VEMU INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL

Prepared By K.Dhanamjay

15A05711- MOBILE APPLICATION DEVELOPMENT LABORATORY

Regulation – R15

Year / Semester: III / II

INDEX

Aim: Creating the Application by using the Activity class
(onCreate(),onStart(),onResume(),onPause(),onStop(),onDestroy(),onRestart()**)**

Theory:

An activity represents a single screen with a user interface just like window or frame of Java.Android activity is the subclass of ContextThemeWrapper class.

Procedure:

Step 1 : You will use Android studio to create an Android application and name it as HelloWorld under a package com.example.helloworld as explained in the Hello World Example chapter.

Step 2 : Modify main activity file MainActivity.java as explained below. Keep rest of the files unchanged.

Step 3: Run the application to launch Android emulator and verify the result of the changes done in the application.

The **Log.d()** method has been used to generate log messages –

Source Code:

MainActivity.java

```
package com.example.helloworld;
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;

public class MainActivity extends Activity {
  String msg = "Android : ";

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d(msg, "The onCreate() event");
  }
  /** Called when the activity is about to become visible. */
  @Override
  protected void onStart() {
    super.onStart();
    Log.d(msg, "The onStart() event");
  }
  /** Called when the activity has become visible. */
  @Override
  protected void onResume() {
    super.onResume();
    Log.d(msg, "The onResume() event");
  }
  /** Called when another activity is taking focus. */
  @Override
  protected void onPause() {
```

```
    super.onPause();
    Log.d(msg, "The onPause() event");
  }
  /** Called when the activity is no longer visible. */
  @Override
  protected void onStop() {
    super.onStop();
    Log.d(msg, "The onStop() event");
  }
  /** Called just before the activity is destroyed. */
  @Override
  public void onDestroy() {
    super.onDestroy();
    Log.d(msg, "The onDestroy() event");
  }
}
```

An activity class loads all the UI component using the XML file available in ***res/layout*** folder of the project. Following statement loads UI components **from *res/layout/activity_main.xml file***:

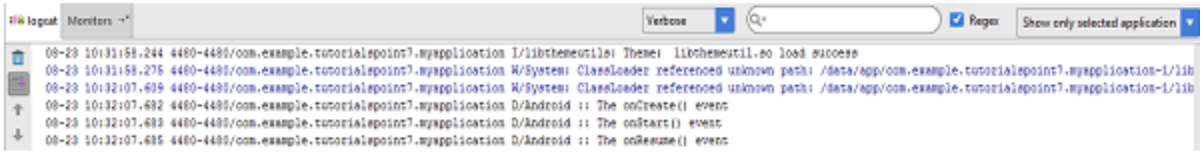
```
setContentView(R.layout.activity_main);
```

Every activity you define for your application must be declared in your ***AndroidManifest.xml*** file and the main activity for your app must be declared in the manifest with an <intent-filter> that includes the MAIN action and LAUNCHER category as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity android:name=".MainActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
                  </intent-filter>
               </activity> </application> </manifest>
```

OUTPUT:

```
08-23 10:32:07.682 4480-4480/com.example.helloworld D/Android :: The onCreate() event
08-23 10:32:07.683 4480-4480/com.example.helloworld D/Android :: The onStart() event
08-23 10:32:07.685 4480-4480/com.example.helloworld D/Android :: The onResume() event
```




```
08-23 10:32:53.230 4480-4480/com.example.helloworld D/Android :: The onPause() event
08-23 10:32:53.294 4480-4480/com.example.helloworld D/Android :: The onStop() event
```

```
08-23 10:34:41.390 4480-4480/com.example.helloworld D/Android :: The onStart() event
08-23 10:34:41.392 4480-4480/com.example.helloworld D/Android :: The onResume() event
```

```
08-23 10:37:24.806 4480-4480/com.example.helloworld D/Android :: The onPause() event
08-23 10:37:25.668 4480-4480/com.example.helloworld D/Android :: The onStop() event
08-23 10:37:25.669 4480-4480/com.example.helloworld D/Android :: The onDestroy() event
```

Aim: Create Application by Using Building Blocks for Android Application Design (Absolute Layout).

Theory:

An Absolute Layout lets you specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.

Following is the content of the modified main activity file **src/com.example.demo/MainActivity.java**.

This file can include each of the fundamental lifecycle methods.

Source Code:

MainActivity.java

