toastTextView.setText("Custom Toast In Android");
                // set the Image in the ImageView
                toastImageView.setImageResource(R.drawable.ic_launcher);
            // create a new Toast using context
            Toast toast = new Toast(getApplicationContext());
            toast.setDuration(Toast.LENGTH_LONG); // set the duration for the Toast
            toast.setView(layout); // set the inflated layout
            toast.show(); // display the custom Toast
         }
      });
   }
}
```



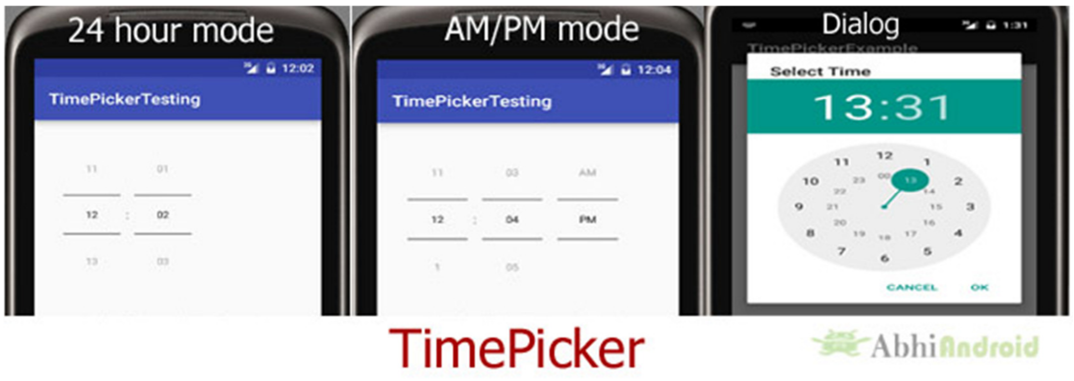
Simple Toast Alert



Custom Toast Alert

TimePicker Tutorial In Android Studio

In Android, TimePicker is a widget used for selecting the time of the day in either AM/PM mode or 24 hours mode. The displayed time consist of hours, minutes and clock format. If we need to show this view as a Dialog then we have to use a TimePickerDialog class.



TimePicker code:

```
<TimePicker
android:id="@+id/simpleTimePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:timePickerMode="spinner"/>
```

Methods of TimePicker:

Let's discuss some common methods of a time picker, which are used to configure a time picker in our application.

1. setCurrentHour(Integer currentHour):

This method is used to set the current hours in a time picker.

setHour(Integer hour): *setCurrentHour() method was deprecated in API level 23. From api level 23 we have to use* ***setHour(Integer hour)***. In this method there is only one parameter of integer type which is used to set the value for hours.

Below we set the 5 value for the current hours.

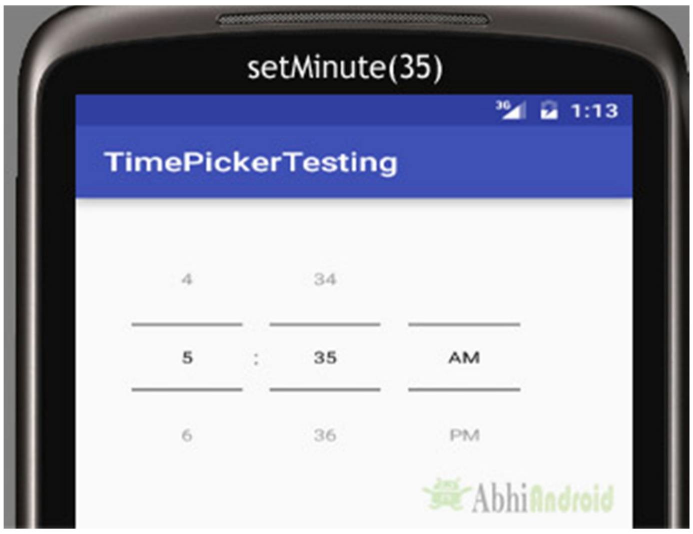
```
TimePicker simpleTimePicker=(TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time pick
er
// set the value for current hours
simpleTimePicker.setCurrentHour(5); // before api level 23
simpleTimePicker.setHour(5);
```

2. setCurrentMinute(Integer currentMinute):
This method is used to set the current minutes in a time picker.

setMinute(Integer minute): *setCurrentMinute() method was deprecated in API level 23. From api level 23 we have to use* ***setMinute(Integer minute)***. In this method there is only one parameter of integer type which set the value for minutes.

Below we set the 35 value for the current minutes.

```
TimePicker simpleTimePicker=(TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time pick
er

// set the value for current hours

simpleTimePicker.setCurrentMinute(35); // before api level 23

simpleTimePicker.setMinute(35); // from api level 23
```



3. getCurrentHour():

This method is used to get the current hours from a time picker.

getCurrentHour(): *getCurrentHour() method was deprecated in API level 23. From api level 23 you have to use* ***getHour().*** This method returns an integer value.

Below we get the value of hours from a timepicker set by user.

