Index

Designing User Interface with View

What is View in android?

In Android, a view is an object that draws on the screen and handles user interaction. A view can represent a widget such as a button, text field, or image, or it can be a container that holds other views.

Each view in Android is a subclass of the View class, and it has its own set of properties and methods that define its behavior and appearance. For example, a Button view has methods to set its text and onClick behavior, while an ImageView view has methods to set its image source and scale type.

Views are arranged in a hierarchy, where each view can have a parent and zero or more children. The top-level view in the hierarchy is usually the ViewGroup that is attached to the activity's window, such as the LinearLayout, RelativeLayout, or ConstraintLayout. The child views are added to the parent view using layout parameters that define their position and size.

The view hierarchy is essential for creating user interfaces in Android. You can define the layout of your user interface using XML files or programmatically by adding views to the hierarchy at runtime. By interacting with the views in the hierarchy, you can respond to user input and update the appearance of the UI dynamically.

Textview:

A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing.

TextView attributes:

Sr.No.	Attribute & Description
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 autom 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. Default is false.
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 small 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
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:minHeight** Makes the TextView be at least this many pixels tall.
12	**android:minWidth** Makes the TextView be at least this many pixels wide.
13	**android:password**

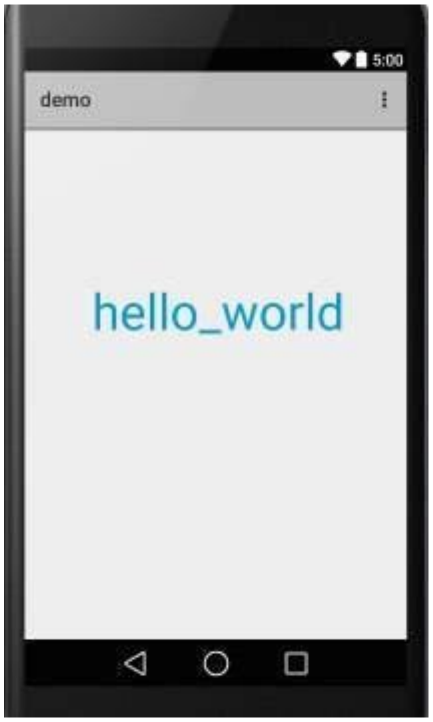
	Whether the characters of the field are displayed as password dots instead of them Possible value either "true" or "false".
14	**android:phoneNumber** If set, specifies that this TextView has a phone number input method. Possible valu "true" or "false".
15	**android:text** Text to display.
16	**android:textAllCaps** Present the text in ALL CAPS. Possible value either "true" or "false".
17	**android:textColor** Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarr
18	**android:textColorHighlight** Color of the text selection highlight.
19	**android:textColorHint** Color of the hint text. May be a color value, in the form of "#rgb", "#argb", "#rrgg "#aarrggbb".
20	**android:textIsSelectable** Indicates that the content of a non-editable text can be selected. Possible value "true" or "false".
21	**android:textSize** Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (ex 15sp).
22	**android:textStyle**

	Style (bold, italic, bolditalic) for the text. You can use or more of the following separated by '|'. • normal - 0 • bold - 1 • italic - 2
23	**android:typeface** Typeface (normal, sans, serif, monospace) for the text. You can use or more of the fol values separated by '|'. • normal - 0 • sans - 1 • serif - 2 • monospace - 3

Xml:

Code:

```
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context=".MainActivity" >

  <TextView
    android:id="@+id/text_id"
    android:layout_width="300dp"
    android:layout_height="200dp"
    android:capitalize="characters"
    android:text="hello_world"
    android:textColor="@android:color/holo_blue_dark"
    android:textColorHighlight="@android:color/primary_text_dark"
    android:layout_centerVertical="true"
    android:layout_alignParentEnd="true"
    android:textSize="50dp"/>

</RelativeLayout>
```



Java:

Grap Textview with Java:

Code:

```
package com.example.demo;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //--- text view---
    TextView txtView = (TextView) findViewById(R.id.text_id);
  }
```

Button:

In Android, a Button is a UI element that users can interact with by clicking or tapping on it to perform an action or trigger an event. It is one of the most common and essential UI components used in Android applications to allow users to interact with the app.

Button attributes:

Attribu	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.

XML:

Code:

```
<?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">
          <Button
          android:id="@+id/addBtn"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Add"
          android:onClick="doOperation"/>
      </LinearLayout>
```

Java:

```
      Button doOp = (Button)findViewById(R.id.doOperation);
      doOp.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                  // Do something in response to button click
                  }
      });
```

or:

```
public void doOperation(){

        // Do something in response to
        button click
}
```

EditText:

In Android, an EditText is a UI element that allows users to enter and edit text. It is one of the most common and essential UI components used in Android applications to collect user input.

An EditText can be used to accept different types of input, such as text, numbers, passwords, or email addresses, depending on the application's needs. It can also be customized with different colors, sizes, and shapes to match the app's theme or branding. You can define the appearance and behavior of an EditText in an XML layout file or programmatically in Java code.

