Android UI Widgets

There are given a lot of **android widgets** with simplified examples such as Button, EditText, AutoCompleteTextView, ToggleButton, DatePicker, TimePicker, ProgressBar etc.

Android widgets are easy to learn. The widely used android widgets with examples are given below:

Android Button Let's learn how to perform event handling on button click.
Android Toast Displays information for the short duration of time.
Custom Toast We are able to customize the toast, such as we can display image on the toast
ToggleButton It has two states ON/OFF.
CheckBox Let's see the application of simple food ordering.
AlertDialog AlertDialog displays a alert dialog containing the message with OK and Cancel buttons.
Spinner Spinner displays the multiple options, but only one can be selected at a time.
AutoCompleteTextView Let's see the simple example of AutoCompleteTextView.
RatingBar RatingBar displays the rating bar.
DatePicker Datepicker displays the datepicker dialog that can be used to pick the date.
TimePicker TimePicker displays the timepicker dialog that can be used to pick the time.
ProgressBar ProgressBar displays progress task.

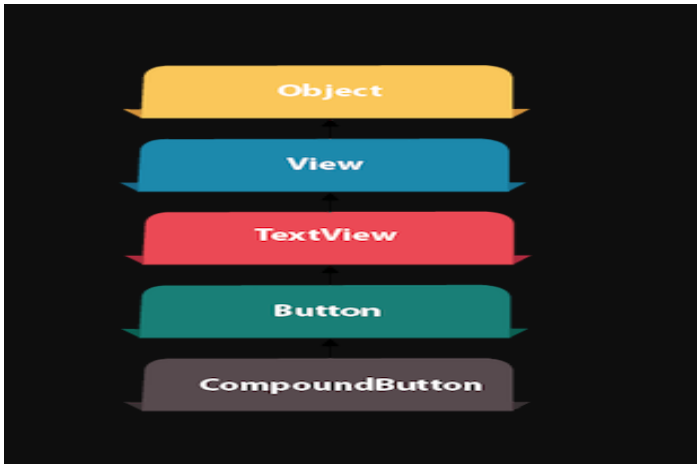
Android Button Example

Android Button represents a push-button. The android.widget.Button is subclass of TextView class and CompoundButton is the subclass of Button class.

There are different types of buttons in android such as RadioButton, ToggleButton, CompoundButton etc.

Android Button Example with Listener

Here, we are going to create two textfields and one button for sum of two numbers. If user clicks button, sum of two input values is displayed on the Toast.



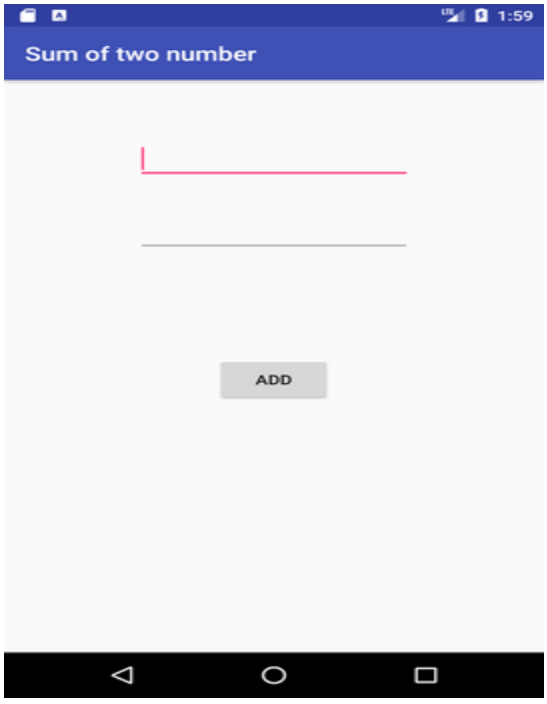
We can perform action on button using different types such as calling listener on button or adding onClick property of button in activity's xml file.

```
button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
          //code
        }
});
```

```
<Button
    android:onClick="methodName"
/>
```

Drag the component or write the code for UI in activity_main.xml

First of all, drag 2 textfields from the Text Fields palette and one button from the Form Widgets palette as shown in the following figure.



The generated code for the ui components will be like this:

File: activity_main.xml

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

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="61dp"
        android:ems="10"
        android:inputType="number"
        tools:layout_editor_absoluteX="84dp"
        tools:layout_editor_absoluteY="53dp" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:inputType="number"
        tools:layout_editor_absoluteX="84dp"
        tools:layout_editor_absoluteY="127dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="109dp"
        android:text="ADD"
        tools:layout_editor_absoluteX="148dp"
        tools:layout_editor_absoluteY="266dp" />
</RelativeLayout>
```

Activity class

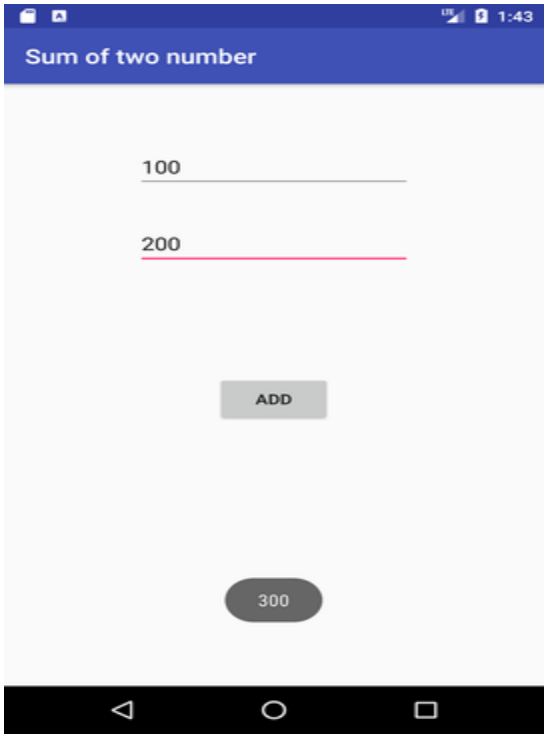
Now write the code to display the sum of two numbers.

File: MainActivity.java

```
1. package example.javatpoint.com.sumoftwonumber;
2.
3. import android.support.v7.app.AppCompatActivity;
4. import android.os.Bundle;
5. import android.view.View;
6. import android.widget.Button;
7. import android.widget.EditText;
8. import android.widget.Toast;
9.
10. public class MainActivity extends AppCompatActivity {
11.     private EditText edittext1, edittext2;
12.     private Button buttonSum;
13.
14.     @Override
15.     protected void onCreate(Bundle savedInstanceState) {
16.         super.onCreate(savedInstanceState);
17.         setContentView(R.layout.activity_main);
18.
19.         addListenerOnButton();
20.     }
21.
22.     public void addListenerOnButton() {
23.         edittext1 = (EditText) findViewById(R.id.editText1);
24.         edittext2 = (EditText) findViewById(R.id.editText2);
25.         buttonSum = (Button) findViewById(R.id.button);
26.
27.         buttonSum.setOnClickListener(new View.OnClickListener() {
28.             @Override
29.             public void onClick(View view) {
30.                 String value1=edittext1.getText().toString();
31.                 String value2=edittext2.getText().toString();
32.                 int a=Integer.parseInt(value1);
33.                 int b=Integer.parseInt(value2);
34.                 int sum=a+b;
35.                 Toast.makeText(getApplicationContext(),String.valueOf(sum), Toast.LENGTH
    _LONG).show();
36.             }
37.         });
38.     }
```

```
39. }
```

Output:

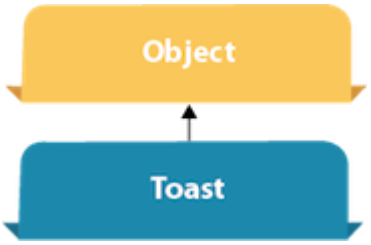


Android Toast Example

Andorid Toast can be used to display information for the short period of time. A toast contains message to be displayed quickly and disappears after sometime.

The android.widget.Toast class is the subclass of java.lang.Object class.

You can also create custom toast as well for example toast displaying image. You can visit next page to see the code for custom toast.



Toast class

Toast class is used to show notification for a particular interval of time. After sometime it disappears. It doesn't block the user interaction.

Constants of Toast class

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

There are only 2 constants of Toast class which are given below.

Methods of Toast class

The widely used methods of Toast class are given below.

Method	Description
public static Toast makeText(Context context, CharSequence text, int duration)	makes the toast containing text and duration.
public void show()	displays toast.
public void setMargin (float horizontalMargin, float verticalMargin)	changes the horizontal and vertical margin difference.

Android Toast Example

```
1. Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT).show
();
```

Another code:

```
1. Toast toast=Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_S
HORT);
2. toast.setMargin(50,50);
3. toast.show();
```

Here, getApplicationContext() method returns the instance of Context.

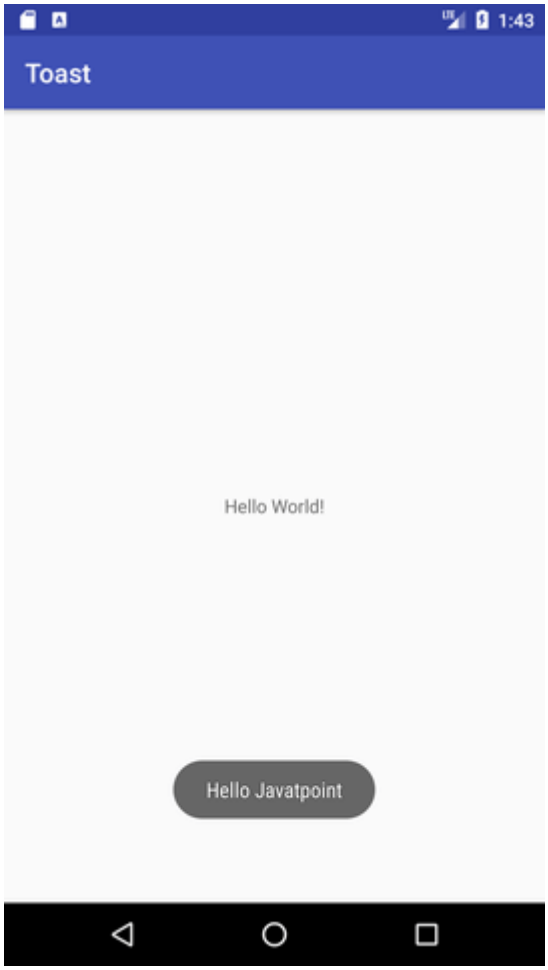
Full code of activity class displaying Toast

Let's see the code to display the toast.

File: MainActivity.java

```
1.  package example.javatpoint.com.toast;
2.
3.  import android.support.v7.app.AppCompatActivity;
4.  import android.os.Bundle;
5.  import android.widget.Toast;
6.
7.  public class MainActivity extends AppCompatActivity {
8.
9.      @Override
10.     protected void onCreate(Bundle savedInstanceState) {
11.         super.onCreate(savedInstanceState);
12.         setContentView(R.layout.activity_main);
13.
14.         //Displaying Toast with Hello Javatpoint message
15.         Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT).
    show();
16.     }
17. }
```

Output:



Android Custom Toast Example

You are able to create custom toast in android. So, you can display some images like congratulations or loss on the toast. It means you are able to customize the toast now.

---

activity_main.xml

Drag the component that you want to display on the main activity.

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.customtoast.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

---

customtoast.xml

Create another xml file inside the layout directory. Here we are having ImageView and TextView in this xml file.

File: customtoast.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/custom_toast_layout"
    android:orientation="vertical"
    android:background="#F14E23"
    >

    <ImageView
        android:id="@+id/custom_toast_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="Hello world"
        android:src="@drawable/jtp_logo"/>

    <TextView
        android:id="@+id/custom_toast_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="To"
        android:text="JavaTpoint custom Toast" />
```

```
</LinearLayout>
```

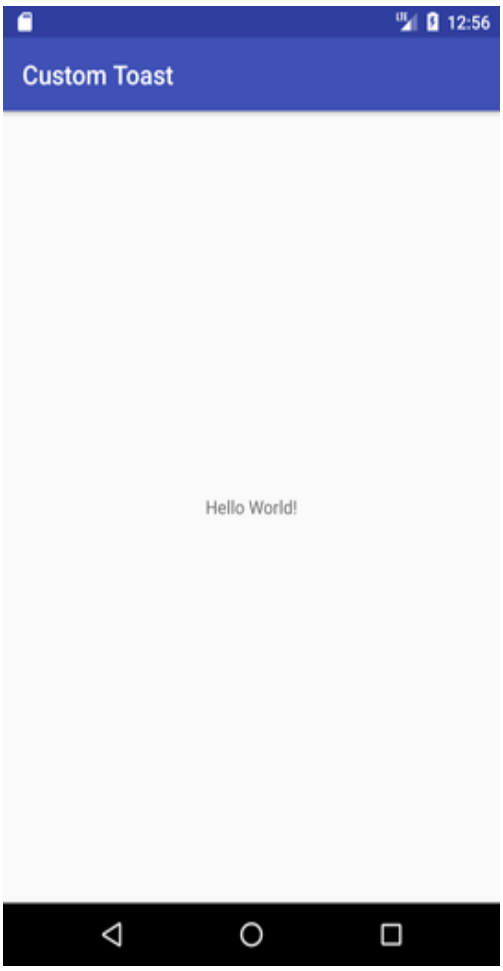
Activity class

Now write the code to display the custom toast.

File: MainActivity.java

```
package example.javatpoint.com.customtoast;

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

public class MainActivity extends AppCompatActivity {

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

        //Creating the LayoutInflater instance
        LayoutInflater li = getLayoutInflater();
        //Getting the View object as defined in the customtoast.xml file
        View layout = li.inflate(R.layout.customtoast,(ViewGroup) findViewById(R.id.custo
m_toast_layout));

        //Creating the Toast object
        Toast toast = new Toast(getApplicationContext());
        toast.setDuration(Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
        toast.setView(layout);//setting the view of custom toast layout
        toast.show();
    }
}
```

Output:



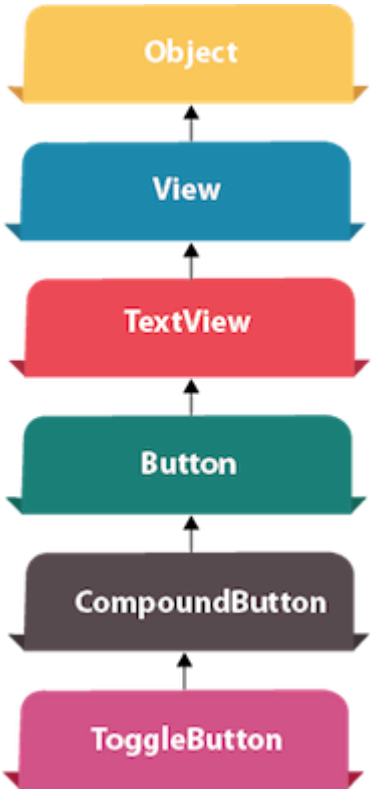
Android ToggleButton Example

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

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

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

Android ToggleButton and Switch both are the subclasses of CompoundButton class.



Android ToggleButton class

ToggleButton class provides the facility of creating the toggle button.

XML Attributes of ToggleButton class

The 3 XML attributes of ToggleButton class.

XML Attribute	Description
android:disabledAlpha	The alpha to apply to the indicator when disabled.
android:textOff	The text for the button when it is not checked.
android:textOn	The text for the button when it is checked.

Methods of ToggleButton class

The widely used methods of ToggleButton class are given below.

Method	Description
CharSequence getTextOff()	Returns the text when button is not in the checked state.
CharSequence getTextOn()	Returns the text for when button is in the checked state.
void setChecked(boolean checked)	Changes the checked state of this button.

Android ToggleButton Example

activity_main.xml

Drag two toggle button and one button for the layout. Now the activity_main.xml file will look like this:

File: activity_main.xml

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

    <ToggleButton
        android:id="@+id/toggleButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="80dp"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintEnd_toStartOf="@+id/toggleButton2"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent" />

    <ToggleButton
        android:id="@+id/toggleButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="60dp"
        android:layout_marginTop="80dp"
        android:text="ToggleButton"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="144dp"
        android:layout_marginLeft="148dp"
        android:text="Submit"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Activity class

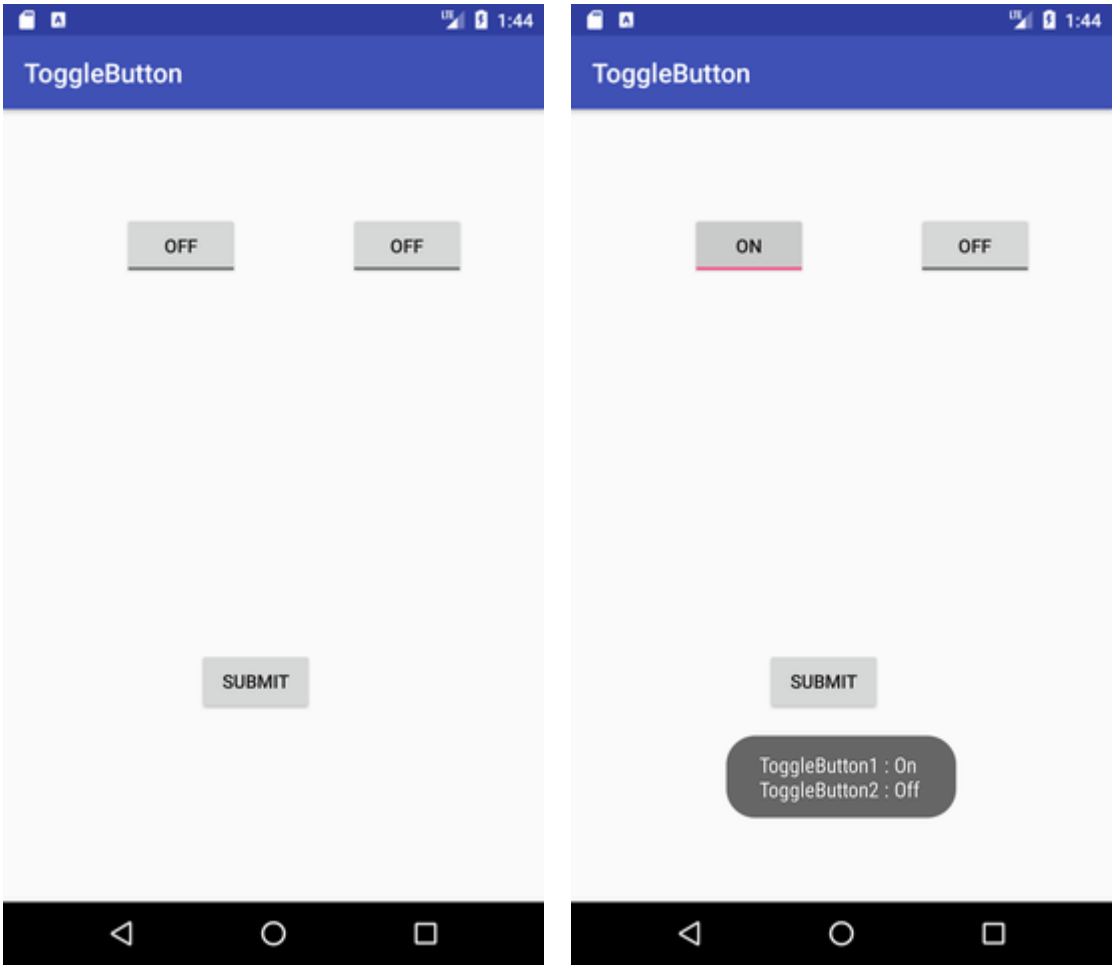
Let's write the code to check which toggle button is ON/OFF.