```
TimePicker simpleTimePicker = (TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time

pickerint hours =simpleTimePicker.getCurrentHour(); // before api level 23

int hours =simpleTimePicker.getHour(); // after api level 23
```

4. getCurrentMinute():

This method is used to get the current minutes from a time picker.

getMinute(): *getCurrentMinute() method was deprecated in API level 23. From api level 23 we have to use* ***getMinute().*** This method returns an integer value.

Below we get the value of minutes from a time picker.

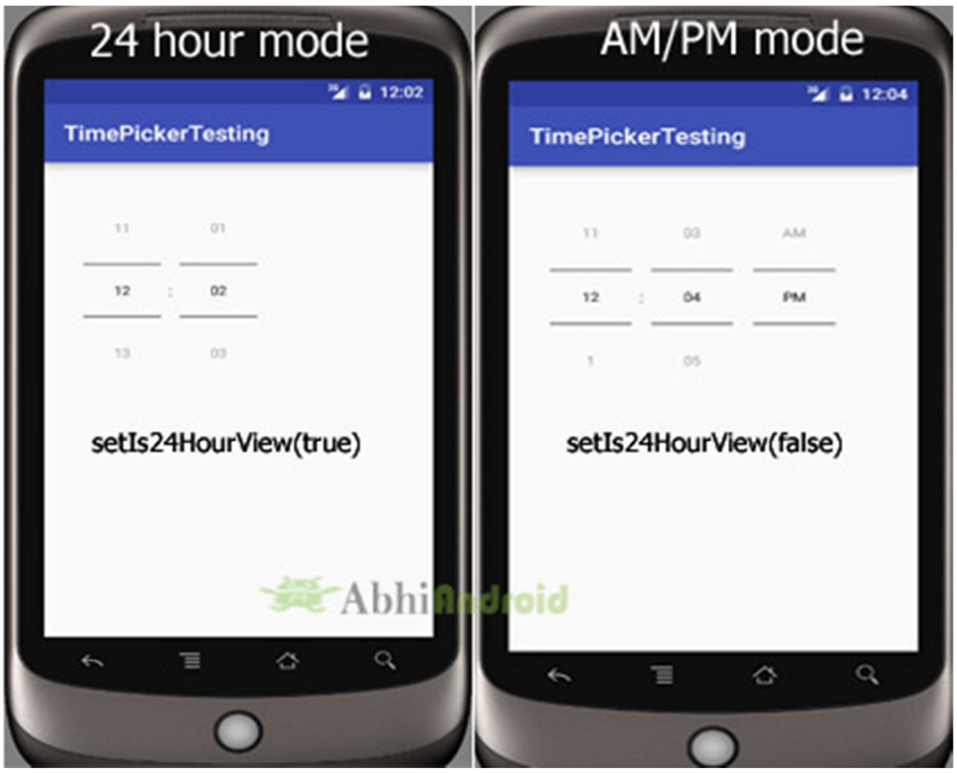
```
TimePicker simpleTimePicker = (TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time pic
ker

int minutes = simpleTimePicker.getCurrentMinute(); // before api level 23

int minutes = simpleTimePicker.getMinute(); // after api level 23
```

5. setIs24HourView(Boolean is24HourView):

This method is used to set the mode of the Time picker either 24 hour mode or AM/PM mode. In this method we set a Boolean value either true or false. True value indicate 24 hour mode and false value indicate AM/PM mode.

Below we set the current mode of the time picker.

```
TimePicker simpleTimePicker = (TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time pic
ker

simpleTimePicker.setIs24HourView(true);
```



6. is24HourView():

This method is used to check the current mode of the time picker. This method returns true if its 24 hour mode or false if AM/PM mode is set.

Below we get the current mode of the time picker:

```
TimePicker simpleTimePicker = (TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time pic
ker

Boolean mode=simpleTimePicker.is24HourView(); // check the current mode of the time picker
```

7.setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener):

This method is used to set the callback that indicates the time has been adjusted by the user. onTimeChanged(TimePicker view, int hourOfDay, int minute) is an override function of this listener in which we have three parameters first is for TimePicker, second for getting hour of the day and last is for getting the minutes after changing the time of the time picker.

Below we show the use of on time changed listener of a time picker.

```
TimePicker simpleTimePicker = (TimePicker)findViewById(R.id.simpleTimePicker); // initiate a time pick
er


simpleTimePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
@Override
public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {


}
});
```

Attributes of TimePicker:

Now let's we discuss about the attributes that helps us to configure a time picker in your xml file (layout).

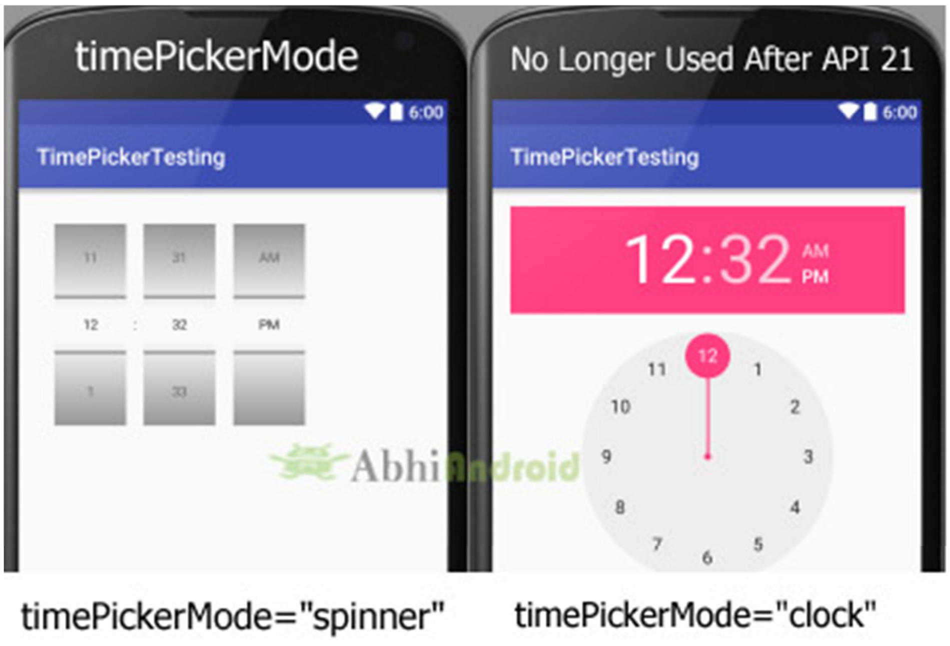
1. id: id is an attribute used to uniquely identify a time picker.