EditText attributes:

Attributes	Description
android:id	It is used to uniquely identify the element
android:width	It is used to set the width of EditText in pixels
android:hight	It is used to set the height of EditText in pixels
android:text	It is used to set the text value inside the EditText
android:inputType	It is used to mention what type of value should pass in EditText, like pla email, password, etc
android:hint	It is used to highlight what should be input when text is empty
android:gravity	gravity is used to align the text like left, right, center, top

android:textSize	It is used to specify the size of the text
android:textStyle	It is used to set the text style like bold, italic, bold italic, etc.
android:textColor	It is used to set the color of the text
android:background	It is used to set the background of your EditText
android:backgroundTint	It is used to set tint to the background of your element
android:maxWidth	It is used to set the maximum width of View in pixel
android:minWidth	It is used to set the minimum width of View in pixel
android:drawableStart	It is used to set the drawable to be drawn to the start of View
android:drawableEnd	It is used to set the drawable to be drawn to the end of View
android:drawableBottom	It is used to set the drawable to be drawn to the bottom of View
android:drawableLeft	It is used to set the drawable to be drawn to the left of View
android:drawableRight	It is used to set the drawable to be drawn to the right of View
android:drawablePadding	It is used to set the drawable to be drawn from the set padding of View

XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >
   <EditText
      android:id="@+id/txtSub"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Subject"
      android:inputType="text"/>
</LinearLayout>
```

Java:

```
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      LinearLayout linearLayout =  (LinearLayout) findViewById(R.id.linearlayout);
      EditText et = new EditText(this);
      et.setHint("Subject");
```

```
        linearLayout.addView(et);
    }
// Get text from editext
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        EditText et = (EditText) findViewById(R.id.txtSub);
        String name = et.getText().toString();
    }
}
```

Checkbox:

In Android, a CheckBox is a UI element that allows the user to select or deselect a particular option. It is commonly used in forms, surveys, or settings screens to offer multiple options to the user. A CheckBox can be customized with text, colors, and styles to match the app's theme or branding.

CheckBox attributes:

Attribute	**Description**
android:id	It is used to uniquely identify the control
android:checked	It is used to specify the current state of checkbox
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 for a checkbox.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of 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 checkbox control.

Attribute	Description
android:padding	It is used to set the padding from left, right, top and bottom.
android:onClick	It's the name of the method to invoke when the checkbox clicked.
android:visibility	It is used to control the visibility of control.

In android, CheckBox is a two-states button that can be either checked (ON) or unchecked (OFF) and it will allow users to toggle between the two states (ON / OFF) based on the requirements.
Generally, we can use multiple CheckBox controls in android application to allow users to select one or more options from the set of values.
By default, the android CheckBox will be in the OFF (Unchecked) state. We can change the default state of CheckBox by using android:checked attribute.
In case, if we want to change the state of CheckBox to ON (Checked), then we need to set android:checked = "true" in our XML layout file.
In android, we can create CheckBox control in two ways either in the XML layout file or create it in the Activity file programmatically.

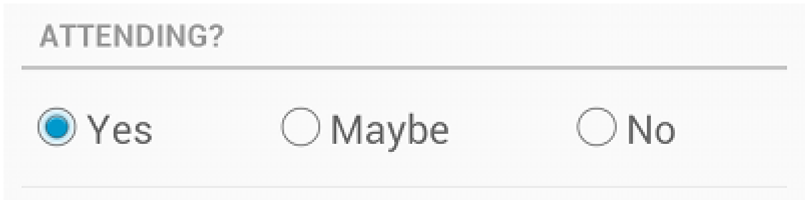
XML:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent" android:layout_height="match_parent">
<CheckBox
  android:id="@+id/chk1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:checked="true"
  android:text="Java"
  android:onClick="onCheckboxClicked" />
</RelativeLayout>
```

Java:

```
LinearLayout layout = (LinearLayout)findViewById(R.id.l_layout);
CheckBox cb = new CheckBox(this);
cb.setText("Madi");
cb.setChecked(true);
layout.addView(cb);
// In Activity that hosts our XML layout file, we need to implement click event method like as shown below to perfom
action.
public void onCheckboxClicked(View view) {
        // Is the view now checked?
        boolean checked = ((CheckBox) view).isChecked();
        // Check which checkbox was clicked
        switch(view.getId()) {
          case R.id.chk1:
            if (checked)
            // Do your coding
          else
            // Do your coding

            break;
          // Perform your logic
        }
      }
```

RadioButton:



In Android, a RadioButton is a UI element that allows the user to select one option from a list of predefined choices. It is commonly used in forms, surveys, or settings screens to offer a set of mutually exclusive options to the user. When one RadioButton is selected, all other RadioButtons in the same group are automatically deselected.

RadioButton attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:checked	It is used to specify the current state of radio button
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 for the radio button.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of the text.

Attribute	Description
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 radio button control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:onClick	It’s the name of the method to invoke when the radio button clicked.
android:visibility	It is used to control the visibility of control.

XML:

```
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Java"
    android:onClick="onRadioButtonClicked"/>

  </RadioGroup>
```

Java:

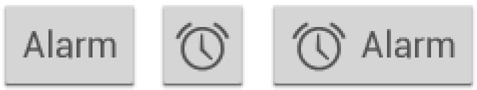
```
LinearLayout layout = (LinearLayout)findViewById(R.id.l_layout);
RadioButton rd = new RadioButton(this);
rd.setText("Madi");
rd.setChecked(true);
layout.addView(rd);
// In Activity that hosts our XML layout file, we need to implement click event method like as shown below.
public void onRadioButtonClicked(View view) {
  // Is the view now checked?
  boolean checked = ((RadioButton) view).isChecked();
  // Check which RadioButton was clicked
  switch(view.getId()) {
    case R.id.chk1:
      if (checked)
      // Do your coding
    else
      // Do your coding

      break;
    // Perform your logic
  }
}
```

RadioGroup:

In Android, a RadioGroup is a UI element that is used to group a set of RadioButton objects together, so that they behave as mutually exclusive options. When a user selects one RadioButton, all other RadioButtons in the same RadioGroup are automatically deselected.
Generally, we use radio buttons within a RadioGroup to combine multiple Radio Buttons into one group and it will make sure that user can select only one option from the group of multiple options.
XML:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent" android:layout_height="match_parent">
<RadioGroup
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
   <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Java"/>
   </RadioGroup> </RelativeLayout>
```

ImageButton:



In Android, an ImageButton is a UI element that is used to display an image that functions as a button. It is similar to a regular Button, but instead of text, it displays an image that can be clicked or tapped to trigger an action.

ImageButton attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:src	It is used to specify the source file of an image
android:background	It is used to set the background color for an image button control.
android:padding	It is used to set the padding from left, right, top and bottom of the image button.

Attribute	Description
android:baseline	It is used to set the offset of the baseline within the view.

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

Java:

```
LinearLayout layout = (LinearLayout)findViewById(R.id.l_layout);
ImageButton btn = new ImageButton(this);
btn.setImageResource(R.drawable.add_icon);
layout.addView(btn);
/** Called when the user touches the button */
public void addOperation(View view) {
   // Do something in response to button click
}
```

ToggleButton:

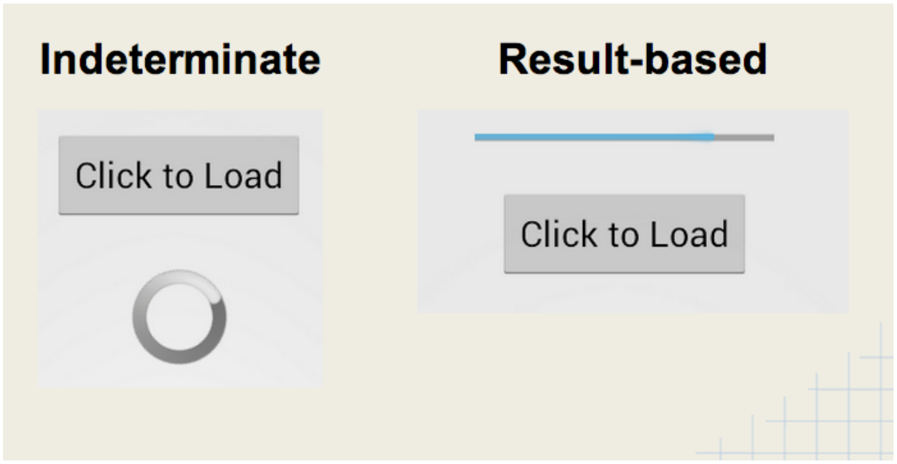
In Android, a ToggleButton is a UI element that represents a two-state button that can be toggled between "on" and "off" states. It is similar to a CheckBox, but it displays a button that can be clicked or tapped to change its state, rather than a checkbox that must be checked or unchecked.

ToggleButton attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:checked	It is used to specify the current state of toggle button
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:textOn	It is used to set the text when the toggle button is in the ON / Checked state.
android:textOff	It is used to set the text when the toggle button is in the OFF / Unchecked state.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of 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 toggle 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 text.
android:drawableRight	It’s drawable to be drawn to the right of the text.
android:drawableLeft	It’s drawable to be drawn to the left of text.

XML:
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent" android:layout_height="match_parent">
   <ToggleButton
      android:id="@+id/toggle1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="100dp"
      android:layout_marginTop="120dp"
      android:checked="true"
      android:textOff="OFF"
      android:textOn="ON"/>
</RelativeLayout>
```
Java:
```
RelativeLayout layout = (RelativeLayout)findViewById(R.id.r_layout);
ToggleButton tb = new ToggleButton(this);
tb.setTextOff("OFF");
tb.setTextOn("ON");
tb.setChecked(true);
layout.addView(tb);

// handle
ToggleButton toggle = (ToggleButton) findViewById(R.id.togglebutton);
toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
   public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
      if (isChecked) {
         // The toggle is enabled
      } else {
         // The toggle is disabled
      }
   }
});
```

ProgressBar:
In Android, a ProgressBar is a UI element that is used to indicate the progress of a task. It can be used to show the progress of a file download, upload, or any other task that takes time to complete.



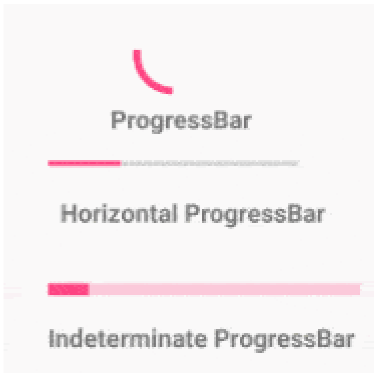
ProgressBar attributes:

Attribute	Description
android:id	It is used to uniquely identify the control
android:minHeight	It is used to set the height of the progress bar.
android:minWidth	It is used to set the width of the progress bar.
android:max	It is used to set the maximum value of the progress bar.
android:progress	It is used to set the default progress value between 0 and max. It must be an integer value.

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.

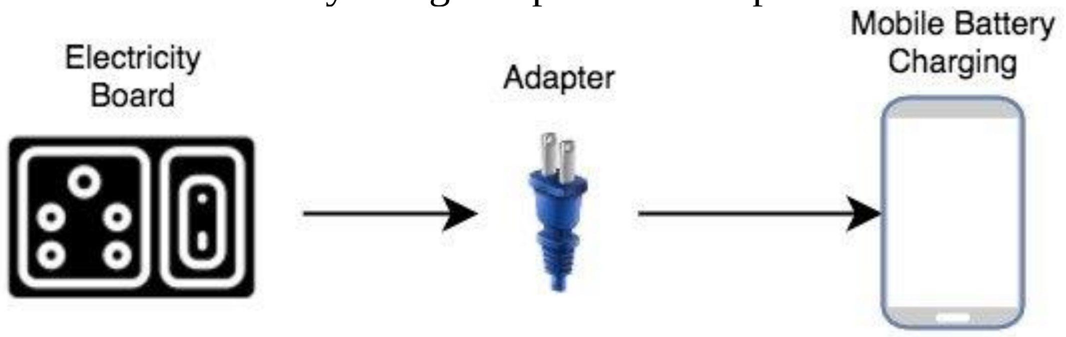
XML:

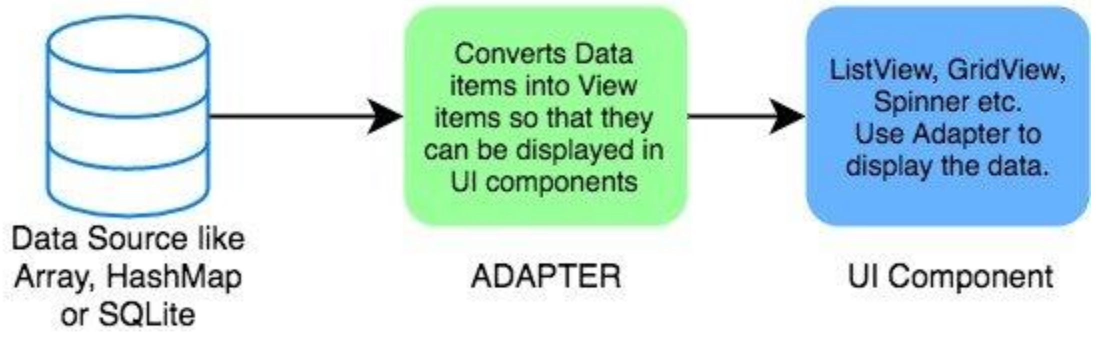
```
<ProgressBar
   android:id="@+id/pBar3"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:minHeight="50dp"
   android:minWidth="250dp"
   android:max="100"
   android:indeterminate="true"
   android:progress="1" />
```



Adapter:

Adapter and Adapter View are so popular, that every time you see any app with a List of items or Grid of items, you can say for sure that it is using Adapter and Adapter View.
Generally, when we create any List or Grid of data, we think we can use a loop to iterate over the data and then set the data to create the list or grid.
But what if the data is a set of 1 million products. Then using a loop will not only consume a lot of time, making the app slow, also it might end up eating all the runtime memory.
All these problems are solved by using Adapter and Adapter View.




Data Source like Array, HashMap or SQLite
Converts Data items into View items so that they can be displayed in UI components
ADAPTER
ListView, GridView, Spinner etc. Use Adapter to display the data.
UI Component

What is an Adapter?

An adapter acts like a bridge between a data source and the user interface. It reads data from various data sources, coverts it into View objects and provide it to the linked Adapter view to create UI components.
The data source or dataset can be an Array object, a List object etc.
You can create your own Adapter class by extending the BaseAdapter class, which is the parent class for all other adapter class. Android SDK also provides some ready-to-use adapter classes, such as ArrayAdapter, SimpleAdapter etc.

What is an Adapter View?

An Adapter View can be used to display large sets of data efficiently in form of List or Grid etc, provided to it by an Adapter.
When we say efficiently, what do we mean?
An Adapter View is capable of displaying millions of items on the User Interface, while keeping the memory and CPU usage very low and without any noticeable lag.
Different Adapters follow different strategies for this, but the default Adapter provided in Android SDK follow the following tricks:
It only renders those View objects which are currently on-screen or are about to some on-screen. Hence no matter how big your data set is, the Adapter View will always load only 5 or 6 or maybe 7 items at once, depending upon the display size. Hence saving memory.
It also reuses the already created layout to populate data items as the user scrolls, hence saving the CPU usage.
Suppose you have a dataset, like a String array with the following contents.

```
String days[] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", 
"Saturday", "Sunday"};
```

Now, what does an Adapter do is that it takes the data from this array and creates a View from this data and then, it gives this View to an AdapterView. The AdapterView then displays the data in the way you want.

Note: Adapter is only responsible for taking the data from a data source and converting it into View and then passing it to the AdapterView. Thus, it is used to manage the data. AdapterView is responsible for displaying the data.
Therefore, you can take the data from a database or an ArrayList or any other data source and then, you can display that data in any arrangement. You can display it vertically (ListView), or in rows and columns (GridView), or in drop-down menu (Spinners), etc.

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. The adapter will hold the data and iterates through an items in data set and generate the views for each item in the list.

Generally, in android we have a different types of adapters available to fetch the data from different data sources to fill the data into adapter views, those are

Adapter	Description
ArrayAdapter	It will expects an Array or List as input.
CurosrAdapter	It will accepts an instance of cursor as an input.
SimpleAdapter	It will accepts 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

ListView:

In Android, a ListView is a UI element that displays a list of items that the user can scroll through vertically. It is often used to display a collection of similar items, such as a list of contacts, emails, or products.
To use a ListView in your Android application, you need to define a layout file that contains the ListView element, and then create an adapter to populate the ListView with data. The adapter is responsible for creating a view for each item in the list and binding data to it.

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="vertical">
  <ListView
    android:id="@+id/userlist"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
  </ListView>
</LinearLayout>
```

Java:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends AppCompatActivity {
  private ListView mListView;
  private ArrayAdapter aAdapter;
  private String[] users = { "Suresh Dasari", "Rohini Alavala", "Trishika Dasari", "Praveen Alavala", "Madav Sai",
"Hamsika Yemineni"};
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mListView = (ListView) findViewById(R.id.userlist);
    aAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, users);
    mListView.setAdapter(aAdapter);
  }
}
```

ListView attributes:

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.

