MOD 2
DESIGNING USER INTERFACE WITH VIEWS

GUI COMPONENTS

- TEXTVIEWS
- BUTTONS
- EDITTEXT
- CHECKBOX
- RADIOBUTTONS
- RADIOGROUPS
- TOGGLEBUTTON
- SPINNER
- LISTVIEW
- GRIDVIEW

TEXTVIEW-displays text to the user and optionally allows them to edit

1	**android:id** This is the ID which uniquely identifies the control.
2	**android:capitalize** If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types. •Don't automatically capitalize anything - 0 •Capitalize the first word of each sentence - 1 •Capitalize the first letter of every word - 2 •Capitalize every character - 3
3	**android:cursorVisible** Makes the cursor visible (the default) or invisible.
4	**android:editable** If set to true, specifies that this TextView has an input method.
5	**android:fontFamily** Font family (named by string) for the text.

6 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.**
7 **android:hint**	Hint text to display when the text is empty.
8 **android:inputType**	The type of data being placed in a text field. Phone, Date, Time, Number, Password etc.
9 **android:maxHeight**	Makes the TextView be at most this many pixels tall.
10 **android:maxWidth**	Makes the TextView be at most this many pixels wide.

11 android:textSize	**Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).**
12 **android:textStyle**	Style (bold, italic, bold|italic) for the text. You can use or more of the following values separated by '|'. •normal - 0 •bold - 1 •italic - 2
13 **android:text**	Text to display.
14 **android:textAllCaps**	Present the text in ALL CAPS. Possible value either "true" or "false".
15 **android:textColor**	Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb"

```
<?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">
    <TextView
        android:id="@+id/text_view_id"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="hello"
        android:textStyle="bold|italic"
        android:textColor="@color/black"
        android:textAllCaps="true"/>
</LinearLayout>
```

Button

A push button which can be pressed or clicked by the user to perform an action.

Attributes Inherited from **android.widget.TextView** Class

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.

Inherited from **android.view.View** Class

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.

```
<Button
  android:layout_height="wrap_content"
  android:layout_width="wrap_content"
  android:text="BUTTON"
  android:id="@+id/button"
  android:layout_below="@+id/simpleImageButton“
android:drawableRight="@mipmap/ic_launcher_round"
android:contentDescription="hghghfggfg"
android:layout_below="@+id/simpleImageButton“/>
```

IMAGE BUTTON

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.