File: MainActivity.java

```
package example.javatpoint.com.togglebutton;

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

public class MainActivity extends AppCompatActivity {
    private ToggleButton toggleButton1, toggleButton2;
    private Button buttonSubmit;
    @Override
```

```
14.    protected void onCreate(Bundle savedInstanceState) {
15.       super.onCreate(savedInstanceState);
16.       setContentView(R.layout.activity_main);
17.
18.       addListenerOnButtonClick();
19.    }
20.
21.    public void addListenerOnButtonClick(){
22.       //Getting the ToggleButton and Button instance from the layout xml file
23.       toggleButton1=(ToggleButton)findViewById(R.id.toggleButton);
24.       toggleButton2=(ToggleButton)findViewById(R.id.toggleButton2);
25.       buttonSubmit=(Button)findViewById(R.id.button);
26.
27.       //Performing action on button click
28.       buttonSubmit.setOnClickListener(new View.OnClickListener(){
29.
30.          @Override
31.          public void onClick(View view) {
32.             StringBuilder result = new StringBuilder();
33.             result.append("ToggleButton1 : ").append(toggleButton1.getText());
34.             result.append("\nToggleButton2 : ").append(toggleButton2.getText());
35.             //Displaying the message in toast
36.             Toast.makeText(getApplicationContext(), result.toString(),Toast.LENGTH_LO
    NG).show();
37.          }
38.
39.       });
40.
41.    }
42. }
```

Output:



Android CheckBox Example

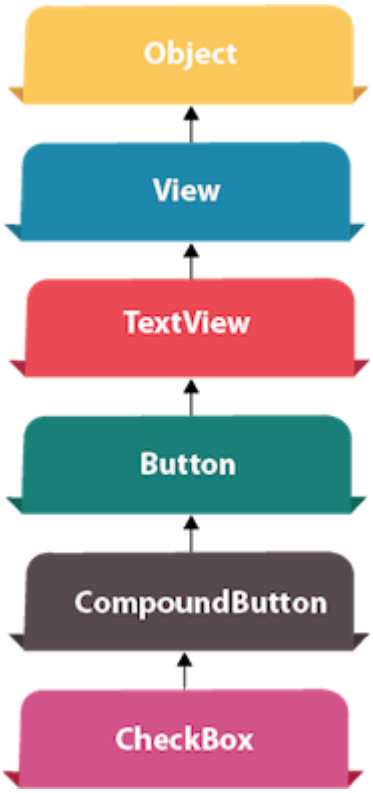
Android CheckBox is a type of two state button either checked or unchecked.

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

Android CheckBox class is the subclass of CompoundButton class.

Android CheckBox class

The android.widget.CheckBox class provides the facility of creating the CheckBoxes.



Methods of CheckBox class

There are many inherited methods of View, TextView, and Button classes in the CheckBox class. Some of them are as follows:

Method	Description
public boolean isChecked()	Returns true if it is checked otherwise false.
public void setChecked(boolean status)	Changes the state of the CheckBox.

Android CheckBox Example

activity_main.xml

Drag the three checkboxes and one button for the layout. Now the activity_main.xml file will look like this:

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.and
roid.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.checkbox.MainActivity">


    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="68dp"
        android:text="Pizza"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="28dp"
        android:text="Coffee"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox" />

    <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="28dp"
        android:text="Burger"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox2" />

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

```
        android:layout_marginLeft="144dp"
        android:layout_marginTop="184dp"
        android:text="Order"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox3" />

</android.support.constraint.ConstraintLayout>
```

Activity class

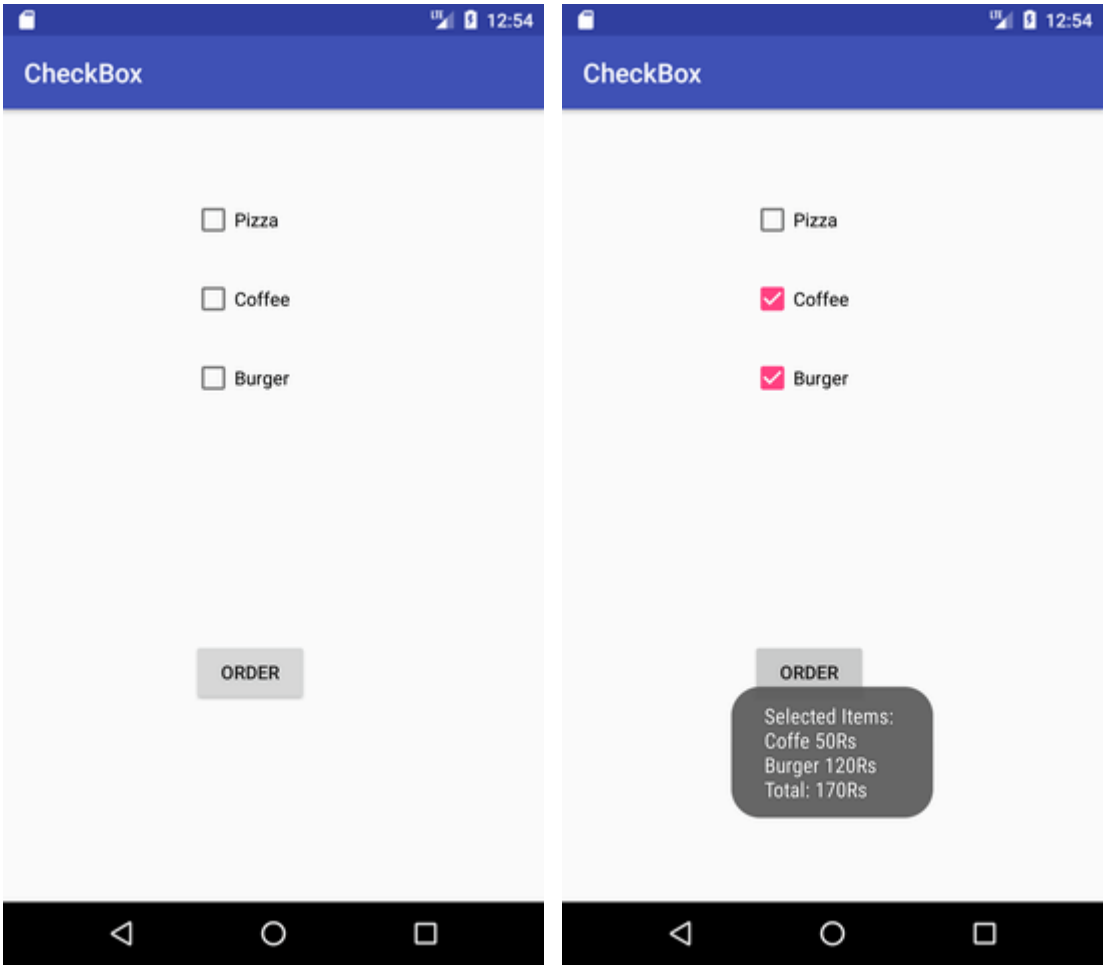
Let's write the code to check which toggle button is ON/OFF.

File: MainActivity.java

```
package example.javatpoint.com.checkbox;

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

public class MainActivity extends AppCompatActivity {
    CheckBox pizza,coffe,burger;
    Button buttonOrder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addListenerOnButtonClick();
    }
    public void addListenerOnButtonClick(){
        //Getting instance of CheckBoxes and Button from the activty_main.xml file
        pizza=(CheckBox)findViewById(R.id.checkBox);
        coffe=(CheckBox)findViewById(R.id.checkBox2);
        burger=(CheckBox)findViewById(R.id.checkBox3);
        buttonOrder=(Button)findViewById(R.id.button);

        //Applying the Listener on the Button click
        buttonOrder.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View view) {
```

```
31.             int totalamount=0;
32.             StringBuilder result=new StringBuilder();
33.             result.append("Selected Items:");
34.             if(pizza.isChecked()){
35.                 result.append("\nPizza 100Rs");
36.                 totalamount+=100;
37.             }
38.             if(coffe.isChecked()){
39.                 result.append("\nCoffe 50Rs");
40.                 totalamount+=50;
41.             }
42.             if(burger.isChecked()){
43.                 result.append("\nBurger 120Rs");
44.                 totalamount+=120;
45.             }
46.             result.append("\nTotal: "+totalamount+"Rs");
47.             //Displaying the message on the toast
48.             Toast.makeText(getApplicationContext(), result.toString(), Toast.LENGTH_LO
    NG).show();
49.         }
50.
51.     });
52.   }
53. }
```

Output:



Android AlertDialog Example

Android AlertDialog can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

Android AlertDialog is composed of three regions: title, content area and action buttons.