ArrayAdapter

You can use this adapter when your data source is an array. By default, ArrayAdapter creates a view for each array item by calling toString() on each item and placing the contents in a TextView. Consider you have an array of strings you want to display in a ListView, initialize a new ArrayAdapter using a constructor to specify the layout for each string and the string array −

```
ArrayAdapter adapter = new
ArrayAdapter<String>(this,R.layout.ListView,StringArray);
```

Here are arguments for this constructor −

First argument this is the application context. Most of the case, keep it this.
Second argument will be layout defined in XML file and having TextView for each string in the array.
Final argument is an array of strings which will be populated in the text view.

Once you have array adapter created, then simply call setAdapter() on your ListView object as follows −

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

SpinnerView:

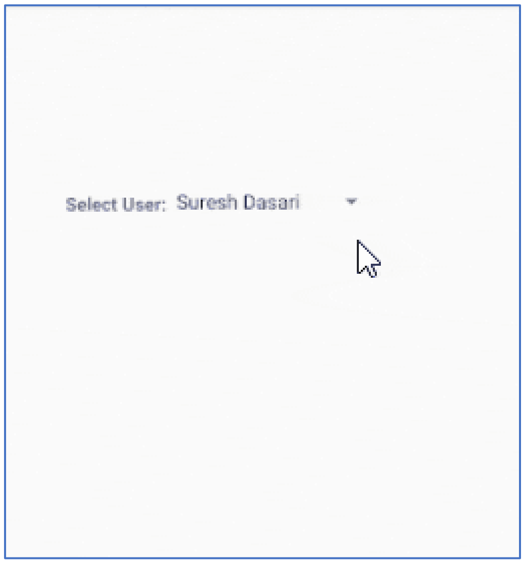
In android, Spinner is a view that allows a user to select one value from the list of values. The spinner in android will behave same as a dropdown list in other programming languages.
Generally, the android spinners will provide a quick way to select one item from the list of values and it will show a dropdown menu with a list of all values when we click or tap on it.
By default, the android spinner will show its currently selected value and by using Adapter we can bind the items to spinner objects.

XML:

```
<Spinner android:id="@+id/spinner1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Java:

```
String[] users = { "Madi", "Trishika Dasari", "Rohini Alavala", "Praveen Kumar", "Madhav Sai" };
Spinner spin = (Spinner) findViewById(R.id.spinner1);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, users);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spin.setAdapter(adapter);
```



Gallery View:

In Android, Gallery is a view used to show items in a center locked, horizontal scrolling list and user will select a view and then user selected view will be shown in the center of the Horizontal list. The items in Gallery are added using Adapter just like in ListView or GridView.

XML:

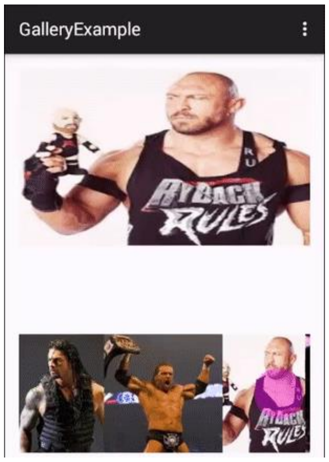
```
<Gallery
android:id="@+id/simpleGallery"
android:layout_width="fill_parent"
android:layout_height="wrap_content"/ >
```

Important Methods Of Gallery In Android:

1. setAnimationDuration(int): This method is used to set the duration for how long a transition animation should run (in milliseconds) when layout has changed.
Below we set the duration for how long a transition animation should run when layout has changed.

Code:

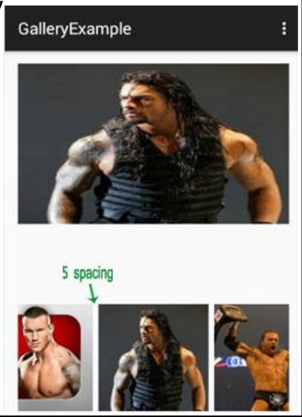
```
Gallery simpleGallery = (Gallery) findViewById(R.id.simpleGallery); // get the reference of Gallery
simpleGallery.setAnimationDuration(3000); // set 3000 milliseconds for animation duration between items of Gallery
```



2. setSpacing(int): This method is used to set the spacing between items in a Gallery.
Below we set the spacing between the items of Gallery.

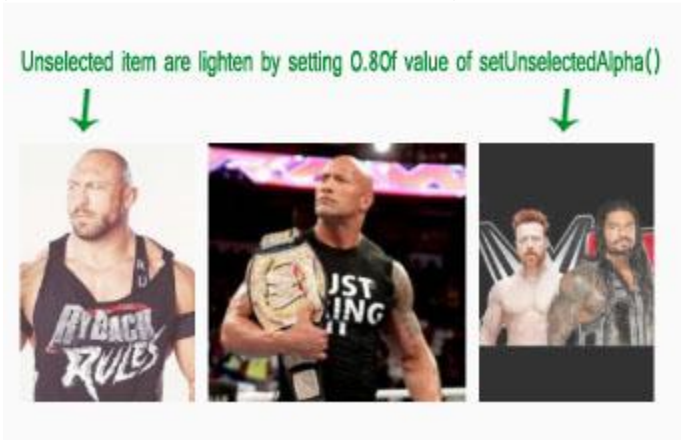
Code:

```
Gallery simpleGallery = (Gallery) findViewById(R.id.simpleGallery); // get the reference of Gallery
simpleGallery.setSpacing(5); // set space between the items of Gallery
```



3.setUnselectedAlpha(float): This method is used to set the alpha on the items that are not selected.
Below we set the alpha value for unselected items of Gallery.

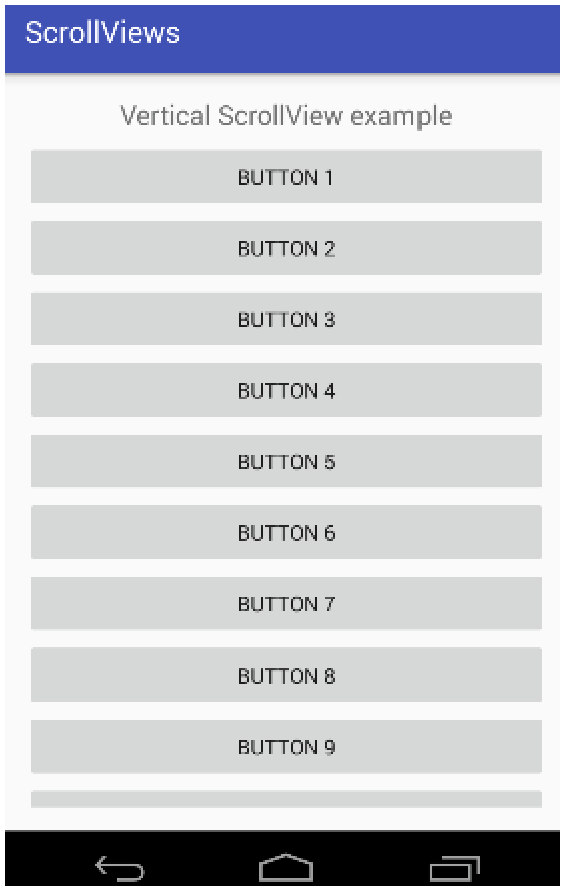
```
Code:
Gallery simpleGallery = (Gallery) findViewById(R.id.simpleGallery); // get the reference of Gallery
simpleGallery.setUnselectedAlpha(0.80f); // set 0.25 value for the alpha of unselected items of Gallery
```


Unselected item are lighten by setting 0.80f value of setUnselectedAlpha()

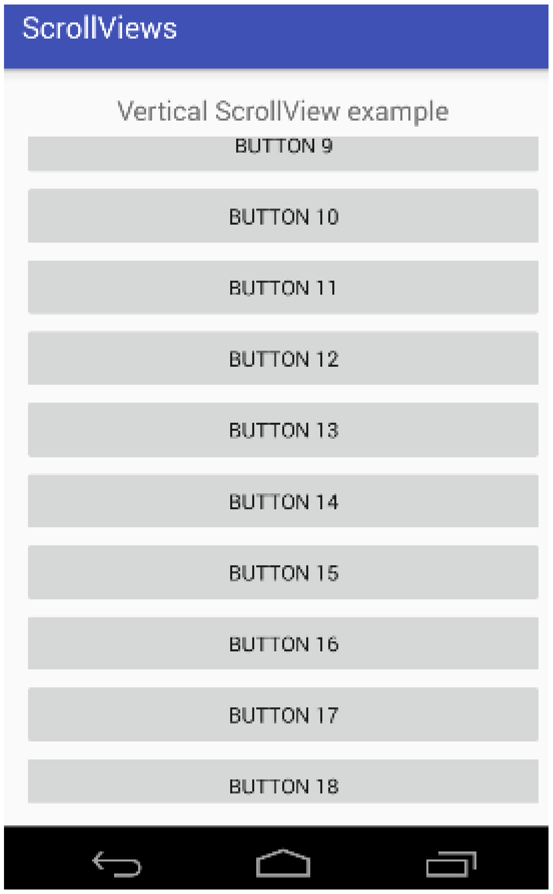
ScrollView:
In android, ScrollView is a kind of layout that is useful to add vertical or horizontal scroll bars to the content which is larger than the actual size of layouts such as linearlayout, relativelayout, framelayout, etc.
Generally, the android ScrollView is useful when we have content that doesn't fit our android app layout screen. The ScrollView will enable a scroll to the content which is exceeding the screen layout and allow users to see the complete content by scrolling.
Ex.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="com.example.test.scrollviews.MainActivity">


  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Vertical ScrollView example"
    android:id="@+id/textView"
    android:layout_gravity="center_horizontal"
    android:layout_centerHorizontal="true"
    android:layout_alignParentTop="true" />
```

```
<ScrollView android:layout_marginTop="30dp"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/scrollView">


    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >

        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 1" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 2" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 3" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 4" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 5" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 6" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 7" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 8" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 9" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 10" />
        <Button
            android:layout_width="fill_parent"
```

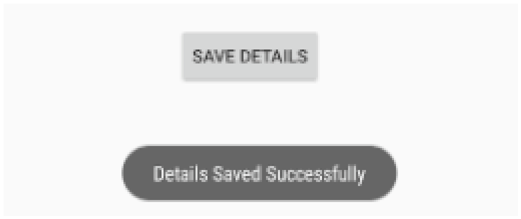


```
            android:layout_height="wrap_content"
            android:text="Button 11" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 12" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 13" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 14" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 15" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 16" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 17" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 18" />

        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 19" />
        <Button
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 20" />

      </LinearLayout>

    </ScrollView>

</RelativeLayout>
```



Toast:

In android, Toast is a small popup notification that is used to display an information about the operation which we performed in our app. The Toast will show the message for a small period of time and it will disappear automatically after a timeout.
Generally, the size of Toast will be adjusted based on the space required for the message and it will be displayed on the top of the main content of activity for a short period of time.
For example, some of the apps will show a message like "Press again to exit" in toast, when we pressed a back button on the home page or showing a message like "saved successfully" toast when we click on the button to save the details.



Create a Toast in Android:

In android, we can create a Toast by instantiating an android.widget.Toast object using makeText() method. The makeText() method will take three parameters: application context, text message and the duration for the toast. We can display the Toast notification by using show() method.
Following is the syntax of creating a Toast in android applications.

