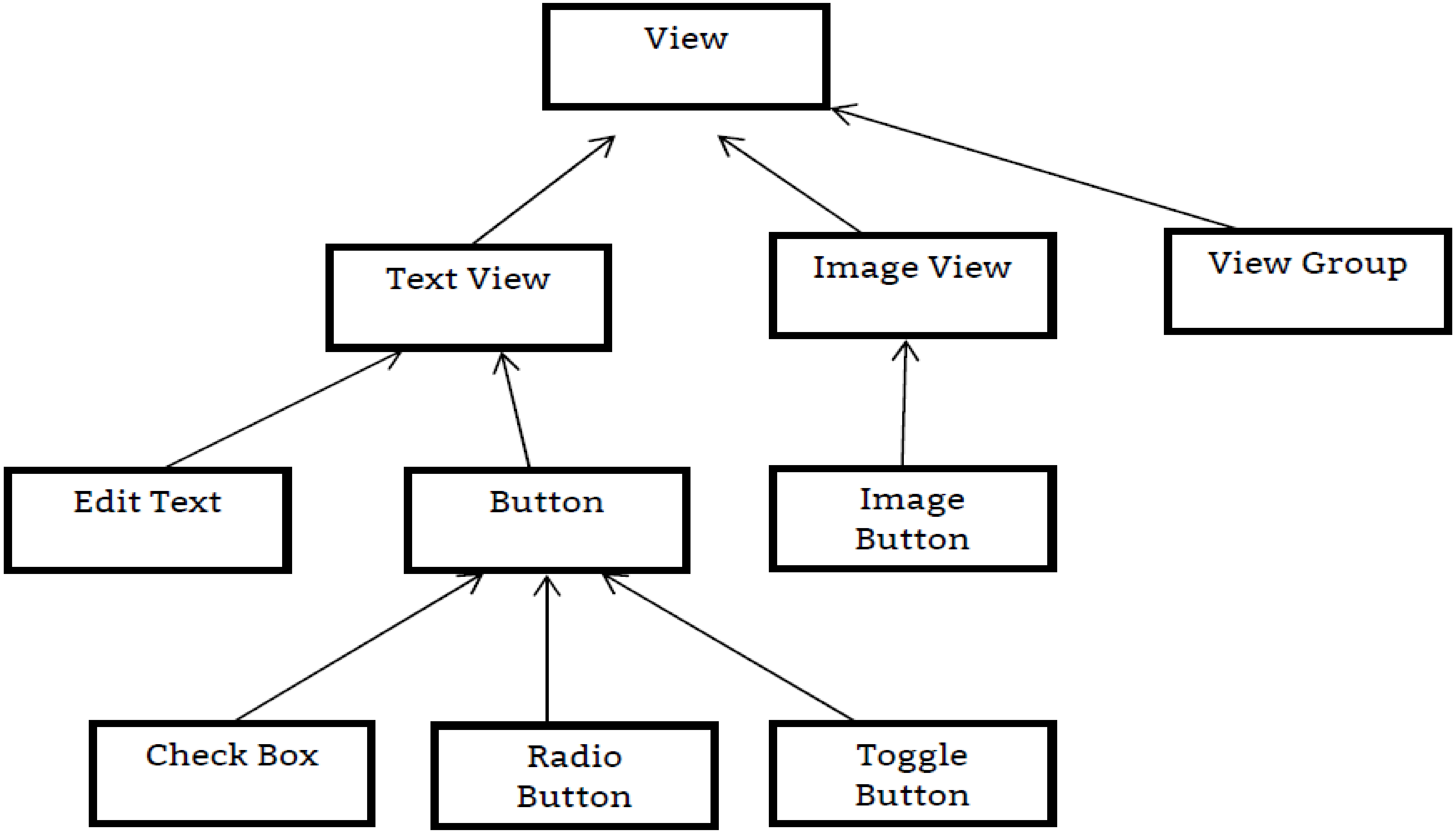
Designing User Interface with View

Prof. Prasad Koyande
Vidyalankar Polytechnic


View
Text View
Image View
View Group
Edit Text
Button
Image Button
Check Box
Radio Button
Toggle Button

TEXT VIEW

- A TextView displays text to the user and optionally allows them to edit it.
- A standard read-only text label that supports string formatting, multiline display, and automatic word wrapping

Attributes/Properties of TextView

- **id:** Supply an identifier name of this view, to later retrieve it with View.findViewByID() or Activity.findViewById()
  - android:id="@+id/simpleTextView“
- **alpha:** alpha property of the view as a value between 0 (entirely transparent) and 1(Completely Opaque).
- **auto link:** Controls whether links such as urls and email addresses are automatically found and converted to clickable links

- **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of the text like left, right, center, top, bottom, center_vertical, center_horizontal etc.
- **text:** text attribute is used to set the text in a text view.
- **textColor:** textColor attribute is used to set the text color of a text view.
- **textSize:** textSize attribute is used to set the size of text of a text view. We can set the text size in sp(scale independent pixel) or dp(density pixel).