```
<TimePicker
android:id="@+id/simpleTimePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/> <!-- id of a time picker -->
```

2. timePickerMode: time picker mode is an attribute of time picker used to set the mode either spinner or clock. Default mode is clock but this mode is no longer used after api level 21, so from api level 21 you have to set the mode to spinner.

Below we set the mode to spinner.

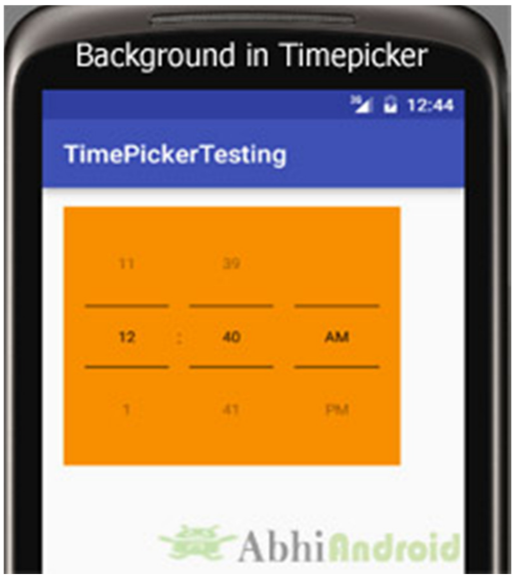
```
<TimePicker
android:id="@+id/simpleTimePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:timePickerMode="spinner" />
```



3. background: background attribute is used to set the background of a time picker. We can set a color or a drawable image in the background. We can also set the background color programmatically means in java class.

Below we set the orange color for the background of a time picker.

```
<TimePicker
   android:id="@+id/simpleTimePicker"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:timePickerMode="spinner"
   android:background="#F88F00" />
```



Setting TimePicker Background In Java Class:

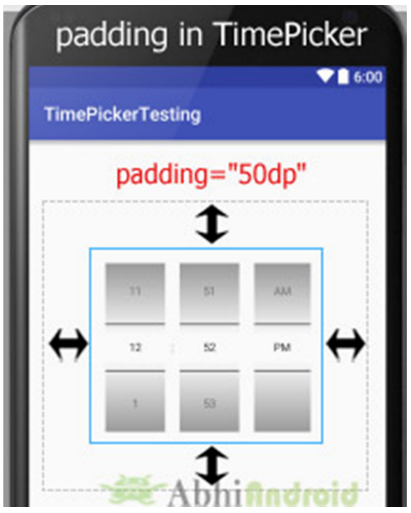
```
TimePicker simpleTimePicker=(TimePicker)findViewById(R.id.simpleTimePicker); //initiate a time pick
er

simpleTimePicker.setBackgroundColor(Color.YELLOW); //Yellow background color for the background o
f a time picker
```

4. padding: padding attribute is used to set the padding from left, right, top or bottom for a time picker.

- **paddingRight:** set the padding from the right side of the time picker**.**
- **paddingLeft:** set the padding from the left side of the time picker**.**
- **paddingTop:** set the padding from the top side of the time picker**.**
- **paddingBottom:** set the padding from the bottom side of the time picker**.**
- **Padding:** set the padding from the all side's of the time picker**.**

Below example we set the 20dp padding from all the side's of the time picker.

```
<TimePicker

android:id="@+id/simpleTimePicker"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:timePickerMode="spinner"

android:layout_centerHorizontal="true"

android:layout_marginTop="50dp"

android:padding="20dp"/>
```



Example of TimePicker in Android Studio:

Example 1: In the below example of time picker we will show you the use of time picker in our application. For that we display simple time picker and a textview in our xml file and perform setOnTimeChangedListener() event, so that whenever a user adjust the time the current displayed time of time picker is displayed by using a Toast and also displayed in the textview.

Step 1: Create a new project and name it **TimePickerExample**

Step 2: Open res -> layout -> **activity_main.xml (or) main.xml** and add following code:

In this step we open an xml file and add the code for displaying a time picker with spinner mode and textview for displaying time of time picker.

```
<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"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <TimePicker
    android:id="@+id/simpleTimePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:background="#090"
    android:padding="20dp"
    android:timePickerMode="spinner" />

  <TextView
    android:id="@+id/time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="Time Is ::"
    android:textColor="#090"
    android:textSize="20sp"
    android:textStyle="bold" />
```

```
</RelativeLayout>
```

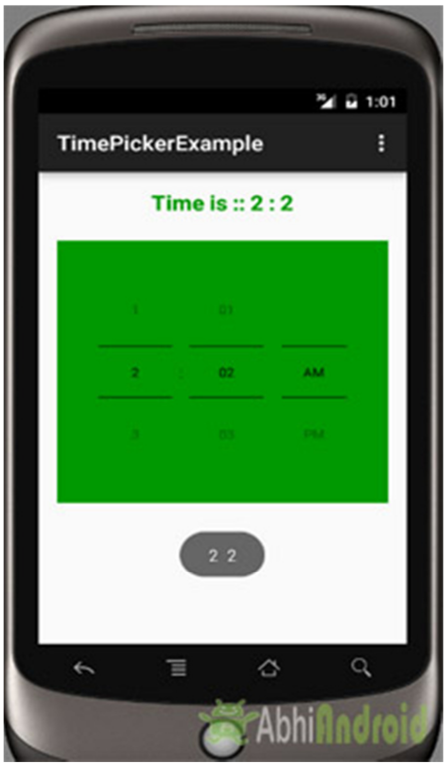
Step 3: Open app -> package -> **MainActivity.java**

In this step we open MainActivity where we add the code to initiate the time picker and a text view to display time of time picker and then we perform setOnTimeChangedListener() event so whenever a user adjust the time the current displayed time of time picker is displayed by using a Toast and also displayed in the textview.