Attributes inherited from **android.widget.ImageView** Class

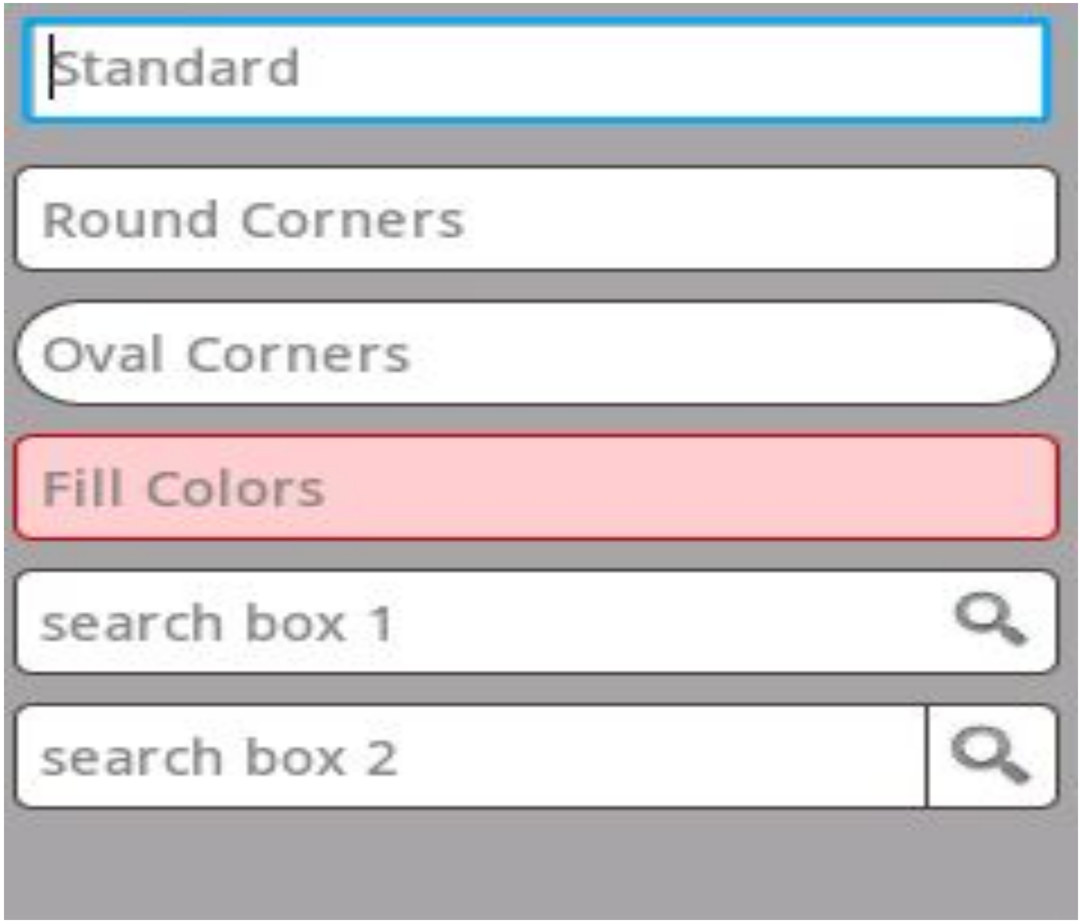
1	**android:adjustViewBounds** Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
2	**android:baseline** This is the offset of the baseline within this view.
3	**android:baselineAlignBottom** If true, the image view will be baseline aligned with 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.

Attributes inherited from **android.view.View** Class –

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.

EditText

A EditText is an overlay over TextView that configures itself to be editable. It is the predefined subclass of TextView that includes rich editing capabilities.



Inherited from **android.widget.TextView** Class

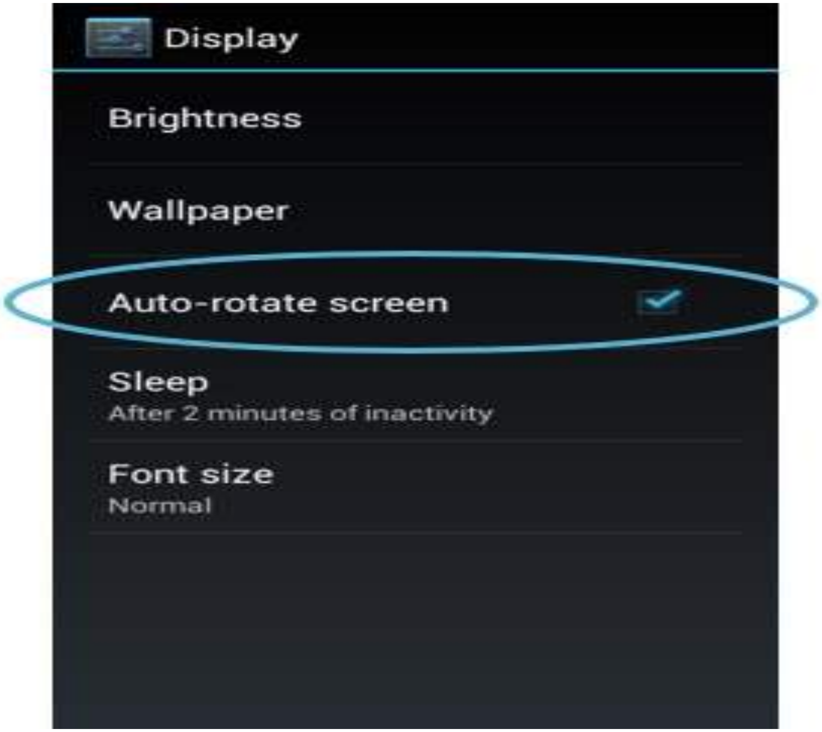
1	**android:autoText** If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
2	**android:drawableBottom** This is the drawable to be drawn below the text.
3	**android:drawableRight** This is the drawable to be drawn to the right of the text.
4	**android:editable** If set, specifies that this TextView has an input method.
5	**android:text** This is the Text to display.