Toast.makeText(context, "message", duration).show();

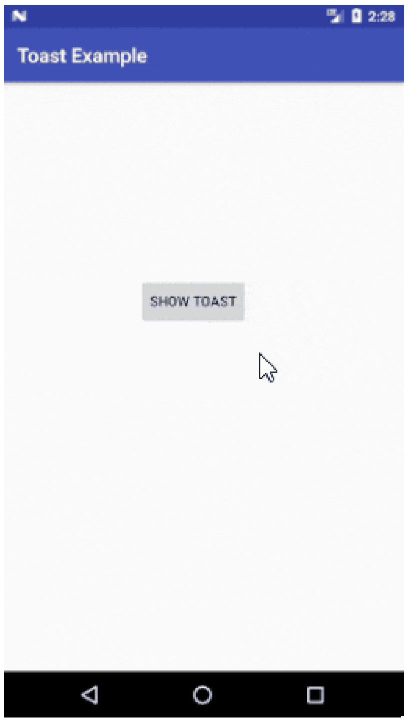
Parameter	Description
context	It's our application context.
message	It's our custom message which we want to show in Toast notification.
duration	It is used to define the duration for notification to display on the screen.

Ex.

```
XML;
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/btnShow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Toast"
        android:layout_marginTop="200dp" android:layout_marginLeft="140dp"/>
</LinearLayout>
```

Java:

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

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = (Button)findViewById(R.id.btnShow);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, "You Clicked on Button..", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```



Change the Position of Android Toast Notification:

By default, the android Toast notification will always appear near the bottom of the screen, centered horizontally like as shown in the above image.
In case if we want to change the position of Toast notification, we can do it by using the setGravity(int, int, int) method. The setGravity() method will accept three parameters: a Gravity constant, an x-position offset, and a y-position offset.
Following is the example of changing the position of android Toast notification to top-right based on offset positions by using setGravity() method.

```
Toast toast =  Toast.makeText(MainActivity.this, "You Clicked on Button..",
Toast.LENGTH_SHORT);
toast.setGravity(Gravity.TOP|Gravity.RIGHT, 100, 250);
toast.show();
```

Custom Toast:

In android, Toast is a small popup notification that is used to display information about the operation which we performed in our app. The Toast will show the message for a small period of time and it will disappear automatically after a timeout.
Generally, the size of Toast will be adjusted based on the space required for the message and it will be displayed on the top of the main content of activity for a short period of time.
Ex.

customToast1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/custom_toast_container"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingLeft="10dp"
  android:paddingRight="10dp"
  android:background="#80CC28">
  <ImageView android:src="@drawable/ic_notification"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="10dp" />
  <TextView android:id="@+id/txtvw"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="13dp"
    android:textColor="#FFF"
    android:textStyle="bold"
    android:textSize="15dp" />
</LinearLayout>
```

Main activity:

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

    <Button
        android:id="@+id/btnShow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Custom Toast"
        android:layout_marginTop="150dp" android:layout_marginLeft="110dp"/>
</LinearLayout>
```

Java:

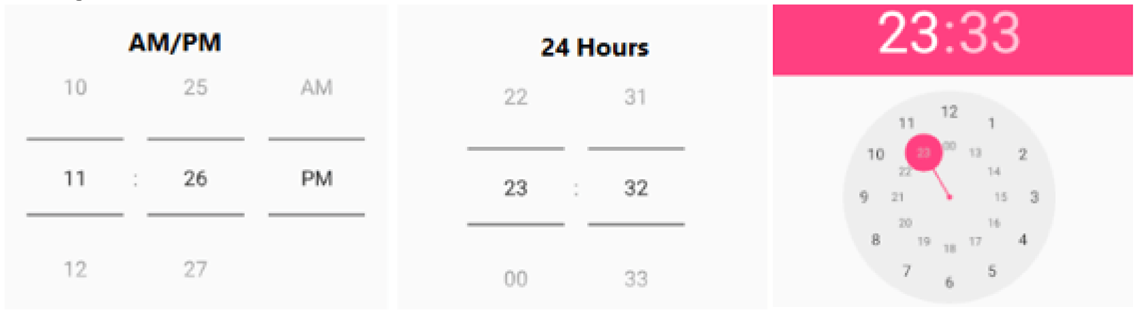
```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = (Button)findViewById(R.id.btnShow);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                LayoutInflater inflater = getLayoutInflater();
                View layout = inflater.inflate(R.layout.custom_toast, (ViewGroup)
findViewById(R.id.custom_toast_layout));
                TextView tv = (TextView) layout.findViewById(R.id.txtvw);
                tv.setText("Custom Toast Notification");
                Toast toast = new Toast(getApplicationContext());
                toast.setGravity(Gravity.CENTER_VERTICAL, 0, 100);
                toast.setDuration(Toast.LENGTH_LONG);
                toast.setView(layout);
                toast.show();
            }
        });
    }
}
```

Picker View:

Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs. Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year). Using these pickers helps ensure that your users can pick a time or date that is valid, formatted correctly, and adjusted to the user's locale.

Time Picker:

In android, TimePicker is a widget for selecting the time of day, in either 24-hour or AM/PM mode.
If we use TimePicker in our application, it will ensure that the users will select a valid time for the day.

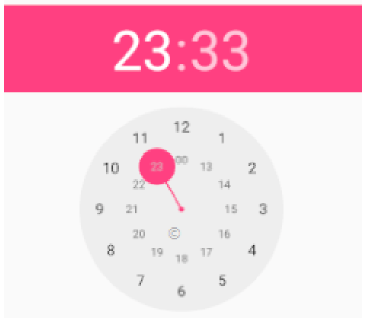


Time Picker Attributes:

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.

