Stream: B.SC (computer science)

Subject : Fundamentals of Mobile Programming

Unit - 3 | Android User Interface Design Essentials

UI Elements :

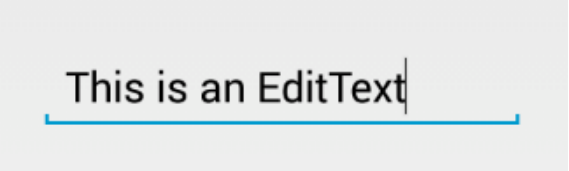
- There are number of UI controls provided by Android that allow you to build the graphical user interface for your app.

Sr.No.	UI Control & Description
1	TextView This control is used to display text to the user.
2	EditText EditText is a predefined subclass of TextView that includes rich editing capabilities.
3	AutoCompleteTextView The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.
4	Button A push-button that can be pressed, or clicked, by the user to perform an action.
5	ImageButton An ImageButton is an AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.
6	CheckBox An on/off switch that can be toggled by the user. You should use check box when presenting users with a group of selectable options that are not mutually exclusive.
7	ToggleButton An on/off button with a light indicator.
8	RadioButton The RadioButton has two states: either checked or unchecked.
9	RadioGroup A RadioGroup is used to group together one or more RadioButtons.
10	ProgressBar The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.

11	Spinner A drop-down list that allows users to select one value from a set.
12	TimePicker The TimePicker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.
13	DatePicker The DatePicker view enables users to select a date of the day.

1) EditText :

- **Edittext** is one of many such widgets which can be used to retrieve text data from user.
- **Edittext** refers to the widget that displays an empty textfield in which a user can enter the required text and this text is further used inside our application.



Syntax :

```
<SomeLayout>
   .
   .
   <Edittext
      android:SomeAttribute1 = "Value of attribute1"
      android:SomeAttribute2 = "Value of attribute2"
      .
      .
      android:SomeAttributeN = "Value of attributeN"/>
   .
   .
</SomeLayout>
```

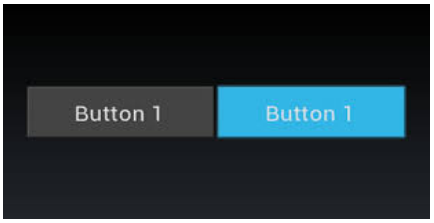
- **XML Attributes of Edittext in Android**

Attributes	Description
android:id	Used to uniquely identify the control
android:gravity	Used to specify how to align the text like left, right, center, top, etc.
android:hint	Used to display the hint text when text is empty

Attributes	Description
android:text	Used to set the text of the EditText
android:textSize	Used to set size of the text.
android:textColor	Used to set color of the text.
android:textStyle	Used to set style of the text. For example, bold, italic, bolditalic, etc.
android:textAllCaps	Used this attribute to show the text in capital letters.
android:width	It makes the TextView be exactly this many pixels wide.
android:height	It makes the TextView be exactly this many pixels tall.
android:maxWidth	Used to make the TextView be at most this many pixels wide.
android:minWidth	Used to make the TextView be at least this many pixels wide.
android:background	Used to set background to this View.
android:backgroundTint	Used to set tint to the background of this view.
android:clickable	Used to set true when you want to make this View clickable. Otherwise, set false.
android:drawableBottom	Used to set drawable to bottom of the text in this view.
android:drawableEnd	Used to set drawable to end of the text in this view.
android:drawableLeft	Used to set drawable to left of the text in this view.
android:drawablePadding	Used to set padding to drawable of the view.
android:drawableRight	Used to set drawable to right of the text in this view.

Attributes	Description
android:drawableStart	Used to set drawable to start of the text in this view.
android:drawableTop	Used to set drawable to top of the text in this view.
android:elevation	Used to set elevation to this view.

2) Button :



- Button is a user interface that is used to perform some action when clicked or tapped.
- It is a very common widget in Android and developers often use it.
- We can perform action on button using different types such as calling listener on button or adding onClick property of button in activity's xml file.