Inherited from **android.view.View** Class –

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.

CHECKBOX

A CheckBox is an on/off switch that can be toggled by the user. You should use check-boxes when presenting users with a group of selectable options that are not mutually exclusive.



Display

Brightness

Wallpaper

Auto-rotate screen ☑

Sleep
After 2 minutes of inactivity

Font size
Normal

Inherited from **android.widget.TextView** Class

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.

Inherited from **android.view.View** Class

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.

Android Toggle Button

Android Toggle Button can be used to display checked/unchecked (On/Off) state on the button.

It is beneficial if user have to change the setting between two states. It can be used to On/Off Sound, Wifi, Bluetooth etc.

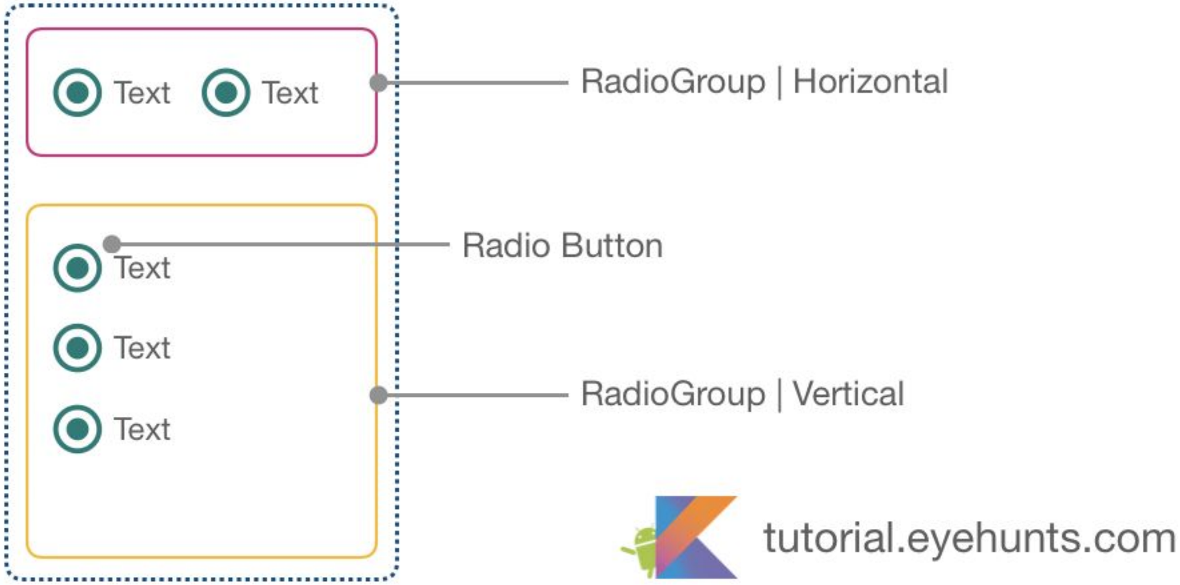
1	**android:autoText** If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
2	**android:background** To use as the background
3	**android:onClick** This is the name of the method invoked when clicked.
4	**android:editable** If set, specifies that this TextView has an input method.
5	**android:text** This is the Text to display.

6	**android:disabledAlpha** This is the alpha to apply to the indicator when disabled.
7	**android:textOff** This is the text for the button when it is not checked.
8	**android:textOn** This is the text for the button when it is checked.

```
<ToggleButton
android:id="@+id/simpleToggleButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:checked="true"
android:gravity="right|center_vertical"/>
```

RADIO GROUP

A RadioGroup class is used for set of radio buttons.
If we check one radio button that belongs to a radio group, it automatically unchecks any previously checked radio button within the same group.