```
package com.example.demo;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }

}
```

Following will be the content of **res/layout/activity_main.xml** file −

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">

   <Button
      android:layout_width="100dp"
      android:layout_height="wrap_content"
      android:text="OK"
      android:layout_x="50px"
      android:layout_y="361px" />
   <Button
      android:layout_width="100dp"
      android:layout_height="wrap_content"
      android:text="Cancel"
      android:layout_x="225px"
      android:layout_y="361px" />
</AbsoluteLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">demo</string>
   <string name="action_settings">Settings</string>
</resources>
```

Output:

Aim: Create the audio Player application

Theory:

We can play and control the audio files in android by the help of MediaPlayer class. Here, we are going to see a simple example to play the audio file. In the next page, we will see the example to control the audio playback like start, stop, pause etc.

Let's write the code of to play the audio file. Here, we are going to play maine.mp3 file located inside the sdcard/Music directory.

Source code:

File: MainActivity.java

```
package com.example.audiomediaplayer1;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.MediaController;
import android.widget.VideoView;

public class MainActivity extends Activity {

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

        MediaPlayer mp=new MediaPlayer();
        try{
            mp.setDataSource("/sdcard/Music/maine.mp3");//Write your location here
            mp.prepare();
            mp.start();

        }catch(Exception e){e.printStackTrace();}

    }
     @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```

Drag three buttons from pallete to start, stop and pause the audio play. Now the xml file will look like this:

Source Code:

activity_main.xml:

```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
 tools:context=".MainActivity" >
 <TextView
  android:id="@+id/textView1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentTop="true"
  android:text="Audio Controller" />
 <Button
  android:id="@+id/button1"
  style="?android:attr/buttonStyleSmall"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignLeft="@+id/textView1"
  android:layout_below="@+id/textView1"
  android:layout_marginTop="48dp"
  android:text="start" />
 <Button
  android:id="@+id/button2"
  style="?android:attr/buttonStyleSmall"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignTop="@+id/button1"
  android:layout_toRightOf="@+id/button1"
  android:text="pause" />
 <Button
  android:id="@+id/button3"
  style="?android:attr/buttonStyleSmall"
  android:layout_width="wrap_content"
    android:layout_height="wrap_content"
          android:layout_alignTop="@+id/button2"
          android:layout_toRightOf="@+id/button2"
          android:text="stop" />
      </RelativeLayout>
```

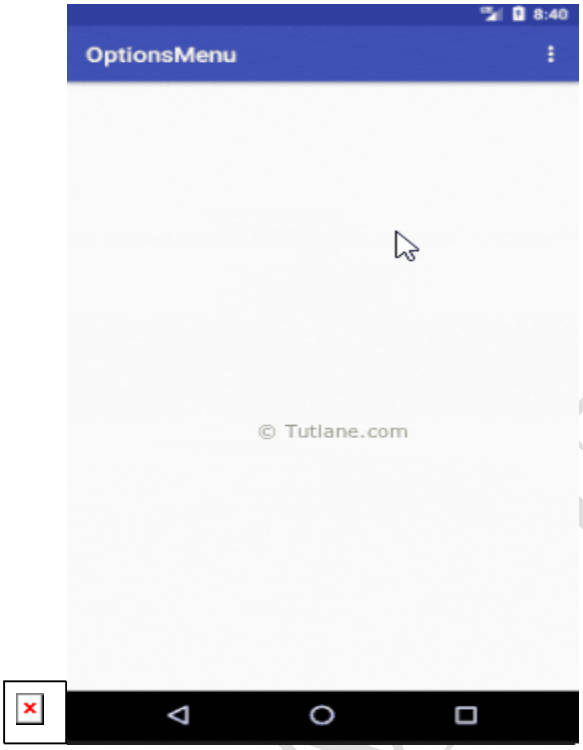
Let's write the code to start, pause and stop the audio player.

File: MainActivity.java

```
package com.example.audioplay;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Environment;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {
    Button start,pause,stop;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        start=(Button)findViewById(R.id.button1);
        pause=(Button)findViewById(R.id.button2);
        stop=(Button)findViewById(R.id.button3);
        //creating media player
        final MediaPlayer mp=new MediaPlayer();
        try{
                //you can change the path, here path is external direc
tory(e.g. sdcard) /Music/maine.mp3
        mp.setDataSource(Environment.getExternalStorageDirectory().get
Path()+"/Music/maine.mp3");

        mp.prepare();
        }catch(Exception e){e.printStackTrace();}

        start.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                mp.start();
            }
        });
        pause.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                mp.pause();
            }
        });
```