```
package example.gb.timepickerexample;


import android.app.TimePickerDialog;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import org.w3c.dom.Text;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    TextView time;
    TimePicker simpleTimePicker;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //  initiate the view's
        time = (TextView) findViewById(R.id.time);
        simpleTimePicker = (TimePicker) findViewById(R.id.simpleTimePicker);
        simpleTimePicker.setIs24HourView(false); // used to display AM/PM mode
        // perform set on time changed listener event
        simpleTimePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
            @Override
            public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
                // display a toast with changed values of time picker
                Toast.makeText(getApplicationContext(), hourOfDay + "  " + minute, Toast.LENGTH_SHORT).sho
w();
                time.setText("Time is :: " + hourOfDay + " : " + minute); // set the current time in text view
            }
        });
    }



}
```

Output:

Now run the App in AVD and you will see TimePicker on the screen. Change the time and it will be displayed as Toast and also in TextView.



Example 2: In the second example of TimePicker we will show the use of time picker dialog in our application. To do that we will display <u>edittext</u> in our xml file and perform a click listener event on it, so whenever a user click on it time picker

dialog will appear and from there user can adjust the time and after selecting the time it will be displayed in the edittext.

Step 1: Create a new project and name it **TimePickerExample**

Step 2: Open res -> layout -> **activity_main.xml (or) main.xml** and add following code:

In this step we open an xml file and add the code for displaying a edittext to display time of time picker.

```
<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"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity">

   <EditText
      android:id="@+id/time"
      android:layout_width="200dp"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:hint="Select Time..."
      android:textColor="#090"
      android:textColorHint="#090"
      android:background="#d4d4d4"
      android:padding="15dp"
      android:textSize="20sp"
      android:textStyle="bold" />

</RelativeLayout>
```

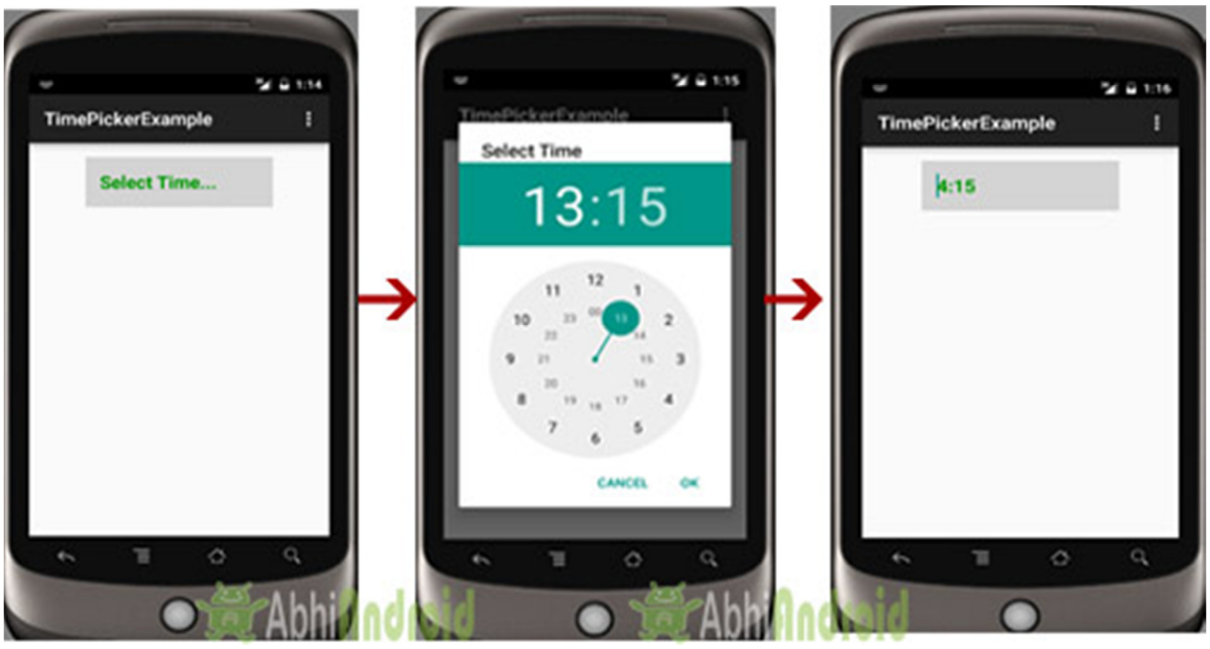
Step 3: Open src -> package -> **MainAcivity.java**

In this step we open MainActivity where we add the code to initiate the edittext to display time of time picker and perform click event on edittext, so whenever a user clicks on edittext a time picker dialog will appear from there user can set the time. And finally then the time will be displayed in the edit text.