Android AlertDialog is the subclass of Dialog class.

Methods of AlertDialog class

Method	Description
public AlertDialog.Builder setTitle(CharSequence)	This method is used to set the title of AlertDialog.
public AlertDialog.Builder setMessage(CharSequence)	This method is used to set the message for AlertDialog.
public AlertDialog.Builder setIcon(int)	This method is used to set the icon over AlertDialog.

Android AlertDialog Example

Let's see a simple example of android alert dialog.

activity_main.xml

You can have multiple components, here we are having only a textview.

File: activity_main.xml

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Close app"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

strings.xml

Optionally, you can store the dialog message and title in the strings.xml file.

File: strings.xml

```
<resources>
    <string name="app_name">AlertDialog</string>
    <string name="dialog_message">Welcome to Alert Dialog</string>
    <string name="dialog_title">Javatpoint Alert Dialog</string>
</resources>
```

Activity class

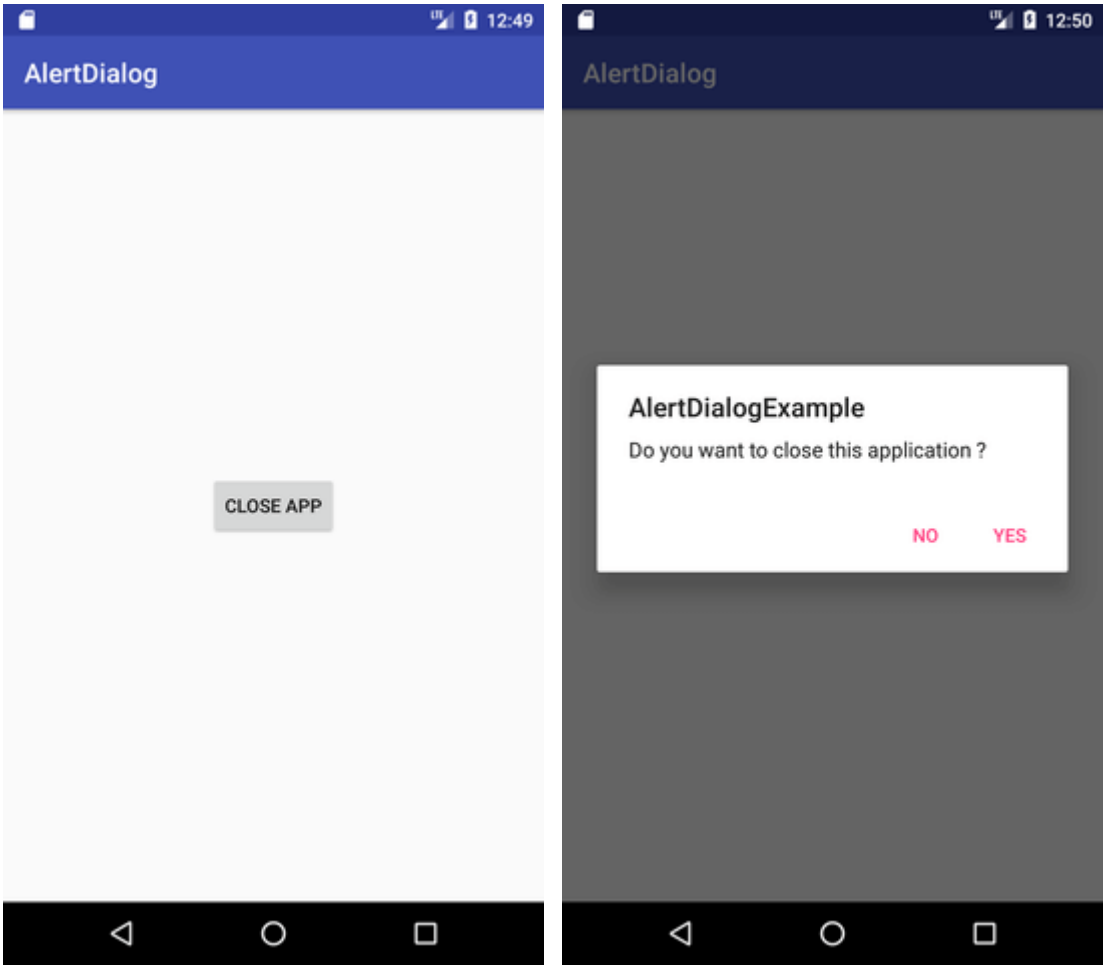
Let's write the code to create and show the AlertDialog.

File: MainActivity.java

```
package example.javatpoint.com.alertdialog;

import android.content.DialogInterface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.app.AlertDialog;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    Button closeButton;
    AlertDialog.Builder builder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        closeButton = (Button) findViewById(R.id.button);
        builder = new AlertDialog.Builder(this);
        closeButton.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
23.         public void onClick(View v) {
24.
25.             //Uncomment the below code to Set the message and title from the strings.x
    ml file
26.             builder.setMessage(R.string.dialog_message) .setTitle(R.string.dialog_title);
27.
28.             //Setting message manually and performing action on button click
29.             builder.setMessage("Do you want to close this application ?")
30.                     .setCancelable(false)
31.                     .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
32.                         public void onClick(DialogInterface dialog, int id) {
33.                             finish();
34.                             Toast.makeText(getApplicationContext(),"you choose yes action fo
    r alertbox",
35.                             Toast.LENGTH_SHORT).show();
36.                         }
37.                     })
38.                     .setNegativeButton("No", new DialogInterface.OnClickListener() {
39.                         public void onClick(DialogInterface dialog, int id) {
40.                             //  Action for 'NO' Button
41.                             dialog.cancel();
42.                             Toast.makeText(getApplicationContext(),"you choose no action for
     alertbox",
43.                             Toast.LENGTH_SHORT).show();
44.                         }
45.                     });
46.             //Creating dialog box
47.             AlertDialog alert = builder.create();
48.             //Setting the title manually
49.             alert.setTitle("AlertDialogExample");
50.             alert.show();
51.         }
52.     });
53.   }
54. }
```

Output:



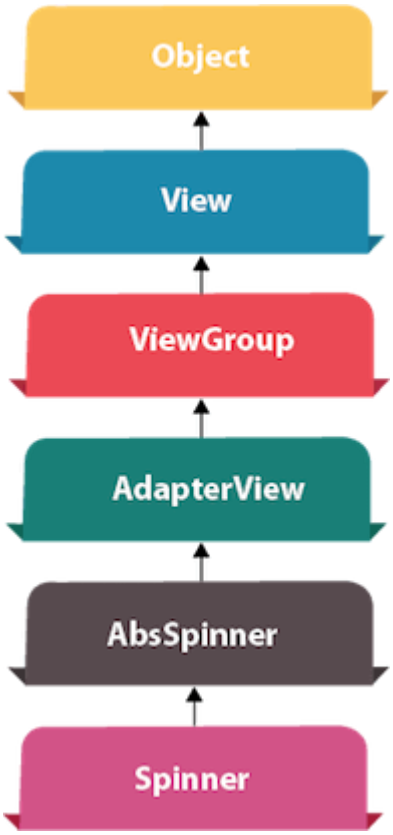
Android Spinner Example

Android Spinner is like the combox box of AWT or Swing. It can be used to display the multiple options to the user in which only one item can be selected by the user.

Android spinner is like the drop down menu with multiple values from which the end user can select only one value.

Android spinner is associated with AdapterView. So you need to use one of the adapter classes with spinner.

Android Spinner class is the subclass of AsbSpinner class.



Android Spinner Example

In this example, we are going to display the country list. You need to use **ArrayAdapter** class to store the country list.

