**1.Android Development Environment**

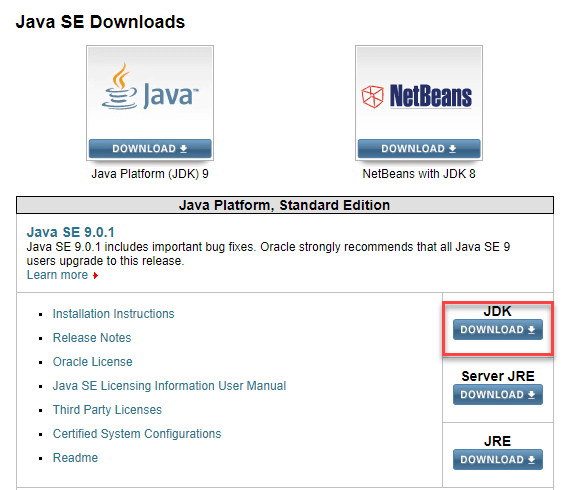
**AIM:**Setting Up the Development Environment

**PROCEDURE:**

This Java Development Kit(JDK) allows you to code and run Java programs. It's possible that you install multiple JDK versions on the same PC. But Its recommended that you install only latest version.

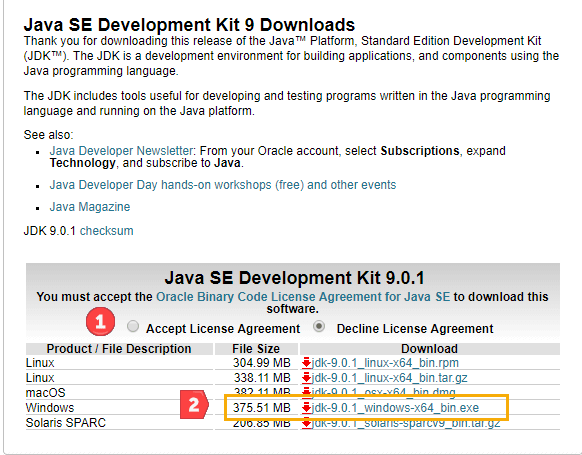
Following are steps to install Java in Windows

**Step 1)** Go to [link](http://www.oracle.com/technetwork/java/javase/downloads/index.html). Click on Download JDK. For java latest version.

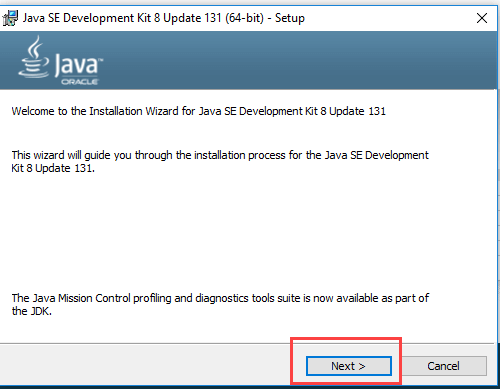
[](https://cdn.guru99.com/images/java/111417_1107_Java21.png)

**Step 2)** Next,

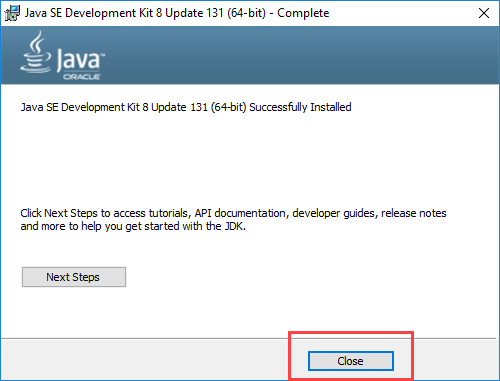
1. Accept License Agreement
2. Download latest Java JDK for your version(32 or 64 bit) of java for Windows.

[](https://cdn.guru99.com/images/java/111417_1107_Java22.png)

**Step 3)** Once the download is complete, run the exe for install JDK. Click Next

[](https://cdn.guru99.com/images/java/111417_1107_Java23.png)

**Step 4)** Once installation is complete click Close

[](https://cdn.guru99.com/images/java/111417_1107_Java24.png)

**How to set Environment Variables in Java: Path and Classpath**

The PATH variable gives the location of executables like javac, java etc. It is possible to run a program without specifying the PATH but you will need to give full path of executable like ***C:\Program Files\Java\jdk1.8.0\_131\bin\javac A.java*** instead of simple ***javac A.java***

The CLASSPATH variable gives location of the Library Files.

Let's look into the steps to set the PATH and CLASSPATH

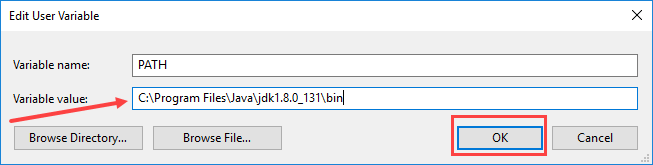
**Step 1)** Right Click on the My Computer and Select the properties

**Step 2)** Click on advanced system settings

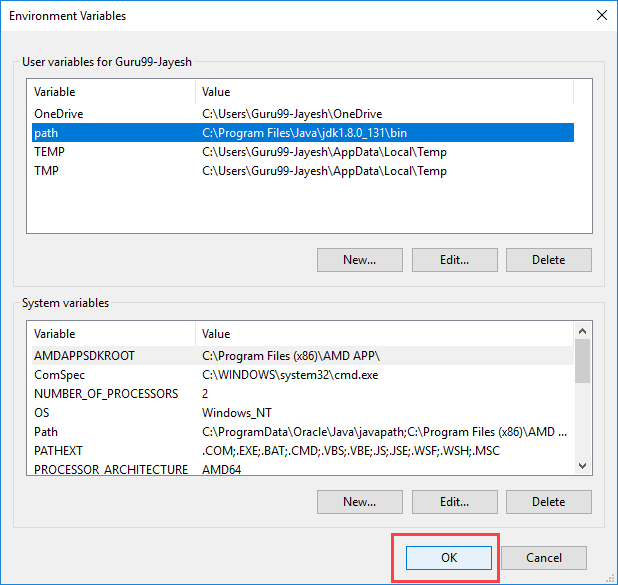
**Step 3)** Click on Environment Variables

[](https://cdn.guru99.com/images/java/111417_1107_Java27.png)

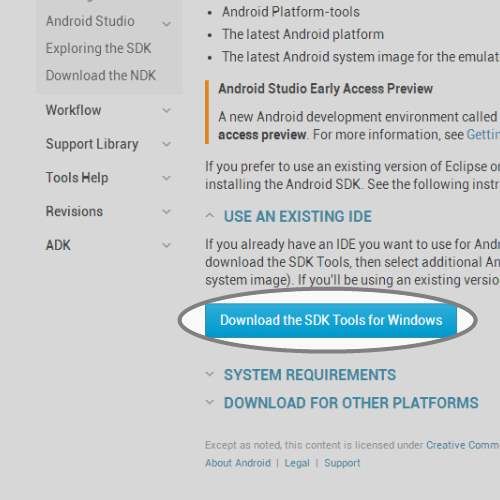
**Step 4**) Paste Path of bin folder in Variable value and click on OK Button.