```
package example.gb.timepickerexample;


import android.app.TimePickerDialog;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

import org.w3c.dom.Text;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    EditText time;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //  initiate the edit text
        time = (EditText) findViewById(R.id.time);
        // perform click event listener on edit text
        time.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Calendar mcurrentTime = Calendar.getInstance();
                int hour = mcurrentTime.get(Calendar.HOUR_OF_DAY);
                int minute = mcurrentTime.get(Calendar.MINUTE);
                TimePickerDialog mTimePicker;
                mTimePicker = new TimePickerDialog(MainActivity.this, new TimePickerDialog.OnTimeSetListe
ner() {
                    @Override
                    public void onTimeSet(TimePicker timePicker, int selectedHour, int selectedMinute) {
                        time.setText(selectedHour + ":" + selectedMinute);
                    }
                }, hour, minute, true);//Yes 24 hour time
                mTimePicker.setTitle("Select Time");
                mTimePicker.show();

            }
        });
    }


}
```

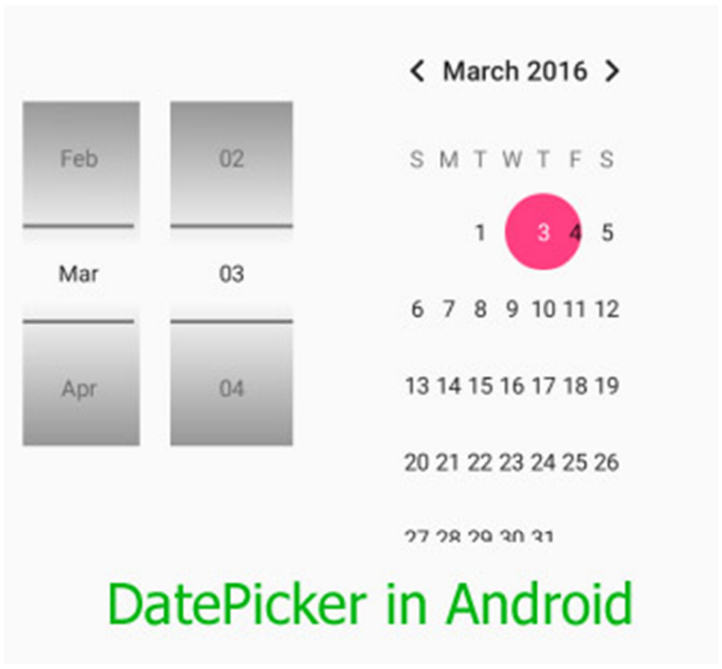
Output:

Now run the App in AVD and you will see edittext asking user to select time. When user click on it, timepicker dialog will open from there user can select time. And then this time will be displayed in EditText.



DatePicker In Android Studio

In Android, DatePicker is a widget used to select a date. It allows to select date by day, month and year in your custom UI (user interface). If we need to show this view as a dialog then we have to use a DatePickerDialog class. For selecting time Android also provides timepicker to select time.



DatePicker code in XML:

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="spinner"/>
```

Methods of DatePicker

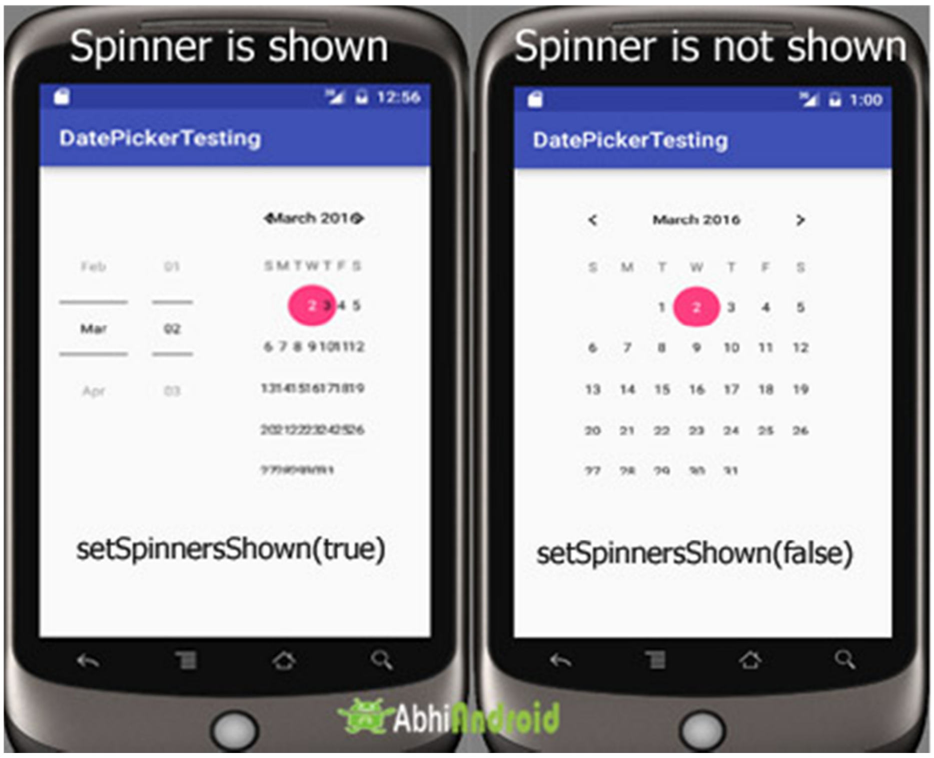
Let's discuss some common methods of a datepicker which are used to configure a DatePicker in our application.

1. setSpinnersShown(boolean shown):

This method is used to set whether the spinner of the date picker in shown or not. In this method you have to set a Boolean value either true or false. True indicates spinner is shown, false value indicates spinner is not shown. Default value for this function is true.