- **textStyle:** textStyle attribute is used to set the text style of a text view. The possible text styles are bold, italic and normal. If we need to use two or more styles for a text view then "|" operator is used for that.
- **background:** background attribute is used to set the background of a text view. We can set a color or a drawable in the background of a text view
- **padding:** padding attribute is used to set the padding from left, right, top or bottom.

BUTTON

- In Android, Button represents a push button. A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.
- There are different types of buttons used in android such as CompoundButton, ToggleButton, RadioButton.
- Button is a subclass of TextView class and compound button is the subclass of Button class.
- On a button we can perform different actions or events like click event, pressed event, touch event etc

Properties of Button:

- **Auto link:**
- **Clickable:** Defines whether this view reacts to click events.
- **id:**
- **gravity:**
- **text:** text attribute is used to set the text in a Button.
- **textColor:**
- **textSize:**
- **textStyle:**
- **background:**
- **padding:**

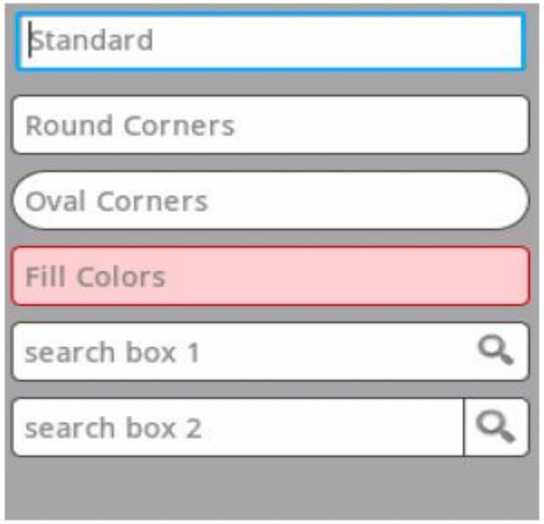
- **drawableBottom:** drawableBottom is the drawable to be drawn to the below of the text.

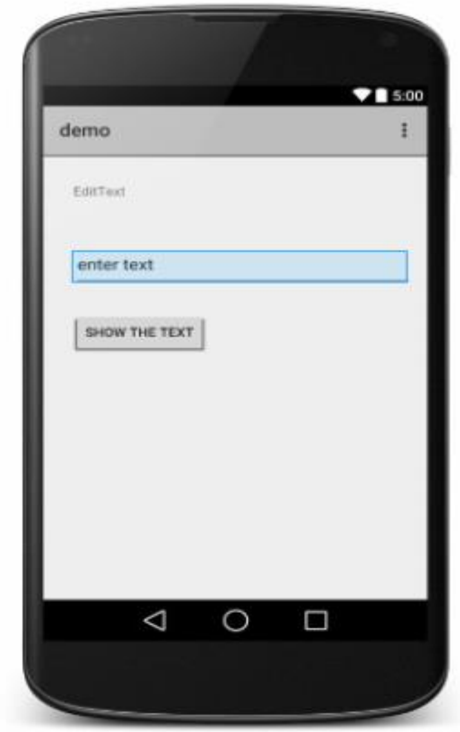
```
<Button
android:id="@+id/simpleButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:background="#147D03"
android:text="Download Code"
android:textSize="20sp"
android:padding="15dp"
android:textStyle="bold|italic"
android:drawableBottom="@drawable/ic_launcher"/>
```

- **drawableTop, drawableRight And drawableLeft:**

EDIT TEXT

- An editable text entry box that accepts multiline entry, word wrapping and hint text.
- 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.





Properties of EditText

- **Alpha:**
- **id:**
- **gravity:**
- **text:**
- **hint:** hint is an attribute used to set the hint i.e. what you want user to enter in this edit text.
- **textColor:**
- **textColorHint:** textColorHint is an attribute used to set the color of displayed hint.
- **textSize:**
- **textStyle:**
- **background:**
- **padding:**