[](https://cdn.guru99.com/images/java/111417_1107_Java211.png)

**Step 5)** Click on OK button

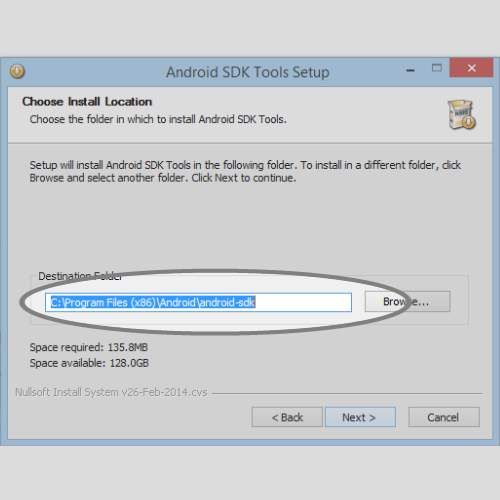
[](https://cdn.guru99.com/images/java/111417_1107_Java213.png)

**Step 1: Obtain the Android SDK**

**[](https://cdn.instructables.com/FZ0/9D35/HSGFXUMG/FZ09D35HSGFXUMG.LARGE.jpg)**

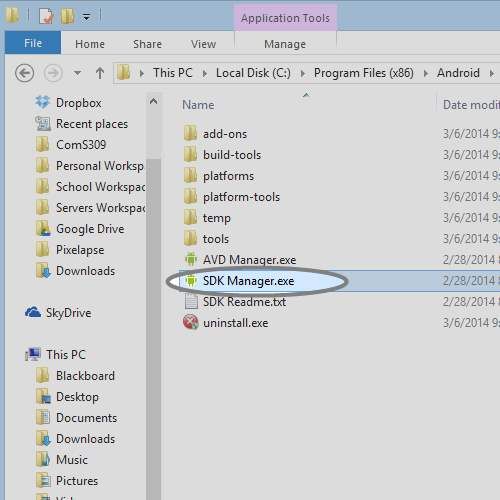
* We need to obtain the Android Software Development Kit. To do that we must first visit the SDK download site below.
* <http://developer.android.com/sdk/index.html>
* Scroll to the bottom of the webpage and select the 'Download the SDK Tools' button.
* This will open a file to be saved somewhere.

**Step 2: Install the Android SDK**

[](https://cdn.instructables.com/FFM/Z698/HSGFXX2A/FFMZ698HSGFXX2A.LARGE.jpg)

* Open the file we just downloaded.
* This will open an executable which will ask you about the installation process.
* When you get to the install location screen, make sure you choose a location you remember (we will need it later).

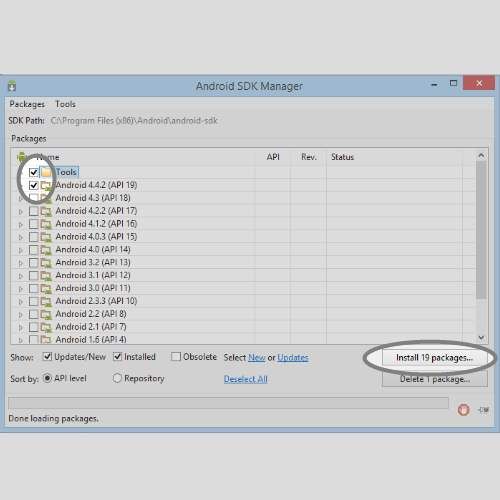
**Step 3: Open Android SDK Manager**

[](https://cdn.instructables.com/FOF/JE1U/HSHDS5JL/FOFJE1UHSHDS5JL.LARGE.jpg)

We'll select the most recent version along with the extra SDK tools so you can build applications for any Android version.

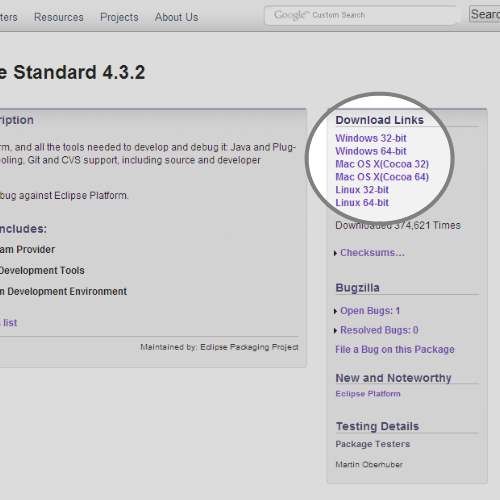
* Open the folder that we installed the SDK into.
* You'll see a executable called 'SDK Manager'. Open it.
* You'll see a window where you can select different versions of Android to develop for.

**Step 4: Install Android Version and Extras for SDK**

[](https://cdn.instructables.com/FOH/ABIZ/HSGFXZH4/FOHABIZHSGFXZH4.LARGE.jpg)

* Select the 'Tools' and 'Android 4.4.2 (API 19)' check boxes.
* If you would like to have extra Android tools you can choose them from the 'Extras' selection.
* You will then be prompted to accept the Android conditions and then the SDK will install. Then installation may take a while depending on your internet connection. The Android SDK is now completely installed.

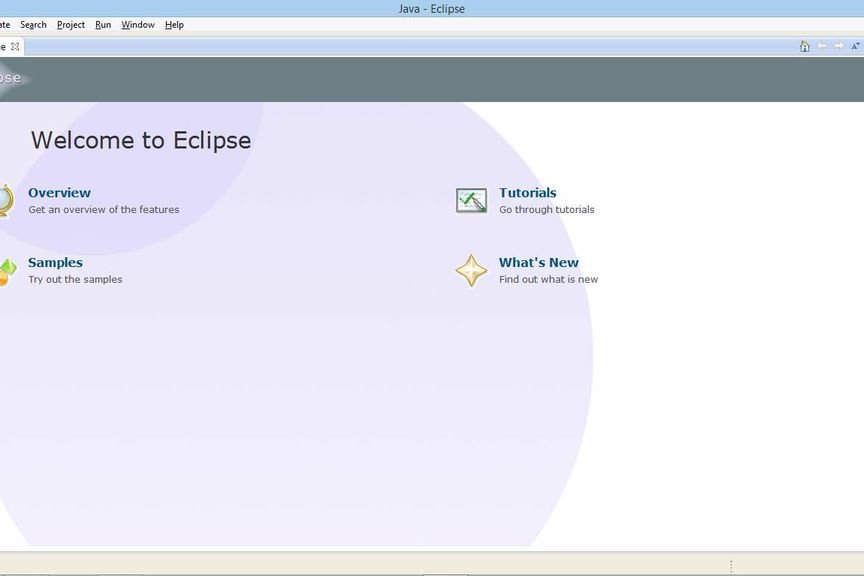
**Step 5: Obtain Eclipse IDE**

[](https://cdn.instructables.com/FJ9/8398/HSGFYBUC/FJ98398HSGFYBUC.LARGE.jpg)

Eclipse is the tool we'll be using to develop in. It is the most popular Android development environment and has officially supported tools from Google.

* Download Eclipse from the website below.
* [http://www.eclipse.org/downloads/packages/eclipse-...](http://www.eclipse.org/downloads/packages/eclipse-standard-432/keplersr2)
* Find the link for your operating system and 32/64 bit version.
* Save the compressed download file.