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

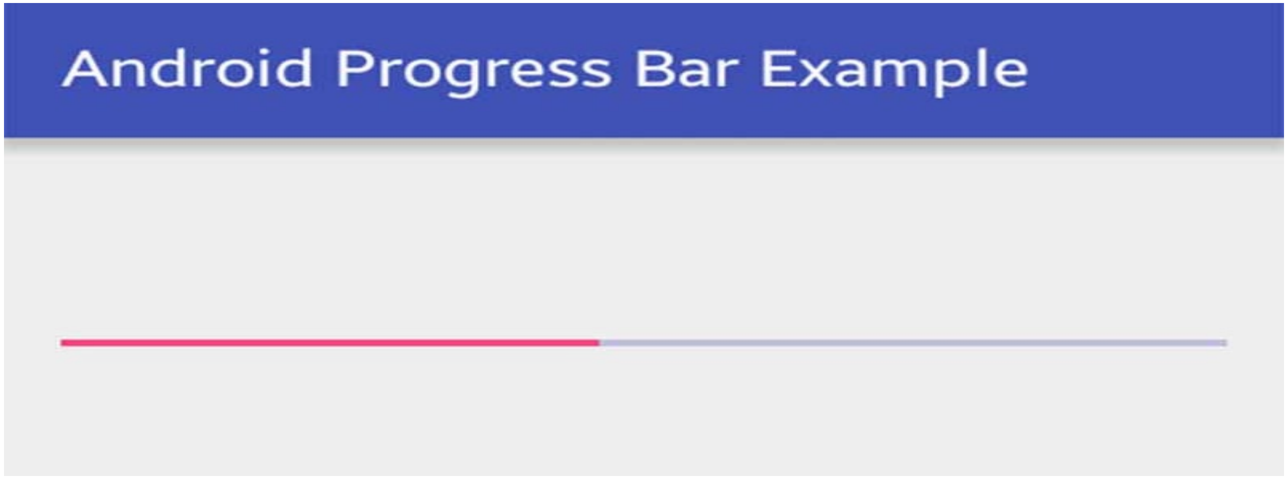
RADIO BUTTON

Radio button is a widget which can have more than one option to choose from.
The user can choose only one option at a time.
Radio Buttons are used inside a RadioGroup.

Attribute	Description
Android:autolink	Checks whether links such as url and email addresses are automatically found and converted to clickable links
Android:cursorVisible	Makes cursor visible or invisible
Android:Button	For button graphics
Android:onclick	Name of the method to be invoked when clicked
Android:checked	Indicates the initial checked state of this button
Android:clickable	This view reacts to click events