```
        stop.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                mp.stop();
            }
        });
    }
}
```

Output:



Aim: Creating the Application using CheckBox

Theory:

- **Android CheckBox** is a type of two state button either checked or unchecked.
- There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.
- Android CheckBox class is the subclass of CompoundButton class.
- Drag the three checkboxes and one button for the layout. Now the activity_main.xml file will look like this:

Source code:

File: activity_main.xml

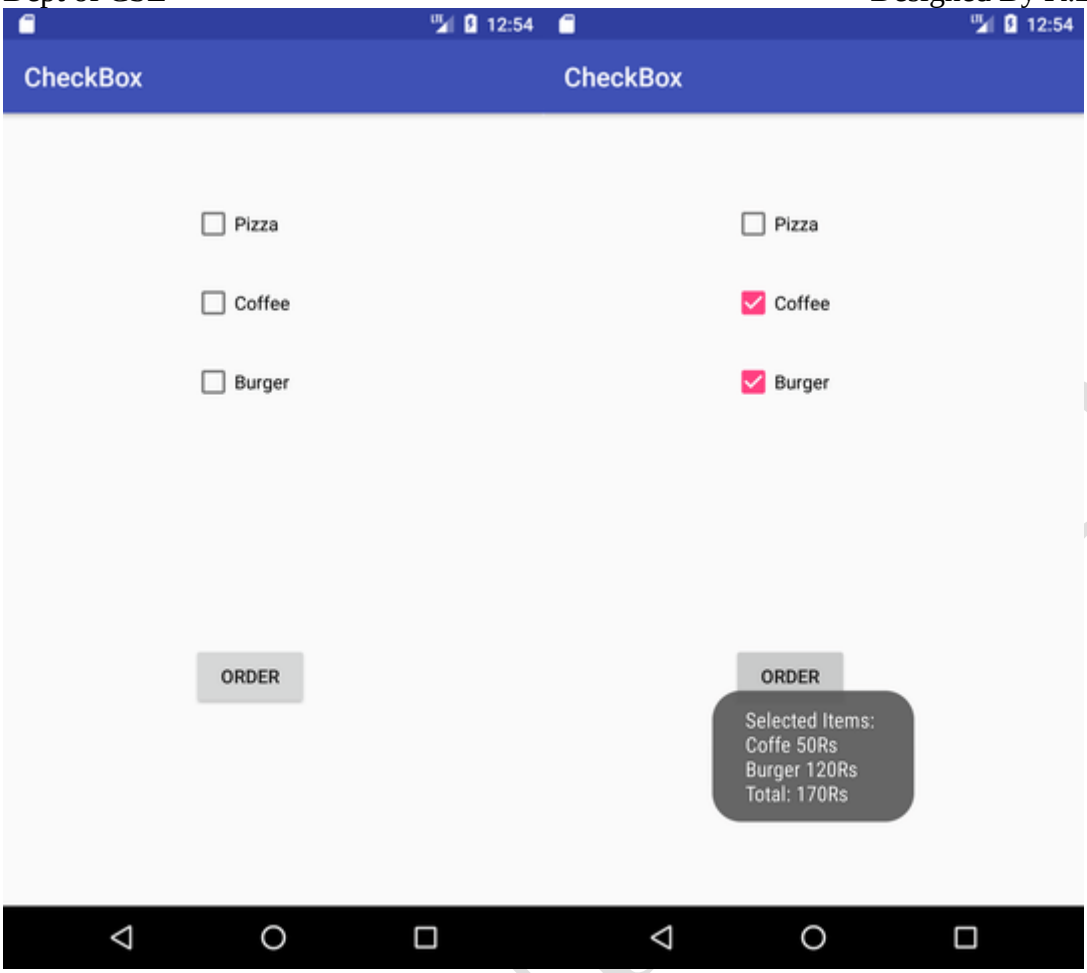
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android
.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
      <CheckBox
      android:id="@+id/checkBox"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
       android:text="Pizza"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
     <CheckBox
      android:id="@+id/checkBox2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Coffee"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/checkBox" />
     <CheckBox
      android:id="@+id/checkBox3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Burger"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/checkBox2" />
   <Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Order"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/checkBox3" />
 </android.support.constraint.ConstraintLayout>
```