Let's see the simple example of spinner in android.

activity_main.xml

Drag the Spinner from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.and
roid.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.javatpoint.com.spinner.MainActivity">

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="149dp"
```

```
        android:layout_height="40dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.498" />

</android.support.constraint.ConstraintLayout>
```

Activity class

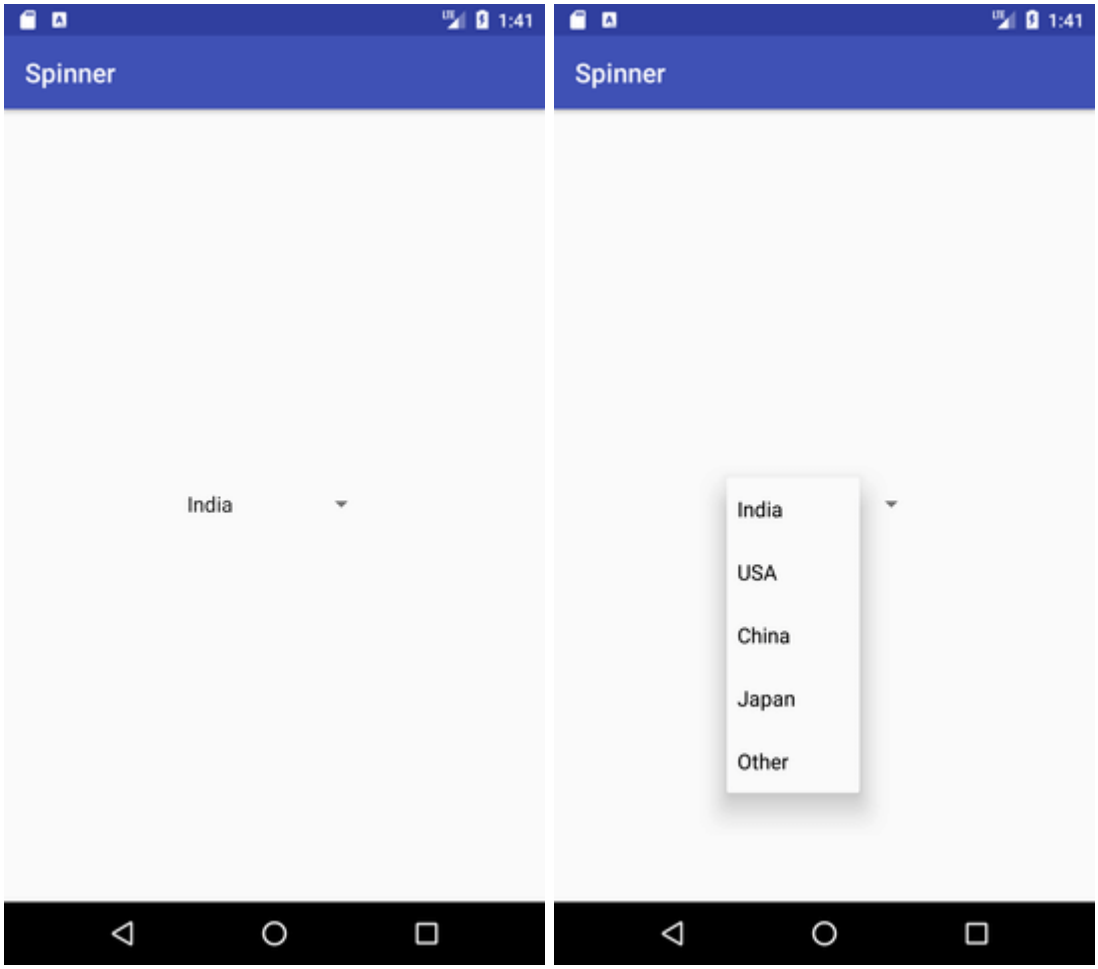
Let's write the code to display item on the spinner and perform event handling.

File: MainActivity.java

```
package example.javatpoint.com.spinner;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements
        AdapterView.OnItemSelectedListener {
    String[] country = { "India", "USA", "China", "Japan", "Other"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       //Getting the instance of Spinner and applying OnItemSelectedListener on it
        Spinner spin = (Spinner) findViewById(R.id.spinner);
        spin.setOnItemSelectedListener(this);

        //Creating the ArrayAdapter instance having the country list
```

```
24.         ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,c
    ountry);
25.         aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
26.         //Setting the ArrayAdapter data on the Spinner
27.         spin.setAdapter(aa);
28.
29.     }
30.
31.     //Performing action onItemSelected and onNothing selected
32.     @Override
33.     public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id)
    {
34.         Toast.makeText(getApplicationContext(),country[position] , Toast.LENGTH_LONG).
    show();
35.     }
36.     @Override
37.     public void onNothingSelected(AdapterView<?> arg0) {
38.         // TODO Auto-generated method stub
39.     }
40. }
```

Output:

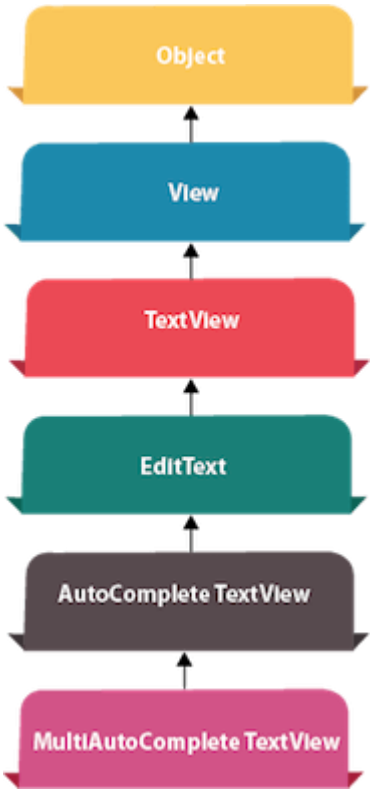

1:41
Spinner
India
1:41
Spinner
India
USA
China
Japan
Other

Android AutoCompleteTextView Example

Android AutoCompleteTextView completes the word based on the reserved words, so no need to write all the characters of the word.

Android AutoCompleteTextView is a editable text field, it displays a list of suggestions in a drop down menu from which user can select only one suggestion or value.

Android AutoCompleteTextView is the subclass of EditText class. The MultiAutoCompleteTextView is the subclass of AutoCompleteTextView class.



Android AutoCompleteTextView Example

In this example, we are displaying the programming languages in the autocompletetextview. All the programming languages are stored in string array. We are using the **ArrayAdapter** class to display the array content.

Let's see the simple example of autocompletetextview in android.

activity_main.xml

Drag the AutoCompleteTextView and TextView from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

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

    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="What is your favourite programming language?"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.032" />

    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="92dp"
        android:layout_marginTop="144dp"
        android:text=""
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Activity class

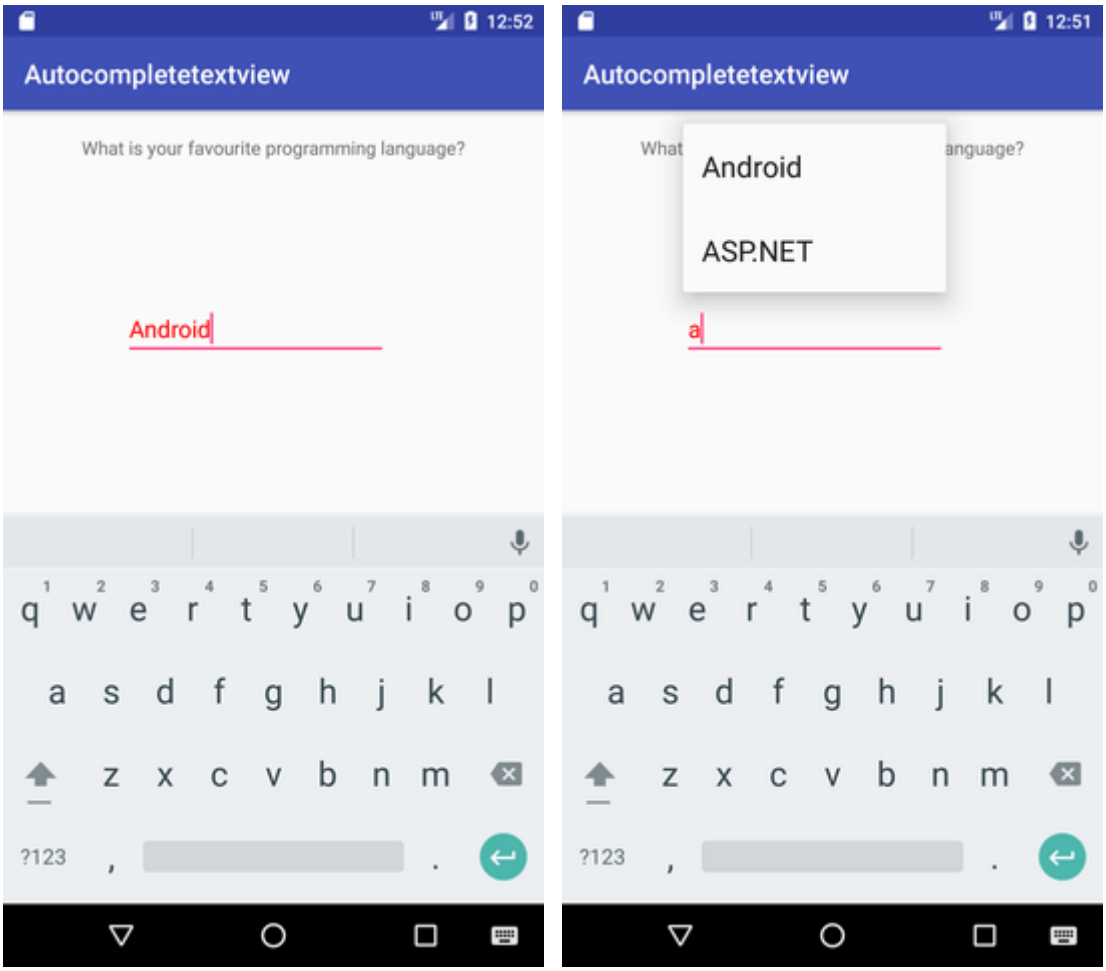
Let's write the code of AutoCompleteTextView.

File: MainActivity.java

```
package example.javatpoint.com.autocompletetextview;

import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class MainActivity extends AppCompatActivity {
    String[] language ={"C","C++","Java",".NET","iPhone","Android","ASP.NET","PHP"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Creating the instance of ArrayAdapter containing list of language names
        ArrayAdapter<String> adapter = new ArrayAdapter<String>
                (this,android.R.layout.select_dialog_item,language);
        //Getting the instance of AutoCompleteTextView
```

