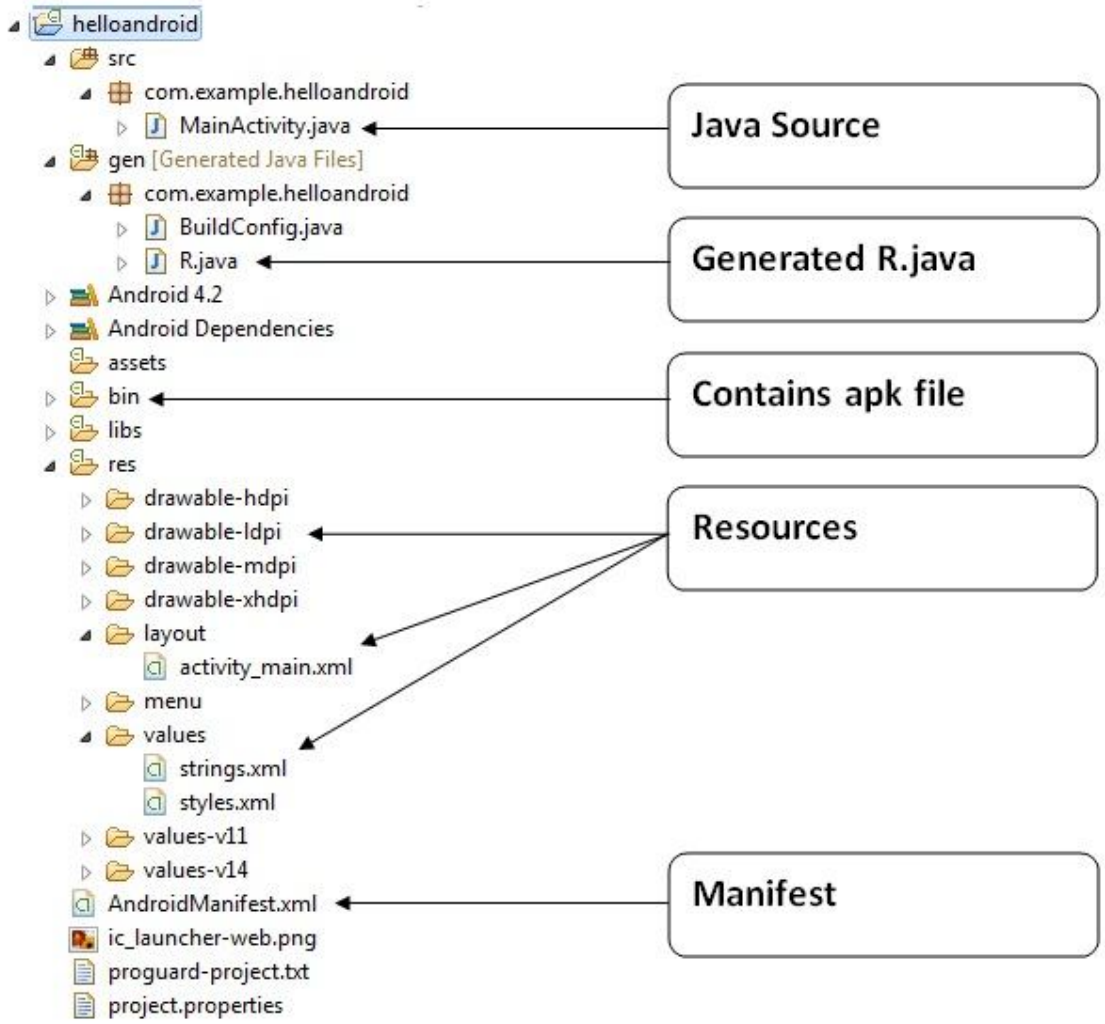
Internal Details of Hello Android Example

Here, we are going to learn the internal details or working of hello android example.

Android application contains different components such as java source code, string resources, images, manifest file, apk file etc. Let's understand the project structure of android application.



Java Source Code

Let's see the java source file created by the Eclipse IDE:

File: MainActivity.java

```
1. package com.example.helloandroid;
2. import android.os.Bundle;
3. import android.app.Activity;
4. import android.view.Menu;
5. import android.widget.TextView;
6. public class MainActivity extends Activity {//(1)
```

```
7.    @Override
8.    protected void onCreate(Bundle savedInstanceState) {//(2)
9.        super.onCreate(savedInstanceState);
10.
11.       setContentView(R.layout.activity_main);//(3)
12.   }
13.   @Override
14.   public boolean onCreateOptionsMenu(Menu menu) {//(4)
15.       // Inflate the menu; this adds items to the action bar if it is present.
16.       getMenuInflater().inflate(R.menu.activity_main, menu);
17.       return true;
18.   }
19. }
```

(1) Activity is a java class that creates and default window on the screen where we can place different components such as Button, EditText, TextView, Spinner etc. It is like the Frame of Java AWT.

It provides life cycle methods for activity such as onCreate, onStop, OnResume etc.

(2) The **onCreate method** is called when Activity class is first created.

(3) The **setContentView(R.layout.activity_main)** gives information about our layout resource. Here, our layout resources are defined in activity_main.xml file.

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.      <TextView
7.          android:layout_width="wrap_content"
8.          android:layout_height="wrap_content"
9.          android:layout_centerHorizontal="true"
10.         android:layout_centerVertical="true"
11.         android:text="@string/hello_world" />
12. </RelativeLayout>
```

As you can see, a textview is created by the framework automatically. But the message for this string is defined in the strings.xml file. The **@string/hello_world** provides information about the textview message. The value of the attribute hello_world is defined in the strings.xml file.

File: strings.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <resources>
```

```
3.     <string name="app_name">helloandroid</string>
4.     <string name="hello_world">Hello world!</string>
5.     <string name="menu_settings">Settings</string>
6.  </resources>
```

You can change the value of the hello_world attribute from this file.

Generated R.java file

It is the auto-generated file that contains IDs for all the resources of res directory. It is generated by aapt(Android Asset Packaging Tool). Whenever you create any component on activity_main, a corresponding ID is created in the R.java file which can be used in the Java Source file later.

File: R.java

```
1.  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
2.   *
3.   * This class was automatically generated by the
4.   * aapt tool from the resource data it found.  It
5.   * should not be modified by hand.
6.   */
7.  package com.example.helloandroid;
8.  public final class R {
9.     public static final class attr {
10.    }
11.    public static final class drawable {
12.       public static final int ic_launcher=0x7f020000;
13.    }
14.    public static final class id {
15.       public static final int menu_settings=0x7f070000;
16.    }
17.    public static final class layout {
18.       public static final int activity_main=0x7f030000;
19.    }
20.    public static final class menu {
21.       public static final int activity_main=0x7f060000;
22.    }
23.    public static final class string {
24.       public static final int app_name=0x7f040000;
25.       public static final int hello_world=0x7f040001;
26.       public static final int menu_settings=0x7f040002;
27.    }
28.    public static final class style {
```

```
29.       /**
30.       Base application theme, dependent on API level. This theme is replaced
31.       by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
32.          Theme customizations available in newer API levels can go in
33.          res/values-vXX/styles.xml, while customizations related to
34.          backward-compatibility can go here.
35.        Base application theme for API 11+. This theme completely replaces
36.       AppBaseTheme from res/values/styles.xml on API 11+ devices.
37.  API 11 theme customizations can go here.
38.       Base application theme for API 14+. This theme completely replaces
39.       AppBaseTheme from BOTH res/values/styles.xml and
40.       res/values-v11/styles.xml on API 14+ devices.
41. API 14 theme customizations can go here.
42.        */
43.       public static final int AppBaseTheme=0x7f050000;
44.       /**  Application theme.
45. All customizations that are NOT specific to a particular API-level can go here.
46.        */
47.       public static final int AppTheme=0x7f050001;
48.    }
49. }
```

APK File

An apk file is created by the framework automatically. If you want to run the android application on the mobile, transfer and install it.

Resources

It contains resource files including activity_main, strings, styles etc.

Manifest file

It contains information about package including components such as activities, services, content providers etc.

AndroidManifest.xml file in android

The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.

It performs some other tasks also:

- It is **responsible to protect the application** to access any protected parts by providing the permissions.
- It also **declares the android api** that the application is going to use.
- It **lists the instrumentation classes**. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.

This is the required xml file for all the android application and located inside the root directory.

A simple AndroidManifest.xml file looks like this:

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Elements of the AndroidManifest.xml file

The elements used in the above xml file are described below.

<manifest>

manifest is the root element of the AndroidManifest.xml file. It has **package** attribute that describes the package name of the activity class.

<application>

application is the subelement of the manifest. It includes the namespace declaration. This element contains several subelements that declares the application component such as activity etc.

The commonly used attributes are of this element are **icon**, **label**, **theme** etc.

android:icon represents the icon for all the android application components.

android:label works as the default label for all the application components.

android:theme represents a common theme for all the android activities.

<activity>

activity is the subelement of application and represents an activity that must be defined in the AndroidManifest.xml file. It has many attributes such as label, name, theme, launchMode etc.

android:label represents a label i.e. displayed on the screen.

android:name represents a name for the activity class. It is required attribute.

<intent-filter>

intent-filter is the sub-element of activity that describes the type of intent to which activity, service or broadcast receiver can respond to.

<action>

It adds an action for the intent-filter. The intent-filter must have at least one action element.

<category>

It adds a category name to an intent-filter.

Android R.java file

Android R.java is *an auto-generated file by aapt* (Android Asset Packaging Tool) that contains resource IDs for all the resources of res/ directory.

If you create any component in the activity_main.xml file, id for the corresponding component is automatically created in this file. This id can be used in the activity source file to perform any action on the component.

Note: If you delete R.jar file, android creates it automatically.

Let's see the android R.java file. It includes a lot of static nested classes such as menu, id, layout, attr, drawable, string etc.

```
1. /* AUTO-GENERATED FILE.  DO NOT MODIFY.
2.  *
3.  * This class was automatically generated by the
4.  * aapt tool from the resource data it found.  It
5.  * should not be modified by hand.
6.  */
7.
8. package com.example.helloandroid;
9.
10. public final class R {
11.     public static final class attr {
12.     }
13.     public static final class drawable {
14.         public static final int ic_launcher=0x7f020000;
15.     }
16.     public static final class id {
17.         public static final int menu_settings=0x7f070000;
18.     }
19.     public static final class layout {
20.         public static final int activity_main=0x7f030000;
21.     }
22.     public static final class menu {
23.         public static final int activity_main=0x7f060000;
24.     }
25.     public static final class string {
26.         public static final int app_name=0x7f040000;
27.         public static final int hello_world=0x7f040001;
28.         public static final int menu_settings=0x7f040002;
29.     }
30.     public static final class style {
31.         /**
32.         Base application theme, dependent on API level. This theme is replaced
33.         by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
34.
35.
36.             Theme customizations available in newer API levels can go in
37.             res/values-vXX/styles.xml, while customizations related to
38.             backward-compatibility can go here.
39.
```

```
40.
41.         Base application theme for API 11+. This theme completely replaces
42.         AppBaseTheme from res/values/styles.xml on API 11+ devices.
43.
44.  API 11 theme customizations can go here.
45.
46.         Base application theme for API 14+. This theme completely replaces
47.         AppBaseTheme from BOTH res/values/styles.xml and
48.         res/values-v11/styles.xml on API 14+ devices.
49.
50.  API 14 theme customizations can go here.
51.          */
52.         public static final int AppBaseTheme=0x7f050000;
53.         /**  Application theme.
54.  All customizations that are NOT specific to a particular API-level can go here.
55.          */
56.         public static final int AppTheme=0x7f050001;
57.     }
58. }
```

Android Hide Title Bar Example

In this example, we are going to explain how to hide the title bar and how to display content in full screen mode.