File: MainActivity.java

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

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

public class MainActivity extends AppCompatActivity {
   CheckBox pizza,coffe,burger;
   Button buttonOrder;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      addListenerOnButtonClick();
   }
   public void addListenerOnButtonClick(){
      //Getting instance of CheckBoxes and Button from the activty_main.xml file
      pizza=(CheckBox)findViewById(R.id.checkBox);
      coffe=(CheckBox)findViewById(R.id.checkBox2);
      burger=(CheckBox)findViewById(R.id.checkBox3);
      buttonOrder=(Button)findViewById(R.id.button);
       //Applying the Listener on the Button click
      buttonOrder.setOnClickListener(new View.OnClickListener(){
          @Override
         public void onClick(View view) {
            int totalamount=0;
            StringBuilder result=new StringBuilder();
            result.append("Selected Items:");
            if(pizza.isChecked()){
               result.append("\nPizza 100Rs");
               totalamount+=100;
            }
            if(coffe.isChecked()){
               result.append("\nCoffe 50Rs");
               totalamount+=50;
            }
            if(burger.isChecked()){
               result.append("\nBurger 120Rs");
               totalamount+=120;
            }
            result.append("\nTotal: "+totalamount+"Rs");
            //Displaying the message on the toast
            Toast.makeText(getApplicationContext(), result.toString(), Toast.LENGTH_LONG).show();
         }

      });
   }
}
```

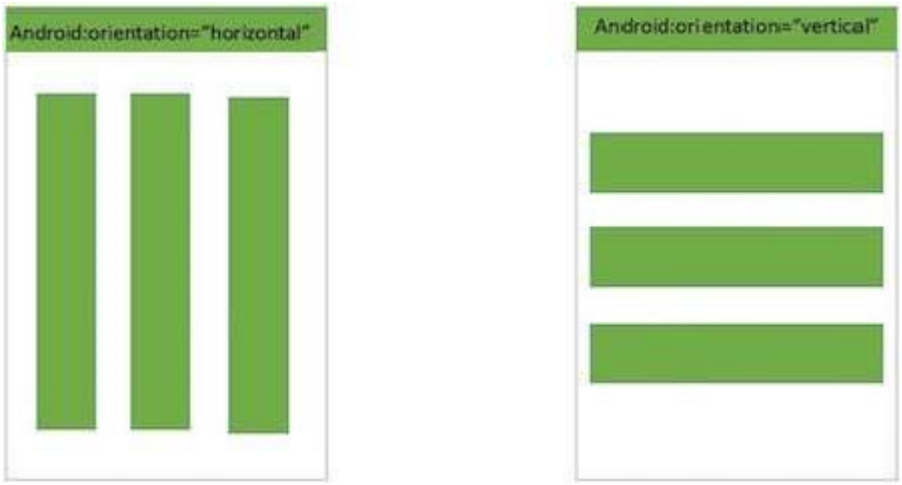
Output:


12:54
CheckBox
Pizza
Coffee
Burger
ORDER
12:54
CheckBox
Pizza
Coffee
Burger
ORDER
Selected Items:
Coffe 50Rs
Burger 120Rs
Total: 170Rs

Aim: Create Application by Using Building Blocks for Android Application Design (Linear Layout).

Theory:

Android LinearLayout is a view group that aligns all children in either *vertically* or *horizontally*.



Source code:

MainActivity.java

```
package com.example.demo;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

Following will be the content of **res/layout/activity_main.xml** file –

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/btnStartService"
      android:layout_width="270dp"
      android:layout_height="wrap_content"
      android:text="start_service"/>
```

```
   <Button android:id="@+id/btnPauseService"
      android:layout_width="270dp"
      android:layout_height="wrap_content"
      android:text="pause_service"/>

   <Button android:id="@+id/btnStopService"
      android:layout_width="270dp"
      android:layout_height="wrap_content"
      android:text="stop_service"/>

</LinearLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">HelloWorld</string>
   <string name="action_settings">Settings</string>
</resources>
```

Output:

Aim: Creating the application using **Android RadioGroup**

Theory:

In android, **Radio Group** is used to group one or more radio buttons into separate groups based on our requirements.

If we group Radio Buttons using **RadioGroup**, at a time only one item can be selected from the group of radio buttons. In case, if we select one radio button that belongs to a radio group will unselect all other previously selected radio buttons within in the same group.

Initially, all the radio buttons of radio group are in unchecked state, once we select a radio button then it's not possible for us to uncheck it like CheckBox control.