```
19.         AutoCompleteTextView actv =  (AutoCompleteTextView)findViewById(R.id.autoCo
    mpleteTextView);
20.         actv.setThreshold(1);//will start working from first character
21.         actv.setAdapter(adapter);//setting the adapter data into the AutoCompleteTextVie
    w
22.         actv.setTextColor(Color.RED);
23.     }
24. }
```

Output:

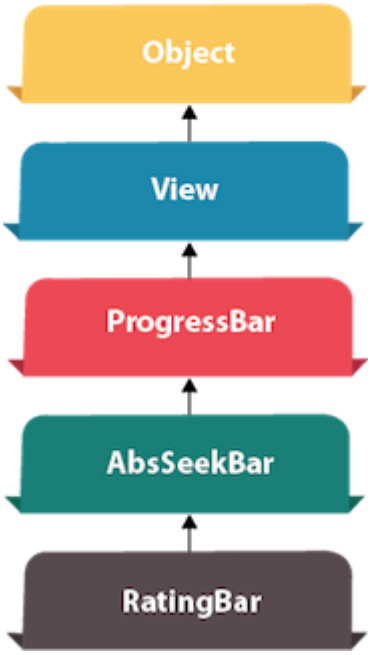


Android RatingBar Example

Android RatingBar can be used to get the rating from the user. The Rating returns a floating-point number. It may be 2.0, 3.5, 4.0 etc.

Android RatingBar displays the rating in stars. Android RatingBar is the subclass of AbsSeekBar class.

The **getRating()** method of android RatingBar class returns the rating number.



Android RatingBar Example

Let's see the simple example of rating bar in android.

activity_main.xml

Drag the RatingBar and Button from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="submit"
        android:id="@+id/button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.615" />

    <RatingBar
        android:id="@+id/ratingBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="72dp"
        android:layout_marginTop="60dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Activity class

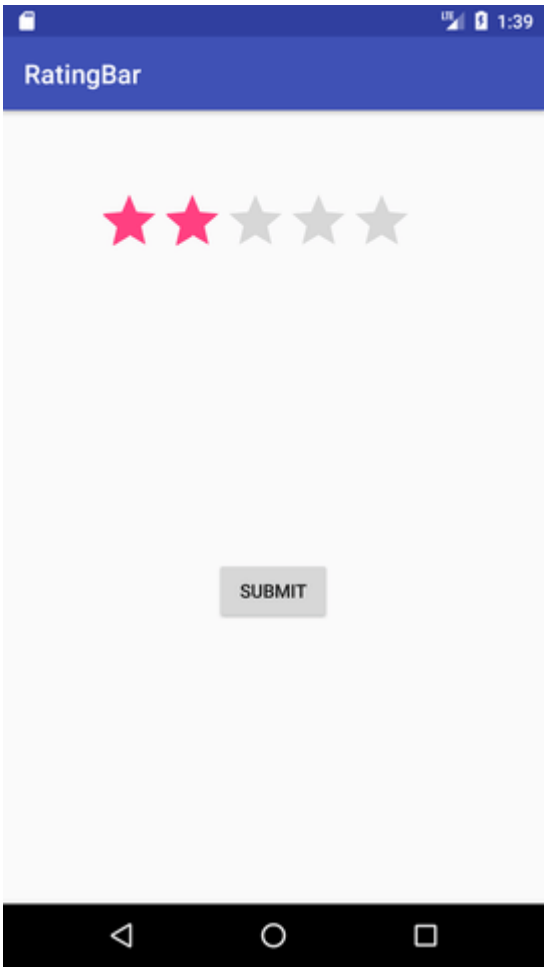
Let's write the code to display the rating of the user.

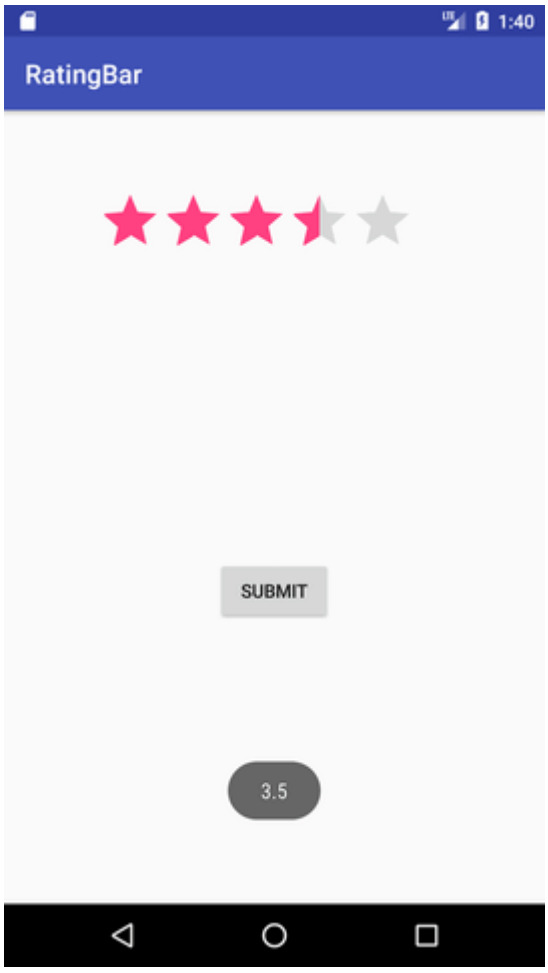
File: MainActivity.java

```
package example.javatpoint.com.ratingbar;

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

public class MainActivity extends AppCompatActivity {
    RatingBar ratingbar;
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addListenerOnButtonClick();
    }
    public void addListenerOnButtonClick(){
        ratingbar=(RatingBar)findViewById(R.id.ratingBar);
        button=(Button)findViewById(R.id.button);
        //Performing action on Button Click
        button.setOnClickListener(new View.OnClickListener(){
```

```
24.
25.         @Override
26.         public void onClick(View arg0) {
27.             //Getting the rating and displaying it on the toast
28.             String rating=String.valueOf(ratingbar.getRating());
29.             Toast.makeText(getApplicationContext(), rating, Toast.LENGTH_LONG).show(
    );
30.         }
31.
32.     });
33.   }
34. }
```

Output:

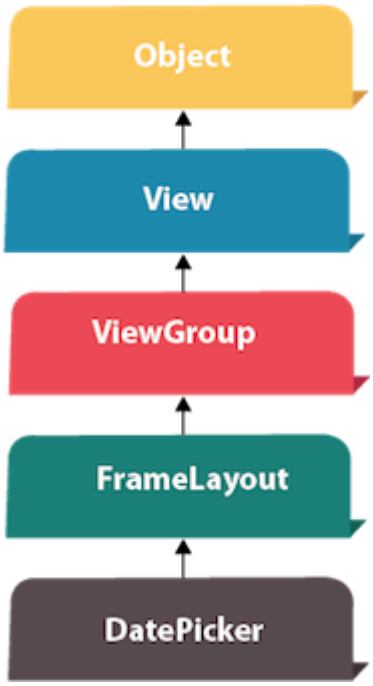




Android DatePicker Example

Android DatePicker is a widget to select date. It allows you to select date by day, month and year. Like DatePicker, android also provides TimePicker to select time.

The android.widget.DatePicker is the subclass of FrameLayout class.



Android DatePicker Example

Let's see the simple example of datepicker widget in android.

activity_main.xml

File: activity_main.xml

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

    <TextView
        android:id="@+id/textView1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button1"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="102dp"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:text="" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp"
        android:text="Change Date" />

    <DatePicker
        android:id="@+id/datePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="36dp" />

</RelativeLayout>
```