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

- **XML Attributes of Button Widget**

XML Attributes	Description
android:id	Used to specify the id of the view.
android:text	Used to the display text of the button.
android:textColor	Used to the display color of the text.
android:textSize	Used to the display size of the text.
android:textStyle	Used to the display style of the text like Bold, Italic, etc.
android:textAllCaps	Used to display text in Capital letters.
android:background	Used to set the background of the view.

XML Attributes	Description
android:padding	Used to set the padding of the view.
android:visibility	Used to set the visibility of the view.
android:gravity	Used to specify the gravity of the view like center, top, bottom, etc

- **Example of Edit Text & Button :**

activity_main.xml

```
<?xmlversion="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:id="@+id/main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">


   <EditText
      android:id="@+id/editText"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:ems="10"
      android:inputType="text"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.5"
      app:layout_constraintVertical_bias="0.3"/>

   <Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Button"
      app:layout_constraintTop_toBottomOf="@id/editText"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.5"/>
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
package com.example.uielements;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
    private EditText editText;
    private Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText
                = (EditText)findViewById(R.id.editText);
        button
                = (Button)findViewById(R.id.button);

        button.setOnClickListener(
                new View.OnClickListener() {

                    @Override
                    public void onClick(View v)
                    {
                        String name
                                = editText.getText()
                                .toString();
                        Toast.makeText(MainActivity.this,
                                       "Welcome to 2025    "
                                               + name,
                                       Toast.LENGTH_SHORT)
                                .show();
                    }
                });
    }
}
```

Output :

3) TextView :

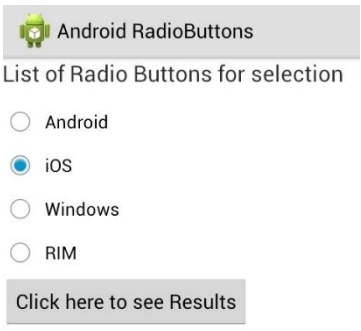
- TextView is a simple widget that is seen in every android application.
- This widget is used to display simple text within the android application. We can add custom styling to the text that we have to show.

- **TextView Attributes :**

Sr.No.	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the control.
2	**android:fontFamily** Font family (named by string) for the text.
3	**android:gravity** Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view.
4	**android:hint** Hint text to display when the text is empty.
5	**android:inputType** The type of data being placed in a text field. Phone, Date, Time, Number, Password etc.
6	**android:maxHeight** Makes the TextView be at most this many pixels tall.
7	**android:maxWidth** Makes the TextView be at most this many pixels wide.
8	**android:minHeight** Makes the TextView be at least this many pixels tall.
9	**android:minWidth**

	Makes the TextView be at least this many pixels wide.
10	**android:password** Whether the characters of the field are displayed as password dots instead of themselves. Possible value either "true" or "false".
11	**android:phoneNumber** If set, specifies that this TextView has a phone number input method. Possible value either "true" or "false".
12	**android:text** Text to display.
13	**android:textAllCaps** Present the text in ALL CAPS. Possible value either "true" or "false".
14	**android:textColor** Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
15	**android:textColorHighlight** Color of the text selection highlight.
16	**android:textSize** Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).
17	**android:textStyle** Style (bold, italic, bolditalic) for the text. You can use or more of the following values separated by '|'. • normal - 0 • bold - 1 • italic - 2
18	**android:typeface** Typeface (normal, sans, serif, monospace) for the text. You can use or more of the following values separated by '|'. • normal - 0 • sans - 1 • serif - 2 • monospace - 3

4) Radio Button :



- RadioButton is a widget used in android which provides functionality to choose a single option from multiple sets of options.
- This is generally used when we want the user to select only one option from the set of given options.

Example :

Activity_main.xml :

```
<?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:id="@+id/main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <!-- Top Section -->
        <TextView
           android:id="@+id/textView1"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_marginTop="30dp"
           android:textSize="22sp"
           android:text="Single Radio Buttons"
           app:layout_constraintTop_toTopOf="parent"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent" />

        <RadioButton
           android:id="@+id/radioButton1"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Radio Button 1"
           android:layout_marginTop="10dp"
           android:textSize="20sp"
           app:layout_constraintTop_toBottomOf="@id/textView1"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent" />

        <RadioButton
           android:id="@+id/radioButton2"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Radio Button 2"
           android:layout_marginTop="10dp"
           android:textSize="20sp"
           app:layout_constraintTop_toBottomOf="@id/radioButton1"
           app:layout_constraintStart_toStartOf="parent"
           app:layout_constraintEnd_toEndOf="parent" />

        <View
           android:id="@+id/separator"
```

