Android Core Building Blocks

An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

#### Activity

An activity is a class that represents a single screen. It is like a Frame in AWT.

#### View

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

#### Intent

Intent is used to invoke components. It is mainly used to:

* Start the service
* Launch an activity
* Display a web page
* Display a list of contacts
* Broadcast a message
* Dial a phone call etc.

For example, you may write the following code to view the webpage.

1. Intent intent=**new** Intent(Intent.ACTION\_VIEW);
2. intent.setData(Uri.parse("http://www.techvisionit.com"));
3. startActivity(intent);

# Android Emulator

**Android Emulator** is used to run, debug and test the android application. If you don't have the real device, it can be the best way to run, debug and test the application.

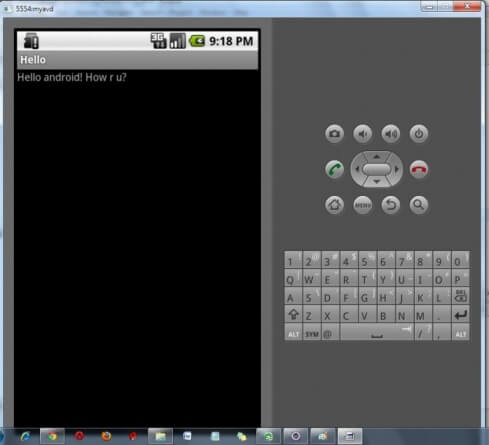
It uses an open source processor emulator technology called **QEMU**.

The emulator tool enables you to start the emulator from the command line. You need to write:

emulator -avd <AVD NAME>

In case of Eclipse IDE, you can create AVD by **Window menu > AVD Manager > New**.

In the given image, you can see the android emulator, it displays the output of the hello android example.



### Write the message

*File: activity\_main.xml*

Android studio auto generates code for activity\_main.xml file. You may edit this file according to your requirement.

1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:layout\_width="match\_parent"
6. android:layout\_height="match\_parent"
7. tools:context="first.tech.com.welcome.MainActivity">
9. <TextView
10. android:layout\_width="wrap\_content"
11. android:layout\_height="wrap\_content"
12. android:text="Hello Android!"
13. app:layout\_constraintBottom\_toBottomOf="parent"
14. app:layout\_constraintLeft\_toLeftOf="parent"
15. app:layout\_constraintRight\_toRightOf="parent"
16. app:layout\_constraintTop\_toTopOf="parent" />
18. </android.support.constraint.ConstraintLayout>
19. }

*File: MainActivity.java*

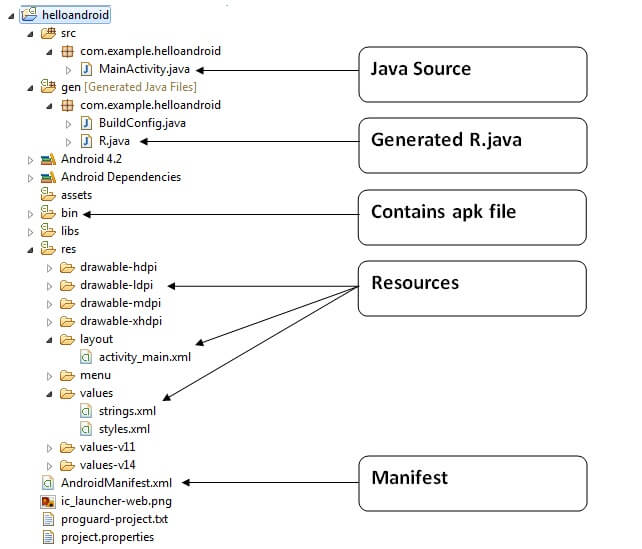
1. **package** first.tech.com.welcome;
3. **import** android.support.v7.app.AppCompatActivity;
4. **import** android.os.Bundle;
6. **public** **class** MainActivity **extends** AppCompatActivity {
7. @Override
8. **protected** **void** onCreate(Bundle savedInstanceState) {
9. **super**.onCreate(savedInstanceState);
10. setContentView(R.layout.activity\_main);
11. }
12. }

[**next →**](https://www.javatpoint.com/dalvik-virtual-machine)[**← prev**](https://www.javatpoint.com/hello-android-example)

# 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);
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. }
20. **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.