IMAGEBUTTON

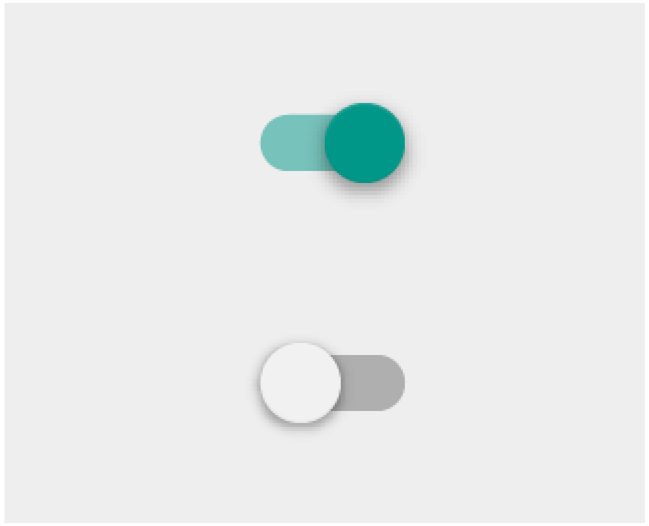
- An ImageButton is an AbsoluteLayout which enables you to specify the exact location of its children.
- ImageButton is used to display a normal button with a custom image in a button
- By default it looks like a normal button with the standard button background that changes the color during different button states

Properties of ImageButton

- **id:**
- **src:**
- **background:**
- **padding, paddingRight, paddingLeft, paddingTop, paddingBottom,**

TOGGLEBUTTON

- ToggleButton is used to display checked and unchecked state of a button
- ToggleButton basically an off/on button with a light indicator which indicate the current state of toggle button

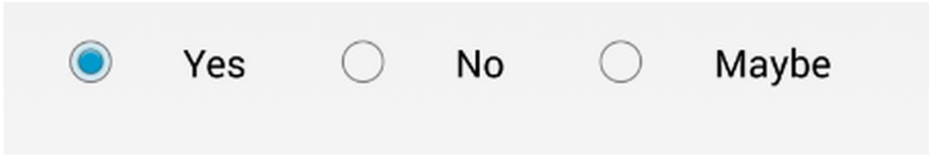


ToggleButton Properties

- **id:**
- **checked:** checked is an attribute of toggle button used to set the current state of a toggle button.(true/false)(Default – false)
- **gravity:**
- **textOn And textOff:**textOn attribute is used to set the text when toggle button is in checked/on state.
- **textColor:**
- **textSize:**
- **textStyle:**
- **background:**
- **padding:**
- **drawableBottom, drawableTop, drawableRight And drawableLeft**

RADIOBUTTON AND RADIO GROUP

- A RadioButton has two states: either checked or unchecked.This allows the user to select one option from a set.
- A RadioGroup class is used for set of radio buttons.
- In RadioGroup checking the one radio button out of several radio button added in it will automatically unchecked all the others.



Properties of RadioButton

- **id:**
- **checked:** checked attribute in radio button is used to set the current state of a radio button.(true/false)
- **gravity:**
- **textColor:**
- **textSize:**
- **textStyle:**
- **background:**
- **padding:**
- **drawableBottom, drawableTop, drawableLeft And drawableRight**

CHECKBOX

- A CheckBox is an on/off switch that can be toggled by the user

Properties of Checkbox

- **id:**
- **checked:**
- **gravity:**
- **text:**
- **textColor:**
- **textSize:**
- **textStyle:**
- **background:**
- **padding:**

PROGRESS BAR

- ProgressBar is used to show the progress of an operation
- **id**
- **max:** Used to define maximum value of the progress can take (like 100, 200 etc)
- **progress:** Used to define the default progress value between 0 and max
- **progressDrawable:** progress drawable is an attribute used in Android to set the custom drawable for the progress mode.

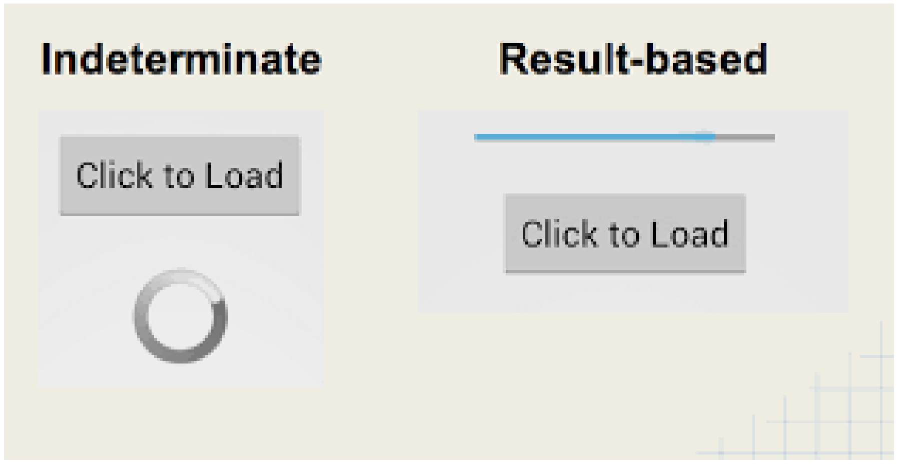
Step 1: Add this code in activity_main.Xml or main.xml inside layout.

```
<ProgressBar
android:id="@+id/simpleProgressBar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:max="100"
android:progress="60"
android:layout_marginTop="100dp"
style="@style/Widget.AppCompat.ProgressBar.Horizontal"
android:progressDrawable="@drawable/custom_progress"/><!--custom progress drawable for
progress mode-->
```

Step 2: Create a new drawable resource xml in drawable folder and name it custom_progress. Here add the below code which creates gradient effect in progressbar.