PROGRESS BAR

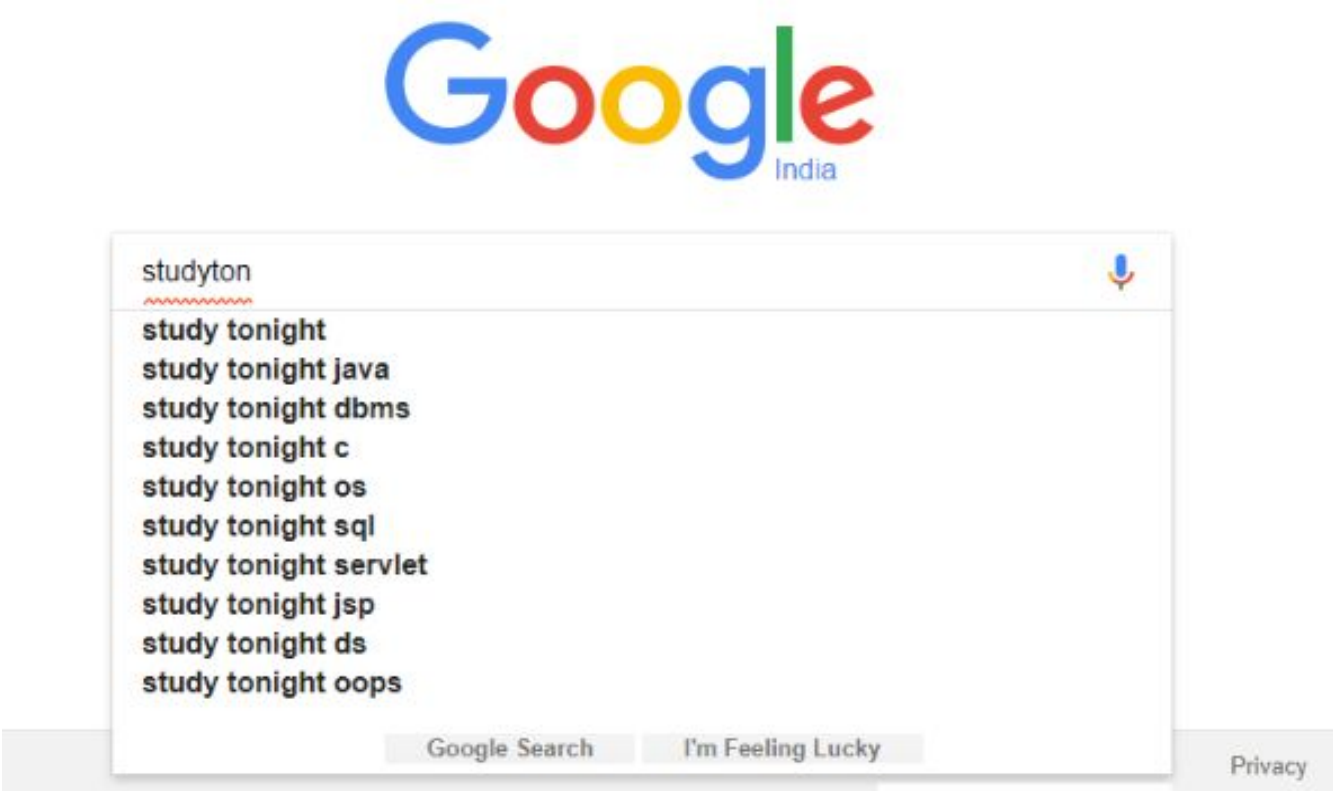
Progress bars are used to show progress of a task. For example, when you are uploading or downloading something from the internet, it is better to show the progress of download/upload to the user.



1	**Android:minHeight-** used to set the height of the progress bar
2	**Android:minWidth-** used to set the width of the progress bar
3	**Android:max-** maximum value of the progress bar
4	**Android:progress-** Used to set default progress value between 0 and max
5	**Android:id-** used to uniquely identify the control

AUTOCOMPLETE TEXTVIEW

A AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.



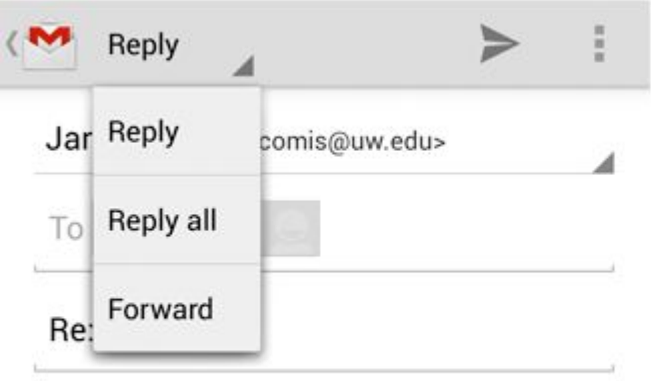
studyton

1	**android:completionHint** This defines the hint displayed in the drop down menu.
2	**android:completionHintView** This defines the hint view displayed in the drop down menu.
3	**android:completionThreshold** This defines the number of characters that the user must type before completion suggestions are displayed in a drop down menu.
4	**android:dropDownAnchor** This is the View to anchor the auto-complete dropdown to.
5	**android:dropDownHeight** This specifies the basic height of the dropdown.
6	**android:dropDownWidth** This specifies the basic width of the dropdown

```
<AutoCompleteTextView
    android:id="@+id/actv"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="100dp"
    android:hint="Enter Country Name"
    android:completionThreshold="1"
    android:completionHint="Select a country"/>
```