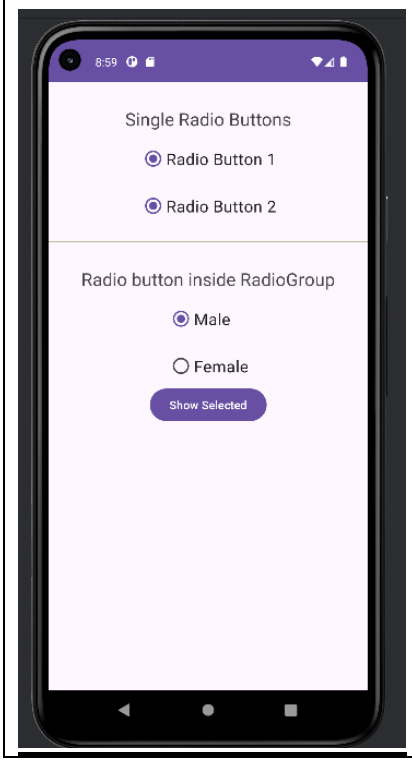
```
        android:layout_width="0dp"
        android:layout_height="1dp"
        android:layout_marginTop="20dp"
        android:background="#B8B894"
        app:layout_constraintTop_toBottomOf="@id/radioButton2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <!-- Bottom Section -->
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:textSize="22sp"
        android:text="Radio button inside RadioGroup"
        app:layout_constraintTop_toBottomOf="@id/separator"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@id/textView2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <RadioButton
            android:id="@+id/radioMale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male"
            android:layout_marginTop="10dp"
            android:textSize="20sp"
            android:contentDescription="Select Male Gender" />

        <RadioButton
            android:id="@+id/radioFemale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"
            android:layout_marginTop="10dp"
            android:textSize="20sp"
            android:contentDescription="Select Female Gender" />
    </RadioGroup>

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

```
            android:layout_height="wrap_content"
            android:text="Show Selected"
            app:layout_constraintTop_toBottomOf="@id/radioGroup"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            android:onClick="onclickbuttonMethod"
            android:contentDescription="Show selected radio button" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

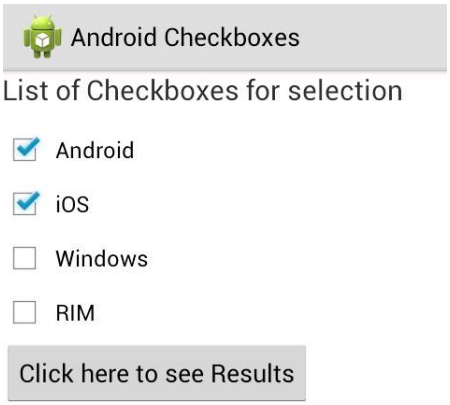
Mainactivity.java

```
package com.example.radio;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
        Button button;
        RadioButton genderradioButton;
        RadioGroup radioGroup;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            radioGroup=(RadioGroup)findViewById(R.id.radioGroup);
        }
        public void onclickbuttonMethod(View v){
            int selectedId = radioGroup.getCheckedRadioButtonId();
            genderradioButton = (RadioButton) findViewById(selectedId);
            if(selectedId==-1){
              Toast.makeText(MainActivity.this,"Nothing selected",
Toast.LENGTH_SHORT).show();
            }
            else{
               Toast.makeText(MainActivity.this,genderradioButton.getText(),
      Toast.LENGTH_SHORT).show();
            }
        }
}
```

Output :



5) Android CheckBox :



- Android CheckBox is a type of two state button either checked or unchecked.
- CheckBox belongs to android.widget.CheckBox class.
- Android CheckBox class is the subclass of **CompoundButton class.**
- It is generally used in a place where user can select one or more than choices from a given list of choices.

- **CheckBox Attributes :**

Sr.No	Attribute & Description
1	**android:autoText** If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
2	**android:drawableBottom** This is the drawable to be drawn below the text.
3	**android:drawableRight** This is the drawable to be drawn to the right of the text.
4	**android:editable** If set, specifies that this TextView has an input method.