The **requestWindowFeature(Window.FEATURE_NO_TITLE)** method of Activity must be called to hide the title. But, it must be coded before the setContentView method.

```
1.      @Override
2.      protected void onCreate(Bundle savedInstanceState) {
3.          super.onCreate(savedInstanceState);
4.
5.  requestWindowFeature(Window.FEATURE_NO_TITLE);//will hide the title not the title bar
6.
7.          setContentView(R.layout.activity_main);
8.
9.      }
10. }
```

The **setFlags()** method of Window class is used to display content in full screen mode. You need to pass the **WindowManager.LayoutParams.FLAG_FULLSCREEN** constant in the setFlags method.

```
1.  @Override
```

```
2.  protected void onCreate(Bundle savedInstanceState) {
3.      super.onCreate(savedInstanceState);
4.
5.      requestWindowFeature(Window.FEATURE_NO_TITLE);
6.      //code that displays the content in full screen mode
7.      this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
8.              WindowManager.LayoutParams.FLAG_FULLSCREEN);//int flag, int mask
9.
10.     setContentView(R.layout.activity_main);
11.
12. }
```

Android Hide Title Bar Example

Let's see the full code to hide the title bar in android.

activity_main.xml

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.      android:paddingBottom="@dimen/activity_vertical_margin"
6.      android:paddingLeft="@dimen/activity_horizontal_margin"
7.      android:paddingRight="@dimen/activity_horizontal_margin"
8.      android:paddingTop="@dimen/activity_vertical_margin"
9.      tools:context=".MainActivity" >
10.
11.     <TextView
12.         android:layout_width="wrap_content"
13.         android:layout_height="wrap_content"
14.         android:text="@string/hello_world" />
15.
16. </RelativeLayout>
```

Activity class

File: MainActivity.java

```
1.  package com.javatpoint.hidetitle;
2.
3.  import android.os.Bundle;
```

```
4. import android.app.Activity;
5. import android.view.Menu;
6. import android.view.Window;
7. import android.view.WindowManager;
8.
9. public class MainActivity extends Activity {
10.
11.    @Override
12.    protected void onCreate(Bundle savedInstanceState) {
13.       super.onCreate(savedInstanceState);
14.
15.       requestWindowFeature(Window.FEATURE_NO_TITLE);
16.
17.     /*this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
18.               WindowManager.LayoutParams.FLAG_FULLSCREEN);//int flag, int mask
19.     */
20.       setContentView(R.layout.activity_main);
21.
22.    }
23.
24.
25. }
```

Android 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.

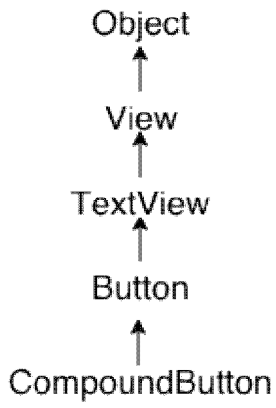
TimePicker

TimePicker displays the timepicker dialog that can be used to pick the time.

ProgressBar

ProgressBar displays progress task.

Android Button Example



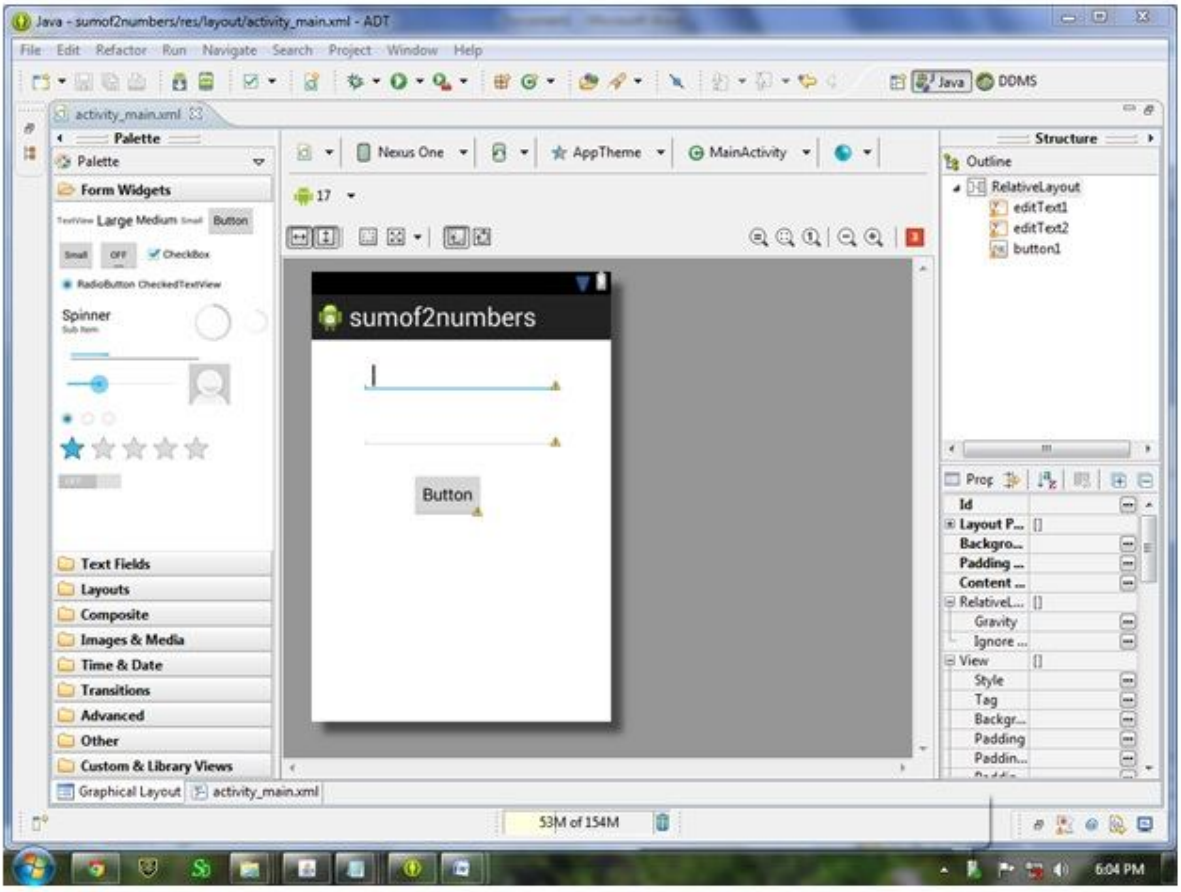
Android Button represents a push-button. The android.widget.Button is subclass of TextView class and CompoundButton is the subclass of Button class.

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

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.

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

```
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.      <EditText
8.          android:id="@+id/editText1"
9.          android:layout_width="wrap_content"
10.         android:layout_height="wrap_content"
```

```
11.     android:layout_alignParentTop="true"
12.     android:layout_centerHorizontal="true"
13.     android:layout_marginTop="24dp"
14.     android:ems="10" />
15.
16.  <EditText
17.     android:id="@+id/editText2"
18.     android:layout_width="wrap_content"
19.     android:layout_height="wrap_content"
20.     android:layout_alignLeft="@+id/editText1"
21.     android:layout_below="@+id/editText1"
22.     android:layout_marginTop="34dp"
23.     android:ems="10" >
24.
25.     <requestFocus />
26.  </EditText>
27.
28.  <Button
29.     android:id="@+id/button1"
30.     android:layout_width="wrap_content"
31.     android:layout_height="wrap_content"
32.     android:layout_centerHorizontal="true"
33.     android:layout_centerVertical="true"
34.     android:text="@string/Button" />
35.
36. </RelativeLayout>
```

Activity class

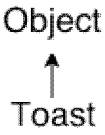
Now write the code to display the sum of two numbers.

File: MainActivity.java

```
1.  package com.example.sumof2numbers;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.view.View;
7.  import android.view.View.OnClickListener;
8.  import android.widget.Button;
9.  import android.widget.EditText;
```

```
10. import android.widget.Toast;
11.
12. public class MainActivity extends Activity {
13.     private EditText edittext1,edittext2;
14.     private Button buttonSum;
15.     @Override
16.     protected void onCreate(Bundle savedInstanceState) {
17.         super.onCreate(savedInstanceState);
18.         setContentView(R.layout.activity_main);
19.
20.         addListenerOnButton();
21.
22.     }
23.     public void addListenerOnButton(){
24.         edittext1=(EditText)findViewById(R.id.editText1);
25.         edittext2=(EditText)findViewById(R.id.editText2);
26.         buttonSum=(Button)findViewById(R.id.button1);
27.
28.         buttonSum.setOnClickListener(new OnClickListener(){
29.
30.             @Override
31.             public void onClick(View view) {
32.                 String value1=edittext1.getText().toString();
33.                 String value2=edittext2.getText().toString();
34.                 int a=Integer.parseInt(value1);
35.                 int b=Integer.parseInt(value2);
36.                 int sum=a+b;
37.     Toast.makeText(getApplicationContext(),String.valueOf(sum),Toast.LENGTH_LONG).show();
38.             }
39.
40.         });
41.
42.     }
43.     @Override
44.     public boolean onCreateOptionsMenu(Menu menu) {
45.         // Inflate the menu; this adds items to the action bar if it is present.
46.         getMenuInflater().inflate(R.menu.activity_main, menu);
47.         return true;
48.     }
49.
50. }
```

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

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

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.

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_SHORT);

2. toast.setMargin(50,50);
3. toast.show();

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

Full code of activity class displaying Toast

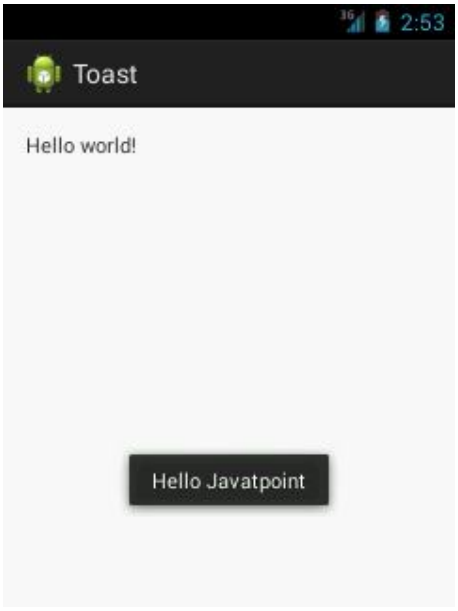
Let's see the code to display the toast.

File: MainActivity.java

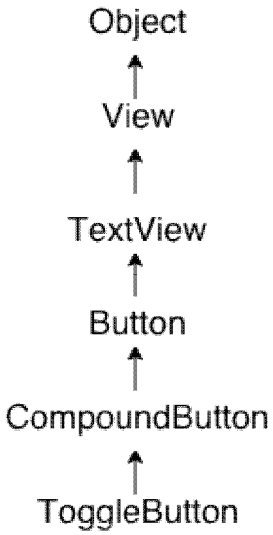
```
1. package com.example.toast;
2. import android.os.Bundle;
3. import android.app.Activity;
4. import android.view.Menu;
5. import android.view.View;
6. import android.widget.Toast;
7.
8. public class MainActivity extends Activity {
9.     @Override
10.     public void onCreate(Bundle savedInstanceState) {
11.         super.onCreate(savedInstanceState);
12.         setContentView(R.layout.activity_main);
13.
14.     //Displaying Toast with Hello Javatpoint message
    Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT).show();
15.     }
16.
17.     @Override
18.     public boolean onCreateOptionsMenu(Menu menu) {
19.         getMenuInflater().inflate(R.menu.activity_main, menu);
```

```
20.             return true;
21.         }
22.
23. }
```

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
CharSequencegetTextOff()	Returns the text when button is not in the checked state.
CharSequencegetTextOn()	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

```
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.    <ToggleButton
8.       android:id="@+id/toggleButton1"
9.       android:layout_width="wrap_content"
10.      android:layout_height="wrap_content"
11.      android:layout_alignParentLeft="true"
12.      android:layout_alignParentTop="true"
13.      android:layout_marginLeft="60dp"
14.      android:layout_marginTop="18dp"
15.      android:text="ToggleButton1"
16.      android:textOff="Off"
17.      android:textOn="On" />
18.
19.   <ToggleButton
20.      android:id="@+id/toggleButton2"
21.      android:layout_width="wrap_content"
22.      android:layout_height="wrap_content"
23.      android:layout_alignBaseline="@+id/toggleButton1"
24.      android:layout_alignBottom="@+id/toggleButton1"
25.      android:layout_marginLeft="44dp"
26.      android:layout_toRightOf="@+id/toggleButton1"
27.      android:text="ToggleButton2"
28.      android:textOff="Off"
29.      android:textOn="On" />
30.
31.   <Button
32.      android:id="@+id/button1"
33.      android:layout_width="wrap_content"
34.      android:layout_height="wrap_content"
35.      android:layout_below="@+id/toggleButton2"
36.      android:layout_marginTop="82dp"
37.      android:layout_toRightOf="@+id/toggleButton1"
38.      android:text="submit" />
39.
40. </RelativeLayout>
```

Activity class

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

File: MainActivity.java