For more information about manifest file visit here: [AndroidManifest.xml file](https://www.javatpoint.com/AndroidManifest-xml-file-in-android).

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:

1. **<manifest** xmlns:android="http://schemas.android.com/apk/res/android"
2. package="com.hello"
3. android:versionCode="1"
4. android:versionName="1.0" **>**
6. **<uses-sdk**
7. android:minSdkVersion="8"
8. android:targetSdkVersion="15" **/>**
10. **<application**
11. android:icon="@drawable/ic\_launcher"
12. android:label="@string/app\_name"
13. android:theme="@style/AppTheme" **>**
14. **<activity**
15. android:name=".MainActivity"
16. android:label="@string/title\_activity\_main" **>**
17. **<intent-filter>**
18. **<action** android:name="android.intent.action.MAIN" **/>**
20. **<category** android:name="android.intent.category.LAUNCHER" **/>**
21. **</intent-filter>**
22. **</activity>**
23. **</application>**
25. **</manifest>**

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 Hide Title Bar and Full Screen 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.

### Code that hides title bar of activity

The **getSupportActionBar()** method is used to retrieve the instance of ActionBar class. Calling the hide() method of ActionBar class hides the title bar.

1. requestWindowFeature(Window.FEATURE\_NO\_TITLE);//will hide the title
2. getSupportActionBar().hide(); //hide the title bar

### Code that enables full screen mode of activity

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. **this**.getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN,
2. WindowManager.LayoutParams.FLAG\_FULLSCREEN); //show the activity in full screen

### Android Hide Title Bar and Full Screen Example

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

#### activity\_main.xml

*File: activity\_main.xml*

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<android.support.constraint.ConstraintLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:layout\_width="match\_parent"
6. android:layout\_height="match\_parent"
7. tools:context="first..com.hidetitlebar.MainActivity"**>**
9. **<TextView**
10. android:layout\_width="wrap\_content"
11. android:layout\_height="wrap\_content"
12. android:text="Hello World!"
13. app:layout\_constraintBottom\_toBottomOf="parent"
14. app:layout\_constraintLeft\_toLeftOf="parent"
15. app:layout\_constraintRight\_toRightOf="parent"
16. app:layout\_constraintTop\_toTopOf="parent" **/>**
18. **</android.support.constraint.ConstraintLayout>**

#### Activity class

*File: MainActivity.java*

1. **package** first..com.hidetitlebar;
3. **import** android.support.v7.app.AppCompatActivity;
4. **import** android.os.Bundle;
5. **import** android.view.Window;
6. **import** android.view.WindowManager;
8. **public** **class** MainActivity **extends** AppCompatActivity {
10. @Override
11. **protected** **void** onCreate(Bundle savedInstanceState) {
12. **super**.onCreate(savedInstanceState);
13. requestWindowFeature(Window.FEATURE\_NO\_TITLE); //will hide the title
14. getSupportActionBar().hide(); // hide the title bar
15. **this**.getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN,
16. WindowManager.LayoutParams.FLAG\_FULLSCREEN); //enable full screen
17. setContentView(R.layout.activity\_main);

20. }
21. }

Android Screen Orientation Example

The **screenOrientation** is the attribute of activity element. The orientation of android activity can be portrait, landscape, sensor, unspecified etc. You need to define it in the AndroidManifest.xml file.

**Syntax:**

1. **<activity** android:name="package\_name.Your\_ActivityName"
2. android:screenOrientation="orirntation\_type"**>**
3. **</activity>**

**Example:**

1. **<activity** android:name=" example.tech.com.screenorientation.MainActivity"
2. android:screenOrientation="portrait"**>**
3. **</activity>**
4. **<activity** android:name=".SecondActivity"
5. android:screenOrientation="landscape"**>**
6. **</activity>**

The common values for screenOrientation attribute are as follows:

|  |  |
| --- | --- |
| **Value** | **Description** |
| unspecified | It is the default value. In such case, system chooses the orientation. |
| portrait | taller not wider |
| landscape | wider not taller |
| sensor | orientation is determined by the device orientation sensor. |

## Android Portrait and Landscape mode screen orientation example

In this example, we will create two activities of different screen orientation. The first activity (MainActivity) will be as "portrait" orientation and second activity (SecondActivity) as "landscape" orientation type.