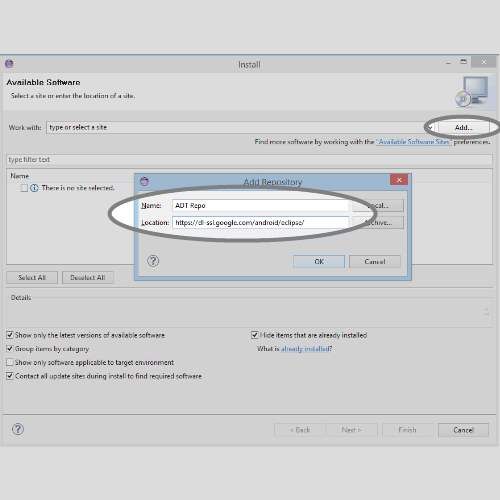
**Step 6: Run Eclipse for First Time**

[](https://cdn.instructables.com/FZM/J8P1/HSGFYLXV/FZMJ8P1HSGFYLXV.LARGE.jpg)

Eclipse does not require installation. It's a folder with all the necessary files and settings. You can run it directly from the Eclipse folder. It's recommended you put it in a safe place with other applications.

* Extract the downloaded Eclipse file into a safe place where you can keep the program.
* Open the extracted folder and open the 'eclipse' executable.

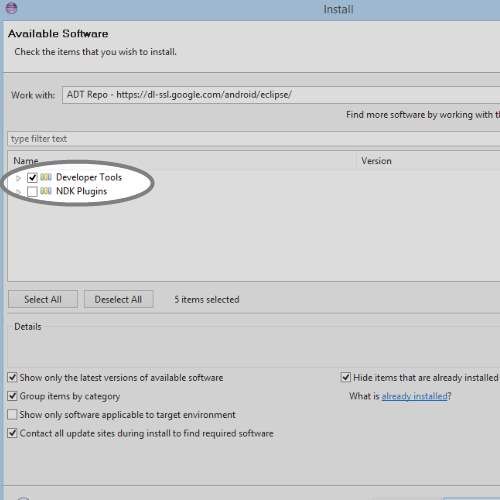
**Step 7: Add ADT Plugin Repository**

[](https://cdn.instructables.com/F11/VNPI/HSHDS5SM/F11VNPIHSHDS5SM.LARGE.jpg)

The ADT (Android Development Tool) Plugin was made specifically for Eclipse to increase productivity and integration with your Android work environment. To use it, we first add the Eclipse plugin repository so it knows where to find it along with updates.

* In the Eclipse application menu, go to 'Help' and then 'Install New Software'.
* Click on the 'Add...' button and you'll see a window appear.
* Give the repository a name like 'ADT Repo'.
* Give it the location [http://dl-ssl.google.com/android/eclipse/.](http://dl-ssl.google.com/android/eclipse/)
* Click 'OK' button.

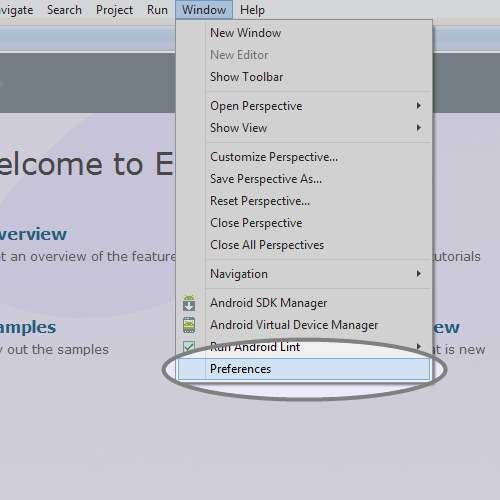
**Step 8: Install ADT Plugin**

[](https://cdn.instructables.com/FCH/UUIF/HSHDS5XF/FCHUUIFHSHDS5XF.LARGE.jpg)

Now that we have the plugin repo setup we need to install the plugin from it.

* On the 'Install Software' screen, select the repo you just created from the 'Work with' selector.
* Select the 'Developer Tools' option from the listed below options.
* Click 'Next' and accept the agreements.
* Click 'Finish' and let it install. It might take a while depending on your internet speed.

**Step 9: Access ADT Plugin Preferences**

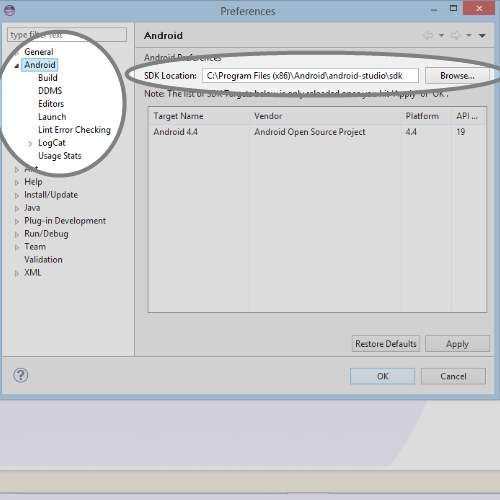
[](https://cdn.instructables.com/FFW/4GU4/HSHDS67B/FFW4GU4HSHDS67B.LARGE.jpg)

You'll most likely have to restart Eclipse after you install the ADT Plugin so do that before you continue.

* Mac/Linux: After Eclipse has restarted, click 'Eclipse' in the application menu.
* Windows: After Eclipse has restarted, click 'Window' in the application menu.
* Then select 'Preferences'.
* In the Preferences window, select the Android tab on the left side and its corresponding drop-down menu.

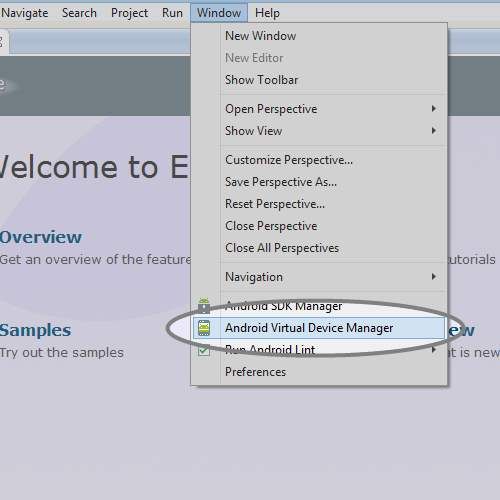
This is the ADT Preferences screen. It will allow you to change settings, remove the SDK, and make editor preferences for your development.

**Step 10: Setup ADT Plugin**

[](https://cdn.instructables.com/FC8/669R/HSHDS6LI/FC8669RHSHDS6LI.LARGE.jpg)

* Click 'Browse' on the right side of the screen.
* Search for the folder in which you installed the Android SDK into and select it.
* Hit the 'Apply' button on the Preferences screen.
* You should see the Android version you installed early to show up if all went well.
* If not, try reselecting the folder. (Make sure the folder contains the folders 'build-tools','platform','extras' and etc.
* Hit the 'OK' button and restart Eclipse.

**Step 11: Access ADT Android Virtual Device Manager**

[](https://cdn.instructables.com/FWO/LZB0/HSHDS6S7/FWOLZB0HSHDS6S7.LARGE.jpg)

RESULT:

**2. HELLO WORLD APPLICATION**

**AIM:** Create "Hello World" Application

**PROCEDURE:**

Step 1: Create a new project and name it Helloworld

In this step we create a new project in android studio by filling all the necessary details of

the app like app name, package name, api versions etc.