```
1.  package com.example.togglebutton;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.view.View;
7.  import android.view.View.OnClickListener;
8.  import android.widget.Button;
9.  import android.widget.Toast;
10. import android.widget.ToggleButton;
11.
12. public class MainActivity extends Activity {
13.     private ToggleButton toggleButton1, toggleButton2;
14.     private Button buttonSubmit;
15.     @Override
16.     protected void onCreate(Bundle savedInstanceState) {
17.         super.onCreate(savedInstanceState);
18.         setContentView(R.layout.activity_main);
19.
20.         addListenerOnButtonClick();
21.     }
22.     public void addListenerOnButtonClick(){
23.         //Getting the ToggleButton and Button instance from the layout xml file
24.         toggleButton1=(ToggleButton)findViewById(R.id.toggleButton1);
25.         toggleButton2=(ToggleButton)findViewById(R.id.toggleButton2);
26.         buttonSubmit=(Button)findViewById(R.id.button1);
27.
28.         //Performing action on button click
29.         buttonSubmit.setOnClickListener(new OnClickListener(){
30.
31.             @Override
32.             public void onClick(View view) {
33.                 StringBuilder result = new StringBuilder();
34.                     result.append("ToggleButton1 : ").append(toggleButton1.getText());
35.                     result.append("\nToggleButton2 : ").append(toggleButton2.getText());
36.                 //Displaying the message in toast
37.         Toast.makeText(getApplicationContext(), result.toString(),Toast.LENGTH_LONG).show();
```

```
38.         }
39.
40.       });
41.
42.   }
43.   @Override
44.   public boolean onCreateOptionsMenu(Menu menu) {
45.       // Inflate the menu; this adds items to the action bar if it is present.
46.       getMenuInflater().inflate(R.menu.activity_main, menu);
47.       return true;
48.   }
49.
50. }
```

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

```
1.  <RelativeLayout xmlns:android="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.      <CheckBox
8.          android:id="@+id/checkBox1"
9.          android:layout_width="wrap_content"
10.         android:layout_height="wrap_content"
11.         android:layout_alignParentLeft="true"
12.         android:layout_alignParentTop="true"
13.         android:text="Pizza" />
14.
15.     <CheckBox
16.         android:id="@+id/checkBox2"
17.         android:layout_width="wrap_content"
18.         android:layout_height="wrap_content"
19.         android:layout_alignParentTop="true"
20.         android:layout_toRightOf="@+id/checkBox1"
21.         android:text="Coffe" />
```

```
22.
23.    <CheckBox
24.        android:id="@+id/checkBox3"
25.        android:layout_width="wrap_content"
26.        android:layout_height="wrap_content"
27.        android:layout_alignParentTop="true"
28.        android:layout_toRightOf="@+id/checkBox2"
29.        android:text="Burger" />
30.
31.    <Button
32.        android:id="@+id/button1"
33.        android:layout_width="wrap_content"
34.        android:layout_height="wrap_content"
35.        android:layout_below="@+id/checkBox2"
36.        android:layout_marginTop="32dp"
37.        android:layout_toLeftOf="@+id/checkBox3"
38.        android:text="Order" />
39.
40. </RelativeLayout>
```

Activity class

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

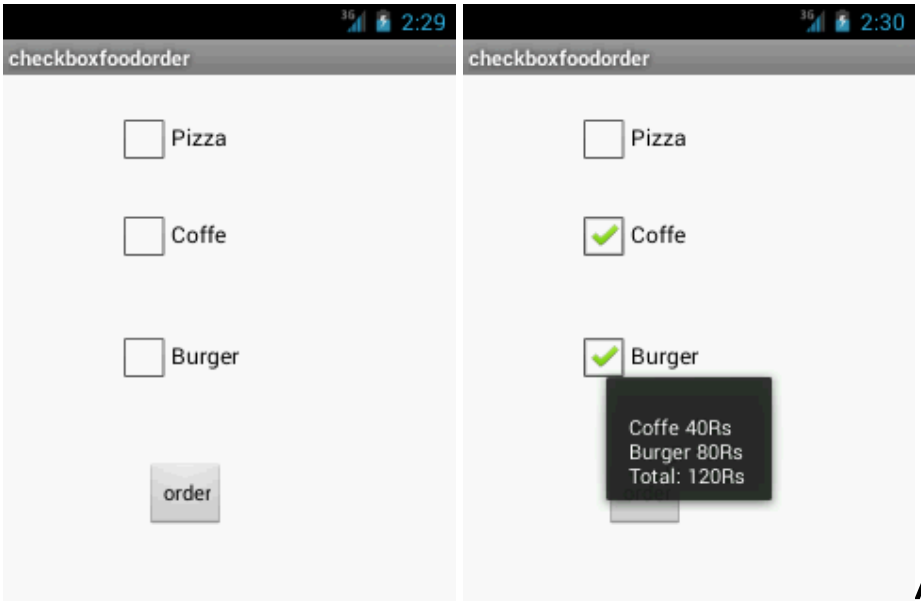
File: MainActivity.java

```
1.  package com.example.checkbox;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.view.View;
7.  import android.view.View.OnClickListener;
8.  import android.widget.*;
9.
10. public class MainActivity extends Activity {
11.     CheckBox pizza,coffe,burger;
12.     Button buttonOrder;
13.     @Override
14.     protected void onCreate(Bundle savedInstanceState) {
15.         super.onCreate(savedInstanceState);
16.         setContentView(R.layout.activity_main);
```

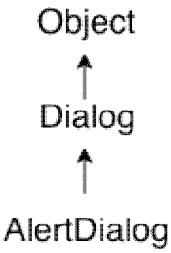
```
17.         addListenerOnButtonClick();
18.     }
19. public void addListenerOnButtonClick(){
20.     //Getting instance of CheckBoxes and Button from the activty_main.xml file
21.     pizza=(CheckBox)findViewById(R.id.checkBox1);
22.     coffe=(CheckBox)findViewById(R.id.checkBox2);
23.     burger=(CheckBox)findViewById(R.id.checkBox3);
24.     buttonOrder=(Button)findViewById(R.id.button1);
25.
26.     //Applying the Listener on the Button click
27.     buttonOrder.setOnClickListener(new OnClickListener(){
28.
29.         @Override
30.         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_LONG).show()
    ;
49.         }
50.
51.     });
52. }
53.     @Override
54.     public boolean onCreateOptionsMenu(Menu menu) {
55.         // Inflate the menu; this adds items to the action bar if it is present.
56.         getMenuInflater().inflate(R.menu.activity_main, menu);
```

```
57.         return true;
58.     }
59.
60. }
```

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.

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

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />

</RelativeLayout>
```

---

strings.xml

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

File: strings.xml

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

    <string name="app_name">alertdialog</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</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

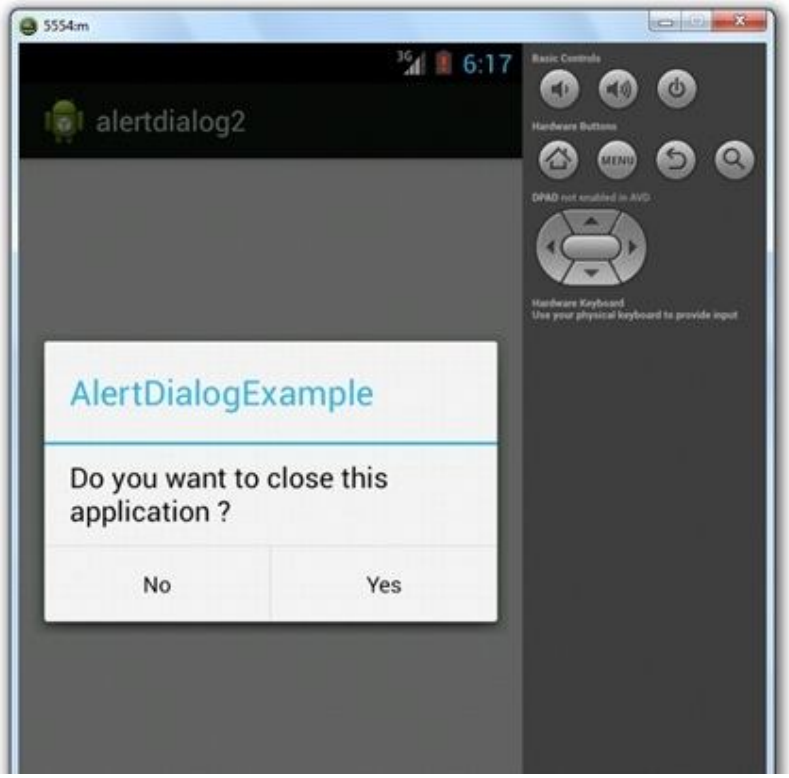
```
1.  package com.example.alertdialog;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.app.AlertDialog;
6.  import android.content.DialogInterface;
7.  import android.view.Menu;
8.
9.  public class MainActivity extends Activity {
10.
11.    @Override
12.    protected void onCreate(Bundle savedInstanceState) {
13.       super.onCreate(savedInstanceState);
14.
15.       AlertDialog.Builder builder = new AlertDialog.Builder(this);
16.       //Uncomment the below code to Set the message and title from the strings.xml file
17.       //builder.setMessage(R.string.dialog_message) .setTitle(R.string.dialog_title);
18.
19.       //Setting message manually and performing action on button click
20.       builder.setMessage("Do you want to close this application ?")
21.          .setCancelable(false)
22.          .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
23.             public void onClick(DialogInterface dialog, int id) {
24.             finish();
25.             }
26.          })
27.          .setNegativeButton("No", new DialogInterface.OnClickListener() {
28.             public void onClick(DialogInterface dialog, int id) {
29.             //  Action for 'NO' Button
30.             dialog.cancel();
31.           }
32.          });
33.
34.       //Creating dialog box
35.       AlertDialog alert = builder.create();
36.       //Setting the title manually
37.       alert.setTitle("AlertDialogExample");
38.       alert.show();
39.       setContentView(R.layout.activity_main);
40.    }
41.
```

```
42.    @Override
43.    public boolean onCreateOptionsMenu(Menu menu) {
44.        // Inflate the menu; this adds items to the action bar if it is present.
45.        getMenuInflater().inflate(R.menu.activity_main, menu);
46.        return true;
47.    }
48.
49. }
```

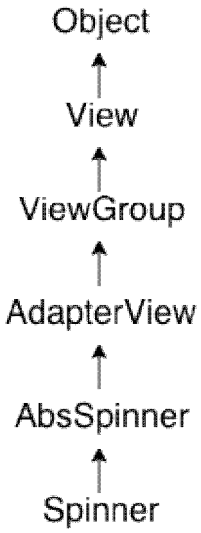
---

download this example

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

```
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.     <Spinner
8.         android:id="@+id/spinner1"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:layout_alignParentTop="true"
12.        android:layout_centerHorizontal="true"
13.        android:layout_marginTop="83dp" />
14.
15.</RelativeLayout>
```

Activity class

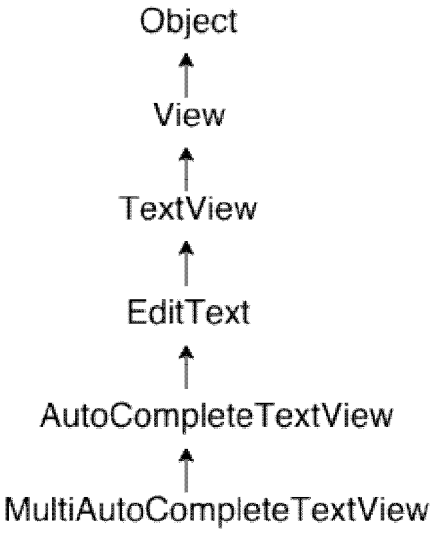
Let's write the code to display item on the spinner and perform event handling.

File: MainActivity.java

```
1. package com.example.spinner;
2. import android.app.Activity;
3. import android.os.Bundle;
4. import android.view.Menu;
5. import android.view.View;
6. import android.widget.AdapterView;
7. import android.widget.ArrayAdapter;
8. import android.widget.Spinner;
9. import android.widget.TextView;
10.import android.widget.Toast;
11.
12.public class MainActivity extends Activity implements
13.AdapterView.OnItemSelectedListener {
14.
15.    String[] country = { "India", "USA", "China", "Japan", "Other",  };
16.
17.    @Override
18.    protected void onCreate(Bundle savedInstanceState) {
19.        super.onCreate(savedInstanceState);
20.        setContentView(R.layout.activity_main);
21.        //Getting the instance of Spinner and applying OnItemSelectedListener on it
```

```
22.       Spinner spin = (Spinner) findViewById(R.id.spinner1);
23.       spin.setOnItemSelectedListener(this);
24.
25.       //Creating the ArrayAdapter instance having the country list
26.       ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,c
    ountry);
27.       aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
28.       //Setting the ArrayAdapter data on the Spinner
29.       spin.setAdapter(aa);
30.    }
31.
32.
33.    //Performing action onItemSelected and onNothing selected
34.    @Override
35.    public void onItemSelected(AdapterView<?> arg0, View arg1, int position,long id)
    {
36.       Toast.makeText(getApplicationContext(),country[position] ,Toast.LENGTH_LONG).
    show();
37.    }
38.
39.    @Override
40.    public void onNothingSelected(AdapterView<?> arg0) {
41.       // TODO Auto-generated method stub
42.
43.    }
44.
45.    @Override
46.    public boolean onCreateOptionsMenu(Menu menu) {
47.       // Inflate the menu; this adds items to the action bar if it is present.
48.       getMenuInflater().inflate(R.menu.activity_main, menu);
49.       return true;
50.    }
51.}
```

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

```
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.     <TextView
```

```
8.      android:id="@+id/textView1"
9.      android:layout_width="wrap_content"
10.     android:layout_height="wrap_content"
11.     android:layout_alignParentLeft="true"
12.     android:layout_alignParentTop="true"
13.     android:layout_marginTop="15dp"
14.     android:text="@string/what_is_your_favourite_programming_language_" />
15.
16.  <AutoCompleteTextView
17.     android:id="@+id/autoCompleteTextView1"
18.     android:layout_width="wrap_content"
19.     android:layout_height="wrap_content"
20.     android:layout_alignParentLeft="true"
21.     android:layout_below="@+id/textView1"
22.     android:layout_marginLeft="36dp"
23.     android:layout_marginTop="17dp"
24.     android:ems="10"
25.     android:text="">
26.
27.     <requestFocus />
28.  </AutoCompleteTextView>
29.
30.</RelativeLayout>
```

Activity class

Let's write the code of AutoCompleteTextView.

File: MainActivity.java

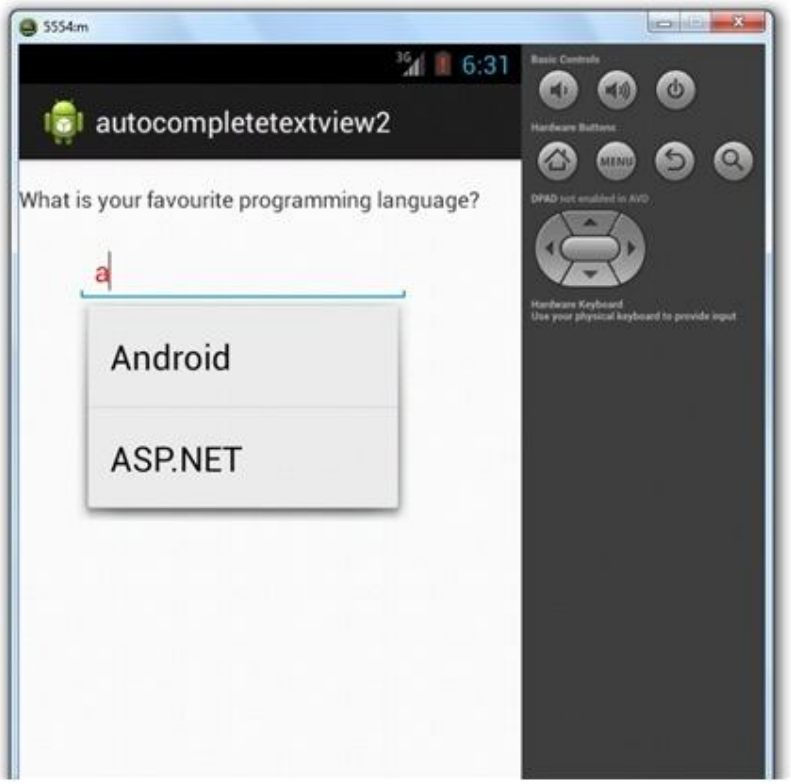
```
1.  package com.example.autocompletetextview;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.graphics.Color;
6.  import android.view.Menu;
7.  import android.widget.ArrayAdapter;
8.  import android.widget.AutoCompleteTextView;
```

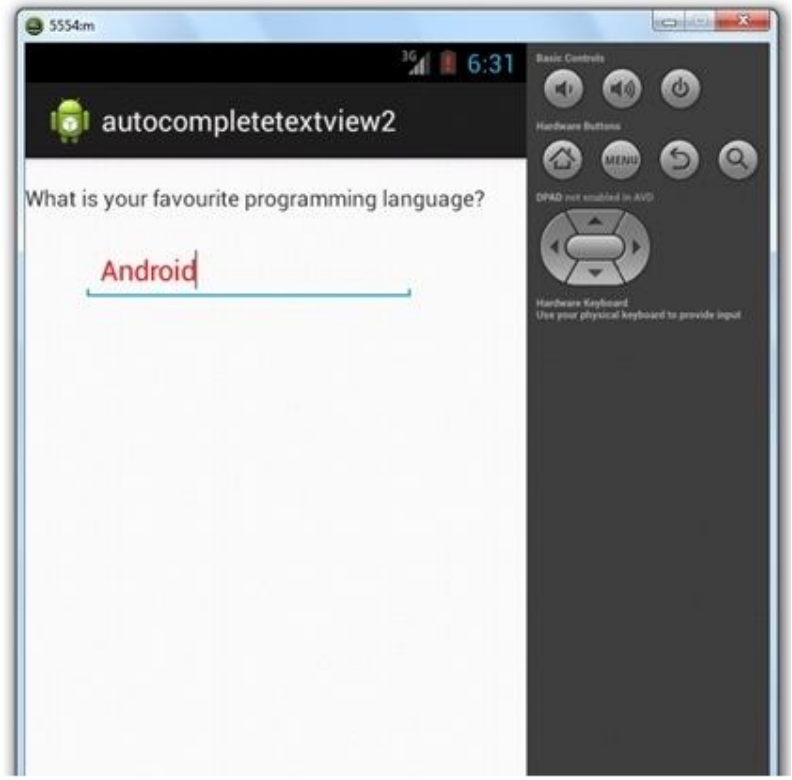
```
9.
10.public class MainActivity extends Activity {
11.    String[] language ={"C","C++","Java",".NET","iPhone","Android","ASP.NET","PHP"};
12.    @Override
13.    protected void onCreate(Bundle savedInstanceState) {
14.        super.onCreate(savedInstanceState);
15.        setContentView(R.layout.activity_main);
16.
17.        //Creating the instance of ArrayAdapter containing list of language names
18.            ArrayAdapter<String> adapter = new ArrayAdapter<String>
19.             (this,android.R.layout.select_dialog_item,language);
20.        //Getting the instance of AutoCompleteTextView
21.            AutoCompleteTextView actv= (AutoCompleteTextView)findViewById(R.id.autoCo
   mpleteTextView1);
22.            actv.setThreshold(1);//will start working from first character
23.            actv.setAdapter(adapter);//setting the adapter data into the AutoCompleteTextVi
   ew
24.            actv.setTextColor(Color.RED);
25.
26.    }
27.
28.    @Override
29.    public boolean onCreateOptionsMenu(Menu menu) {
30.        // Inflate the menu; this adds items to the action bar if it is present.
31.        getMenuInflater().inflate(R.menu.activity_main, menu);
32.        return true;
33.    }
34.
35.}
```

---

download this example

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

```
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.     <RatingBar
8.         android:id="@+id/ratingBar1"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:layout_alignParentTop="true"
12.        android:layout_centerHorizontal="true"
13.        android:layout_marginTop="44dp" />
```

```
14.
15.    <Button
16.        android:id="@+id/button1"
17.        android:layout_width="wrap_content"
18.        android:layout_height="wrap_content"
19.        android:layout_alignLeft="@+id/ratingBar1"
20.        android:layout_below="@+id/ratingBar1"
21.        android:layout_marginLeft="92dp"
22.        android:layout_marginTop="66dp"
23.        android:text="submit" />
24.
25.</RelativeLayout>
```

Activity class

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

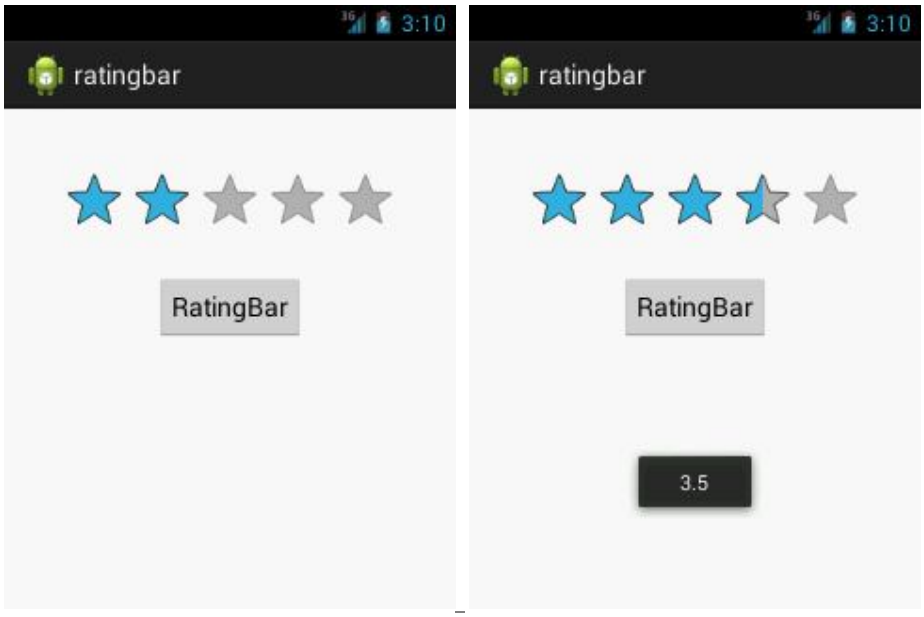
File: MainActivity.java

```
1.  package com.example.rating;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.view.View;
7.  import android.view.View.OnClickListener;
8.  import android.widget.Button;
9.  import android.widget.RatingBar;
10. import android.widget.Toast;
11.
12. public class MainActivity extends Activity {
13.     RatingBar ratingbar1;
14.     Button button;
15.     @Override
16.     protected void onCreate(Bundle savedInstanceState) {
17.         super.onCreate(savedInstanceState);
18.         setContentView(R.layout.activity_main);
19.         addListenerOnButtonClick();
```

```
20.    }
21.
22.    public void addListenerOnButtonClick(){
23.        ratingbar1=(RatingBar)findViewById(R.id.ratingBar1);
24.        button=(Button)findViewById(R.id.button1);
25.        //Performing action on Button Click
26.        button.setOnClickListener(new OnClickListener(){
27.
28.            @Override
29.            public void onClick(View arg0) {
30.                //Getting the rating and displaying it on the toast
31.                String rating=String.valueOf(ratingbar1.getRating());
32.                Toast.makeText(getApplicationContext(), rating, Toast.LENGTH_LONG).show
    ();
33.            }
34.
35.        });
36.    }
37.    @Override
38.    public boolean onCreateOptionsMenu(Menu menu) {
39.        // Inflate the menu; this adds items to the action bar if it is present.
40.        getMenuInflater().inflate(R.menu.activity_main, menu);
41.        return true;
42.    }
43.
44.}
```

download this example

Output:



Android WebView Example

Android WebView is used to display web page in android. The web page can be loaded from same application or URL. It is used to display online content in android activity.

Android WebView uses webkit engine to display web page.

The android.webkit.WebView is the subclass of AbsoluteLayout class.

The **loadUrl()** and **loadData()** methods of Android WebView class are used to load and display web page.