#### activity\_main.xml

*File: activity\_main.xml*

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<android.support.constraint.ConstraintLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:layout\_width="match\_parent"
6. android:layout\_height="match\_parent"
7. tools:context="example.tech.com.screenorientation.MainActivity"**>**

10. **<Button**
11. android:id="@+id/button1"
12. android:layout\_width="wrap\_content"
13. android:layout\_height="wrap\_content"
14. android:layout\_marginBottom="8dp"
15. android:layout\_marginTop="112dp"
16. android:onClick="onClick"
17. android:text="Launch next activity"
18. app:layout\_constraintBottom\_toBottomOf="parent"
19. app:layout\_constraintEnd\_toEndOf="parent"
20. app:layout\_constraintHorizontal\_bias="0.612"
21. app:layout\_constraintStart\_toStartOf="parent"
22. app:layout\_constraintTop\_toBottomOf="@+id/editText1"
23. app:layout\_constraintVertical\_bias="0.613" **/>**
25. **<TextView**
26. android:id="@+id/editText1"
27. android:layout\_width="wrap\_content"
28. android:layout\_height="wrap\_content"
29. android:layout\_centerHorizontal="true"
30. android:layout\_marginEnd="8dp"
31. android:layout\_marginStart="8dp"
32. android:layout\_marginTop="124dp"
33. android:ems="10"
34. android:textSize="22dp"
35. android:text="This activity is portrait orientation"
36. app:layout\_constraintEnd\_toEndOf="parent"
37. app:layout\_constraintHorizontal\_bias="0.502"
38. app:layout\_constraintStart\_toStartOf="parent"
39. app:layout\_constraintTop\_toTopOf="parent" **/>**
40. **</android.support.constraint.ConstraintLayout>**

#### Activity class

*File: MainActivity.java*

1. **package** example.tech.com.screenorientation;
3. **import** android.content.Intent;
4. **import** android.support.v7.app.AppCompatActivity;
5. **import** android.os.Bundle;
6. **import** android.view.View;
7. **import** android.widget.Button;
9. **public** **class** MainActivity **extends** AppCompatActivity {
11. Button button1;
12. @Override
13. **protected** **void** onCreate(Bundle savedInstanceState) {
14. **super**.onCreate(savedInstanceState);
15. setContentView(R.layout.activity\_main);
17. button1=(Button)findViewById(R.id.button1);
18. }
19. **public** **void** onClick(View v) {
20. Intent intent = **new** Intent(MainActivity.**this**,SecondActivity.**class**);
21. startActivity(intent);
22. }
23. }

#### activity\_second.xml

*File: activity\_second.xml*

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<android.support.constraint.ConstraintLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:layout\_width="match\_parent"
6. android:layout\_height="match\_parent"
7. tools:context="example.tech.com.screenorientation.SecondActivity"**>**
9. **<TextView**
10. android:id="@+id/textView"
11. android:layout\_width="wrap\_content"
12. android:layout\_height="wrap\_content"
13. android:layout\_marginEnd="8dp"
14. android:layout\_marginStart="8dp"
15. android:layout\_marginTop="180dp"
16. android:text="this is landscape orientation"
17. android:textSize="22dp"
18. app:layout\_constraintEnd\_toEndOf="parent"
19. app:layout\_constraintHorizontal\_bias="0.502"
20. app:layout\_constraintStart\_toStartOf="parent"
21. app:layout\_constraintTop\_toTopOf="parent" **/>**
22. **</android.support.constraint.ConstraintLayout>**

#### SecondActivity class

*File: SecondActivity.java*

1. **package** example.tech.com.screenorientation;
3. **import** android.support.v7.app.AppCompatActivity;
4. **import** android.os.Bundle;
6. **public** **class** SecondActivity **extends** AppCompatActivity {
8. @Override
9. **protected** **void** onCreate(Bundle savedInstanceState) {
10. **super**.onCreate(savedInstanceState);
11. setContentView(R.layout.activity\_second);
13. }
14. }

#### AndroidManifest.xml

*File: AndroidManifest.xml*

In AndroidManifest.xml file add the screenOrientation attribute in activity and provides its orientation. In this example, we provide "portrait" orientation for MainActivity and "landscape" for SecondActivity.