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

Step 2: Now Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add the following

code:

2. Create "Hello World" Application

activity\_main.[xml](https://abhiandroid.com/ui/xml)

*<?*xml version="1.0" encoding="utf-8"*?>*<android.support.constraint.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.example.lenovo.helloworld.MainActivity">  
<TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello World!"  
app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
</android.support.constraint.ConstraintLayout>

**MainActivity.java**

packagecom.example.lenovo.helloworld;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
 @Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 }  
}

EXPECTED OUTPUT:

EXECUTED OUTPUT:

RESULT:

**3.APPLICATION CREATION USING WIDGETS**

**AIM:**Creating the Application by using the Activity class

**PROCEDURE:**

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

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

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

**Step 2:** Now Open res -> layout -> **activity\_main.**[**xml**](https://abhiandroid.com/ui/xml)**(or) main.**[**xml**](https://abhiandroid.com/ui/xml)and add the following code:

3. Creating the Application by using the Activity class.

**Activity\_main.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.example.lenovo.prog2.MainActivity">  
  
<TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello World!"  
app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</android.support.constraint.ConstraintLayout>

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

In this step we add the code to initiate the activity of application

MainActivity.java

packagecom.example.lenovo.prog2;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.widget.Toast;  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 }  
 @Override  
protected void onStart() {  
super.onStart();  
 Toast.*makeText*(this, "on start", Toast.*LENGTH\_SHORT*).show();  
 }  
 @Override  
protected void onPause() {  
super.onPause();  
 Toast.*makeText*(this, "on pause", Toast.*LENGTH\_SHORT*).show();  
 }

@Override  
protected void onRestart() {  
super.onRestart();  
 Toast.*makeText*(this, "on restart", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
protected void onResume() {  
super.onResume();  
 Toast.*makeText*(this, "on resume", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
protected void onDestroy() {  
super.onDestroy();  
 Toast.*makeText*(this, "on destroy", Toast.*LENGTH\_SHORT*).show();  
 }  
}

EXPECTED OUTPUT:

EXECUTED OUTPUT:

RESULT:

**4.CREATE APPLICATION BY USING TEXTEDIT CONTROL**

**PROCEDURE**:

Step 1: Create a new project and name it TexteditExample

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

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

Step 2: Now Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add the following code:

4. Creating the Application by using Text Edit control.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/user\_name"

android:text="Enter your name:"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/response"/>

</LinearLayout>

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

In this step we add the code to initiate the check boxes we created. And then we perform click event on button and display the text for selected check boxes using a toast

Package.com.androidunleashed.edittextapp;

importandroid.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

import android.widget.EditText;

import android.view.View;

import android.view.View.OnKeyListener;

import android.view.KeyEvent;

public class EditTextAppActivity extends Activity

{

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_edit\_text\_app);

final TextView resp= (TextView) this.findViewById (R.id.response);

final EditText username=(EditText) findViewById(R.id.user\_name);

username.setOnKeyListener(new OnKeyListener() {

public boolean onKey(View v, int keyCode, KeyEvent event) {

if((event.getAction()== KeyEvent.ACTION\_UP) &

(keyCode==(KeyEvent.KEYCODE\_ENTER))){

resp.setText("Welcome "+username.getText()+"!");

return true;

}

return false;

}});}}

**EXPECTED OUT PUT:**

EXECUTED OUTPUT:

RESULT:

**5. Creating the Application Choosing Options**

**Aim**: To Create Applications with Choosing Options

(i) CheckBox (ii) RadioButton (iii) RadioGroup (iv) Spinner

(i) CheckBox

**PROCEDURE**:

Step 1: Create a new project and name it CheckBoxExample

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

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

Step 2: Now Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add the following code:

5. Creating the Application by using CheckBox control.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
<TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Select Items you Want"/>  
<CheckBox  
 android:id="@+id/checkbox\_pizza"  
 android:layout\_height="wrap\_content"  
 android:text="pizza $15"  
 android:layout\_width="match\_parent"/>  
<CheckBox  
 android:id="@+id/checkbox\_hotdog"  
 android:layout\_height="wrap\_content"  
 android:text="Hot Dog $5"  
 android:layout\_width="match\_parent"/>  
<CheckBox  
 android:id="@+id/checkbox\_burger"  
 android:layout\_height="wrap\_content"  
 android:text="Burger $10"  
 android:layout\_width="match\_parent"/>  
<Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/bill\_btn"  
 android:text="Calculate Bill"/>  
<TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/amount"/>  
</LinearLayout>

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

In this step we add the code to initiate the check boxes we created. And then we perform click event on button and display the text for selected check boxes using a TextView

package com.androidunleashed.checkboxapp;

import android.app.Activity;  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.CheckBox;  
import android.view.View;  
import android.view.View.OnClickListener;  
public class CheckBoxAppActivity extends Activity implements OnClickListener{  
 CheckBox c1,c2,c3;  
 TextView resp;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_check\_box\_app);  
 Button b=(Button)this.findViewById(R.id.bill\_btn);  
 resp=(TextView)this.findViewById(R.id.amount);  
 c1=(CheckBox)this.findViewById(R.id.checkbox\_pizza);  
 c2=(CheckBox)this.findViewById(R.id.checkbox\_hotdog);  
 c3=(CheckBox)this.findViewById(R.id.checkbox\_burger);  
 b.setOnClickListener(this);  
 }  
 public void onClick(View v){  
 int amt=0;  
 if(c1.isChecked()){  
 amt=amt+15;

}  
if(c2.isChecked()){  
 amt=amt+5;

}  
if(c3.isChecked()){  
 amt=amt+10;

}  
resp.setText("Bill is"+Integer.toString(amt));  
 }  
}

**EXPECTED OUTPUT:**

EXECUTED OUTPUT:

RESULT:

(ii) RadioButton

**PROCEDURE**:

Step 1: Create a new project and name it RadioButtonExample

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

Step 2: Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add following code:

In this step we open an [xml](https://abhiandroid.com/ui/xml) file and add the code for displaying 5 RadioButton and one normal button.

6. Creating the Application by using RadioButton control.

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Select the type of hotel"/>

<RadioGroup

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<RadioButton

android:id="@+id/radio\_fivestar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Five Star"/>

<RadioButton

android:id="@+id/radio\_threestar"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Three Star"/>

</RadioGroup>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/hoteltype"/>

</LinearLayout>

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

In this step we open MainActivity and add the code to initiate the RadioButton and normal button.

package com.androidunleashed.radiobutttonapp;

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

import android.widget.RadioButton;

import android.view.View;

import android.view.View.OnClickListener;

public class RadioButtonAppActivty extends Activity{

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activty\_radio\_button\_app);

RadioButton radioFivestar=(RadioButton)findViewById(R.id.radio\_fivestar);

RadioButton radioThreestar=(RadioButton)findViewById(R.id.radio\_threestar);

radioFivestar.setOnClickListener(radioListener);

radioThreestar.setOnClickListener(radioListener);

}

private OnClickListener radioListener=new OnClickListener(){

public void onClick(View v){

TextView selectedHotel=(TextView)findViewById(R.id.hoteltype);

RadioButton rb=(RadioButton) v;

selectedHotel.setText("The hotel type selected is:" + rb.getText());

}

};

}

**EXPECTED OUTPUT:**

EXECUTED **OUTPUT:**

**RESULT:**

(iii) RadioGroup

**PROCEDURE**:

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

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

Step 2: Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add following code.