Let's see the simple code to **display javatpoint.com web page** using web view.

```
1. WebView mywebview = (WebView) findViewById(R.id.webView1);
2. mywebview.loadUrl("http://www.javatpoint.com/");
```

Let's see the simple code to **display HTML web page** using web view. In this case, html file must be located inside the asset directory.

```
1. WebView mywebview = (WebView) findViewById(R.id.webView1);
2. mywebview.loadUrl("file:///android_asset/myresource.html");
```

Let's see another code to display **HTML code of a string**.

```
1. String data = "<html><body><h1>Hello, Javatpoint!</h1></body></html>";
2. mywebview.loadData(data, "text/html", "UTF-8");
```

Android WebView Example

Let's see a simple example of android webview.

activity_main.xml

File: activity_main.xml

```
1. <RelativeLayout xmlns:android="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.     <WebView
8.         android:id="@+id/webView1"
9.         android:layout_width="match_parent"
10.        android:layout_height="match_parent"
11.        android:layout_alignParentTop="true"
12.        android:layout_centerHorizontal="true"
13.        android:layout_marginTop="42dp" />
14.
15. </RelativeLayout>
```

Activity class

File: MainActivity.java

```
1. package com.example.webview;
2.
3. import android.os.Bundle;
4. import android.app.Activity;
5. import android.view.Menu;
6. import android.webkit.WebView;
7.
8. public class MainActivity extends Activity {
9.
```

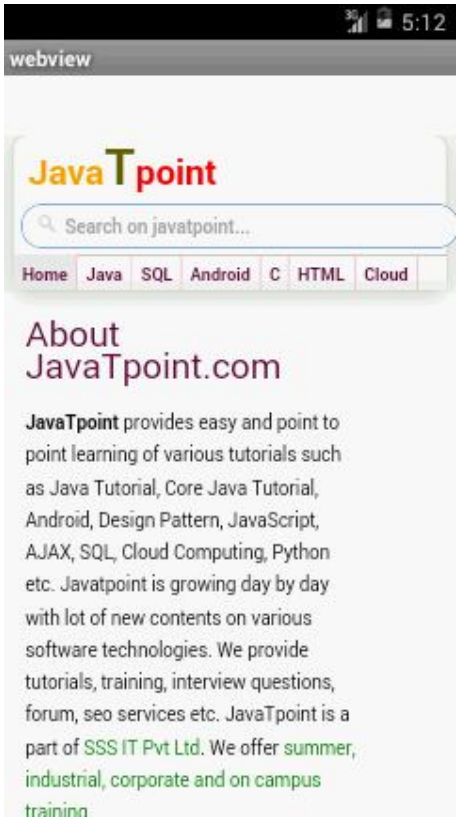
```
10.    @Override
11.    protected void onCreate(Bundle savedInstanceState) {
12.        super.onCreate(savedInstanceState);
13.        setContentView(R.layout.activity_main);
14.
15.        WebView mywebview = (WebView) findViewById(R.id.webView1);
16.         //mywebview.loadUrl("http://www.javatpoint.com/");
17.
18.        /*String data = "<html><body><h1>Hello, Javatpoint!</h1></body></html>";
19.        mywebview.loadData(data, "text/html", "UTF-8"); */
20.
21.        mywebview.loadUrl("file:///android_asset/myresource.html");
22.    }
23.
24.    @Override
25.    public boolean onCreateOptionsMenu(Menu menu) {
26.        // Inflate the menu; this adds items to the action bar if it is present.
27.        getMenuInflater().inflate(R.menu.activity_main, menu);
28.        return true;
29.    }
30.
31.}
```

download this android example

Output:

Let's see the output if you load the HTML page.

Let's see the output if you load the javatpoint.com web page.



Android SeekBar Example

Android SeekBar is a kind of ProgressBar with draggable thumb. The end user can drag the thum left and right to move the progress of song, file download etc.

The SeekBar.OnSeekBarChangeListener interface provides methods to perform even handling for seek bar.

Android SeekBar and RatingBar classes are the sub classes of AbsSeekBar.

Android SeekBar Example

activity_main.xml

Drag the seek bar from the pallete, now activity_main.xml will look like this:

File: activity_main.xml

```
1. <RelativeLayout xmlns:android="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.     android:paddingBottom="@dimen/activity_vertical_margin"
6.     android:paddingLeft="@dimen/activity_horizontal_margin"
7.     android:paddingRight="@dimen/activity_horizontal_margin"
8.     android:paddingTop="@dimen/activity_vertical_margin"
9.     tools:context=".MainActivity" >
10.
11.    <SeekBar
12.        android:id="@+id/seekBar1"
13.        android:layout_width="match_parent"
```

```
14.       android:layout_height="wrap_content"
15.       android:layout_alignParentTop="true"
16.       android:layout_centerHorizontal="true"
17.       android:layout_marginTop="39dp" />
18.
19. </RelativeLayout>
```

Activity class

Let's see the Activity class displaying seek bar and performing event handling.

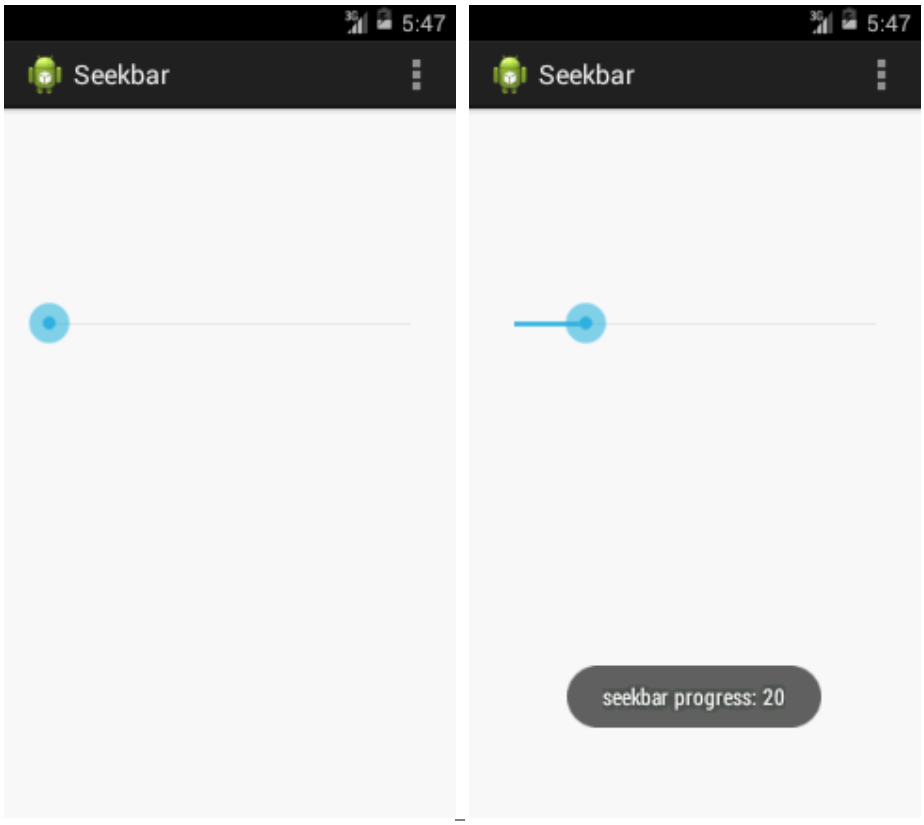
File: MainActivity.java

```
1.  package com.example.seekbar;
2.  import android.os.Bundle;
3.  import android.app.Activity;
4.  import android.view.Menu;
5.  import android.widget.SeekBar;
6.  import android.widget.SeekBar.OnSeekBarChangeListener;
7.  import android.widget.Toast;
8.  public class MainActivity extends Activity implements OnSeekBarChangeListener{
9.      SeekBar seekBar1;
10.     @Override
11.     protected void onCreate(Bundle savedInstanceState) {
12.         super.onCreate(savedInstanceState);
13.         setContentView(R.layout.activity_main);
14.
15.         seekBar1=(SeekBar)findViewById(R.id.seekBar1);
16.         seekBar1.setOnSeekBarChangeListener(this);
17.     }
18.     @Override
19.     public void onProgressChanged(SeekBar seekBar, int progress,
20.             boolean fromUser) {
21.         Toast.makeText(getApplicationContext(),"seekbar progress: "+progress, Toast.LEN
    GTH_SHORT).show();
22.     }
23.     @Override
24.     public void onStartTrackingTouch(SeekBar seekBar) {
```

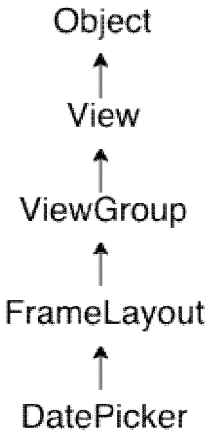
```
25.       Toast.makeText(getApplicationContext(),"seekbar touch started!", Toast.LENGTH
    SHORT).show();
26.    }
27.    @Override
28.    public void onStopTrackingTouch(SeekBar seekBar) {
29.       Toast.makeText(getApplicationContext(),"seekbar touch stopped!", Toast.LENGTH_
    SHORT).show();
30.    }
31.    @Override
32.    public boolean onCreateOptionsMenu(Menu menu) {
33.       // Inflate the menu; this adds items to the action bar if it is present.
34.       getMenuInflater().inflate(R.menu.main, menu);
35.       return true;
36.    }
37.}
```

download this android example

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

```
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.     <TextView
8.         android:id="@+id/textView1"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:layout_alignParentLeft="true"
12.        android:layout_alignParentTop="true"
13.        android:layout_marginLeft="50dp"
14.        android:layout_marginTop="36dp"
```

```
15.     android:text="Current Date:" />
16.
17.  <Button
18.     android:id="@+id/button1"
19.     android:layout_width="wrap_content"
20.     android:layout_height="wrap_content"
21.     android:layout_alignParentBottom="true"
22.     android:layout_centerHorizontal="true"
23.     android:layout_marginBottom="140dp"
24.     android:text="Change Date" />
25.
26.  <DatePicker
27.     android:id="@+id/datePicker1"
28.     android:layout_width="wrap_content"
29.     android:layout_height="wrap_content"
30.     android:layout_above="@+id/button1"
31.     android:layout_centerHorizontal="true"
32.     android:layout_marginBottom="30dp" />
33.
34.</RelativeLayout>
```

Activity class

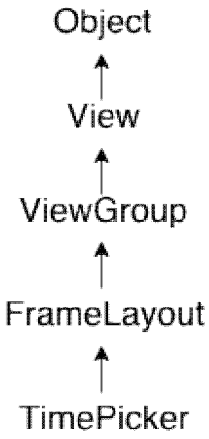
File: MainActivity.java

```
1.  package com.example.datepicker2;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.view.View;
7.  import android.view.View.OnClickListener;
8.  import android.widget.Button;
9.  import android.widget.DatePicker;
10.import android.widget.TextView;
11.import android.widget.Toast;
12.
```

```
13.public class MainActivity extends Activity {
14.    DatePicker picker;
15.    Button displayDate;
16.    TextView textview1;
17.    @Override
18.    protected void onCreate(Bundle savedInstanceState) {
19.        super.onCreate(savedInstanceState);
20.        setContentView(R.layout.activity_main);
21.
22.        textview1=(TextView)findViewById(R.id.textView1);
23.        picker=(DatePicker)findViewById(R.id.datePicker1);
24.        displayDate=(Button)findViewById(R.id.button1);
25.
26.        textview1.setText(getCurrentDate());
27.
28.        displayDate.setOnClickListener(new OnClickListener(){
29.            @Override
30.            public void onClick(View view) {
31.                textview1.setText(getCurrentDate());
32.            }
33.
34.        });
35.    }
36.    public String getCurrentDate(){
37.        StringBuilder builder=new StringBuilder();
38.        builder.append("Current Date: ");
39.        builder.append((picker.getMonth() + 1)+"/");//month is 0 based
40.        builder.append(picker.getDayOfMonth()+"/");
41.        builder.append(picker.getYear());
42.        return builder.toString();
43.    }
44.    @Override
45.    public boolean onCreateOptionsMenu(Menu menu) {
46.        // Inflate the menu; this adds items to the action bar if it is present.
47.        getMenuInflater().inflate(R.menu.activity_main, menu);
48.        return true;
49.    }
```

```
50.
51.}
```

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

```
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.     <TimePicker
8.         android:id="@+id/timePicker1"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:layout_alignParentTop="true"
12.        android:layout_centerHorizontal="true"
13.        android:layout_marginTop="86dp" />
```

```
14.
15.    <TextView
16.        android:id="@+id/textView1"
17.        android:layout_width="wrap_content"
18.        android:layout_height="wrap_content"
19.        android:layout_alignLeft="@+id/timePicker1"
20.        android:layout_alignParentTop="true"
21.        android:layout_marginTop="17dp"
22.        android:text="Current Time:" />
23.
24.    <Button
25.        android:id="@+id/button1"
26.        android:layout_width="wrap_content"
27.        android:layout_height="wrap_content"
28.        android:layout_alignLeft="@+id/timePicker1"
29.        android:layout_below="@+id/timePicker1"
30.        android:layout_marginLeft="37dp"
31.        android:layout_marginTop="55dp"
32.        android:text="Change Time" />
33.
34.</RelativeLayout>
```

Activity class

File: MainActivity.java

```
1.  package com.example.timepicker1;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.view.View;
7.  import android.view.View.OnClickListener;
8.  import android.widget.Button;
9.  import android.widget.TextView;
10. import android.widget.TimePicker;
11. import android.widget.Toast;
12.
```

```
13.public class MainActivity extends Activity {
14.    TextView textview1;
15.    TimePicker timepicker1;
16.    Button changetime;
17.
18.    @Override
19.    protected void onCreate(Bundle savedInstanceState) {
20.        super.onCreate(savedInstanceState);
21.        setContentView(R.layout.activity_main);
22.
23.        textview1=(TextView)findViewById(R.id.textView1);
24.        timepicker1=(TimePicker)findViewById(R.id.timePicker1);
25.        //Uncomment the below line of code for 24 hour view
26.        timepicker1.setIs24HourView(true);
27.        changetime=(Button)findViewById(R.id.button1);
28.
29.        textview1.setText(getCurrentTime());
30.
31.        changetime.setOnClickListener(new OnClickListener(){
32.            @Override
33.            public void onClick(View view) {
34.                textview1.setText(getCurrentTime());
35.            }
36.        });
37.
38.    }
39.
40.    public String getCurrentTime(){
41.        String currentTime="Current Time: "+timepicker1.getCurrentHour()+":"+timepick
    er1.getCurrentMinute();
42.        return currentTime;
43.    }
44.    @Override
45.    public boolean onCreateOptionsMenu(Menu menu) {
46.        // Inflate the menu; this adds items to the action bar if it is present.
47.        getMenuInflater().inflate(R.menu.activity_main, menu);
48.        return true;
```

```
49.    }
50.
51.}
```

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.