Spinner

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



Spinner attributes

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:visibilty	Used to set the visibility of the view.
android:gravity	Used to specify the gravity of the view like center, top, bottom, etc

EXAMPLE

Strings.xml

```
<resources>
    <string name="app_name">spinner</string>
    <string-array name="Spinner_items">
        <item>Spinner Item one</item>
        <item>Spinner Item two</item>
        <item>Spinner Item three</item>
        <item>Spinner Item four</item>
        <item>Spinner Item five</item>
    </string-array>
</resources>
```

activity_main.xml

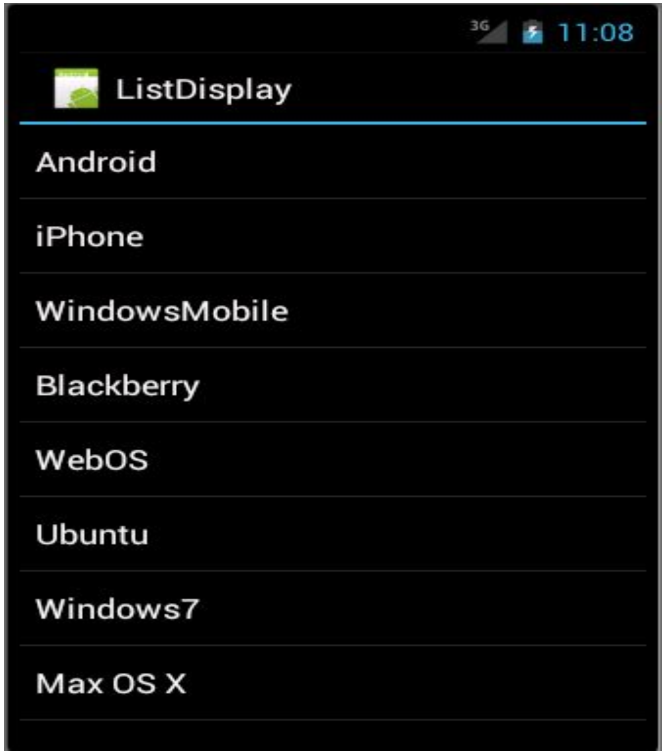
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="20dp"
   android:paddingRight="20dp"
   android:orientation="vertical"
   android:id="@+id/linear_Layout">


   <Spinner
      android:id="@+id/aSpinner"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="100dp"
      android:entries="@array/Spinner_items"
      />
</LinearLayout>
```

Main_activity.java