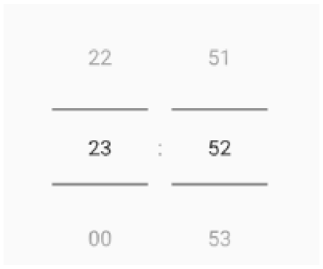
Xml:

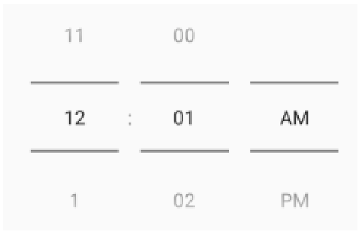
```
<TimePicker android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="clock" />
```



Xml:

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



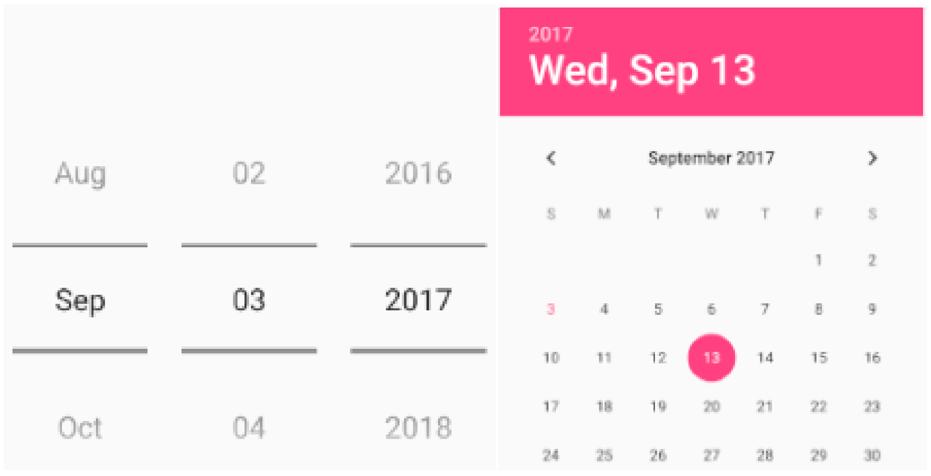


Java:

```
TimePicker picker=(TimePicker)findViewById(R.id.timePicker1);
picker.setIs24HourView(true);
hour = picker.getCurrentHour();
minute = picker.getCurrentMinute();
```

Date Picker:

n android, DatePicker is a control that will allow users to select the date by a day, month and year in our application user interface.
If we use DatePicker in our application, it will ensure that the users will select a valid date.

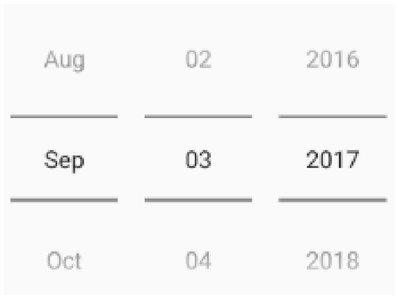


Date Picker Attributes:

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.

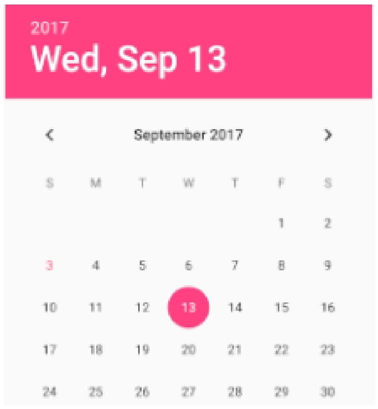
Attribute	Description
android:padding	It is used to set the padding for left, right, top or bottom of the date picker.

Xml:



```
<DatePicker
  android:id="@+id/datePicker1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:datePickerMode="spinner"
  android:calendarViewShown="false"/>
```

Xml:



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

Ex.

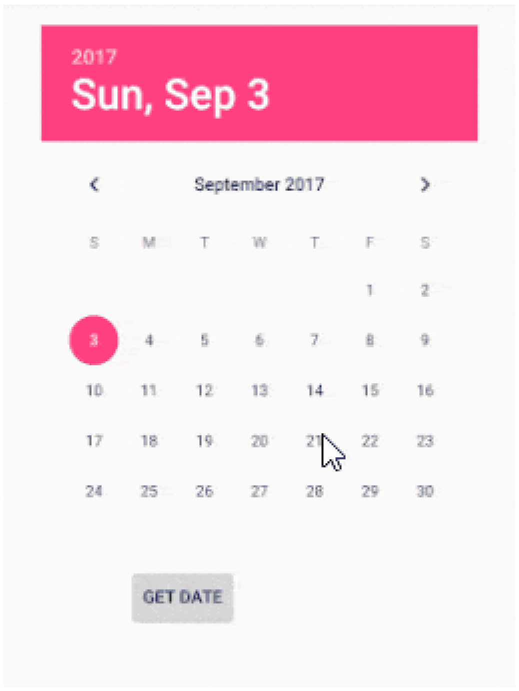
Xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent" android:layout_height="match_parent">
  <DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp" />
  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/datePicker1"
    android:layout_marginLeft="100dp"
    android:text="Get Date" />
  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button1"
    android:layout_marginLeft="100dp"
    android:layout_marginTop="10dp"
```

```
        android:textStyle="bold"
        android:textSize="18dp"/>
</RelativeLayout>
```

Java:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    DatePicker picker;
    Button btnGet;
    TextView tvw;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvw=(TextView)findViewById(R.id.textView1);
        picker=(DatePicker)findViewById(R.id.datePicker1);
        btnGet=(Button)findViewById(R.id.button1);
        btnGet.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                tvw.setText("Selected Date: "+ picker.getDayOfMonth()+"/"+ (picker.getMonth() +
1)+"/"+picker.getYear());
            }
        });
    }
}
```


2017
Sun, Sep 3
September 2017
S M T W T F S
GET DATE