7. Creating the Application by using RadioGroup control

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
<TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Select the type of hotel"/>  
<RadioGroup  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
<RadioButton  
 android:id="@+id/radio\_fivestar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Five Star"/>  
<RadioButton  
 android:id="@+id/radio\_threestar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Three Star"/>  
</RadioGroup>  
<TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/hoteltype"/>  
<TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Select the type of room"/>  
<RadioGroup  
 android:id="@+id/group\_room"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
<RadioButton  
 android:id="@+id/radio\_suite"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Grand Suite"/>  
<RadioButton  
 android:id="@+id/radio\_luxury"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Luxury Room"/>  
<RadioButton  
 android:id="@+id/radio\_ordinary"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="ordinary Room"/>  
</RadioGroup>  
<TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/roomtype" />  
</LinearLayout>

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

Package com.androidunleashed.radiobuttonapp;

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

import android.widget.RadioButton;

import android.view.View;

import android.view.View.OnClickListener;

public class RadioButtonAppActivity extends Activity{

String str1=" ";

String str2=" ";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_radio\_button\_app);

RadioButton radioFivestar = (RadioButton) findViewById(R.id.radio\_fivestar);

RadioButton radioThreestar=(RadioButton) findViewById(R.id.radio\_threestar);

RadioButton radioSuite = (RadioButton) findViewById(R.id.radio\_suite);

RadioButton radioLuxury = (RadioButton) findViewById(R.id.radio\_luxury);

RadioButton radioOrdinary = (RadioButton) findViewById(R.id.radio\_ordinary);

radioFivestar.setOnClickListener(radioListener1);

radioThreestar.setOnClickListener(radioListener1);

radioSuite.setOnClickListener(radioListener2);

radioLuxury.setOnClickListener(radioListener2);

radioOrdinary.setOnClickListener(radioListener2);

private OnClickListener radioListener1 = new OnClickListener(){

public void onClick(View v) {

TextView selectedOptions = (TextView) findViewById(R.id.hoteltype);

RadioButton rb = (RadioButton) v;

str1 = "The hotel type selected is: " + rb.getText();

selectedOptions.setText(str1 + "\n" + str2);

} };

private OnClickListener radioListener2 = new OnClickListener(){

public void onClick(View v){

TextView selectedOptions = (TextView) findViewById(R.id.hoteltype);

RadioButton rb = (RadioButton) v;

str2="Room type selected is: "+rb.getText();

selectedOptions.setText(str1+"\n"+str2);

}

};

}

**EXPECTED OUT PUT:**

EXECUTED **OUTPUT:**

**RESULT:**

(iv) Spinner

**PROCEDURE**:

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

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

Step 2: Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add following code. Here we will create a Spinner inside [Relative Layout](https://abhiandroid.com/ui/relative-layout).

8. Creating the Application by using Spinner control.

STRINGS.XML:

<resources

<string name="app\_name">SpinnerAppActivity</string>

<string name="choose\_msg">Choose a fruit</string>

</resources>

ARRAYS.XML

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

<resources>

<string-array name="fruits">

<item>Apple</item>

<item>Mango</item>

<item>Banana</item>

<item>Orange</item>

<item>Grapes</item>

</string-array>

</resources>

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<Spinner

android:id="@+id/spinner"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:prompt="@string/choose\_msg"

android:entries="@array/fruits">

</Spinner>

<TextView

android:id="@+id/selectedopt"

androd:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

Step 3: Now open app-> java -> package -> MainActivity.java and add the following code. Here we will use Adapter to fill the data in Spinner.

package com.example.spinnerappactivity;  
  
import android.os.Bundle;  
import androidx.appcompat.app.AppCompatActivity;  
  
  
  
import android.widget.TextView;  
  
import android.widget.Spinner;  
  
import android.widget.AdapterView;  
  
import android.view.View;  
  
import android.widget.AdapterView.OnItemSelectedListener;  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 final TextView selectedOpt=(TextView)findViewById(R.id.*selectedopt*);  
  
 Spinner spin=(Spinner)findViewById(R.id.*spinner*);  
  
  
 final String[] fruitsArray = getResources().getStringArray(R.array.*fruits*);  
  
 spin.setOnItemSelectedListener(new OnItemSelectedListener() {  
  
 @Override  
  
 public void onItemSelected(AdapterView<?> parent, View view, int position, long id)  
 {  
  
 selectedOpt.setText("You have selected " + fruitsArray[position]);  
  
 }  
  
  
  
 @Override  
  
 public void onNothingSelected(AdapterView<?> parent) {  
  
 selectedOpt.setText(" ");  
  
 }  
  
 });  
  
  
  
 }  
  
}

**EXPECTED OUT PUT:**

**EXECUTED OUTPUT:**

**RESULT:**

6. **DIFFERENT ANDROID APPLICATION LAYOUTS**

AIM: To Create Application by Using Building Blocks for Android Application Design. Design the Application by using

(i) Linear Layout (ii) Relative Layout (iii) Absolute Layout

**(i) Linear Layout:**

**AIM:**Design the Application by usingLinear layout

**PROCEDURE:**

Step 1: Create a new project and name it LinearlayoutExample

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

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

Step 2: Now Open res -> layout -> activity\_main.xml (or) main.xml and add the following code:

9. Design the Application by using Linear Layout.

activity\_main.xml:

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<Button

android:id="@+id/Apple"

android:text="Apple"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_vertical"

android:layout\_weight="1.0" />

<Button

android:id="@+id/Mango"

android:text="Mango"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:gravity="center"

android:layout\_gravity="center"

android:layout\_weight="1.0"/>

<Button

android:id="@+id/Banana"

android:text="Banana"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_vertical|right"

android:layout\_weight="1.0"

android:layout\_gravity="right"/>

</LinearLayout>

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

In this step we add the code to initiate the linear layout fruit names

LinearLayoutApp.java

packagecom.example.lenovo.LinearLayoutApp;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.layout\_app\_linear);

}}

EXPECTED OUT PUT:

EXECUTED OUTPUT:

RESULT:

**(ii) Relative Layout**

**AIM:** Design the Application by using (Relative Layout)

**PROCEDURE:**

Step 1: Create a new project and name it RelativelayoutExample

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

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

Step 2: Now Open res -> layout -> activity\_main.xml (or) main.xml and add the following code:

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal">

<TextView

android:id="@+id/sign\_msg"

android:text="Sign In"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:typeface="serif"

android:textSize="25dip"

android:textStyle="bold"

android:padding="10dip"

android:layout\_centerHorizontal="true" />

<TextView

android:id="@+id/user\_msg"

android:text="User ID"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="10dip"

android:layout\_below="@id/sign\_msg"/>

<EditText

android:id="@+id/user\_ID"

android:layout\_width="250dip"

android:layout\_height="wrap\_content"

android:layout\_below="@id/sign\_msg"

android:layout\_toRightOf="@id/user\_msg"

android:maxLines="1"/>

<TextView

android:id="@+id/psw\_msg"

android:text="Password"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/user\_msg"

android:layout\_margin="10dip"

android:paddingTop="10dip"/>

<EditText

android:id="@+id/psw"

android:layout\_width="250dip"

android:layout\_height="wrap\_content"

android:maxLines="1"

android:layout\_below="@id/user\_ID"

android:layout\_toRightOf="@id/psw\_msg"

android:inputType="textPassword"/>

<Button

android:id="@+id/login\_button"

android:text="Sign In"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="10dip"

android:layout\_below="@+id/psw\_msg/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/resp"

android:layout\_below="@id/login\_button"/>

</RelativeLayout>

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

In this step we add the code to initiate Relative layout using a textview

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View.OnClickListener;

import android.widget.EditText;

import android.widget.TextView;

import android.view.View;

import android.widget.Button;

public class MainActivity extends AppCompatActivity implements OnClickListener {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button b=(Button)this.findViewById(R.id.login\_button);

b.setOnClickListener(this);

}

public void onClick(View v)

{

EditText uid= (EditText)findViewById(R.id.user\_ID);

EditText psw= (EditText)findViewById(R.id.psw);

TextView resp= (TextView)this.findViewById(R.id.resp);

String usr=uid.getText().toString();

String pswd = psw.getText().toString();

if (usr.trim().length()==0 || pswd.trim().length()==0) {

String str = "The User ID or Password is left blank \n Please Try Again.";

resp.setText(str);

}

else

{

if(usr.equals("venki")&& pswd.equals("venki"))

resp.setText("Welcome "+usr+" ! ");

else resp.setText("The User ID or Password is Incorrect\nPlease Try Again.");

}

}

}

**EXPECTED OUT PUT :**

**EXECUTED** **OUTPUT:**

**RESULT:**

**(iii) Absolute Layout**

**AIM:**Design the Application by usingAbsolute Layout

**PROCEDURE:**

Step 1: Create a new project and name it AbsolutelayoutExample

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

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

Step 2: Now Open res -> layout -> activity\_main.xml (or) main.xml and add the following code:

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

<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="90dip"

android:layout\_y="2dip"

android:text="New Product Form"

android:textSize="20sp"

android:textStyle="bold" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Product Code"

android:layout\_x="5dip"

android:layout\_y="40dip" />

<EditText

android:id="@+id/p\_code"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="110dip"

android:layout\_y="30dip"

android:minWidth="100dip" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Product Name"

android:layout\_x="5dip"

android:layout\_y="90dip" />

<EditText

android:id="@+id/p\_name"

android:layout\_width="200dip"

android:layout\_height="wrap\_content"

android:layout\_x="110dip"

android:layout\_y="80dip"

android:minWidth="200dip"

android:scrollHorizontally="true" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Product Price"

android:layout\_x="5dip"

android:layout\_y="140dip" />

<EditText

android:id="@+id/p\_price"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="110dip"

android:layout\_y="130dip"

android:minWidth="100dip" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/click\_btn"

android:text="Add New Product"

android:layout\_x="80dip"

android:layout\_y="190dip"/>

</AbsoluteLayout>

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

In this step we add the code to initiate Absolute layout using a Textview

AbsoluteLayoutApp.java

packagecom.example.lenovo.AbsoluteLayoutApp;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.layout\_app\_absolute);

}

}

**EXPECTED RESULT:**

**EXECUTED** **OUTPUT:**

**RESULT:**

**7. CREATE APPLICATION BY USING AUDIO & VIDEO CLIPS**

**AIM:**Create the Application to play the Audio clip

**PROCEDURE**:

**Step 1: Create a new project in Android Studio and name it Audioclip**

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

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

**In this step we open an xml file and add the code to display a toggle button in our activity.**

**Create the Application to play the Audio.**

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width= "match\_parent"  
 android:layout\_height="match\_parent">  
<TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/response"  
 android:gravity="center" />  
<ToggleButton  
 android:id="@+id/playstop\_btn"  
 android:layout\_width= "wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:textOn= " "  
 android:textOff=""  
 android:background="@drawable/play" />  
</LinearLayout>

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

**In this step we open MainActivity and add the code to initiate the video view and create an object of MediaController to control the video playback.**

**Add raw directory under resource and paste audio under raw directory**

package com.example.lenovo.playaudio;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.widget.ToggleButton;  
import android.view.View;  
import android.widget.TextView;  
import android.media.MediaPlayer;  
import android.view.View.OnClickListener;  
  
public class PlayAudioApp extends AppCompatActivity {  
  
 @Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_play\_audio\_app*);  
final TextView response = (TextView)this.findViewById(R.id.*response*);  
 response.setText("Select Play button to play audio");  
final MediaPlayer mp = MediaPlayer.*create*(PlayAudioApp.this,R.raw.*song1*);  
final ToggleButton playStopButton = (ToggleButton)findViewById(R.id.*playstop\_btn*);  
 playStopButton.setOnClickListener(new OnClickListener() {  
public void onClick(View v) {  
if (playStopButton. isChecked()) {  
 response.setText("Select Stop button to stop audio");

playStopButton.setBackgroundDrawable(getResources().getDrawable(R.drawable.*stop*));  
 mp.start();  
 }  
else {  
 response.setText("Select Play button to play audio");  
 playStopButton.setBackgroundDrawable(getResources().getDrawable(R. drawable.*play*));  
 mp.pause ();  
 } } }); }}

**Expected OUT PUT:**

**EXECUTED** Output:

Result:

**Create the Application to play the video clip**

**AIM:Create the Application to play the video clip**

**PROCEDURE**:

**Step 1: Create a new project in Android Studio and name it VideoViewExample**

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

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

**In this step we open an xml file and add the code to display a VideoView in our activity.**

**Create the Application to play the Video clips.**

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas .android.com/tools"  
 android:orientation="vertical"  
 android:layout\_width= "match\_parent"  
 android:layout\_height="match\_parent" >  
<VideoView android:id="@+id/video"  
 android:layout\_width="match\_parent"  
 android:layout\_height="420dip" />  
<Button android:id= "@+id/playvideo"  
 android:text="Play Video"  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent" />  
</LinearLayout>

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

**In this step we open MainActivity and add the code to initiate the video view and create an object of MediaController to control the video playback.**

**Add raw directory under resource and paste video under raw directory**

**In this class we also set the uri for the video and perform set on error and completion listener events and display Toast message when video is completed or an error is occur while playing thee video.**

**Also make sure to create a new directory in res folder and name it raw. Save a video name vid in raw folder.**

package com.example.lenovo.playingvideo;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.MediaController;  
import android.view.View.OnClickListener;  
import android.widget.VideoView;  
  
public class PlayingVideo extends AppCompatActivity {  
  
 @Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_playing\_video*);  
 Button playVideoButton = (Button) findViewById(R.id.*playvideo*);  
 playVideoButton.setOnClickListener(new OnClickListener() {  
public void onClick(View view) {  
 VideoView videoView = (VideoView) findViewById(R.id.*video*);  
 videoView.setMediaController(new  
MediaController(PlayingVideo.this));  
 videoView.setVideoPath(

"android.resource://com.example.lenovo.playingvideo/"+R.raw.*vid*);  
videoView.requestFocus();  
 videoView.start();  
 }  
 });  
 }  
}

**EXPECTED OUT PUT:**

**EXECUTED OUTPUT:**

**RESULT:**

**8. CREATE APPLICATION BY USING BUILDING MENUS AND STORING DATA**

**AIM:**Design the Application for Menus and Action Bar

**PROCEDURE**:

Step 1: Create a new project and name it MenuExample

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

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

Step 2: Now Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add the following code:

14. Design the Application for Menus and Action Bar.

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

<android.support.constraint.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.lenovo.menuactiobar.MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_conten"

android:text="Hello World!"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="paren"

app:layout\_constraintTop\_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>

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

In this step we add the code to display menu on action bar

package com.example.lenovo.menuactionbar;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle

import android.view.Menu;

import android.view.MenuItem;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main\_menu,menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch(item.getItemId()){

case R.id.settings:

Toast.makeText(this,"Settings Selected" , Toast.LENGTH\_LONG).show();

return true;

case R.id.share:

Toast.makeText(this,"Share Selected" , Toast.LENGTH\_LONG).show();

return true;

case R.id.refresh:

Toast.makeText(this,"Refresh Selected" , Toast.LENGTH\_LONG).show();

return true;

case R.id.search:

Toast.makeText(this,"Search Selected" , Toast.LENGTH\_LONG).show();

return true;

case R.id.help:

Toast.makeText(this,"Help Selected" , Toast.LENGTH\_LONG).show();

return true;

default:

return super.onOptionsItemSelected(item);

}

}

}

Step 4: Open res ->values -> strings.[xml](https://abhiandroid.com/ui/xml)

In this step we show string file which is used to store string data of an app.

strings.[xml](https://abhiandroid.com/ui/xml)

<resources>

<string name="app\_name">Menuactiobar</string>

<string name="settings">Settings</string>

<string name="share">Share</string>

<string name="search">Search</string>

<string name="refresh">Refresh</string>

<string name="help">Help</string>

</resources>

Step 5:

Add menu resource directory then add menu resource file

Open res ->menu-> Main\_menu.xml

In this step we show string file which is used to store string data of an app.