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. It represents the timing of the current device.

activity_main.xml

Now, drag the analog and digital clocks, now the 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.     <AnalogClock
8.         android:id="@+id/analogClock1"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:layout_alignParentTop="true"
12.        android:layout_centerHorizontal="true"
13.        android:layout_marginTop="22dp" />
```

```
14.
15.    <DigitalClock
16.        android:id="@+id/digitalClock1"
17.        android:layout_width="wrap_content"
18.        android:layout_height="wrap_content"
19.        android:layout_below="@+id/analogClock1"
20.        android:layout_centerHorizontal="true"
21.        android:layout_marginTop="81dp"
22.        android:text="DigitalClock" />
23.
24.</RelativeLayout>
```

Activity class

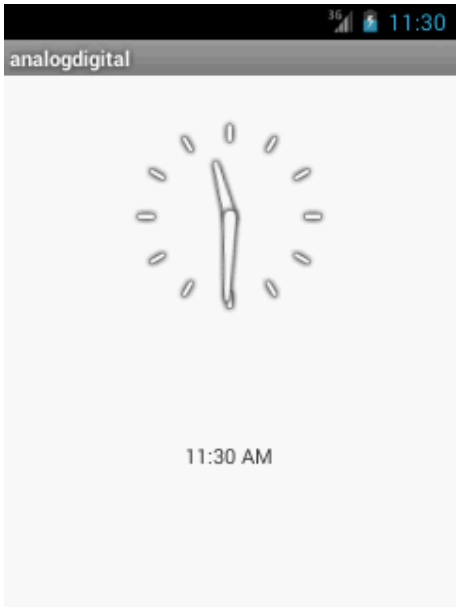
We have not write any code here.

File: MainActivity.java

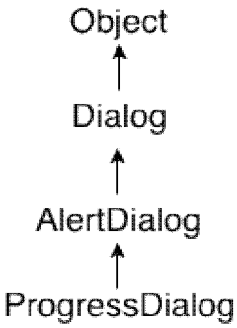
```
1.  package com.example.analogdigital;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.
7.  public class MainActivity extends Activity {
8.
9.      @Override
10.     protected void onCreate(Bundle savedInstanceState) {
11.         super.onCreate(savedInstanceState);
12.         setContentView(R.layout.activity_main);
13.     }
14.
15.     @Override
16.     public boolean onCreateOptionsMenu(Menu menu) {
17.         // Inflate the menu; this adds items to the action bar if it is present.
18.         getMenuInflater().inflate(R.menu.activity_main, menu);
19.         return true;
20.     }
```

```
21.}
```

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"
9.         android:layout_width="wrap_content"
10.        android:layout_height="wrap_content"
11.        android:layout_alignParentTop="true"
12.        android:layout_centerHorizontal="true"
13.        android:layout_marginTop="116dp"
14.        android:text="download file" />
15.
16. </RelativeLayout>
```

Activity class

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

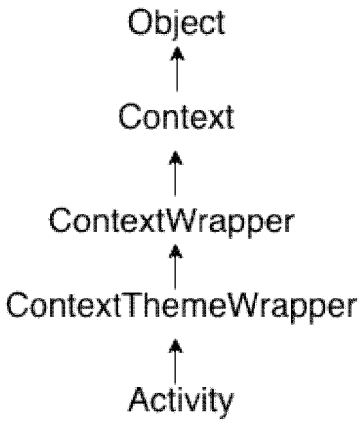
File: MainActivity.java

```
1. package com.example.progressbar1;
2. import android.app.Activity;
3. import android.app.ProgressDialog;
4. import android.os.Bundle;
5. import android.os.Handler;
6. import android.widget.Button;
7. import android.view.Menu;
8. import android.view.View;
9. import android.view.View.OnClickListener;
10.public class MainActivity extends Activity {
11.    Button btnStartProgress;
12.    ProgressDialog progressBar;
13.    private int progressBarStatus = 0;
14.    private Handler progressBarHandler = new Handler();
15.        private long fileSize = 0;
16.    @Override
17.    protected void onCreate(Bundle savedInstanceState) {
18.        super.onCreate(savedInstanceState);
19.        setContentView(R.layout.activity_main);
20.        addListenerOnButtonClick();
21.    }
22.    public void addListenerOnButtonClick() {
23.        btnStartProgress = (Button) findViewById(R.id.button1);
24.        btnStartProgress.setOnClickListener(new OnClickListener(){
25.
26.            @Override
27.            public void onClick(View v) {
28.             // creating progress bar dialog
29.             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 {Thread.sleep(1000);} catch (InterruptedException e) {e.printStackTra
    ce();}
46.               // Updating the progress bar
47.               progressBarHandler.post(new Runnable() {
48.                 public void run() {
49.                   progressBar.setProgress(progressBarStatus);
50.                 }
51.               });
52.             }
53.             // performing operation if file is downloaded,
54.             if (progressBarStatus >= 100) {
55.               // sleeping for 1 second after operation completed
56.               try {Thread.sleep(1000);} catch (InterruptedException e) {e.printStackTr
    ace();}
57.               // close the progress bar dialog
58.               progressBar.dismiss();
59.             }
60.           }
61.         }).start();
62.         }//end of onClick method
63.       });
64.     }
65.   // checking how much file is downloaded and updating the filesize
66.   public int doOperation() {
67.     //The range of ProgressDialog starts from 0 to 10000
68.     while (fileSize <= 10000) {
69.       fileSize++;
70.       if (fileSize == 1000) {
71.         return 10;
```

```
72.         } else if (fileSize == 2000) {
73.             return 20;
74.         } else if (fileSize == 3000) {
75.             return 30;
76.         } else if (fileSize == 4000) {
77.         return 40;//you can add more else if
78.         } else{
79.             return 100;
80.         }
81.       }//end of while
82.       return 100;
83.    }//end of doOperation
84.
85.    @Override
86.    public boolean onCreateOptionsMenu(Menu menu) {
87.       // Inflate the menu; this adds items to the action bar if it is present.
88.       getMenuInflater().inflate(R.menu.main, menu);
89.       return true;
90.    }
91.}
```

Android Activity Lifecycle



Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

An activity is the single screen in android. It is like window or frame of Java.

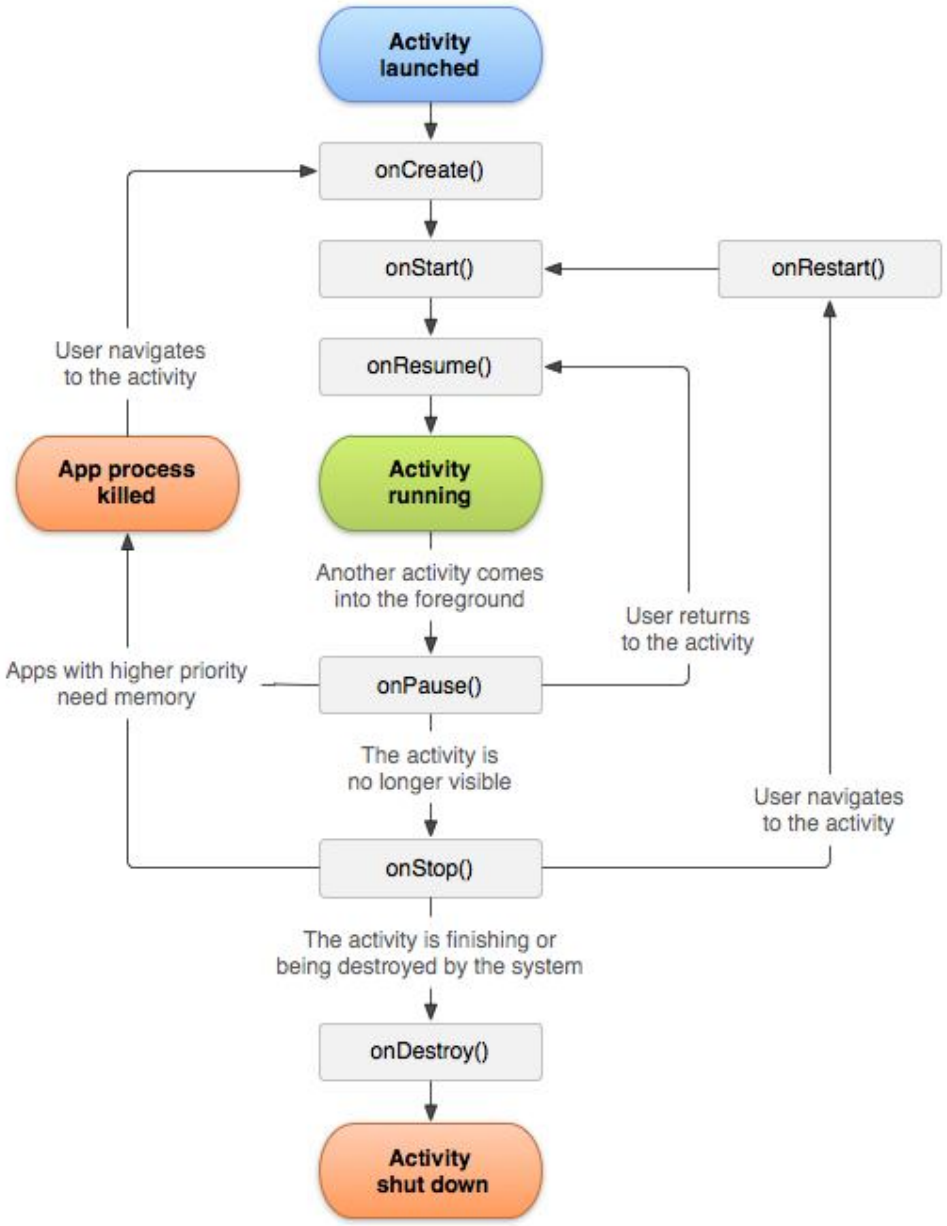
By the help of activity, you can place all your UI components or widgets in a single screen.

The 7 lifecycle method of Activity describes how activity will behave at different states.

Android Activity Lifecycle methods

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

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



Android Activity Lifecycle Example

It provides the details about the invocation of life cycle methods of activity. In this example, we are displaying the content on the logcat.