Below we show the use of setSpinnerShown() function by setting false value.

```
DatePicker simpleDatePicker = (DatePicker)findViewById(R.id.simpleDatePicker);
simpleDatePicker.setSpinnersShown(false);
```



2. getDayOfMonth():

This method is used to get the selected day of the month from a date picker. This method returns an integer value.

Below we get the selected day of the month from a date picker.

```
/*Add in Oncreate() funtion after setContentView()*/

DatePicker simpleDatePicker = (DatePicker) findViewById(R.id.simpleDatePicker); // initiate a date pick
er

int day = simpleDatePicker.getDayOfMonth(); // get the selected day of the month
```

3. getMonth():

This method is used to get the selected month from a date picker. This method returns an integer value.

Below we get the selected month from a date picker.

```
DatePicker simpleDatePicker = (DatePicker)findViewById(R.id.simpleDatePicker); // initiate a date picke
r

int month = simpleDatePicker.getMonth(); // get the selected month
```

4. getYear():

This method is used to get the selected year from a date picker. This method returns an integer value.

Below code is used to get the selected year from a date picker.

```
DatePicker simpleDatePicker = (DatePicker)findViewById(R.id.simpleDatePicker); // initiate a date picke
r
```

```
int year = simpleDatePicker.getYear(); // get the selected year
```

5. getFirstDayOfWeek():

This method is used to get the first day of the week. This method returns an integer value.

Below code is used to get the first day of the week.

```
DatePicker simpleDatePicker = (DatePicker)findViewById(R.id.simpleDatePicker); // initiate a date picke
r


int firstDay=simpleDatePicker.getFirstDayOfWeek(); // get the first day of the week
```

Attributes of DatePicker

Now let’s we discuss some important attributes that helps us to configure a DatePicker in your XML file (layout).

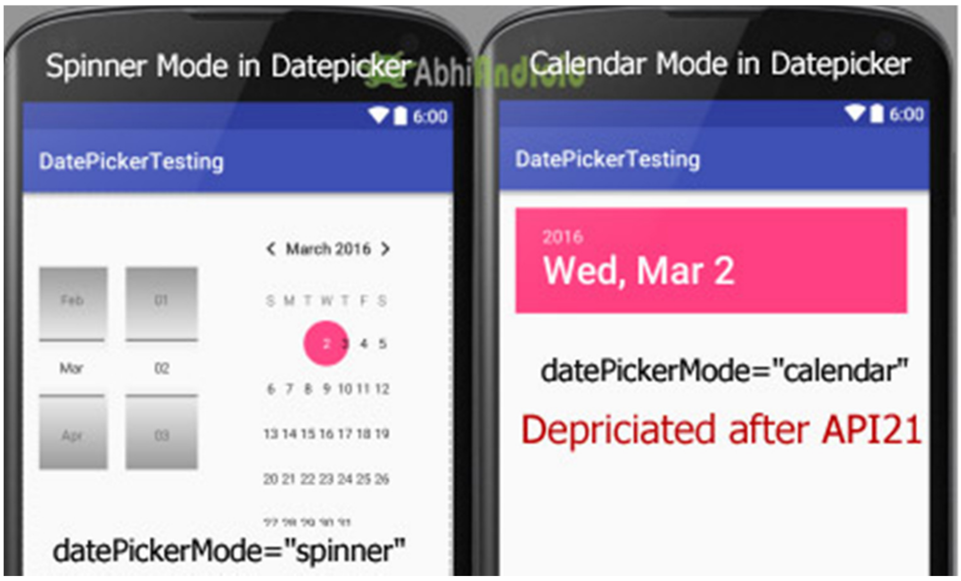
1. id: id is an attribute used to uniquely identify a date picker.

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
```

2. datePickerMode: This attribute is used to set the Date Picker in mode either spinner or calendar. Default mode is calendar but this mode is not used after api level 21, so from api level 21 you have to set the mode to spinner.

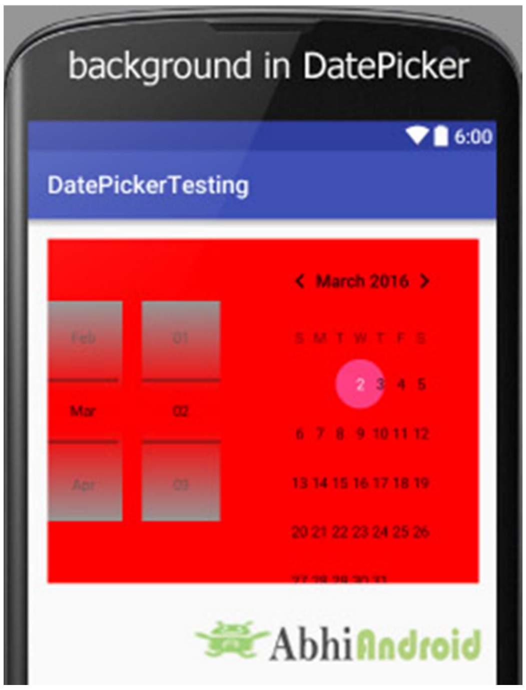
Below is an example code in which we set the mode to spinner for a date picker.

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="spinner" />
```



3. background: background attribute is used to set the background of a date picker. We can set a color or a drawable image in the background.
Below we set the red color for the background of a date picker.