 Main\_menu.xml

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:id="@+id/settings"

app:showAsAction="never"

android:title="@string/settings" />

<item

android:id="@+id/refresh"

app:showAsAction="never"

android:title="@string/refresh" />

<item

android:id="@+id/share"

app:showAsAction="never"

android:title="@string/share" />

<item

android:id="@+id/search"

app:showAsAction="never"

android:title="@string/search" />

<item

android:id="@+id/help"

app:showAsAction="never"

android:title="@string/help" />

</menu>

**EXPECTED OUT PUT:**

**EXECUTED OUTPUT:**

**RESULT:**

**9. Design the Application for Menus and Action Bar.**

**AIM**: Design the application to Menus and Action Bar

* The following steps involved in creating a custom ActionBar for the MainActivity of an application.

**Procedure**:

Step 1: Default ActionBar

As mentioned earlier, every android app contains an ActionBar by default. This pre-included ActionBar display title for the current activity that is managed by the Main.xml file. The string value of the application’s title is provided by @string/app\_name resource present under the application nodes.

<application

…..

…..

android:label=”@string/app\_name”

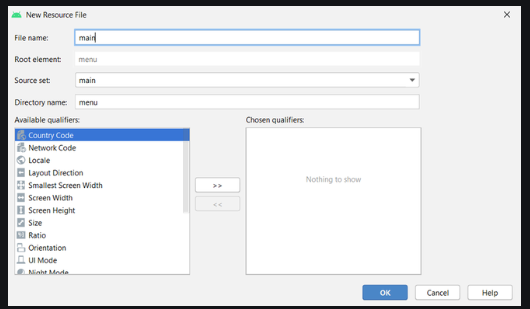
…..

</application>

Step 2: Creating a new directory and design items of ActionBar

To code the elements of ActionBar, create a new directory in the resource folder of the application project files. Right-click on the res folder and selects New -> Directory. Give the name “menu” to the new directory.

Further, create a new Menu Resource File by right click on the menu directory. As the ActionBar is being created for the main Activity, type the name as “main” to the Menu Resource File. With this, a new file named “main.xml” must be created under the menu directory. In this file, one can declare the items which will be displayed as the action buttons of the ActionBar.



For every menu items, the following attributes are needed to be configured:

android:title: Its value contains the title of the menu item that will be displayed when a user clicks and holds that item in the app.

android:id: A unique ID for the menu item that will be used to access it anywhere in the whole application files.

android:orderInCategory: The value of this attribute specify the item’s position in the ActionBar. There are two ways to define the position of different menu items. The first one is to provide the same value of this attribute for all items and the position will be defined in the same order as they are declared in the code. The second way is to provide a different numeric value for all items and then the items will position themselves according to ascending order of this attribute’s value.

app:showAsAction: This attribute defines how the item is going to be present in the action bar. There are four possible flags to choose from:

a. always: To display the item in the ActionBar all the time.

b. ifRoom: To keep the item if space is available.

c. never: With this flag, the item will be not be displayed as an icon in ActionBar, but will be present in the overflow menu.

d. withText: To represent an item as both icon and the title, one can append this flag with the always or ifRoom flag(always|withText or ifRoom|withText).

android:icon: The icon of an item is referenced in the drawable directories through this attribute.

Menu.XML

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

<menu xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:android="http://schemas.android.com/apk/res/android">

<!-- action button for search -->

<item android:title="search"

android:id="@+id/search"

android:orderInCategory="100"

app:showAsAction="ifRoom"

android:icon="@drawable/search\_icon"/>

<!-- action button for refresh -->

<item android:title="refresh"

android:id="@+id/refresh"

android:orderInCategory="100"

app:showAsAction="ifRoom"

android:icon="@drawable/refresh\_icon"/>

<!-- action button for copy -->

<item android:title="copy"

android:id="@+id/copy"

android:orderInCategory="100"

app:showAsAction="never"

android:icon="@drawable/copy\_icon"/>

</menu>

Step 3: Working with activity\_main.xml file

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

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#168BC34A"

tools:context=".MainActivity">

<!-- Adding a TextView in the activity -->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello!!"

android:textColor="#000000"

android:textSize="24sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

Step 4: Working with the Activity File

The items of an ActionBar is designed with a purpose to perform some operations. Those operations/actions of the items are declared in that Activity file for which the ActionBar has been designed. In this example, the target activity is the MainActivity file. Further, the custom title, subtitle, and application logo are also defined in this file. Below is the proper code to design all mentioned items and to display a toast message when a user clicks on the items of ActionBar.

MainActivity.java

import androidx.annotation.NonNull;

import androidx.appcompat.app.ActionBar;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate( Bundle savedInstanceState ) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// calling this activity's function to

// use ActionBar utility methods

ActionBar actionBar = getSupportActionBar();

// providing title for the ActionBar

actionBar.setTitle(" GfG | Action Bar");

// providing subtitle for the ActionBar

actionBar.setSubtitle(" Design a custom Action Bar");

// adding icon in the ActionBar

actionBar.setIcon(R.drawable.app\_logo);

// methods to display the icon in the ActionBar

actionBar.setDisplayUseLogoEnabled(true);

actionBar.setDisplayShowHomeEnabled(true);

}

// method to inflate the options menu when

// the user opens the menu for the first time

@Override

public boolean onCreateOptionsMenu( Menu menu ) {

getMenuInflater().inflate(R.menu.main, menu);

return super.onCreateOptionsMenu(menu);

}

// methods to control the operations that will

// happen when user clicks on the action buttons

@Override

public boolean onOptionsItemSelected( @NonNull MenuItem item ) {

switch (item.getItemId()){

case R.id.search:

Toast.makeText(this, "Search Clicked", Toast.LENGTH\_SHORT).show();

break;

case R.id.refresh:

Toast.makeText(this, "Refresh Clicked", Toast.LENGTH\_SHORT).show();

break;

case R.id.copy:

Toast.makeText(this, "Copy Clicked", Toast.LENGTH\_SHORT).show();

break;

}

return super.onOptionsItemSelected(item);

}

}

**EXPECTED OUT PUT:**

**EXECUTED OUTPUT:**

**RESULT:**

**10. CREATE APPLICATIONBY USING DROP-DOWN LIST ACTION BAR**

**AIM:**Design the application to display the Drop-Down List Action Bar

**PROCEDURE**:

Step 1: Create a new project and name it Dropdownlistactionbarexample

In this step we create a new project in android studio by filling all the necessary details of the app like app name, package name, api versions etc.