5	**android:text** This is the Text to display.
6	**android:background** This is a drawable to use as the background.
7	**android:contentDescription** This defines text that briefly describes content of the view.
8	**android:id** This supplies an identifier name for this view.
9	**android:onClick** This is the name of the method in this View's context to invoke when the view is clicked.
10	**android:visibility** This controls the initial visibility of the view.

- **Methods of CheckBox class :**

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

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

Example :

Activity_main.xml

```
<?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:id="@+id/main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <CheckBox
      android:id="@+id/checkBox"
      android:layout_width="wrap_content"
```

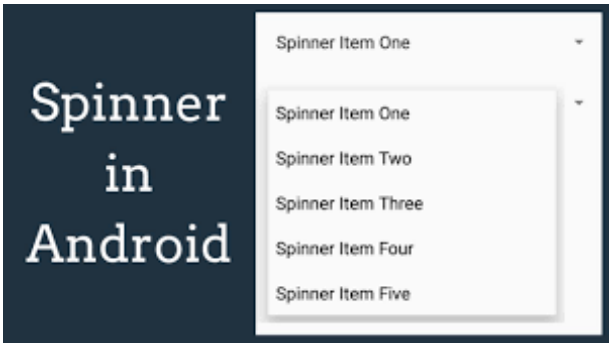
```
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="68dp"
        android:text="Pizza"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

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

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="184dp"
        android:text="Order"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox3" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
package com.example.checkbox;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
```

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
    CheckBox pizza,coffe,burger;
    Button buttonOrder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        addListenerOnButtonClick();
    }
    public void addListenerOnButtonClick() {
        pizza=(CheckBox)findViewById(R.id.checkBox);
        coffe=(CheckBox)findViewById(R.id.checkBox2);
        burger=(CheckBox)findViewById(R.id.checkBox3);
        buttonOrder=(Button)findViewById(R.id.button);

        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();
            }

        });

    }
}
```

6) Android Spinner :



- Android Spinner is a view similar to the dropdown list which is used to select one option from the list of options.
- It provides an easy way to select one item from the list of items and it shows a dropdown list of all values when we click on it.
- The default value of the android spinner will be the currently selected value and by using Adapter we can easily bind the items to the spinner objects.
- Android spinner is associated with **AdapterView**. So you need to use one of the adapter classes with spinner.

- **Different Attributes for Spinner Widget :**

XML attributes	Description
android:id	Used to specify the id of the view.
android:textAlignment	Used to the text alignment in the dropdown list.
android:background	Used to set the background of the view.
android:padding	Used to set the padding of the view.
android:visibility	Used to set the visibility of the view.
android:gravity	Used to specify the gravity of the view like center, top, bottom, etc

Example :

Activity_main.xml :

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

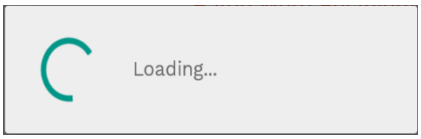
```
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="149dp"
        android:layout_height="40dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        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" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java :

```
package com.example.spinnner;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        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, country);
```

```
        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(),"you selected "+country[position] ,
Toast.LENGTH_LONG).show();
    }
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
    }
}
```

7) Android ProgressBar :



- ProgressBars are used as loading indicators in android applications.
- These are generally used when the application is loading the data from the server or database.
- There are different types of progress bars used within the android application as loading indicators.

Example :

Activity_main.xml

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!-- Heading TextView -->
    <TextView
        android:id="@+id/idTVHeading"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:padding="10dp"
        android:text="Progress Bar in Android"
        android:textAlignment="center"
        android:textColor="@color/black"
```

```
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <!-- Progress Bar -->
    <ProgressBar
        android:id="@+id/idPBLoading"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:visibility="gone"
        app:layout_constraintTop_toBottomOf="@id/idTVHeading"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <!-- Button -->
    <Button
        android:id="@+id/idBtnDisplayProgress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Show Progress Bar"
        android:textAllCaps="false"
        app:layout_constraintTop_toBottomOf="@id/idPBLoading"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
package com.example.progressbar;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
    private Button showProgressBtn;
    private ProgressBar loadingPB;
    boolean isProgressVisible = false;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        showProgressBtn = findViewById(R.id.idBtnDisplayProgress);
        loadingPB = findViewById(R.id.idPBLoading);
        showProgressBtn.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                if (isProgressVisible) {
                    showProgressBtn.setText("Show Progress Bar");
                    loadingPB.setVisibility(View.GONE);
                    isProgressVisible = false;
                } else {
                    isProgressVisible = true;
                    showProgressBtn.setText("Hide Progress Bar");
                    loadingPB.setVisibility(View.VISIBLE);
                }
            }
        });
    }
}
```

8) Android ToggleButton :

- ToggleButton is used to allow users to perform two operations on a single button.
- These are used to perform on and off operations like that of Switch.
- ToggleButton can perform two different operations on clicking on it.

Example :

Activity_main.xml

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!-- Heading TextView -->
    <TextView
```

```
        android:id="@+id/idTVHeading"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:padding="10dp"
        android:text="Toggle Button in Android"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <!-- Status TextView -->
    <TextView
        android:id="@+id/idTVStatus"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:padding="10dp"
        android:text="Status"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintTop_toBottomOf="@id/idTVHeading"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <!-- Toggle Button -->
    <ToggleButton
        android:id="@+id/idBtnToggle"
        android:layout_width="0dp"
        android:layout_height="65dp"
        android:layout_margin="20dp"
        android:padding="10dp"
        android:textAllCaps="true"
        android:textColor="@color/black"
        android:textOff="Off"
        android:textOn="On"
        app:layout_constraintTop_toBottomOf="@id/idTVStatus"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
package com.example.toggle;

import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.ToggleButton;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
    private ToggleButton toggleBtn;
    private TextView statusTV;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        toggleBtn = findViewById(R.id.idBtnToggle);
        statusTV = findViewById(R.id.idTVStatus);
        if (toggleBtn.isChecked()) {
            statusTV.setText("Toggle Button is On");
        } else {
            statusTV.setText("Toggle Button is Off");
        }
        toggleBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (toggleBtn.isChecked()) {
                    statusTV.setText("Toggle Button is On");
                } else {
                    statusTV.setText("Toggle Button is Off");
                }
            }
        });
    }
}
```

9) Android ListView :

- A ListView is a type of **AdapterView** that displays a vertical list of scroll-able views and each view is placed one below the other.
- Using an adapter, items are inserted into the list from an array or database.
- For displaying the items in the list method setAdaptor() is used.
- **setAdaptor()** method conjoins an adapter with the list.

- Android ListView is a ViewGroup that is used to display the list of items in multiple rows and contains an adapter that automatically inserts the items into the list.
- **ListView Attributes :**
  - Following are the important attributes specific to GridView −

Sr.No	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:divider** This is drawable or color to draw between list items.
3	**android:dividerHeight** This specifies height of the divider. This could be in px, dp, sp, in, or mm.
4	**android:entries** Specifies the reference to an array resource that will populate the ListView.
5	**android:footerDividersEnabled** When set to false, the ListView will not draw the divider before each footer view. The default value is true.
6	**android:headerDividersEnabled** When set to false, the ListView will not draw the divider after each header view. The default value is true.

Example :

Activity_main.xml

```
<?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:id="@+id/main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <ListView
      android:id="@+id/list"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_marginTop="200dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Items.xml :

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

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:ellipsize="marquee"
   android:singleLine="true"
   android:textAppearance="?android:attr/textAppearanceListItemSmall"
   android:paddingStart="16dp"
   android:paddingEnd="16dp"
   android:gravity="center_vertical" />
```

MainActivity.java :

```
package com.example.listview;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {
   ListView l;
   String languages[] = { "Web Development", "Web Design",
         "Python", "PHP",
         "C#", "Android",
         "Asp.net", "C++",
         "JAVA" };

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      EdgeToEdge.enable(this);
      setContentView(R.layout.activity_main);
      l = findViewById(R.id.list);
      ArrayAdapter<String> arr;

      arr = new ArrayAdapter<String>(this,
            R.layout.items,
            languages);
      l.setAdapter(arr);
   }
}
```