```
package com.example.spinner;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.Spinner;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Spinner aSpinner=findViewById(R.id.aSpinner);
    }
}
```

LISTVIEW

Android **ListView** is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an **Adapter** that pulls content from a source such as an array or database.



Android

iPhone

WindowsMobile

Blackberry

WebOS

Ubuntu

Windows7

Max OS X

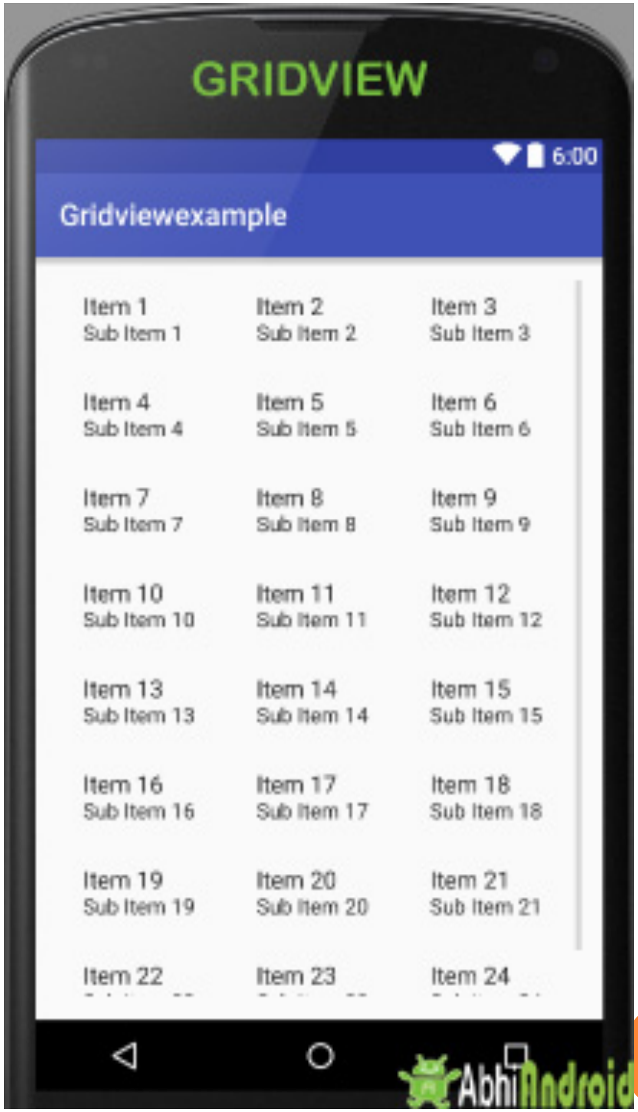
Attributes

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.

GridView

Android **GridView** shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a **ListAdapter.**

The **ListView** and **GridView** are subclasses of **AdapterView** and they can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.



Item 1 Sub Item 1	Item 2 Sub Item 2	Item 3 Sub Item 3
Item 4 Sub Item 4	Item 5 Sub Item 5	Item 6 Sub Item 6
Item 7 Sub Item 7	Item 8 Sub Item 8	Item 9 Sub Item 9
Item 10 Sub Item 10	Item 11 Sub Item 11	Item 12 Sub Item 12
Item 13 Sub Item 13	Item 14 Sub Item 14	Item 15 Sub Item 15
Item 16 Sub Item 16	Item 17 Sub Item 17	Item 18 Sub Item 18
Item 19 Sub Item 19	Item 20 Sub Item 20	Item 21 Sub Item 21
Item 22	Item 23	Item 24

1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:columnWidth** This specifies the fixed width for each column. This could be in px, dp, sp, in, or mm.
3	**android:gravity** Specifies the gravity within each cell. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
4	**android:horizontalSpacing** Defines the default horizontal spacing between columns. This could be in px, dp, sp, in, or mm.
5	**android:numColumns** Defines how many columns to show. May be an integer value, such as "100" or auto_fit which means display as many columns as possible to fill the available space.
6	**android:verticalSpacing** Defines the default vertical spacing between rows. This could be in px, dp, sp, in, or mm.

IMAGE VIEW

In Android, ImageView class is used to display an image file in application.

```
<ImageView android:id="@+id/simpleImageView"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:src="@drawable/lion"