```
<?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">
<RadioGroup
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical">
  <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Java"/>
  </RadioGroup>
</RelativeLayout>
```

Source code:

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?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">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="150dp"
    android:layout_marginLeft="100dp"
    android:textSize="18dp"
    android:text="Select Your Course"
    android:textStyle="bold"
    android:id="@+id/txtView"/>
<RadioGroup
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  android:id="@+id/rdGroup"
  android:layout_below="@+id/txtView">
```

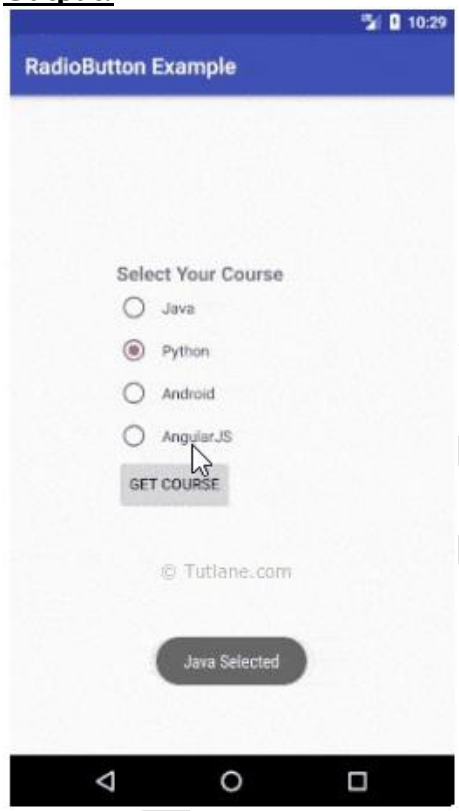
```
    <RadioButton
        android:id="@+id/rdbJava"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginLeft="100dp"
        android:text="Java"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton
        android:id="@+id/rdbPython"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginLeft="100dp"
        android:text="Python"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton
        android:id="@+id/rdbAndroid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginLeft="100dp"
        android:text="Android"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton
        android:id="@+id/rdbAngular"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginLeft="100dp"
        android:text="AngularJS"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
    <Button
        android:id="@+id/getBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_below="@+id/rdGroup"
        android:text="Get Course" />
</RelativeLayout>
```

MainActivity.java

```
package com.tutlane.radiobuttonexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
   RadioButton android, java, angular, python;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      android = (RadioButton)findViewById(R.id.rdbAndroid);
      angular = (RadioButton)findViewById(R.id.rdbAngular);
      java = (RadioButton)findViewById(R.id.rdbJava);
      python = (RadioButton)findViewById(R.id.rdbPython);
      Button btn = (Button)findViewById(R.id.getBtn);
      btn.setOnClickListener(new View.OnClickListener() {


         @Override
         public void onClick(View v) {
            String result = "Selected Course: ";
            result+=
(android.isChecked())?"Android":(angular.isChecked())?"AngularJS":(java.isChecked())?"Java":(python.is
Checked())?"Python":"";
            Toast.makeText(getApplicationContext(), result, Toast.LENGTH_SHORT).show();
         }
      });
   }
   public void onRadioButtonClicked(View view) {
      boolean checked = ((RadioButton) view).isChecked();
      String str="";
      // Check which radio button was clicked
      switch(view.getId()) {
         case R.id.rdbAndroid:
            if(checked)
            str = "Android Selected";
            break;
         case R.id.rdbAngular:
            if(checked)
            str = "AngularJS Selected";
```

```
            break;


          case R.id.rdbJava:
            if(checked)
            str = "Java Selected";
            break;
          case R.id.rdbPython:
            if(checked)
            str = "Python Selected";
            break;
      }
      Toast.makeText(getApplicationContext(), str, Toast.LENGTH_SHORT).show();
   }
}
```

Output:



Aim: Create Application by Using Building Blocks for Android Application Design (RelativeLayout).