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<manifest** xmlns:android="http://schemas.android.com/apk/res/android"
3. package="example.tech.com.screenorientation"**>**
5. **<application**
6. android:allowBackup="true"
7. android:icon="@mipmap/ic\_launcher"
8. android:label="@string/app\_name"
9. android:roundIcon="@mipmap/ic\_launcher\_round"
10. android:supportsRtl="true"
11. android:theme="@style/AppTheme"**>**
12. **<activity**
13. android:name="example.tech.com.screenorientation.MainActivity"
14. android:screenOrientation="portrait"**>**
15. **<intent-filter>**
16. **<action** android:name="android.intent.action.MAIN" **/>**
18. **<category** android:name="android.intent.category.LAUNCHER" **/>**
19. **</intent-filter>**
20. **</activity>**
21. **<activity** android:name=".SecondActivity"
22. android:screenOrientation="landscape"**>**
23. **</activity>**
24. **</application>**
26. **</manifest>**

# Android Button Example

## Android Button Example with Listener

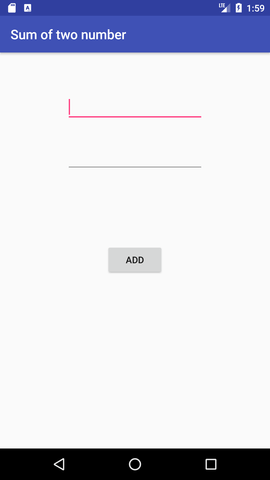
Here, we are going to create two textfields and one button for sum of two numbers. If user clicks button, sum of two input values is displayed on the Toast.

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.

1. button.setOnClickListener(new View.OnClickListener() {
2. @Override
3. public void onClick(View view) {
4. //code
5. }
6. });
7. **<Button**
8. android:onClick="methodName"
9. **/>**

### 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.



*File: activity\_main.xml*

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<RelativeLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:layout\_width="match\_parent"
6. android:layout\_height="match\_parent"
7. tools:context="example.tech.com.sumoftwonumber.MainActivity"**>**
9. **<EditText**
10. android:id="@+id/editText1"
11. android:layout\_width="wrap\_content"
12. android:layout\_height="wrap\_content"
13. android:layout\_alignParentTop="true"
14. android:layout\_centerHorizontal="true"
15. android:layout\_marginTop="61dp"
16. android:ems="10"
17. android:inputType="number"
18. tools:layout\_editor\_absoluteX="84dp"
19. tools:layout\_editor\_absoluteY="53dp" **/>**
21. **<EditText**
22. android:id="@+id/editText2"
23. android:layout\_width="wrap\_content"
24. android:layout\_height="wrap\_content"
25. android:layout\_below="@+id/editText1"
26. android:layout\_centerHorizontal="true"
27. android:layout\_marginTop="32dp"
28. android:ems="10"
29. android:inputType="number"
30. tools:layout\_editor\_absoluteX="84dp"
31. tools:layout\_editor\_absoluteY="127dp" **/>**
33. **<Button**
34. android:id="@+id/button"
35. android:layout\_width="wrap\_content"
36. android:layout\_height="wrap\_content"
37. android:layout\_below="@+id/editText2"
38. android:layout\_centerHorizontal="true"
39. android:layout\_marginTop="109dp"
40. android:text="ADD"
41. tools:layout\_editor\_absoluteX="148dp"
42. tools:layout\_editor\_absoluteY="266dp" **/>**
43. **</RelativeLayout>**

### Activity class

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

*File: MainActivity.java*

1. **package** example.tech.com.sumoftwonumber;
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;
10. **public** **class** MainActivity **extends** AppCompatActivity {
11. **private** EditText edittext1, edittext2;
12. **private** Button buttonSum;
14. @Override
15. **protected** **void** onCreate(Bundle savedInstanceState) {
16. **super**.onCreate(savedInstanceState);
17. setContentView(R.layout.activity\_main);
19. addListenerOnButton();
20. }
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);
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:

