Unit : 2 – Android User Interface Design

User Interface Screen elements

1. **Button**:

- In android, Button is a user interface control that is used to perform an action whenever the user clicks or tap on it
- Attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for button control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:drawableBottom	It's drawable to be drawn to the below of text.
android:drawableRight	It's drawable to be drawn to the right of text.
android:drawableLeft	It's drawable to be drawn to the left of the text.

- Example:

XML Code:
```
android:id="@+id/addBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add"
    android:onClick="addOperation"/>
```
Java Code:
```
Button btnAdd = (Button)findViewById(R.id.addBtn);
btnAdd.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
// Do something in response to button click
}
});
}
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-button-with-examples

2. **EditText**:

- In android, EditText is a user interface control which is used to allow the user to enter or modify the text.
- Attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text.
android:hint	It is used to display the hint text when text is empty
android:textColor	It is used to change the color of the text.
android:textColorHint	It is used to change the text color of hint text.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for edit text control
android:ems	It is used to make the textview be exactly this many ems wide.
android:textAllCaps	It is used to present the text in all CAPS
android:typeface	It is used to specify the Typeface (normal, sans, serif, monospace) for the text.
android:inputType	It is used to specify the type of text being placed in text fields.
android:fontFamily	It is used to specify the fontFamily for the text.
android:editable	If we set false, EditText won't allow us to enter or modify the text

- Example:

XML Code:

```
<EditText
    android:id="@+id/txtName"
    android:ems="10"
    android:hint="Name"
    android:inputType="text | Password | Email | Date | Pone"
    android:selectAllOnFocus="true"/>
```

Java Code

```
EditText et = (EditText)findViewById(R.id.editText1);
et.setText("Welcome to Tutlane");
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-edittext-with-examples

3. TextView:

- In android, TextView is a user interface control that is used to set and display the text to the user based on our requirements.
- Attributes:

Attribute	Description
android: id	It is used to uniquely identify the control
android:text	It is used to display the text.
android:textColor	It is used to change the color of the text.
android:gravity	It is used to specify how to align the text by the view's x and y-axis.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:typeface	It is used to specify the Typeface (normal, sans, serif, monospace) for the text.
android:textColor	It is used to change the color of the text.
android:textColorHighlight	It is used to change the color of text selection highlight.
android:textColorLink	It is used to change the text color of links.

android:fontFamily	It is used to specify the fontFamily for the text.

- Example:

```
XML Code:

<TextView
    android:id="@+id/textView1"
    android:text="Welcome to Tutlane"
    android:textColor="#86AD33"
    android:textSize="20dp"
    android:textStyle="bold" />

Java Code

TextView tv = (TextView)findViewById(R.id.textView2);
tv.setText("Welcome to Tutlane");
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-textview-with-examples

4. **DatePicker**:

- In android, DatePicker is a control that will allow users to select the date by a day, month and year in our application user interface.
- Attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:datePickerMode	It is used to specify datepicker mode either spinner or calendar
android:background	It is used to set the background color for the date picker.
android:padding	It is used to set the padding for left, right, top or bottom of the date picker.

- Example:

```
XML Code:

<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
      android:layout_centerHorizontal="true"
      android:layout_marginTop="20dp" />

Java Code

picker=(DatePicker)findViewById(R.id.datePicker1);
btnGet.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            tvw.setText("Selected Date: "+ picker.getDayOfMonth()+"/"+ (picker.getMonth()
+ 1)+"/"+picker.getYear());
         }
      });
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-datepicker-with-examples

5. **TimePicker** :

- In android, TimePicker is a widget for selecting the time of day, in either 24-hour or AM/PM mode.
- Attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:timePickerMode	It is used to specify timepicker mode, either spinner or clock
android:background	It is used to set the background color for the date picker.
android:padding	It is used to set the padding for left, right, top or bottom of the date picker.

- Example:

```
XML Code:

<TimePicker
   android:id="@+id/datePicker1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:timePickerMode="spinner"/>

Java Code

TimePicker picker=(TimePicker)findViewById(R.id.timePicker1);
picker.setIs24HourView(true);
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-timepicker-with-examples

6. ProgressBar:

- In android, ProgressBar is a user interface control that is used to indicate the progress of an operation. For example, downloading a file, uploading a file.
- Attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:max	It is used to specify the maximum value of the progress can take
android:progress	It is used to specify default progress value.
android:background	It is used to set the background color for a progress bar.
android:indeterminate	It is used to enable the indeterminate progress mode.
android:padding	It is used to set the padding for left, right, top or bottom of a progress bar.

- Example:

XML Code:

```
<ProgressBar
   android:id="@+id/pBar3"
   style="?android:attr/progressBarStyleHorizontal"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:minHeight="50dp"
   android:minWidth="250dp"
   android:max="100"
   android:indeterminate="true"
   android:progress="1" />
```

Java Code:

```
 pgsBar = (ProgressBar) findViewById(R.id.pBar);
      txtView = (TextView) findViewById(R.id.tView);
      Button btn = (Button)findViewById(R.id.btnShow);
      btn.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
            i = pgsBar.getProgress();
            new Thread(new Runnable() {
                public void run() {
                    while (i < 100) {
                        i += 1;
                        // Update the progress bar and display the current value in text view
                        hdlr.post(new Runnable() {
                            public void run() {
                                pgsBar.setProgress(i);
                                txtView.setText(i+"/"+pgsBar.getMax());
                            }
                        });
                        try {
                            // Sleep for 100 milliseconds to show the progress slowly.
                            Thread.sleep(100);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            }).start();
        }
    });
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-progressbar-with-examples

7. ListView:

- In android, ListView is a ViewGroup that is used to display the list of scrollable of items in multiple rows and the list items are automatically inserted to the list using an adapter.
- Attributes

Attribute	Attribute & Description
android:id	This is the ID which uniquely identifies the layout.
android:divider	This is drawable or color to draw between list items.
android:dividerHeight	This specifies height of the divider. This could be in px, dp, sp, in, or mm.
android:entries	Specifies the reference to an array resource that will populate the ListView.
android:footerDividersEnabled	When set to false, the ListView will not draw the divider before each footer view. The default value is true.

android:headerDividersEnabled	When set to false, the ListView will not draw the divider after each header view. The default value is true.

- Example:

```
XML Code:

<ListView
    android:id="@+id/mobile_list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</ListView>


Java Code:

ListView listView = (ListView) findViewById(R.id.mobile_list);
listView.setAdapter(adapter);
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-listview-with-examples

8. GridView:

- In android, Grid View is a ViewGroup that is used to display items in a two dimensional, scrollable grid and grid items are automatically inserted to the gridview layout using a list adapter.

- Android Adapter
- In android, Adapter will act as an intermediate between the data sources and adapter views such as ListView, Gridview to fill the data into adapter views.

Adapter	Description
ArrayAdapter	It will expect an Array or List as input.
CurosrAdapter	It will accept an instance of a cursor as an input.
SimpleAdapter	It will accept a static data defined in the resources.
BaseAdapter	It is a generic implementation for all three adapter types and it can be used for ListView, Gridview or Spinners based on our requirements

- Example:

```
XML Code:

<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="110dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />

Java Code:

GridView gv = (GridView) findViewById(R.id.gvDetails);
        gv.setAdapter(new ImageAdapter(this));
        gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
                Toast.makeText(MainActivity.this, "Image Position: " + position,
Toast.LENGTH_SHORT).show();
            }
        });
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-gridview-with-examples

9. RadioGroup:

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

Attribute	Description
android:checkedButton	This is the id of child radio button that should be checked by default within this radio group.

- Inherited from android.view.View Class –

Sr.No.	Attribute & Description
1	android:background This is a drawable to use as the background.
2	android:contentDescription This defines text that briefly describes content of the view.
3	android:id This supplies an identifier name for this view
4	android:onClick

	This is the name of the method in this View's context to invoke when the view is clicked.
5	android:visibility This controls the initial visibility of the view.

- Example:

Java Code:

```
radioSexGroup=(RadioGroup)findViewById(R.id.radioGroup);

    btnDisplay=(Button)findViewById(R.id.button);

    btnDisplay.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        int selectedId=radioSexGroup.getCheckedRadioButtonId();
        radioSexButton=(RadioButton)findViewById(selectedId);
        Toast.makeText(MainActivity.this,radioSexButton.getText(),Toast.LENGTH_SHORT).show();
      }
    });
```

XML Code:

```
<RadioButton
      android:layout_width="wrap_content"
      android:layout_height="55dp"
      android:text="Male"
      android:id="@+id/radioButton"
      android:layout_gravity="center_horizontal"
      android:checked="false"
      android:textSize="25dp" />
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-radiogroup-with-examples

10.ImageButton:

- In android, Image Button is a user interface control that is used to display a button with an image and to perform an action when a user clicks or taps on it.
- In android, we can add an image to the button by using <ImageButton> attribute android:src in XML layout file or by using the setImageResource() method.
- Attrubutes:

Sr.No	Attribute & Description
1	android:adjustViewBounds

	Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
2	android:baseline This is the offset of the baseline within this view.
3	android:baselineAlignBottom If true, the image view will be baseline aligned with based on its bottom edge.
4	android:cropToPadding If true, the image will be cropped to fit within its padding.
5	android:src This sets a drawable as the content of this ImageView.

Inherited from android.view.View Class –

Sr.No	Attribute & Description
1	android:background This is a drawable to use as the background.
2	android:contentDescription This defines text that briefly describes content of the view.
3	android:id This supplies an identifier name for this view
4	android:onClick This is the name of the method in this View's context to invoke when the view is clicked.
5	android:visibility This controls the initial visibility of the view.

- Example:

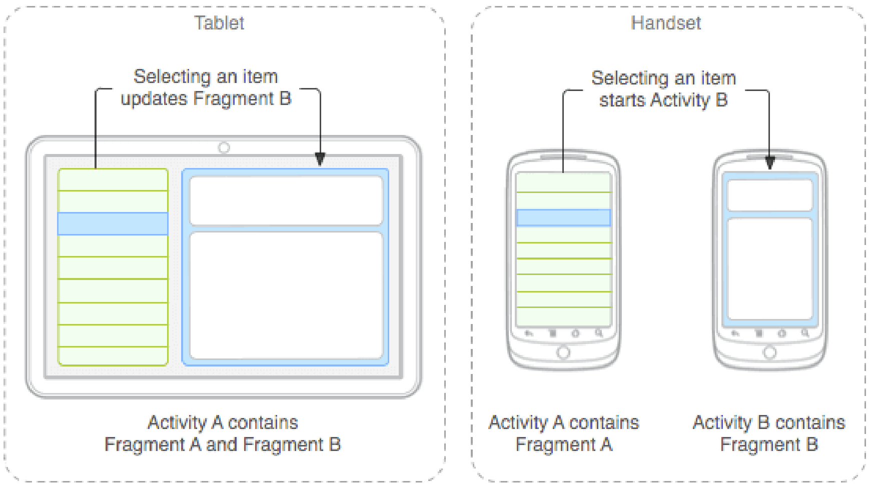
XML Code:

```
<ImageButton
    android:id="@+id/addBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/add_icon"
    android:onClick="addOperation"/>

Java Code:

final EditText firstNum = (EditText)findViewById(R.id.firstNum);
        final EditText secNum = (EditText)findViewById(R.id.secondNum);
        ImageButton btnAdd = (ImageButton)findViewById(R.id.addBtn);
        btnAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(firstNum.getText().toString().isEmpty() || secNum.getText().toString().isEmpty())
                {
                    Toast.makeText(getApplicationContext(), "Please fill all the fields",
Toast.LENGTH_SHORT).show();
                }
                else {
                    int num1 = Integer.parseInt(firstNum.getText().toString());
                    int num2 = Integer.parseInt(secNum.getText().toString());
                    Toast.makeText(getApplicationContext(), "SUM = " + (num1 + num2),
Toast.LENGTH_SHORT).show();
                }
            }
        });
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-imagebutton-with-examples

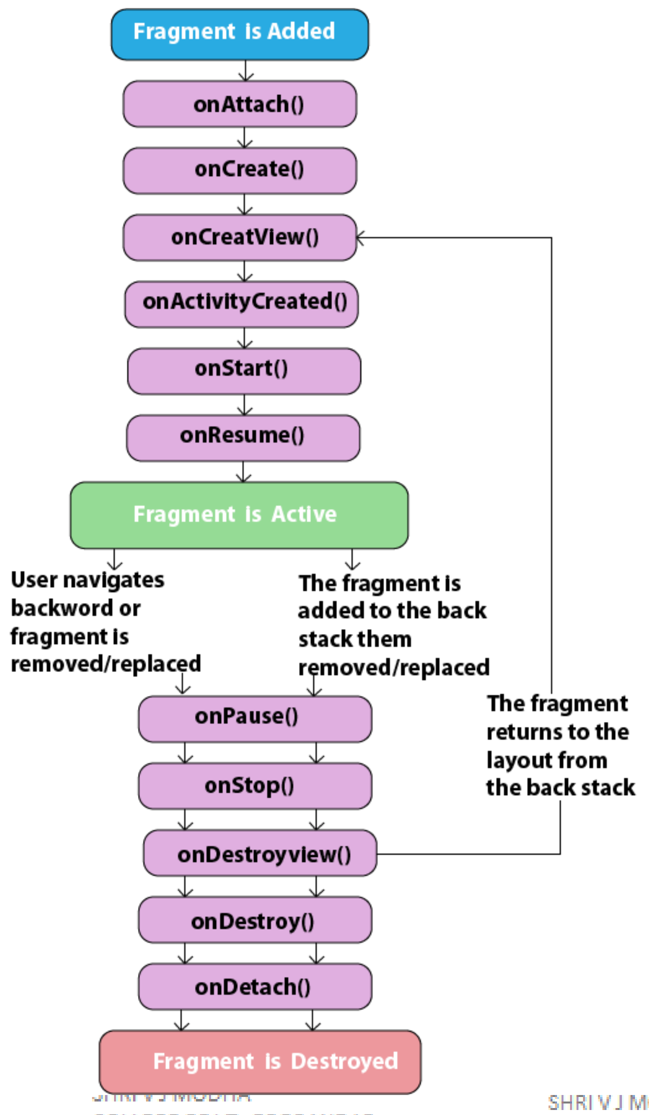
11. **Fragement**:

- Android Fragment is the part of activity, it is also known as sub-activity. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

- Android Fragment Lifecycle

- The lifecycle of android fragment is like the activity lifecycle. There are 12 lifecycle methods for fragment.



- Android Fragment Lifecycle Methods

No.	Method	Description
1)	onAttach(Activity)	it is called only once when it is attached with activity.
2)	onCreate(Bundle)	It is used to initialize the fragment.
3)	onCreateView(LayoutInflater, ViewGroup, Bundle)	creates and returns view hierarchy.
4)	onActivityCreated(Bundle)	It is invoked after the completion of onCreate() method.
5)	onViewStateRestored(Bundle)	It provides information to the fragment that all the saved state of fragment view hierarchy has been restored.

6)	onStart()	makes the fragment visible.
7)	onResume()	makes the fragment interactive.
8)	onPause()	is called when fragment is no longer interactive.
9)	onStop()	is called when fragment is no longer visible.
10)	onDestroyView()	allows the fragment to clean up resources.
11)	onDestroy()	allows the fragment to do final clean up of fragment state.
12)	onDetach()	It is called immediately prior to the fragment no longer being associated with its activity.

- Example:

Listitems_info.xml

```
<?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">
    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@android:id/list" />
</LinearLayout>
```

Details_info.xml:

```
<?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"
    android:background="#0079D6">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#ffffff"
        android:layout_marginTop="200px"
        android:layout_marginLeft="200px"
        android:id="@+id/Name"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="200px"
        android:textColor="#ffffff"
    android:id="@+id/Location"/>
</LinearLayout>
```

DetailFragment.java

```
package com.tutlane.fragmentsexample;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

/**
 * Created by tutlane on 06-08-2017.
 */

public class DetailsFragment extends Fragment {
  TextView name,location;
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    View view = inflater.inflate(R.layout.details_info, container, false);
    name = (TextView)view.findViewById(R.id.Name);
    location = (TextView)view.findViewById(R.id.Location);
    return view;
  }
  public void change(String uname, String ulocation){
    name.setText(uname);
    location.setText(ulocation);
  }
}
```

ListMenuFragment.java:

```
package com.tutlane.fragmentsexample;
import android.app.ListFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;

/**
```

```
 * Created by tutlane on 06-08-2017.
 */
public class ListMenuFragment extends ListFragment {
    String[] users = new String[] { "Suresh","Rohini","Trishika","Praveen","Sateesh","Madhav" };
String[] location = new String[]{"Hyderabad","Guntur","Hyderabad","Bangalore","Vizag","Nagpur"};
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view =inflater.inflate(R.layout.listitems_info, container, false);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getActivity(),
                android.R.layout.simple_list_item_1, users);
        setListAdapter(adapter);
        return view;
    }
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        DetailsFragment txt =
(DetailsFragment)getFragmentManager().findFragmentById(R.id.fragment2);
        txt.change("Name: "+ users[position],"Location : "+ location[position]);
        getListView().setSelector(android.R.color.holo_blue_dark);
    }
}
```

Activity_main.xml

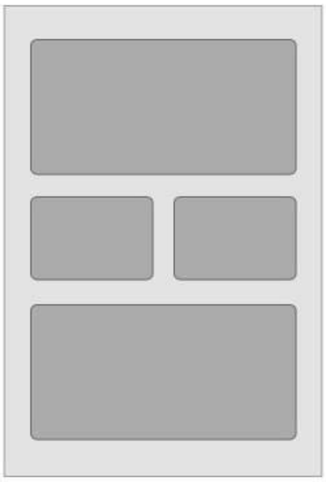
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    tools:context="com.tutlane.fragmentsexample.MainActivity">

    <fragment
        android:layout_height="match_parent"
        android:layout_width="350px"
        class="com.tutlane.fragmentsexample.ListMenuFragment"
        android:id="@+id/fragment"/>
    <fragment
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.tutlane.fragmentsexample.DetailsFragment"
        android:id="@+id/fragment2"/>
</LinearLayout>
```

- **For Full Code: https://www.tutlane.com/tutorial/android/android-fragments-with-examples**

• Designing User Interfaces with Layouts:

1. Relative Layout:

- 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.



- RelativeLayout Attributes:
- Following are the important attributes specific to RelativeLayout –

Sr.No.	Attribute & Description
1	android:id This is the ID which uniquely identifies the layout.
2	android:gravity This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
3	android:ignoreGravity This indicates what view should not be affected by gravity.

- Other Important Attribute:

Sr.No.	Attribute & Description
1	android:layout_above Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name"

2	android:layout_alignBottom Makes the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
3	android:layout_alignLeft Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
4	android:layout_alignParentBottom If true, makes the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".
5	android:layout_alignParentEnd If true, makes the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".
6	android:layout_alignParentLeft If true, makes the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".
7	android:layout_alignParentRight If true, makes the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".
8	android:layout_alignParentStart If true, makes the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".
9	android:layout_alignParentTop If true, makes the top edge of this view match the top edge of the parent. Must be a boolean value, either "true" or "false".
10	android:layout_alignRight Makes the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
11	android:layout_alignStart Makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
12	android:layout_alignTop Makes the top edge of this view match the top edge of the given anchor view ID and must be a

	reference to another resource, in the form "@[+][package:]type:name".
13	android:layout_below Positions the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
14	android:layout_centerHorizontal If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".
15	android:layout_centerInParent If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".
16	android:layout_centerVertical If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".
17	android:layout_toEndOf Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
18	android:layout_toLeftOf Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
19	android:layout_toRightOf Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
20	android:layout_toStartOf Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

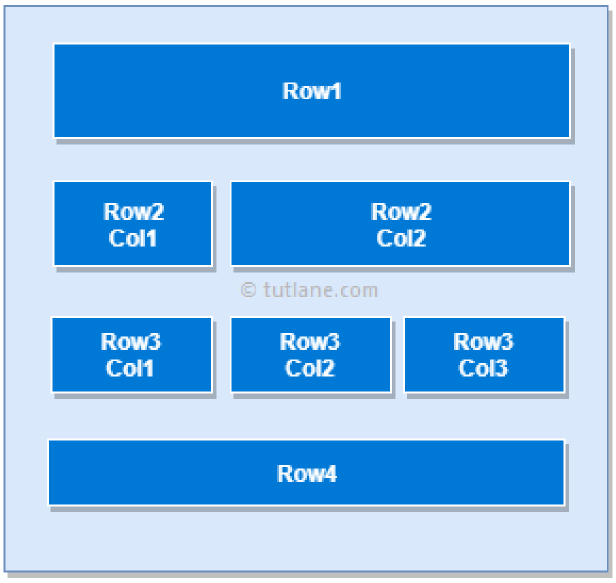
- Example:

```
<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>
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-relativelayout-with-examples

2. Table Layout:

- In android, TableLayout is a ViewGroup subclass that is used to display the child View elements in rows and columns.



- Example:

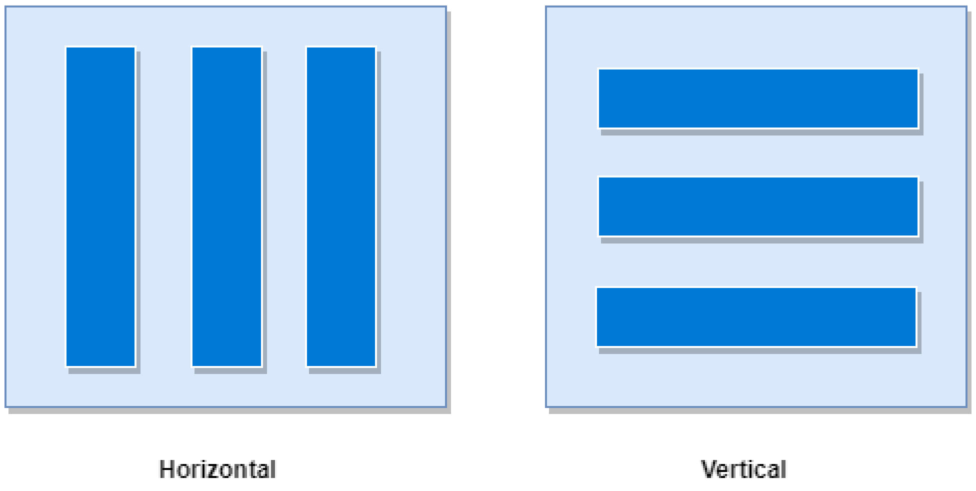
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:layout_marginTop="100dp"
  android:paddingLeft="10dp"
  android:paddingRight="10dp" >
  <TableRow android:background="#0079D6" android:padding="5dp">
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="UserId" />
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="User Name" />
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="Location" />
  </TableRow>
  <TableRow android:background="#DAE8FC" 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="Suresh Dasari" />
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:text="Hyderabad" />
  </TableRow>
  <TableRow android:background="#DAE8FC" 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="Rohini Alavala" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Guntur" />
    </TableRow>
    <TableRow android:background="#DAE8FC" 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="Trishika Dasari" />
      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Guntur" />
    </TableRow>
</TableLayout>
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-tablelayout-with-examples

3. Linear Layout:

- In android, LinearLayout is a ViewGroup subclass which is used to render all child View instances one by one either in Horizontal direction or Vertical direction based on the orientation property.



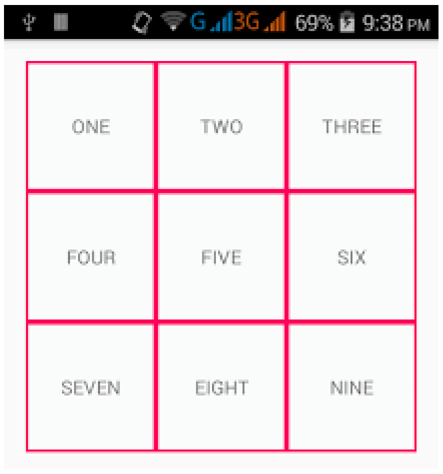
- Example:

```
<?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" >
   <!-- Add Child Views Here -->
</LinearLayout>
```

- **For Full Code:** https://www.tutlane.com/tutorial/android/android-linearlayout-with-examples

4. GridLayout:

- It is a layout that places its children in a rectangle grid. The grid is composed of a set of infinitely thin lines that separate the viewing area into cells. Through the API, grid lines are referenced by grid indices.



- Example:

```
<GridLayout
```

```
android:id="@+id/grid_layout"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:columnCount="2"

android:layout_margin="80dp">

<TextView
   android:id="@+id/textView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="User Name:" />

<EditText
   android:id="@+id/textview_user_name"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:ems="10"
   android:inputType="textPersonName"
   android:hint="Input user name " />

<TextView
   android:id="@+id/textView2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Password:" />

<EditText
   android:id="@+id/edittext_password"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:ems="10"
   android:inputType="textPassword" />

<Button
   android:id="@+id/button_add_password"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Add Password"
   android:layout_columnSpan="2"
   android:textAllCaps="false"/>

<Button
   android:id="@+id/button_login"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Login"
        android:layout_columnSpan="2"
        android:textAllCaps="false"/>

</GridLayout>
```

• Dialogs:

1. Android AlertDialog:

- Android AlertDialog can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.
- Android AlertDialog is composed of three regions: title, content area and action buttons.
- Methods of AlertDialog class:

Method	Description
public AlertDialog.Builder setTitle(CharSequence)	This method is used to set the title of AlertDialog.
public AlertDialog.Builder setMessage(CharSequence)	This method is used to set the message for AlertDialog.
public AlertDialog.Builder setIcon(int)	This method is used to set the icon over AlertDialog.

- Example:

```
closeButton = (Button) findViewById(R.id.button);
builder = new AlertDialog.Builder(this);
closeButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {

//Uncomment the below code to Set the message and title from the strings.xml file
builder.setMessage(R.string.dialog_message) .setTitle(R.string.dialog_title);

//Setting message manually and performing action on button click
builder.setMessage("Do you want to close this application ?")
.setCancelable(false)
.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int id) {
```

```
finish();
Toast.makeText(getApplicationContext(),"you choose yes action for alertbox",
Toast.LENGTH_SHORT).show();


.setNegativeButton("No", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int id) {
//  Action for 'NO' Button
dialog.cancel();
Toast.makeText(getApplicationContext(),"you choose no action for alertbox",
Toast.LENGTH_SHORT).show();
}
});
//Creating dialog box
AlertDialog alert = builder.create();
//Setting the title manually
alert.setTitle("AlertDialogExample");
alert.show();
}
});
```

- **For Full Code:** https://www.javatpoint.com/android-alert-dialog-example

2. **Custom DialogBox:**

- Custom_Layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

- Activity_Main.xml:

```
<?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:gravity="center"
    android:id="@+id/root"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="showAlertDialogButtonClicked"
        android:text="Show Dialog"
        />
</LinearLayout>
```

- MainActivity.java:

```
package com.example.dialogdemo;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

    }

    public void showAlertDialogButtonClicked(View view) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Name");
        final View customLayout = getLayoutInflater().inflate(R.layout.custom_layout, null);
        builder.setView(customLayout);
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                EditText editText = customLayout.findViewById(R.id.editText);
                Toast.makeText(getApplicationContext(), editText.getText().toString(), Toast.LENGTH_SHORT).show();
            }
        });
        AlertDialog dialog = builder.create();
        dialog.show();
    }
}
```

- For Full Code: https://www.geeksforgeeks.org/how-to-create-a-custom-alertdialog-in-android/

Drawing and Working with Animation

1. Frame By Frame Animation:

- 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:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <ImageView
    android:id="@+id/imageview_animation_list_filling"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/animation_list_filling"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.259" />

  <ImageView
    android:id="@+id/imageview_animation_list_emptying"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/animation_list_emptying"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.706" />

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

- MainActivity.Java:

```
package com.example.framebyframeanimationdemo;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.widget.ImageView;
```

```
public class MainActivity extends AppCompatActivity {

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

        ImageView mImageViewFilling = (ImageView) findViewById(R.id.imageview_animation_list_filling);
        ((AnimationDrawable) mImageViewFilling.getBackground()).start();

        ImageView mImageViewEmpty = (ImageView) findViewById(R.id.imageview_animation_list_emptying);
        ((AnimationDrawable) mImageViewEmpty.getBackground()).start();
    }
}
```

- Animation_list_filling.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_0"/>

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_25"/>

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_50"/>

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_75"/>

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_100"/>

</animation-list>
```

- Animation_list_emptying.xml:

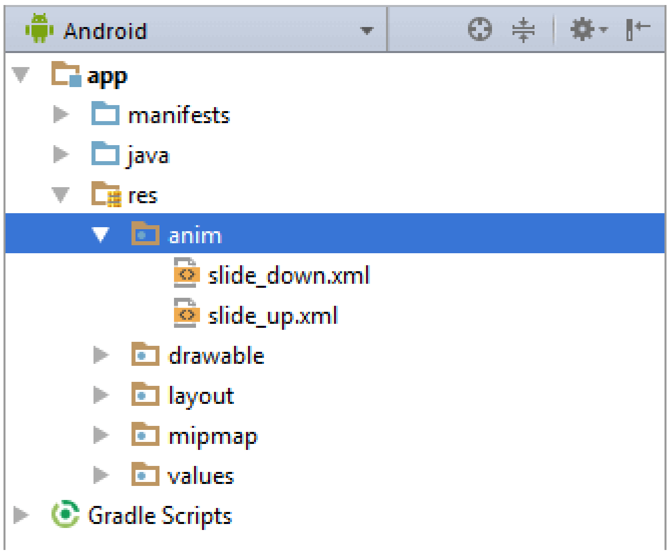
```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_100"/>

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_75"/>
    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_50"/>

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_25"/>

    <item
        android:duration="500"
        android:drawable="@drawable/ic_heart_0"/>

</animation-list>
```

For Full Code: https://www.bignerdranch.com/blog/frame-animations-in-android/

2. Twined Animation

Fade In
Fade Out
Cross Fading
Blink
Zoom In
Zoom Out
Rotate
Move
Slide Up
Slide Down

- **Tween Animation**
- In android, Animations are used to change the appearance and behavior of the objects over a particular interval of time. The animations will provide a better look and feel high-quality user interface for our applications.
- We need to create an XML file that defines the type of animation to perform in a new folder anim under res directory (res anim animation.xml) with the required properties. In case, anim folder not exists in res directory, create a new one.
- Following is the example of creating XML files under anim folder to define slide up / down animation properties.



- Attributes:

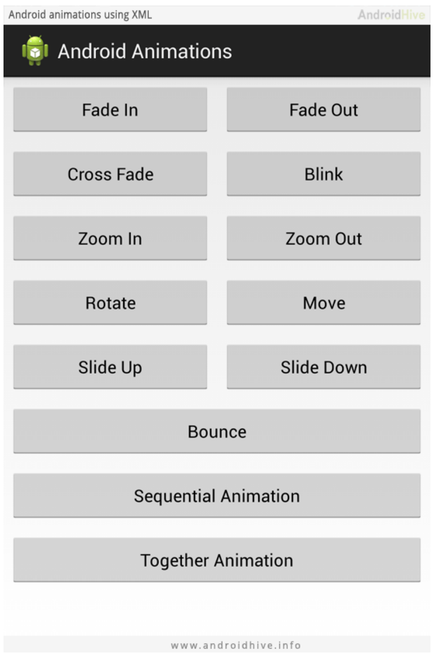
Attributes	Description
android:duration	It is used to define the duration of the animation to complete.
android:startOffset	It is used to define the waiting time before the animation starts.
android:interpolator	It is used to define the rate of change in animation.
android:repeatMode	It is useful when we want to repeat our animation.
android:repeatCount	It is used to define the number of times the animation repeats. In case if we set infinite, the animation will repeat infinite times.
android:fillAfter	It is used to define whether to apply animation transformation after the animation completes or not.

- Methods:

Sr.No	Method & Description
1	start() This method starts the animation.
2	setDuration(long duration) This method sets the duration of an animation.
3	getDuration() This method gets the duration which is set by above method
4	end()

	This method ends the animation.
5	cancel() This method cancels the animation.

- **Types Of Animation:**



i. Fade In :

- For fade in animation you can use <alpha> tag which defines alpha value. Fade in animation is nothing but increasing alpha value from 0 to 1.

fade_in.xml
```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true" >

    <alpha
        android:duration="1000"
        android:fromAlpha="0.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="1.0" />

</set>
```

ii. Fade Out:

- Fade out is exactly opposite to fade in, where we need to decrease the alpha value from 1 to 0

fade_out.xml
```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
```

```
   android:fillAfter="true" >

   <alpha
      android:duration="1000"
      android:fromAlpha="1.0"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:toAlpha="0.0" />

</set>
```

iii. Cross Fading:

- Cross fading is performing fade in animation while other element is fading out. For this you don't have to create separate animation file, you can just use fade_in.xml and fade_out.xml files.
- In the following code i loaded fade in and fade out, then performed them on two different UI elements.

```
TextView txtView1, txtView2;
Animation animFadeIn, animFadeOut;
.
.
// load animations
animFadeIn = AnimationUtils.loadAnimation(getApplicationContext(),
            R.anim.fade_in);
animFadeOut = AnimationUtils.loadAnimation(getApplicationContext(),
            R.anim.fade_out);
.
.
// set animation listeners
animFadeIn.setAnimationListener(this);
animFadeOut.setAnimationListener(this);


.
.
// Make fade in elements Visible first
txtMessage2.setVisibility(View.VISIBLE);

// start fade in animation
txtMessage2.startAnimation(animFadeIn);

// start fade out animation
txtMessage1.startAnimation(animFadeOut);
```

iv. Blink:

- Blink animation is animating fade out or fade in animation in repetitive fashion. For this you will have to set android:repeatMode="reverse" and android:repeatCount attributes.

```
blink.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <alpha android:fromAlpha="0.0"
      android:toAlpha="1.0"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:duration="600"
      android:repeatMode="reverse"
      android:repeatCount="infinite"/>
</set>
```

v. Zoom In:

- For zoom use <scale> tag. Use pivotX=”50%” and pivotY=”50%” to perform zoom from the center of the element. Also you need to use fromXScale, fromYScale attributes which defines scaling of the object. Keep these value lesser than toXScale, toYScale

zoom_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <scale
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:duration="1000"
      android:fromXScale="1"
      android:fromYScale="1"
      android:pivotX="50%"
      android:pivotY="50%"
      android:toXScale="3"
      android:toYScale="3" >
   </scale>

</set>
```

vi. Zoom Out:

- Zoom out animation is same as zoom in but toXScale, toYScale values are lesser than fromXScale, fromYScale

zoom_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <scale
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:duration="1000"
      android:fromXScale="1.0"
      android:fromYScale="1.0"
      android:pivotX="50%"
      android:pivotY="50%"
      android:toXScale="0.5"
      android:toYScale="0.5" >
    </scale>

</set>
```

vii. Rotate:

- Rotate animation uses <rotate> tag. For rotate animation required tags are android:fromDegrees and android:toDegrees which defines rotation angles.
- Clock wise – use positive toDegrees value
  Anti clock wise – use negative toDegrees value

rotate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <rotate android:fromDegrees="0"
      android:toDegrees="360"
      android:pivotX="50%"
      android:pivotY="50%"
      android:duration="600"
      android:repeatMode="restart"
      android:repeatCount="infinite"
      android:interpolator="@android:anim/cycle_interpolator"/>

</set>
```

viii. Move:

- In order to change position of object use <translate> tag. It uses fromXDelta, fromYDelta for X-direction and toXDelta, toYDelta attributes for Y-direction.

move.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:interpolator="@android:anim/linear_interpolator"
   android:fillAfter="true">

  <translate
     android:fromXDelta="0%p"
     android:toXDelta="75%p"
     android:duration="800" />
</set>
```

ix. Slide Up:

- Sliding animation uses <scale> tag only. Slide up can be achieved by setting android:fromYScale=”1.0” and android:toYScale=”0.0”

slide_up.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <scale
      android:duration="500"
      android:fromXScale="1.0"
      android:fromYScale="1.0"
      android:interpolator="@android:anim/linear_interpolator"
      android:toXScale="1.0"
      android:toYScale="0.0" />

</set>
```

x. Slide Down:

- Slide down is exactly opposite to slide down animation. Just interchange android:fromYScale and android:toYScale values.

slide_down.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true">

   <scale
      android:duration="500"
      android:fromXScale="1.0"
      android:fromYScale="0.0"
      android:interpolator="@android:anim/linear_interpolator"
      android:toXScale="1.0"
      android:toYScale="1.0" />

</set>
```

xi. Bounce:

- Bounce is just an animation effect where animation ends in bouncing fashion. For this set android:interpolator value to @android:anim/bounce_interpolator. This bounce can be used with any kind animation. Following slide down example uses bounce effect.

bounce.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:fillAfter="true"
   android:interpolator="@android:anim/bounce_interpolator">

   <scale
      android:duration="500"
      android:fromXScale="1.0"
      android:fromYScale="0.0"
```

```
        android:toXScale="1.0"
        android:toYScale="1.0" />

</set>
```

xii. Sequential Animation:

- If you want to perform multiple animation in a sequential manner you have to use android:startOffset to give start delay time. The easy way to calculate this value is to add the duration and startOffset values of previous animation. Following is a sequential animation where set of move animations performs in sequential manner.

sequential.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true"
    android:interpolator="@android:anim/linear_interpolator" >

    <!-- Use startOffset to give delay between animations -->


    <!-- Move -->
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromXDelta="0%p"
        android:startOffset="300"
        android:toXDelta="75%p" />
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromYDelta="0%p"
        android:startOffset="1100"
        android:toYDelta="70%p" />
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromXDelta="0%p"
        android:startOffset="1900"
        android:toXDelta="-75%p" />
    <translate
        android:duration="800"
        android:fillAfter="true"
        android:fromYDelta="0%p"
        android:startOffset="2700"
        android:toYDelta="-70%p" />

    <!-- Rotate 360 degrees -->
    <rotate
```

```
        android:duration="1000"
        android:fromDegrees="0"
        android:interpolator="@android:anim/cycle_interpolator"
        android:pivotX="50%"
        android:pivotY="50%"
        android:startOffset="3800"
        android:repeatCount="infinite"
        android:repeatMode="restart"
        android:toDegrees="360" />

</set>
```

xiii. Together Animation:

- Performing all animation together is just writing all animations one by one without using android:startOffset

together.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true"
    android:interpolator="@android:anim/linear_interpolator" >

    <scale
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:duration="4000"
        android:fromXScale="1"
        android:fromYScale="1"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="4"
        android:toYScale="4" >
    </scale>

    <!-- Rotate 180 degrees -->
    <rotate
        android:duration="500"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="infinite"
        android:repeatMode="restart"
        android:toDegrees="360" />

</set>
```

- **Java Code:**

```
package com.example.animationexample;
```

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button btnFadein = findViewById(R.id.btnFadeIn);
    btnFadein.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        TextView tvAnimation = findViewById(R.id.textView);
        Animation fadin = AnimationUtils.loadAnimation
            (getApplicationContext(), R.anim.fade_in);
        tvAnimation.startAnimation(fadin);
      }
    });
    Button btnFadeout = findViewById(R.id.btnFadeOut);
    btnFadeout.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        TextView tvAnimation = findViewById(R.id.textView);
        Animation fadout = AnimationUtils.loadAnimation
            (getApplicationContext(), R.anim.fade_out);
        tvAnimation.startAnimation(fadout);
      }
    });
  }
}
```

- **For Full Code: https://www.androidhive.info/2013/06/android-working-with-xml-animations/**

Important Questions

1Marks:

Year	Questions
2015	1. List out the types of the Layouts.
2016	1. In Linear Layout set the orientation,_____is used at runtime? 2. ______ is not the Layout in android. 3. Toast is _____ in android.
2018	1. When the user checks the checkbox, the checkbox object receives an_______ event.
2019	1. In which folder animation file is stored? 2. Which layout is used to arrange control by horizontal or vertical way? 3. What is the name of the textbox control in android? 4. What is the name of the label control in android?
Expected in 2021	1. Which control is used to display the List of items? 2. Which attribute is used to specify how to align the text by the view's x and y-axis? 3. ____________ is also known as Sub-Activity. 4. What is used to display multiple screens inside single Activity? 5. Which lay out allows to set the control as per its constraints? 6. Which layout is used to set the control on the basis of row and column? 7. What is used to take the confirmations from the users? 8. Which type of dialog box can be created as per the user's needs & requirements? 9. Define Schemas.

3/5Marks:

Year	Questions
2015	1. Explain types of resources in brief. 2. List out various types of layouts and Define any one. 3. Write a note on Spinner widget.
2016	1. Explain progress dialog in android. 2. Explain layouts in android. 3. Explain date picker in detail. 4. Explain types of resources in brief. 5. Explain Listview in brief 6. Explain how to notify with Status Bar.
2018	1. Explain onCreate(). 2. What is textfield? 3. Explain Linear layout in details. 4. Explain the components available with notifications. 5. Explain difference between ID & UI.
2019	1. Explain Progressbar Element in android. 2. Explain RadioGroup Element in android. 3. Explain Relative layout. 4. Explain Table Layout. 5. Explain Dialog in Android with suitable example. 6. Explain twined animation with suitable example.
Expected in 2021	1. Explain UI components available in the android & give brief about any 4. 2. Explain Layouts with its types. 3. Explain Linear Layout 4. Explain Grid Layout with its example 5. What is animation? Explain any 4 animations.