File: MainActivity.java

```
1. package com.example.activitylifecycle;
2. import android.os.Bundle;
3. import android.app.Activity;
4. import android.util.Log;
5. import android.view.Menu;
```

```
6.  public class MainActivity extends Activity {
7.      @Override
8.      protected void onCreate(Bundle savedInstanceState) {
9.          super.onCreate(savedInstanceState);
10.         setContentView(R.layout.activity_main);
11.         Log.d("lifecycle","onCreate invoked");
12.     }
13.     @Override
14.     protected void onStart() {
15.         super.onStart();
16.         Log.d("lifecycle","onStart invoked");
17.     }
18.     @Override
19.     protected void onResume() {
20.         super.onResume();
21.         Log.d("lifecycle","onResume invoked");
22.     }
23.     @Override
24.     protected void onPause() {
25.         super.onPause();
26.         Log.d("lifecycle","onPause invoked");
27.     }
28.     @Override
29.     protected void onStop() {
30.         super.onStop();
31.         Log.d("lifecycle","onStop invoked");
32.     }
33.     @Override
34.     protected void onRestart() {
35.         super.onRestart();
36.         Log.d("lifecycle","onRestart invoked");
37.     }
38.     @Override
39.     protected void onDestroy() {
40.         super.onDestroy();
41.         Log.d("lifecycle","onDestroy invoked");
42.     }
```

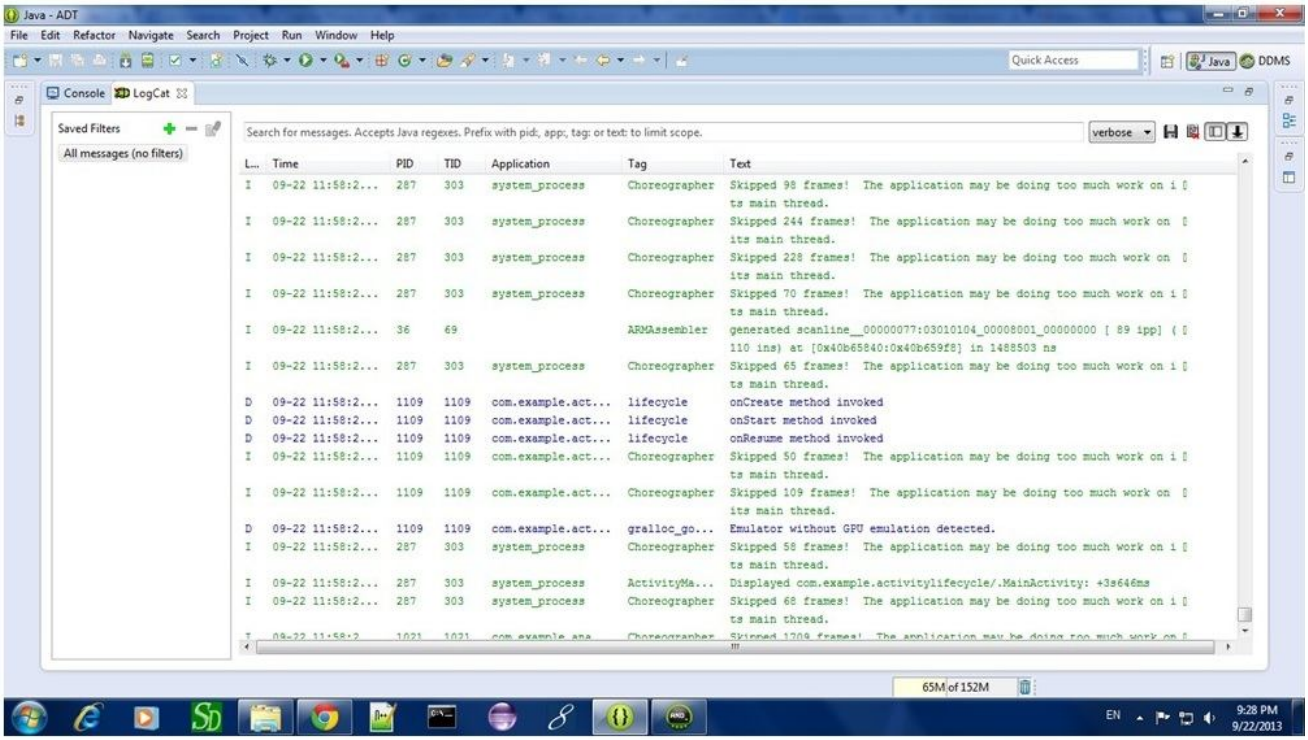
```
43.}
```

download this example

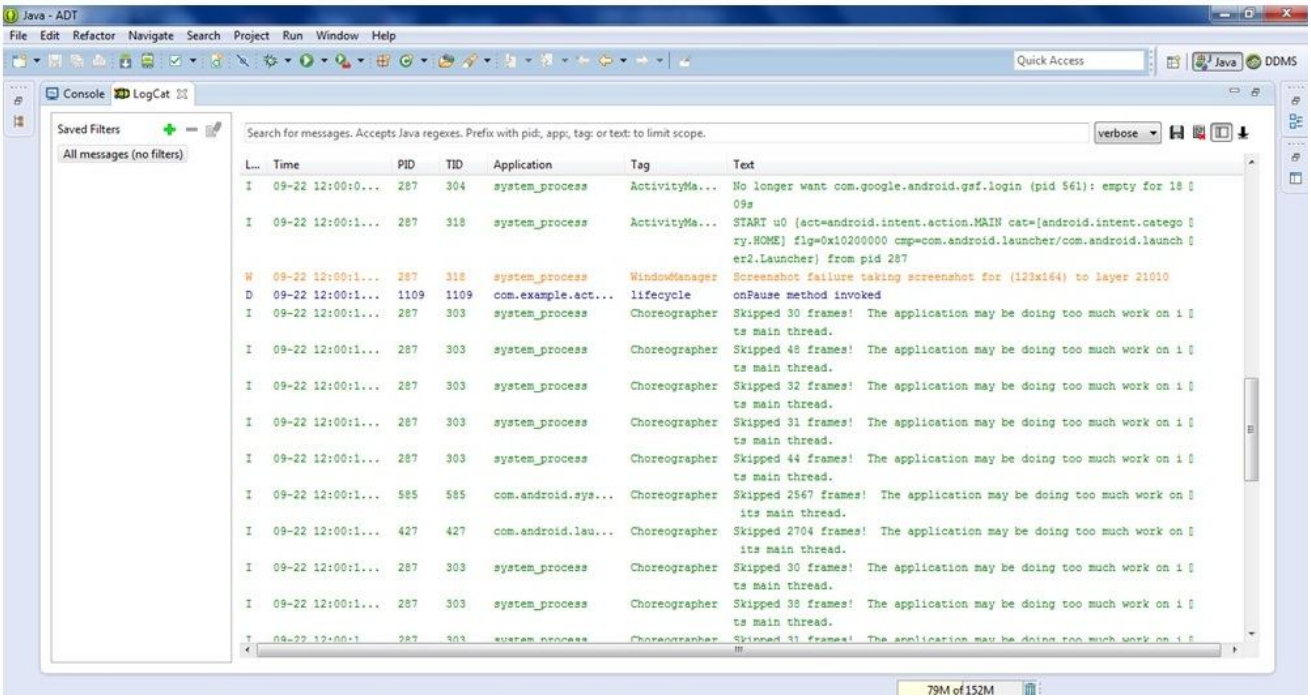
Output:

You will not see any output on the emulator or device. You need to open logcat.

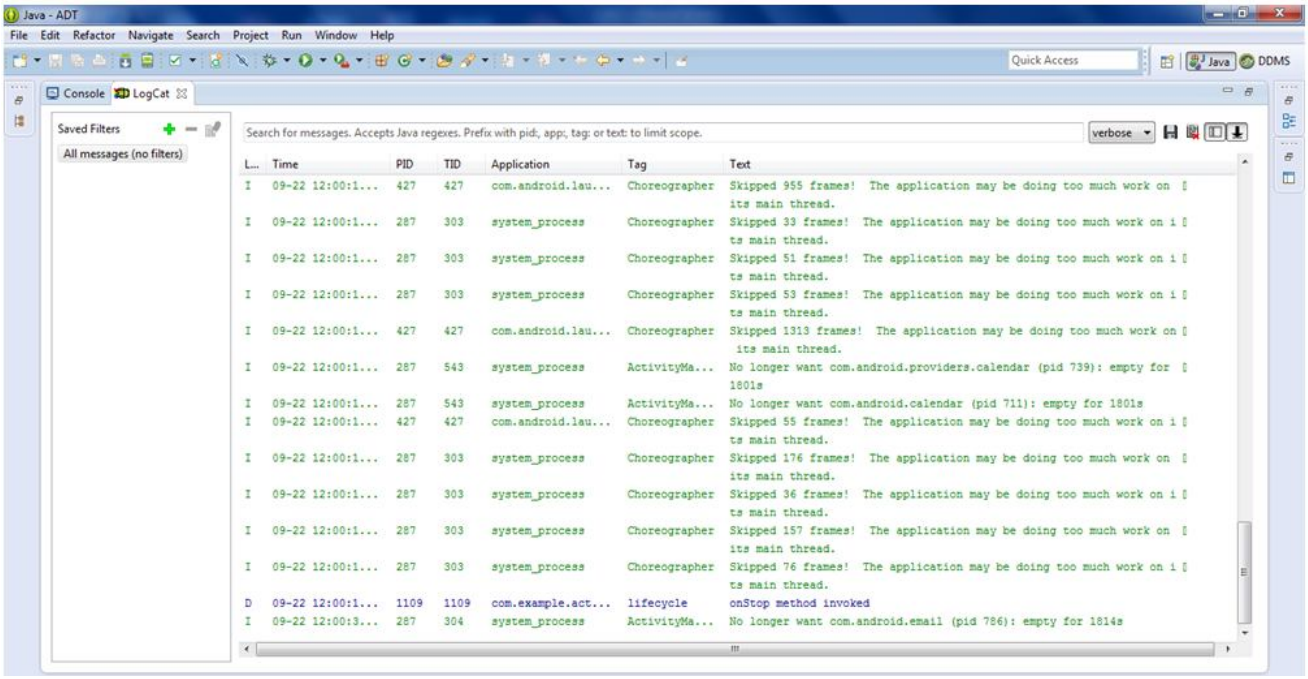
Now see on the logcat: onCreate, onStart and onResume methods are invoked.



Now click on the HOME Button. You will see onPause method is invoked.



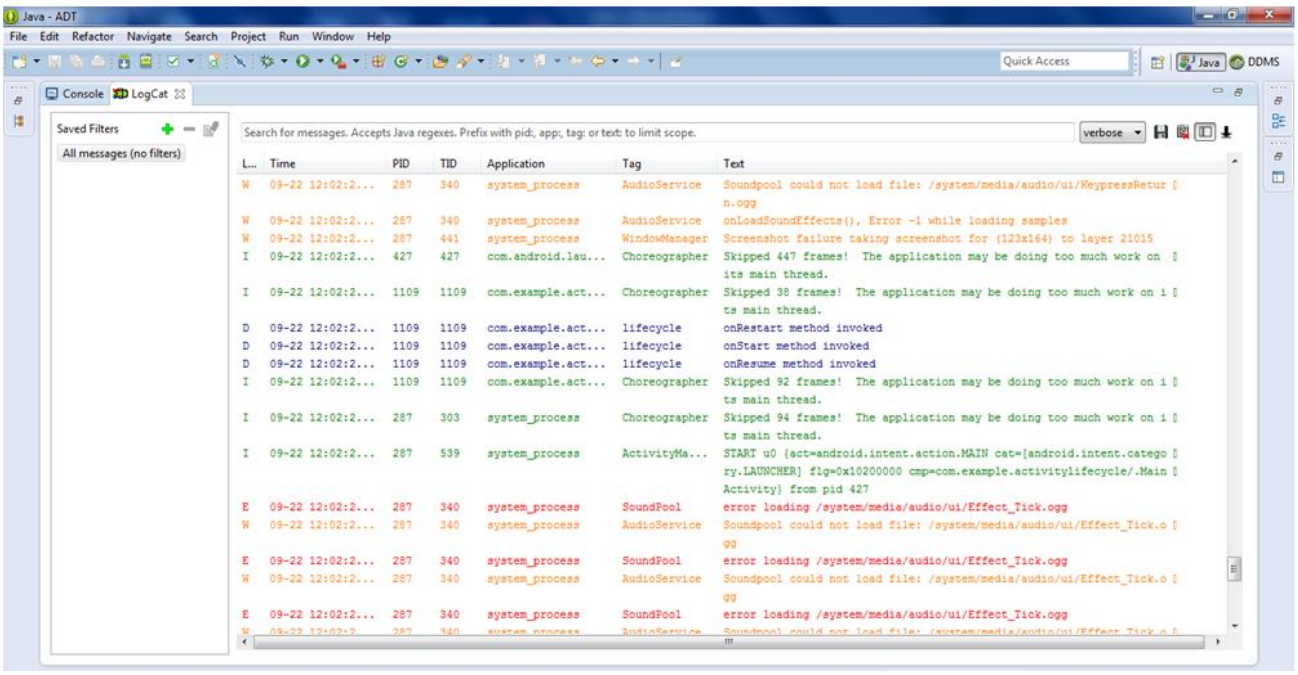
After a while, you will see onStop method is invoked.



Now see on the emulator. It is on the home. Now click on the center button to launch the app again.