Designing User Interfaces with Layouts :

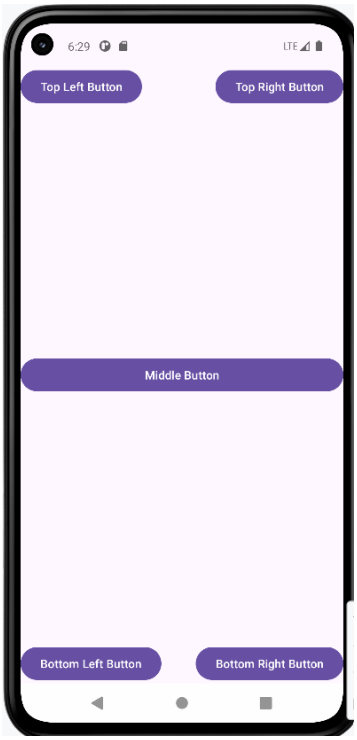
1) Relative Layout :

- The Relative Layout is used to arrange the Child Views in a proper order which means arranging the child objects relative to each other.
- There are so many **properties** that are supported by relative layouts. some of the most used properties are listed below :
    - layout_alignParentTop
    - layout_alignParentBottom
    - layout_alignParentRight
    - layout_alignParentLeft
    - layout_centerHorozontal
    - layout_centerVertical
    - layout_above
    - layout_below

Example :

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Top Left Button"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"/>
        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Top Right Button"
            android:layout_alignParentTop="true"
            android:layout_alignParentRight="true"/>
        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Bottom Left Button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"/>

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom Right Button"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"/>
    <Button
        android:id="@+id/button5"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Middle Button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"/>

  </RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```



2) Linear Layout :

- LinearLayout is the most basic layout in android studio, that aligns all the children sequentially either in a horizontal manner or a vertical manner by specifying the **android:orientation** attribute.
- If one applies **android:orientation=”vertical”** then elements will be arranged one after another in a vertical manner.
- If you apply **android:orientation=”horizontal”** then elements will be arranged one after another in a horizontal manner.

- **Some Important Attributes of LinearLayout :**

Attributes	Description
android:layout_weight	It is defined individually to the child's views to specify how LinearLayout divides the remaining space amongst the views it contains
android:orientation	How the elements should be arranged in the layout. It can be horizontal or vertical.
android:layout_gravity	Sets the gravity of the View or Layout relative to its parent. Possible values are – center_vertical, fill, center, bottom, end, etc.
android:id	This gives a unique id to the layout.

Example :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   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:orientation="vertical"
   tools:context=".MainActivity">
   <!-- Add vertical in the android:orientation-->

   <!-- Add Button-->
   <Button
      android:layout_width="match_parent"
      android:layout_margin="10dp"
      android:layout_height="wrap_content"/>

   <!-- Add Button-->
   <Button
      android:layout_width="match_parent"
      android:layout_margin="10dp"
      android:layout_height="wrap_content"/>

</LinearLayout>
```

3) Table Layout :

- Android TableLayout is a ViewGroup subclass that is used to display the child View elements in rows and columns.
- It will arrange all the children's elements into rows and columns and does not display any border lines between rows, columns or cells.
- The working of Android Table Layout is almost similar to an HTML table and it contains as many columns as a row with the most cells.

Example :

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout 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:id="@+id/main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <TextView
      android:id="@+id/txt"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="ICC Ranking of Players:"
      android:textSize = "20dp"
      android:textStyle="bold">

   </TextView>


   <TableRow android:background="#51B435" android:padding="10dp">
      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="Rank" />
      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="Player" />
      <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="Team" />
      <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Points" />
    </TableRow>
    <TableRow android:background="#F0F7F7" android:padding="5dp">
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="1" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Virat Kohli" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="IND" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="895" />
    </TableRow>
    <TableRow android:background="#F0F7F7" android:padding="5dp">
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="2" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Rohit Sharma" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="IND" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="863" />

    </TableRow>
```

```
    <TableRow android:background="#F0F7F7" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="3" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Mohammand Hafiz" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="PAK" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="834" />

    </TableRow>

</TableLayout>
```