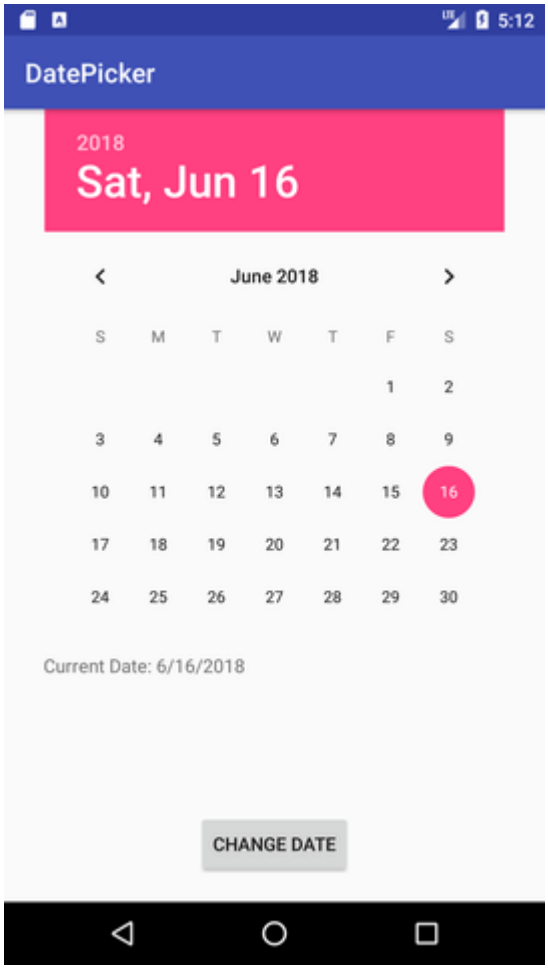
Activity class

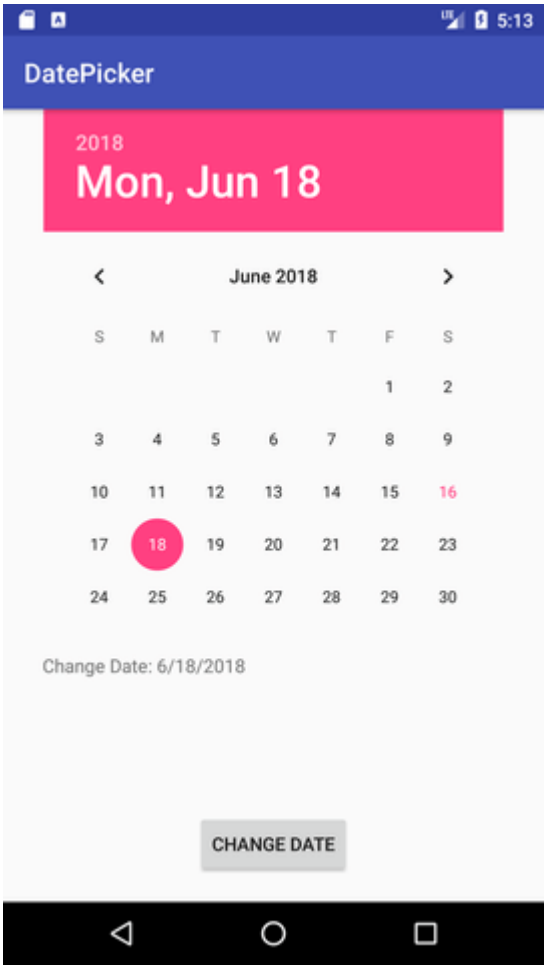
File: MainActivity.java

```
package example.javatpoint.com.datepicker;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    DatePicker picker;
```

```
12.    Button displayDate;
13.    TextView textview1;
14.    @Override
15.    protected void onCreate(Bundle savedInstanceState) {
16.        super.onCreate(savedInstanceState);
17.        setContentView(R.layout.activity_main);
18.
19.        textview1=(TextView)findViewById(R.id.textView1);
20.        picker=(DatePicker)findViewById(R.id.datePicker);
21.        displayDate=(Button)findViewById(R.id.button1);
22.
23.        textview1.setText("Current Date: "+getCurrentDate());
24.
25.        displayDate.setOnClickListener(new View.OnClickListener(){
26.            @Override
27.            public void onClick(View view) {
28.
29.                textview1.setText("Change Date: "+getCurrentDate());
30.            }
31.
32.        });
33.
34.    }
35.    public String getCurrentDate(){
36.        StringBuilder builder=new StringBuilder();;
37.        builder.append((picker.getMonth() + 1)+"/");//month is 0 based
38.        builder.append(picker.getDayOfMonth()+"/");
39.        builder.append(picker.getYear());
40.        return builder.toString();
41.    }
42. }
```

Output:

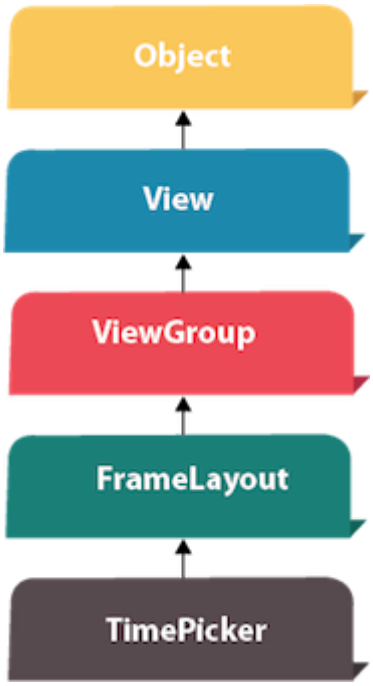

5:12
DatePicker
2018
Sat, Jun 16
June 2018
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
Current Date: 6/16/2018
CHANGE DATE


5:13
DatePicker
2018
Mon, Jun 18
June 2018
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
Change Date: 6/18/2018
CHANGE DATE

Android TimePicker Example

Android TimePicker widget is used to select date. It allows you to select time by hour and minute. You cannot select time by seconds.

The android.widget.TimePicker is the subclass of FrameLayout class.



Android TimePicker Example

Let's see a simple example of android time picker.

activity_main.xml

File: activity_main.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_above="@+id/button1"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="102dp"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:text="" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp"
        android:text="Change Time" />

    <TimePicker
        android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="36dp" />
</RelativeLayout>
```

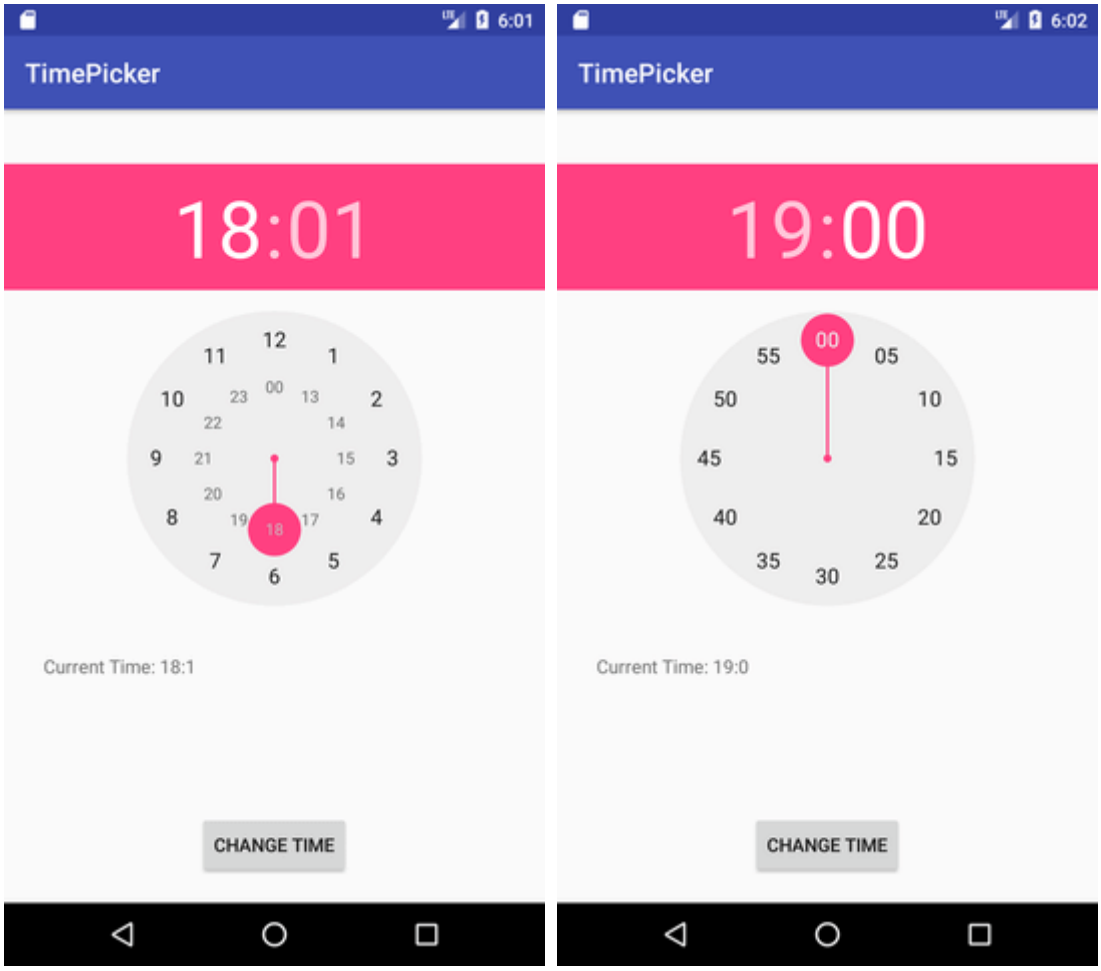
Activity class

File: MainActivity.java