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

Step 2: Now Open res -> layout -> activity\_main.[xml](https://abhiandroid.com/ui/xml) (or) main.[xml](https://abhiandroid.com/ui/xml) and add the following code:

15. Design the application to display the Drop-Down List Action Bar.

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
<android.support.v7.widget.Toolbar  
android:id="@+id/toolbar"  
android:layout\_width="match\_parent"  
android:layout\_height="wrap\_content"  
android:layout\_alignParentLeft="true"  
android:layout\_alignParentStart="true"  
android:layout\_alignParentTop="true"  
android:background="@color/colorPrimary"  
app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">  
  
<Spinner  
 android:id="@+id/spinner"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignLeft="@+id/toolbar"  
 android:layout\_alignStart="@+id/toolbar"  
 android:layout\_below="@+id/toolbar"  
 android:layout\_gravity="end"  
 android:layout\_marginEnd="10dp"  
 android:layout\_marginRight="10dp"  
 app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>  
</android.support.v7.widget.Toolbar>  
</RelativeLayout>

Step 3: Open res ->values -> Cust0m spinner item.[xml](https://abhiandroid.com/ui/xml)

Custom\_spinner\_item.xml

<?xml version="1.0" encoding="utf-8"?>  
<TextView android:textColor="@android:color/white"  
 android:layout\_height="match\_parent"  
 android:layout\_width="match\_parent"  
 xmlns:android="http://schemas.android.com/apk/res/android">  
</TextView>

Step 4: Open res ->values -> strings.[xml](https://abhiandroid.com/ui/xml)

In this step we show string file which is used to store string data of an app.

Strings.xml

<?xml version="1.0"?>  
<resources>  
<string name="app\_name">DropDown ActionBar</string>  
<string-array name="names">  
<item>IV-CSE-A</item>  
<item>IV-CSE-B</item>  
<item>IV-CSE-C</item>  
<item>IV-CSE-D</item>  
</string-array>  
</resources>

Styles.xml

<resources>  
<!-- Base application theme. -->  
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">  
<!-- Customize your theme here. -->  
<item name="colorPrimary">@color/colorPrimary</item>  
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>  
<item name="colorAccent">@color/colorAccent</item>  
</style>  
</resources>

Step 5: Now Open  app -> java-> package -> MainActivity.java

In this step we add the code to initiate the check boxes we created. And then we perform click event on button and display the text for selected check boxes using a TextView

package com.example.lenovo.dropdownlist;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.support.v7.widget.Toolbar;  
import android.widget.Toast;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.Spinner;  
  
public class ListActionBarAppActivity extends AppCompatActivity {  
 Toolbar myToolbar;  
 Spinner mySpinner;  
 @Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_list\_action\_bar\_app);  
myToolbar = (Toolbar) findViewById(R.id.toolbar);  
mySpinner = (Spinner) findViewById(R.id.spinner);  
myToolbar.setTitle(getResources().getString(R.string.app\_name));  
 ArrayAdapter<String>myAdapter = new ArrayAdapter<String>(ListActionBarAppActivity.this,  
 R.layout.custom\_spinner\_item,getResources().getStringArray(R.array.names));  
 myAdapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);  
mySpinner.setAdapter(myAdapter);  
mySpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
public void onItemSelected(AdapterView<?>adapterView, View view, int i, long l) {  
 Toast.makeText(ListActionBarAppActivity.this,  
mySpinner.getSelectedItem().toString(),  
 Toast.LENGTH\_SHORT).show();  
 }  
 @Override  
public void onNothingSelected(AdapterView<?>adapterView) {  
 }  
 });  
 }  
}

**EXPECTED OUT PUT:**

**EXECUTED OUTPUT:**

**RESULT:**

**11. Develop an Android Application to Display Options Elements Using ListView**

**Aim:** To Develop an Android Application to Display Options Elements Using ListView

**Procedure**: Create a new Android project called ListViewApp with two controls: ListView and TextView. ListView is populated through string resources to display a list of items for user to select from. The item or option selected from ListView is displayed via TextView control.

* Open the String resource file /res/values/array.xml, and add a string-array structure called fruits that list various fruits that we want to display via the ListView control. After we add a string-array, the strings.xml file appears as shown below.

**array.xml**

<resources>

<string name="app\_name">ListViewApp</string>

<string name="menu\_settings">Settings</string>

<string name="title\_activity\_list\_view\_app">ListViewAppActivity</string>

<string-array name="fruits">

<item>Apple</item>

<item>Mango</item>

<item>Orange</item>

<item>Grapes</item>

<item>Banana</item>

</string-array>

</resources>

* After we define fruits, Open the layout file main.xml and define ListView and TextView in activity\_iist\_view\_app.xml as shown below.

activity\_iist\_view\_app.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools=<http://schemas.android.com/tools>

android:orientation="vertical" android:layout\_width="match\_parent" android:layout\_height="wrap\_content">

<ListView android:id="@+id/fruits\_list"

android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:entries="@array/fruits" android:drawSelectorOnTop="false"/>

<TextView android:id="@+id/selectedopt" android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"/>

</LinearLayout>

* To display the option selected from the ListView in the TextView control, we write the code shown below into the Java activity file ListViewAppActivity.java.

package com.androidunleashed.listviewapp;

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

import android.widget.ListView;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.view.View;

public class ListViewAppActivity extends Activity

{

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState); setContentView(R.layout.activity\_list\_view\_app);

final String[] fruitsArray = getResources().getStringArray(R.array.*fruits*); final TextView selectedOpt=(TextView)findViewById(R.id.*selectedopt*); ListView fruitsList = (ListView)findViewById(R.id.*fruits\_list*); fruitsList.setOnItemClickListener(new OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View v, int position,

long id)

{

selectedOpt.setText("You have selected "+fruitsArray[position]);

}

});

}

}

**EXPECTED OUT PUT:**

**EXECUTED OUTPUT:**

**RESULT:**

**12. Develop an Android Application to Display Options Elements Using Spinner Control.**

* We define two resources, one to display a prompt in the spinner control and the other to display a list of choices. To display a prompt in the spinner control, we define a string resource. Open the strings.xml file from the res/values folder and define a string resource in it.

**strings.xml**

<resources>

<string name="app\_name">SpinnerApp</string>

<string name="menu\_settings">Settings</string>

<string name="title\_activity\_spinner\_app">SpinnerAppActivity</string>

<string name="choose\_msg">Choose a fruit</string>

</resources>

* To add a new xml file to the res/values folder, right-click on the res/values folder in the package explorer window and select the new, android xml file option. Call the file arrays and then click the finish button. The code written in arrays.xml is shown below

**arrays.xml**

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

<resources>

<string-array name="fruits">

<item>Apple</item>

<item>Mango</item>

<item>Orange</item>

<item>Grapes</item>

<item>Banana</item>

</string-array>

</resources>

* We see that a string-array called fruits is defined, consisting of five elements, Apple, Mango, Orange, Grapes and Banana. These array elements are used to display options in the spinner control. To display spinner control, write code in layout file activity\_spinner\_app.xml, as shown below

activity\_spinner\_app.xml

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<Spinner android:id="@+id/spinner"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:prompt="@string/choose\_msg"

android:entries="@array/fruits"/>

<TextView android:id="@+id/selectedopt"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"/>

</LinearLayout>

* To do, write the code shown below in spinnerAppActivity.java

package com.androidunleashed.Spinnerapp;

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

import android.widget.Spinner;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemSelectedListener;

public class SpinnerAppActivity extends Activity

{

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_spinner\_app);

final TextView selectedOpt=(TextView)findViewById(R.id.*selectedopt*); Spinner

spin=(Spinner)findViewById(R.id.spinner);

final String[] fruitArray=getResources().getStringArray(R.array.*fruits*); spin.setOnItemSelectedListener(new OnItemSelectedListener(){

public void onItemSelected(AdapterView<?> parent, View v,int position,long id) { selectedOpt.setText("you have selected" + fruitArray[position]);

}

public void onNothingSelected(AdapterView<?> parent)  
 {

selectedOpt.setText("");

}

});

}

}

**EXPECTED OUT PUT:**

**EXECUTED OUTPUT:**

**RESULT:**

**============================THE END======================**