/>
```

Attribute	Description
android:maxHeight	Used to specify a maximum height for this view.
android:maxWidth	Used to specify a maximum width for this view.
android:src	Sets a drawable as the content for this ImageView.
android:scaleType	Controls how the image should be resized or moved to match the size of the ImageView.
android:tint	Tints the color of the image in the ImageView.

android:background:

This property gives a background color to your ImageView. When your image is not able to entirely fill the ImageView, then background color is used to fill the area not covered by the image.

android:padding:

To provide padding or extra space inside the ImageView for the image.

SCALE_TYPE

CENTER: Places the image in center, but does not scale it.
CENTER_CROP: Scales the image uniformly.
CENTER_INSIDE: This will place the image inside the container and the edges of the image will not overlap with that of the container, the image will be inside it.
FIT_CENTER: Scale the image from the center.
FIT_END: Scale the image from the end of the container, i.e from the right hand side.
FIT_START: Scale the image from the start of the container, i.e from the left hand side.
FIT_XY: This will fill the complete container with the image. This generally distorts the image by stretching/sqeezing it in disproportionate ratios.
MATRIX: Used to scale the image using the image matrix when drawing.

SCROLL VIEW

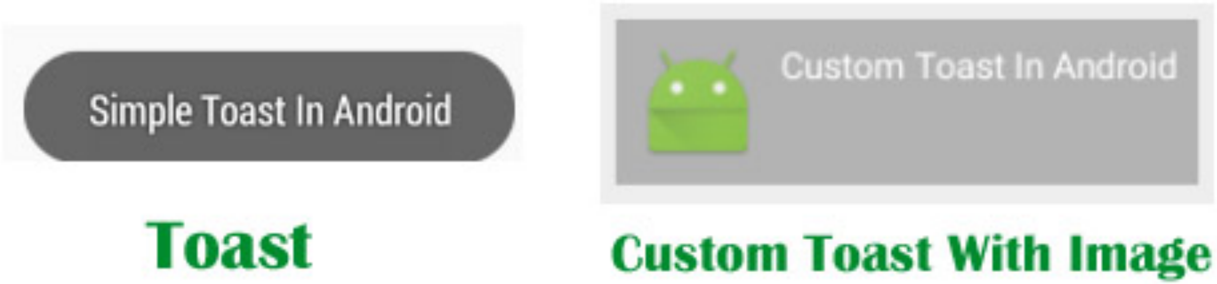
A ScrollView is a view group that is used to make vertically scrollable views. A scroll view contains a single direct child only. In order to place multiple views in the scroll view, one needs to make a view group(like LinearLayout) as a direct child and then we can define many views inside it.

A ScrollView supports Vertical scrolling only, so in order to create a horizontally scrollable view, Horizontal scroll view is used.

android:addStatesFromChildren	Sets whether this ViewGroup’s drawable states also include its children’s drawable states.
android:animateLayoutChanges	Defines whether changes in layout should cause a LayoutTransition to run.
android:clipChildren	Defines whether a child is limited to draw inside of its bounds or not.
android:clipToPadding	Defines whether the ViewGroup will clip its children and resize any EdgeEffect to its padding, if padding is not zero.
android:layoutAnimation	Defines the layout animation to use the first time the ViewGroup is laid out.
android:fillViewport	Defines whether the scrollview should stretch its content to fill the viewport.

CUSTOM TOAST ALERT

In Android, Toast is used to display information for a period of time. It contains a message to be displayed quickly and disappears after specified period of time. Toast is a subclass of Object class.



Simple Toast In Android

Custom Toast In Android

Toast

Custom Toast With Image

Methods

1. **makeText(Context context, CharSequence text, int duration):** This method is used to initiate the Toast. This method take three parameters First is for the application Context, Second is text message and last one is duration for the Toast.

a. LENGTH_LONG: It is used to display the Toast for a long period of time. When we set this duration the Toast will be displayed for a long duration.

b. LENGTH_SHORT: It is used to display the Toast for short period of time. When we set this duration the Toast will be displayed for short duration.

2. show(): This method is used to display the Toast on the screen.

3. setGravity(int,int,int): This method is used to set the gravity for the Toast. This accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

4. setText(CharSequence s): This method is used to set the text for the Toast.

DATE PICKER

Android Date Picker allows you to select the date consisting of day, month and year in your custom user interface.

1	**getDayOfMonth()** This method gets the selected day of month
2	**getMonth()** This method gets the selected month
3	**getYear()** This method gets the selected year
7	**updateDate(int year, int month, int dayOfMonth)** This method updates the current date
8	**getCalendarView()** This method returns calendar view
9	**getFirstDayOfWeek()** This Method returns first day of the week

TIME PICKER

Android Time Picker allows you to select the time of day in either 24 hour or AM/PM mode. The time consists of hours, minutes and clock format. Android provides this functionality through **TimePicker** class.

1	**is24HourView()** This method returns true if this is in 24 hour view else false
2	**isEnabled()** This method returns the enabled status for this view
3	**setCurrentHour(Integer currentHour)** This method sets the current hour
4	**setCurrentMinute(Integer currentMinute)** This method sets the current minute
5	**setIs24HourView(Boolean is24HourView)** This method set whether in 24 hour or AM/PM mode