Theory:

Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

Procedure:

Step 1	You will use Android Studio IDE to create an Android application and name it as *demo* under a package *com.example.demo* as explained in the *Hello World Example* chapter.
Step 2	Modify the default content of *res/layout/activity_main.xml* file to include few widgets in Relative layout.
Step 3	Define required constants in *res/values/strings.xml* file
Step 4	Run the application to launch Android emulator and verify the result of the changes done in the application.

Source Code :

MainActivity.java

```
package com.example.demo;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }

}
```

Following will be the content of **res/layout/activity_main.xml** file −

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:paddingLeft="16dp"
  android:paddingRight="16dp" >
    <EditText
    android:id="@+id/name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/reminder" />
      <LinearLayout
```

```
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/name">
    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Button"
      android:id="@+id/button" />
     <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Button"
      android:id="@+id/button2" />
     </LinearLayout>
</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants –

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="action_settings">Settings</string>
   <string name="reminder">Enter your name</string>
</resources>
```

Output:

Aim: Creating the Application using Spinner

Theory:

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

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

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

Source code:

File: activity_main.xml

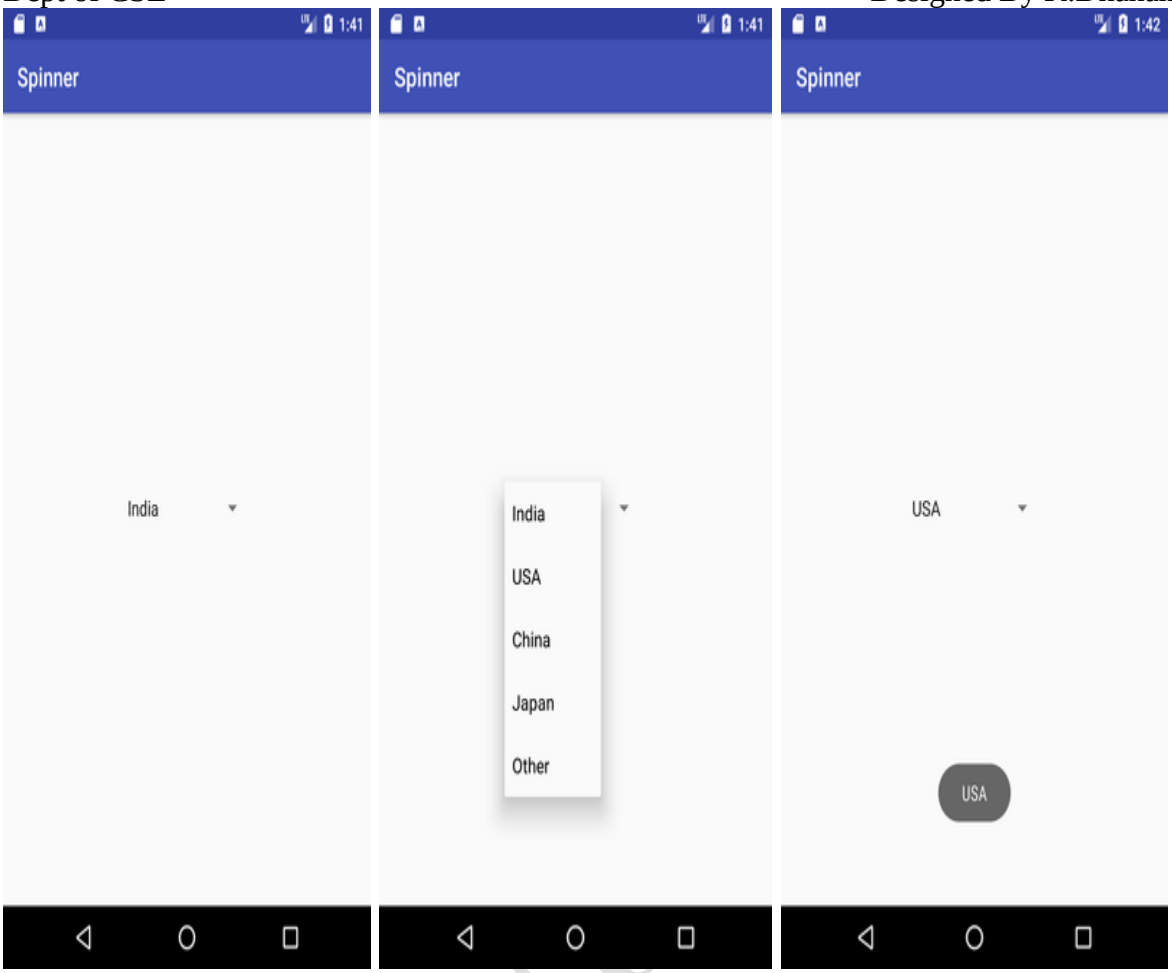
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.and
roid.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

     <Spinner
      android:id="@+id/spinner"
      android:layout_width="149dp"
      android:layout_height="40dp"
      android:layout_marginBottom="8dp"
      android:layout_marginEnd="8dp"
      android:layout_marginStart="8dp"
      android:layout_marginTop="8dp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.502"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.498" />

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

File: MainActivity.java

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

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity implements
        AdapterView.OnItemSelectedListener {
    String[] country = { "India", "USA", "China", "Japan", "Other"};
     @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       //Getting the instance of Spinner and applying OnItemSelectedListener on it
        Spinner spin = (Spinner) findViewById(R.id.spinner);
        spin.setOnItemSelectedListener(this);
         //Creating the ArrayAdapter instance having the country list
        ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,c
ountry);
        aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        //Setting the ArrayAdapter data on the Spinner
        spin.setAdapter(aa);
      }
      //Performing action onItemSelected and onNothing selected
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id)
{
        Toast.makeText(getApplicationContext(),country[position] , Toast.LENGTH_LONG).
show();
    }
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
    }
}
```

Output:


1:41
Spinner
India
1:41
Spinner
India
USA
China
Japan
Other
1:42
Spinner
USA
USA

Aim: Create Video Player android application

Theory:

By the help of **MediaController** and **VideoView** classes, we can play the video files in android.

MediaController class

The **android.widget.MediaController** is a view that contains media controls like play/pause, previous, next, fast-forward, rewind etc.

VideoView class

The **android.widget.VideoView** class provides methods to play and control the video player.

activity_main.xml

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

Source Code:

File: activity_main.xml

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

   <VideoView
      android:id="@+id/videoView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_centerVertical="true" />

</RelativeLayout>
```

File: MainActivity.java

```
package com.example.video1;

import android.net.Uri;
import android.os.Bundle;
```

```
import android.app.Activity;
import android.view.Menu;
import android.widget.MediaController;
import android.widget.VideoView;

public class MainActivity extends Activity {

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

        VideoView videoView =(VideoView)findViewById(R.id.videoView1);
                    //Creating MediaController
        MediaController mediaController= new MediaController(this);
            mediaController.setAnchorView(videoView);

             //specify the location of media file
       Uri uri=Uri.parse(Environment.getExternalStorageDirectory().getPath()+"/media/1.mp4");
                               //Setting MediaController and URI, then starting the videoView
          videoView.setMediaController(mediaController);
          videoView.setVideoURI(uri);
          videoView.requestFocus();
          videoView.start();

    }
      @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```

Aim: Creating the Application by using Text Edit control

Theory:

EditText is a standard entry widget in android apps. It is an overlay over TextView that configures itself to be editable. EditText is a subclass of TextView with text editing operations. **We often use EditText in our applications in order to provide an input or text field, especially in forms.**

Procedure:

Step1: Create a new project in Android Studio and name it EditTextExample.
Step2: Now open res -> layout -> xml (or) **activity_main.xml** and add following code. In this code we have added multiple edittext and a button with onclick functionality.

Step3: Now open app -> java -> package -> **MainActivity.java** and add the below code. In this we just fetch the text from the edittext, further with the button click event a toast will show the text fetched before.

Source code:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.edittextexample.MainActivity">

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/name"
        android:inputType="textPersonName"
        android:selectAllOnFocus="true" />

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/password_0_9"
        android:inputType="numberPassword" />

    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/e_mail"
        android:inputType="textEmailAddress" />

    <EditText
```

```
        android:id="@+id/editText4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/date"
        android:inputType="date" />

    <EditText
        android:id="@+id/editText5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/contact_number"
        android:inputType="phone" />

    <Button
        android:id="@+id/button"
        style="@android:style/Widget.Button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/submit"
        android:textSize="16sp"
        android:textStyle="normal|bold" />
</RelativeLayout>
```

MainActivity.java

```
package com.example.edittextexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

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

        submit.setOnClickListener(new View.OnClickListener() {
```

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

Output:

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