```
package example.javatpoint.com.timepicker;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.TimePicker;

public class MainActivity extends AppCompatActivity {
    TextView textview1;
    TimePicker timepicker;
    Button changetime;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
16.         super.onCreate(savedInstanceState);
17.         setContentView(R.layout.activity_main);
18.
19.         textview1=(TextView)findViewById(R.id.textView1);
20.         timepicker=(TimePicker)findViewById(R.id.timePicker);
21.         //Uncomment the below line of code for 24 hour view
22.         timepicker.setIs24HourView(true);
23.         changetime=(Button)findViewById(R.id.button1);
24.
25.         textview1.setText(getCurrentTime());
26.
27.         changetime.setOnClickListener(new View.OnClickListener(){
28.             @Override
29.             public void onClick(View view) {
30.                 textview1.setText(getCurrentTime());
31.             }
32.         });
33.
34.     }
35.
36.     public String getCurrentTime(){
37.         String currentTime="Current Time: "+timepicker.getCurrentHour()+":"+timepicker
    .getCurrentMinute();
38.         return currentTime;
39.     }
40.
41. }
```

Output:



Android Analog clock and Digital clock example

The **android.widget.AnalogClock** and **android.widget.DigitalClock** classes provides the functionality to display analog and digital clocks.

Android analog and digital clocks are used to show time in android application.

Android AnalogClock is the subclass of View class.

Android DigitalClock is the subclass of TextView class. Since Android API level 17, it is *deprecated*. You are recommended to use **TextClock** Instead.

The AnalogClock was deprecated in API level 23. This widget is no longer supported. Instead if you want to use AnalogClock in your application you need to hard code. It does not appear in API level 27 to drag from palette.

Note: Analog and Digital clocks cannot be used to change the time of the device. To do so, you need to use DatePicker and TimePicker.

In android, you need to drag analog and digital clocks from the pallet to display analog and digital clocks.

activity_main.xml

Now, drag the analog and digital clocks, now the xml file will look like this.

File: activity_main.xml

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

    <AnalogClock
        android:id="@+id/analogClock1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginLeft="136dp"
        android:layout_marginTop="296dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <DigitalClock
        android:id="@+id/digitalClock1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/analogClock1"
        android:layout_centerHorizontal="true"
        android:layout_marginLeft="176dp"
        android:layout_marginTop="84dp"
        android:text="DigitalClock"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
31.
32. </android.support.constraint.ConstraintLayout>
```

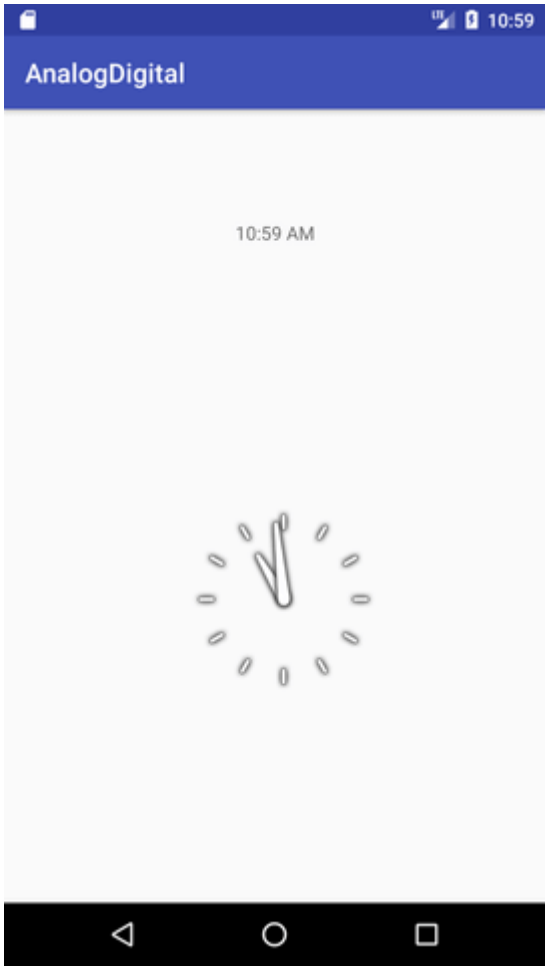
Activity class

We have not write any code here.

File: MainActivity.java

```
1.  package example.javatpoint.com.analogdigital;
2.
3.  import android.support.v7.app.AppCompatActivity;
4.  import android.os.Bundle;
5.
6.  public class MainActivity extends AppCompatActivity {
7.
8.      @Override
9.      protected void onCreate(Bundle savedInstanceState) {
10.         super.onCreate(savedInstanceState);
11.         setContentView(R.layout.activity_main);
12.     }
13. }
```

Output:



Android ProgressBar Example

We can display the **android progress bar** dialog box to display the status of work being done e.g. downloading file, analyzing status of work etc.

In this example, we are displaying the progress dialog for dummy file download operation.

Here we are using **android.app.ProgressDialog** class to show the progress bar. Android ProgressDialog is the subclass of AlertDialog class.

The **ProgressDialog** class provides methods to work on progress bar like setProgress(), setMessage(), setProgressStyle(), setMax(), show() etc. The progress range of Progress Dialog is 0 to 10000.

Let's see a simple example to display progress bar in android.

```
1. ProgressDialog progressBar = new ProgressDialog(this);
2. progressBar.setCancelable(true);//you can cancel it by pressing back button
3. progressBar.setMessage("File downloading ...");
4. progressBar.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
5. progressBar.setProgress(0);//initially progress is 0
6. progressBar.setMax(100);//sets the maximum value 100
7. progressBar.show();//displays the progress bar
```

Android Progress Bar Example by ProgressDialog

Let's see a simple example to create progress bar using ProgressDialog class.

activity_main.xml

Drag one button from the pallete, now the activity_main.xml file will look like this:

File: activity_main.xml

```
1. <RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     tools:context=".MainActivity" >
6.
7.     <Button
8.         android:id="@+id/button1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="116dp"
        android:text="download file" />

</RelativeLayout>
```

Activity class

Let's write the code to display the progress bar dialog box.

File: MainActivity.java

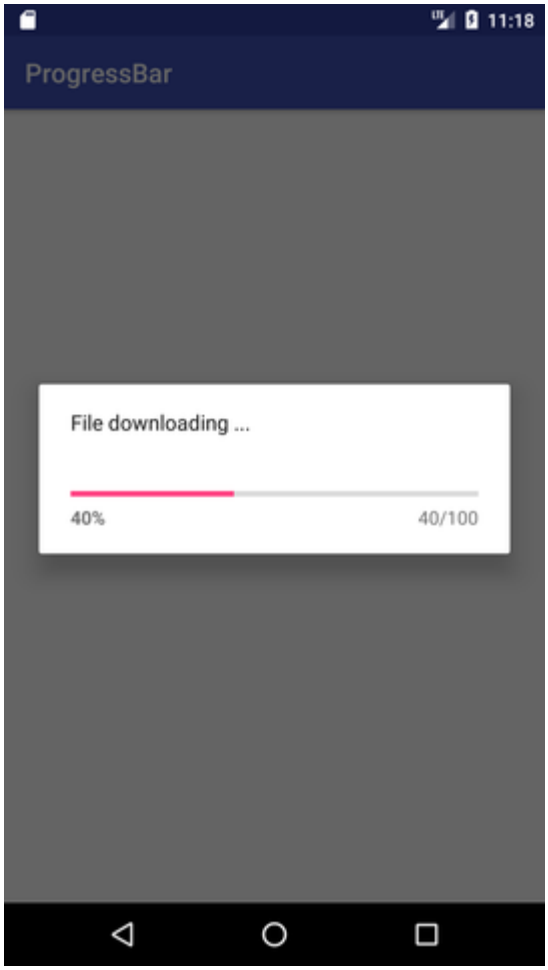
```
package example.javatpoint.com.progressbar;

import android.app.ProgressDialog;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
    Button btnStartProgress;
    ProgressDialog progressBar;
    private int progressBarStatus = 0;
    private Handler progressBarHandler = new Handler();
    private long fileSize = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addListenerOnButtonClick();
    }
    public void addListenerOnButtonClick() {
        btnStartProgress = findViewById(R.id.button);
        btnStartProgress.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View v) {
                // creating progress bar dialog
                progressBar = new ProgressDialog(v.getContext());
```

```
30.             progressBar.setCancelable(true);
31.             progressBar.setMessage("File downloading ...");
32.             progressBar.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
33.             progressBar.setProgress(0);
34.             progressBar.setMax(100);
35.             progressBar.show();
36.             //reset progress bar and filesize status
37.             progressBarStatus = 0;
38.             fileSize = 0;
39.
40.             new Thread(new Runnable() {
41.                 public void run() {
42.                     while (progressBarStatus < 100) {
43.                         // performing operation
44.                         progressBarStatus = doOperation();
45.                         try {
46.                             Thread.sleep(1000);
47.                         } catch (InterruptedException e) {
48.                             e.printStackTrace();
49.                         }
50.                         // Updating the progress bar
51.                         progressBarHandler.post(new Runnable() {
52.                             public void run() {
53.                                 progressBar.setProgress(progressBarStatus);
54.                             }
55.                         });
56.                     }
57.                     // performing operation if file is downloaded,
58.                     if (progressBarStatus >= 100) {
59.                         // sleeping for 1 second after operation completed
60.                         try {
61.                             Thread.sleep(1000);
62.                         } catch (InterruptedException e) {
63.                             e.printStackTrace();
64.                         }
65.                         // close the progress bar dialog
66.                         progressBar.dismiss();
67.                     }
68.                 }
69.             }).start();
70.         }//end of onClick method
71.     });
72. }
73. // checking how much file is downloaded and updating the filesize
```

```
74.    public int doOperation() {
75.       //The range of ProgressDialog starts from 0 to 10000
76.       while (fileSize <= 10000) {
77.          fileSize++;
78.          if (fileSize == 1000) {
79.             return 10;
80.          } else if (fileSize == 2000) {
81.             return 20;
82.          } else if (fileSize == 3000) {
83.             return 30;
84.          } else if (fileSize == 4000) {
85.             return 40; // you can add more else if
86.          }
87.        /* else {
88.             return 100;
89.          }*/
90.       }//end of while
91.          return 100;
92.    }//end of doOperation
93. }
```

Output:


11:17
ProgressBar
DOWNLOAD FILE


11:18
ProgressBar
File downloading ...
40%
40/100