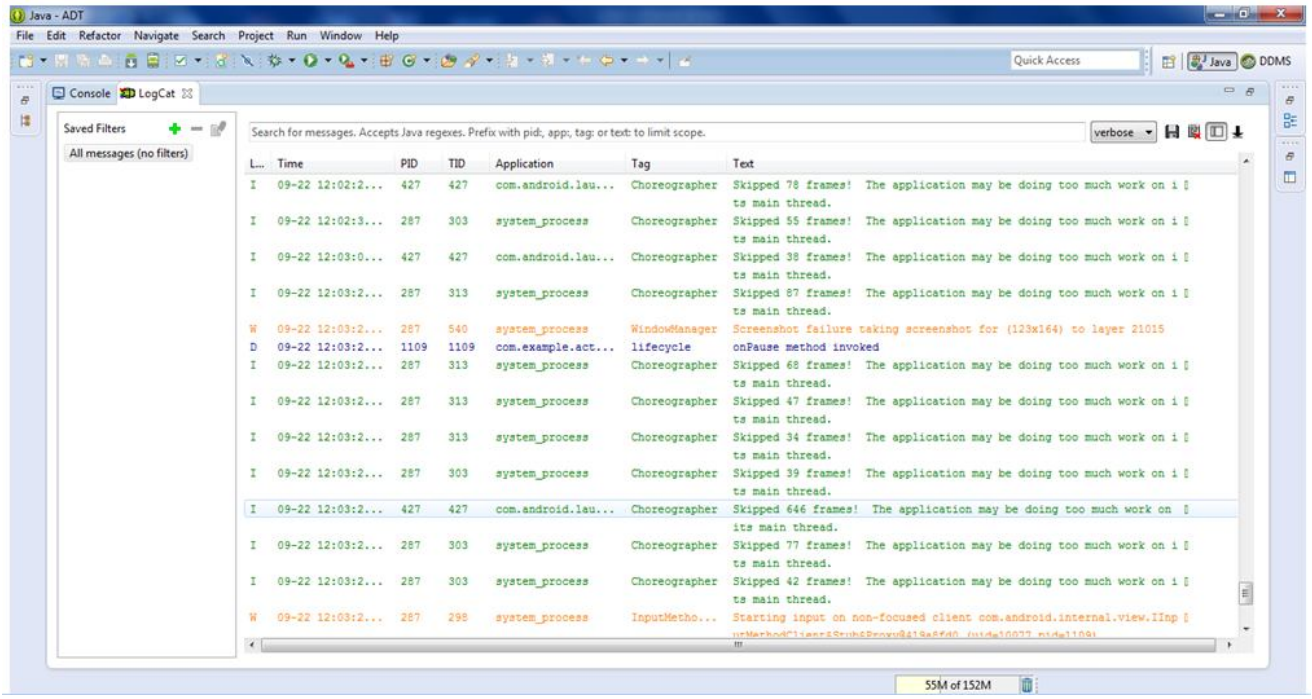
Now click on the lifecycleactivity icon.

Now see on the logcat: onRestart, onStart and onResume methods are invoked.

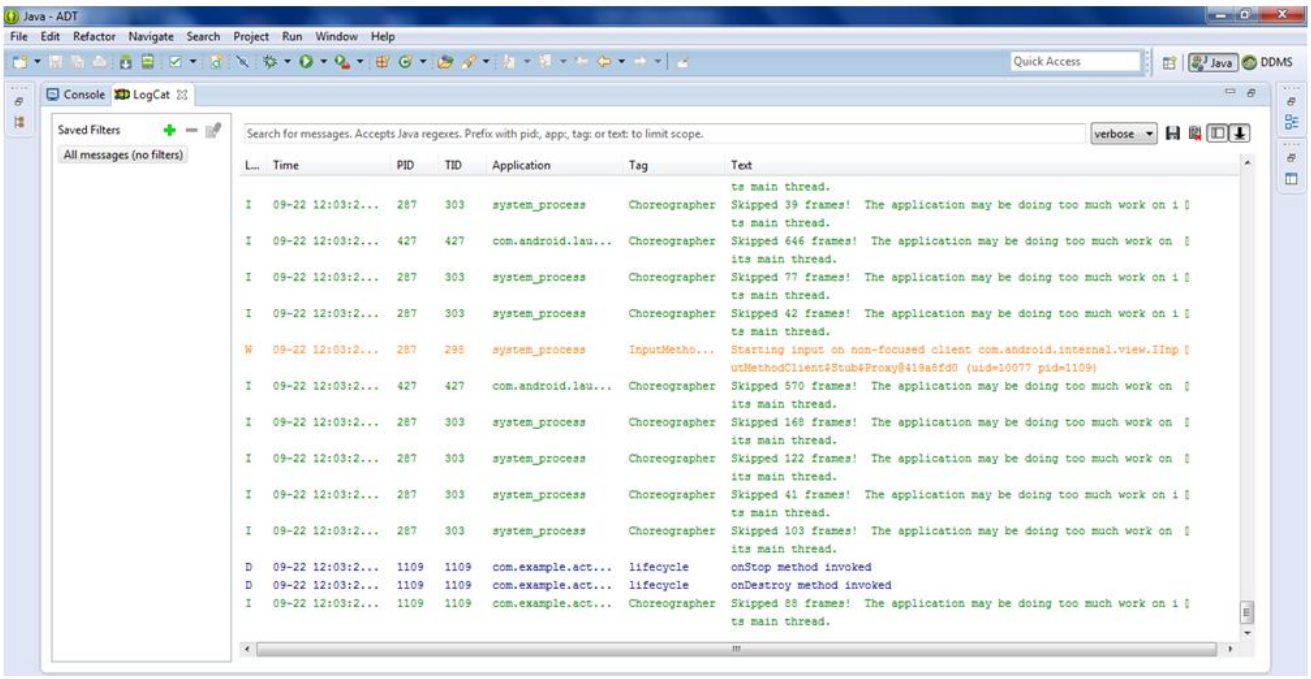


If you see the emulator, application is started again.

Now click on the back button. Now you will see onPause methods is invoked.



After a while, you will see onStop and onDestroy methods are invoked.



The onCreate() and onDestroy() methods are called only once throughout the activity lifecycle.

next →← prev

Android Option Menu Example

Android Option Menus are the primary menus of android. They can be used for settings, search, delete item etc.

Here, we are going to see two examples of option menus. First, the simple option menus and second, options menus with images.

Here, we are inflating the menu by calling the **inflate()** method of **MenuInflater** class. To perform event handling on menu items, you need to override **onOptionsItemSelected()** method of Activity class.

Android Option Menu Example

Let's see how to create menu in android. Let's see the simple option menu example that contains three menu items.

activity_main.xml

We have only one textview in this file.

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.     android:paddingBottom="@dimen/activity_vertical_margin"
6.     android:paddingLeft="@dimen/activity_horizontal_margin"
7.     android:paddingRight="@dimen/activity_horizontal_margin"
8.     android:paddingTop="@dimen/activity_vertical_margin"
9.     tools:context=".MainActivity" >
10.
11.    <TextView
12.        android:layout_width="wrap_content"
13.        android:layout_height="wrap_content"
14.        android:text="@string/hello_world" />
15.
16. </RelativeLayout>
```

menu_main.xml

It contains three items as show below. It is created automatically inside the res/menu directory.

File: menu_main.xml

```
1. <menu xmlns:androclass="http://schemas.android.com/apk/res/android" >
2.     <item  android:id="@+id/item1"
3.         android:title="Item 1" />
4.     <item  android:id="@+id/item2"
5.         android:title="Item 2" />
6.     <item  android:id="@+id/item3"
7.         android:title="Item 3" />
8. </menu>
```

Activity class

This class displays the content of menu.xml file and performs event handling on clicking the menu items.

File: MainActivity.java

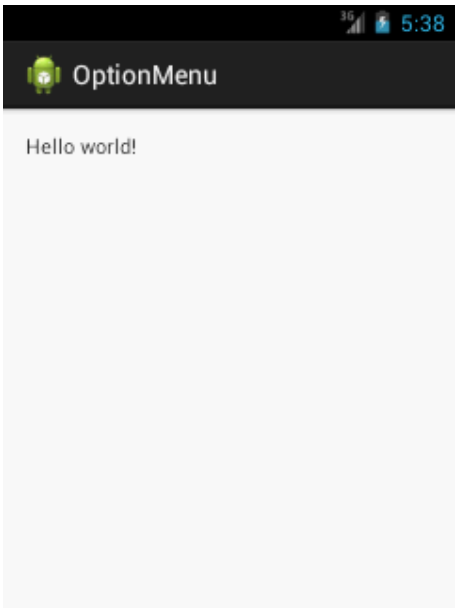
```
1. package com.javatpoint.optionmenu;
2. import android.os.Bundle;
3. import android.app.Activity;
4. import android.view.Menu;
5. import android.view.MenuItem;
6. import android.widget.Toast;
7. public class MainActivity extends Activity {
8.     @Override
9.     protected void onCreate(Bundle savedInstanceState) {
10.         super.onCreate(savedInstanceState);
11.         setContentView(R.layout.activity_main);
12.     }
13.     @Override
14.     public boolean onCreateOptionsMenu(Menu menu) {
15.         // Inflate the menu; this adds items to the action bar if it is present.
16.         getMenuInflater().inflate(R.menu.main, menu);//Menu Resource, Menu
17.         return true;
18.     }
19.     @Override
20.     public boolean onOptionsItemSelected(MenuItem item) {
21.         switch (item.getItemId()) {
22.             case R.id.item1:
23.                 Toast.makeText(getApplicationContext(),"Item 1 Selected",Toast.LENGTH_L
    ONG).show();
24.             return true;
25.             case R.id.item2:
26.                 Toast.makeText(getApplicationContext(),"Item 2 Selected",Toast.LENGTH
    _LONG).show();
27.                 return true;
28.             case R.id.item3:
```

```
29.             Toast.makeText(getApplicationContext(),"Item 3 Selected",Toast.LENGTH
    _LONG).show();
30.         return true;
31.         default:
32.             return super.onOptionsItemSelected(item);
33.     }
34.  }
35.}
```

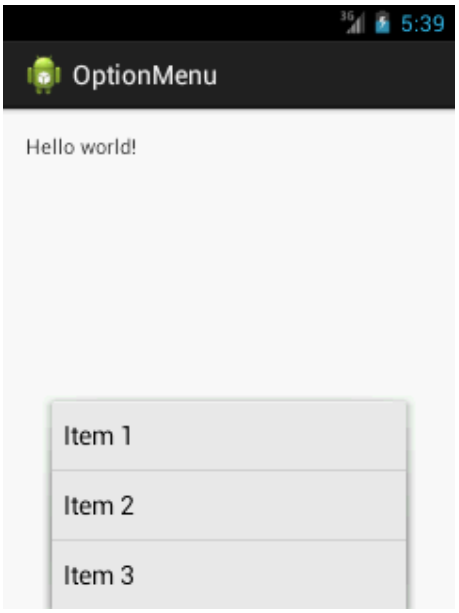
download this android example

Output:

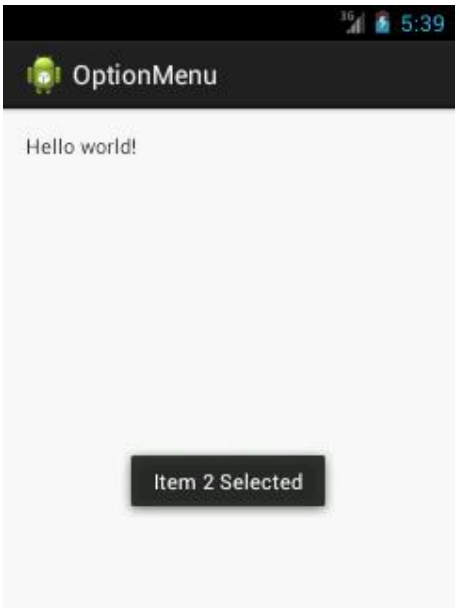
Output without clicking on the menu button.



Output after clicking on the menu button.



Output after clicking on the second menu item .



Option Menu with Icon

You need to have icon images inside the res/drawable directory. The android:icon element is used to display the icon on the option menu. You can write the string information in the strings.xml file. But we have written it inside the menu_main.xml file.

File: menu_main.xml

```
1. <menu xmlns:androclass="http://schemas.android.com/apk/res/android" >
2.     <item  android:id="@+id/item1"
3.         android:icon="@drawable/add"
4.         android:title="Item 1" />
5.     <item  android:id="@+id/item2"
6.         android:icon="@drawable/minus"
7.         android:title="Item 2" />
8.     <item  android:id="@+id/item3"
9.         android:icon="@drawable/delete"
10.        android:title="Item 3" />
11.</menu>
```