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="spinner"
android:background="#f00"/>
```



Setting background of DatePicker In Java Class:

```
DatePicker simpleDatePicker=(DatePicker)findViewById(R.id.simpleDatePicker); // initiate a date picker
```

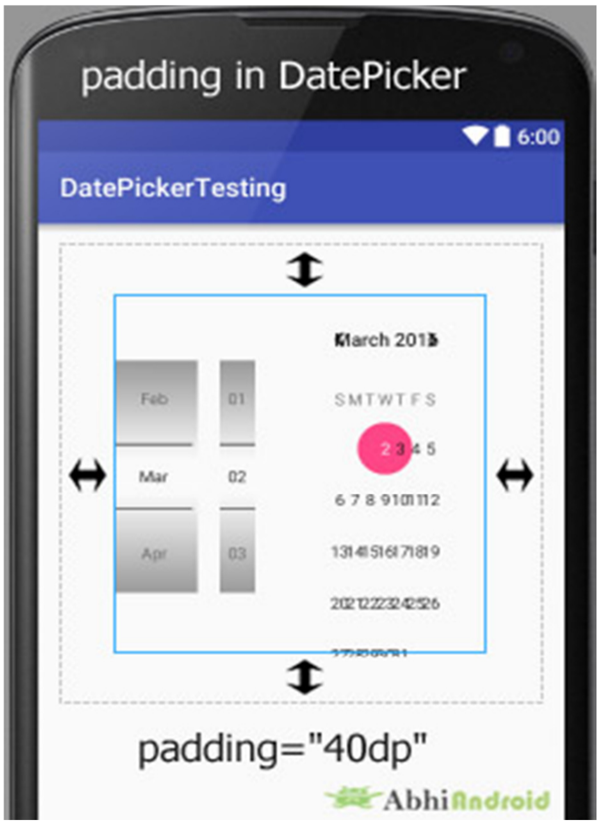
```
simpleDatePicker.setBackgroundColor(Color.RED); //  red color for the background of a date picker
```

4. padding: padding attribute is used to set the padding from left, right, top or bottom for a date picker.

- **paddingRight:** set the padding from the right side of the date picker.
- **paddingLeft:** set the padding from the left side of the date picker.
- **paddingTop:** set the padding from the top side of the date picker.
- **paddingBottom:** set the padding from the bottom side of the date picker.
- **Padding:** set the padding from the all side's of the date picker.

Below code of padding attribute set the 40dp padding from all the side's of the date picker.

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="spinner"
android:padding="40dp"/>
```



DatePicker Example in Android Studio:

Example 1: In the first example of DatePicker we show simple date picker and a Button in our xml file and perform click event on button. So whenever a user clicks on a button the day of the month, month and year will be displayed by using a Toast. Below is the final output, download code and step by step explanation:

Step 1: Create a new project and name it **DatePickerExample**

Step 2: Open res -> layout -> **activity_main.xml (or) main.xml** and add following code:

In this step we open an xml file and add the code for displaying a datepicker with spinner mode and a button for getting the date from the datepicker.

```
<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"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <DatePicker
    android:id="@+id/simpleDatePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#150"
    android:datePickerMode="spinner" />

  <Button
    android:id="@+id/submitButton"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_below="@+id/simpleDatePicker"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:background="#150"
    android:text="SUBMIT"
    android:textColor="#fff"
    android:textSize="20sp"
    android:textStyle="bold" />
</RelativeLayout>
```

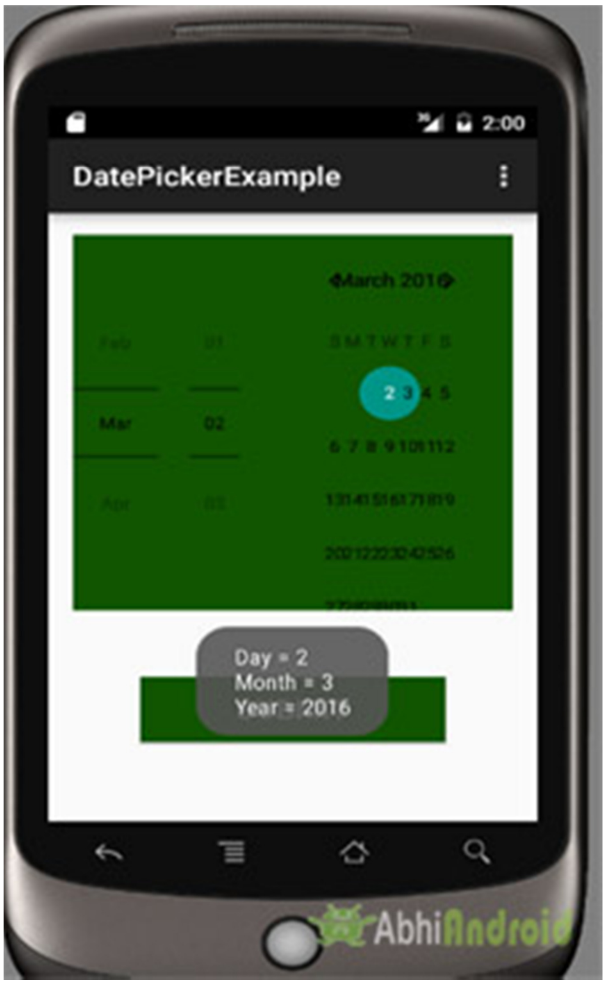
Step 3: Open app -> package -> **MainActivity.java**

In this step we open MainActivity where we add the code to initiate the datepicker & a button and then perform onClickListener() event on button so whenever a user clicks on the button the day of the month, month and year will be displayed by using a Toast.

```
package example.abhiandroid.datepickerexample;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.DatePicker;
import android.widget.Button;
import android.widget.Toast;


public class MainActivity extends AppCompatActivity {


  DatePicker simpleDatePicker;
  Button submit;


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // initiate the date picker and a button
    simpleDatePicker = (DatePicker) findViewById(R.id.simpleDatePicker);
    submit = (Button) findViewById(R.id.submitButton);
    // perform click event on submit button
    submit.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
            // get the values for day of month , month and year from a date picker
            String day = "Day = " + simpleDatePicker.getDayOfMonth();
            String month = "Month = " + (simpleDatePicker.getMonth() + 1);
            String year = "Year = " + simpleDatePicker.getYear();
            // display the values by using a toast
            Toast.makeText(getApplicationContext(), day + "\n" + month + "\n" + year, Toast.LENGTH_LONG
).show();
        }
      });


   }




}
```

Output:

Now run the App in AVD and you will see datepicker will appear on the screen. Choose the date, month & year and click submit. The date you selected will appear on Screen.



```
Day = 2
Month = 3
Year = 2016
```

Example 2: In the second example we show the use of date picker dialog in our application, for that we display edittext in our xml file and perform a onClickListener() event on it. So whenever a user click on it date picker dialog is appeared and from there user can adjust the date and after selecting the date it will be displayed in the edit text. Below is the final output, download code and step by step tutorial:

Step 1: Create a new project and name it **DatePickerExample**

Step 2: Open res – > layout – > **activity_main.xml (or) main.xml** and add following code:

In this step we open xml file and add the code for displaying edittext which will be used to display the date of datepicker.

```
<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"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">


  <EditText
    android:id="@+id/date"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:background="#d4d4d4"
    android:hint="Select Date..."
    android:padding="15dp"
    android:textColor="#897"
    android:textColorHint="#090"
    android:textSize="20sp"
    android:textStyle="bold" />


</RelativeLayout>
```

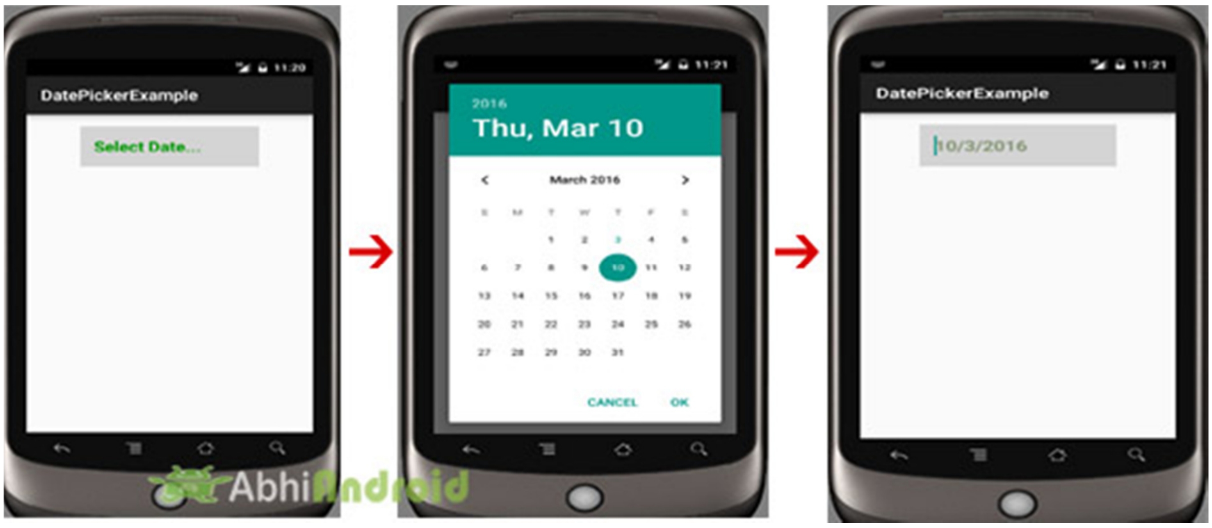
Step 3: Open src -> package -> **MainActvity.java**

In this step we open MainActivity where we add the code to initiate the edittext to display date(day of month, month and year) from a date picker and perform click event on edit text so whenever a user clicks on edit text a date picker dialog is appeared from there user can set the date by choosing day of month , month and year , after setting the date will be displayed in the edit text.

```
package example.abhiandroid.datepickerexample;


import android.app.DatePickerDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;


import java.util.Calendar;


public class MainActivity extends AppCompatActivity {


    EditText date;
    DatePickerDialog datePickerDialog;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // initiate the date picker and a button
        date = (EditText) findViewById(R.id.date);
        // perform click event on edit text
        date.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // calender class's instance and get current date , month and year from calender
                final Calendar c = Calendar.getInstance();
                int mYear = c.get(Calendar.YEAR); // current year
                int mMonth = c.get(Calendar.MONTH); // current month
                int mDay = c.get(Calendar.DAY_OF_MONTH); // current day
            // date picker dialog
            datePickerDialog = new DatePickerDialog(MainActivity.this,
                  new DatePickerDialog.OnDateSetListener() {


                     @Override
                     public void onDateSet(DatePicker view, int year,  int monthOfYear, int dayOfMonth) {
            date.setText(dayOfMonth + "/"
                              + (monthOfYear + 1) + "/" + year);
                     }
                  }, mYear, mMonth, mDay);
            datePickerDialog.show();
         }
      });
   }
}
```

Output:

Now run the App in Emulator and fill the date in EditText option.


DatePickerExample
Select Date...
2016
Thu, Mar 10
March 2016
CANCEL
OK
DatePickerExample
10/3/2016
AbhiAndroid