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

<item>
<shape>
<gradient
                android:endColor="#fff"
                android:startColor="#1f1"
                android:useLevel="true" />

</shape>
</item>



</layer-list>
```

- **background:**
- **indeterminate:**
  - In this mode a progress bar shows a cyclic animation without an indication of progress.
  - This mode is used in application when we don’t know the amount of work to be done.
- **padding:**



LIST VIEW

- ListView displays a vertically-scrollable collection of views, where each view is positioned Immediately below the previous view in the list.
- 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.

- **id:**
- **divider:**
- **dividerHeight:**

```
<ListView
android:id="@+id/simpleListView"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:divider="#f00"
android:dividerHeight="1dp"
/>
```

- **listSelector:**
  - listSelector property is used to set the selector of the listView.
  - It is generally orange or Sky blue color mostly but we can also define your custom color or an image as a list selector as per our design.

Adapters Use in ListView

- An adapter is a bridge between UI component and data source that helps us to fill data in UI component.
- It holds the data and send the data to adapter view then view can takes the data from the adapter view and shows the data on different views like as list view, grid view, spinner etc.
- Two types of Adapter
  - Array Adapter
  - Base Adapter

Array Adapter

- Whenever we have a list of single items which is backed by an array, we can use ArrayAdapter. For instance, list of phone contacts, countries or names.
- ArrayAdapter expects a Layout with a single TextView, If you want to use more complex views means more customization in list items, we have to use custom adapters.

Base Adapter:

- BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView.
- Whenever you need a customized list we create our own adapter and extend base adapter in that.
- Base Adapter can be extended to create a custom Adapter for displaying a custom list item.

GRID VIEW

- Android GridView shows items in two-dimensional scrolling grid (rows and columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter.
- GridView will fill the entire screen. Android has a GridView control that can display data in the form of a grid.
- Content of the grid can be text, images,

IMAGE VIEW

- In android, we can use “android.widget.ImageView” class to display an image file.

SCROLLVIEW

- A view group that allows the view hierarchy placed within it to be scrolled. Scroll view may have only one direct child placed within it.
- Scroll view supports vertical scrolling only. For horizontal scrolling, use HorizontalScrollView
- SrollView is a control for development a View container with a vertical scrollbar. This is useful when we have too much to fit onto a single screen.

CUSTOM TOAST ALERT

- 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.
- It does not block the user interaction. Toast is a subclass of Object class.
- In this we use two constants for setting the duration for the Toast.
- Toast notification in android always appears near the bottom of the screen.
- We can also create our custom toast by using custom layout(xml file).

TIMEPICKER

- TimePicker is a widget used for selecting the time of the day in either AM/PM mode or 24 hours mode.
- The displayed time consist of hours, minutes and clock format. If we need to show this view as a Dialog then we have to use a TimePickerDialog class.
- 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.

Methods of TimePicker

- **setCurrentHour(Integer currentHour):**
  - **setHour(Integer hour):** setCurrentHour() method was deprecated in API level 23.
- **setCurrentMinute(Integer currentMinute):**
  - setMinute(Integer minute): setCurrentMinute() method was deprecated in API level 23.
- **getCurrentMinute():**
  - **getMinute():** getCurrentMinute() method was deprecated in API level 23.
- **setIs24HourView(Boolean is24HourView):**
  - This method is used to set the mode of the Time picker either 24 hour mode or AM/PM mode.
  - In this method we set a Boolean value either true or false.

- **is24HourView():**
  - This method is used to check the current mode of the time picker.
  - This method returns true if its 24 hour mode or false if AM/PM mode is set.
- **setOnTimeChangedListener(TimePicker.OnTimeChangedListener onTimeChangedListener):**
  - This method is used to set the callback that indicates the time has been adjusted by the user.

Attributes of TimePicker

- **id:**
- **timePickerMode:** time picker mode is an attribute of time picker used to set the mode either spinner or clock.
- **background:**
- **padding:**

DATE PICKER

- In Android, DatePicker is a widget used to select a date.
- It allows to select date by day, month and year in our custom UI (user interface).

Methods of DatePicker:

- **setSpinnersShown(boolean shown):**
  - This method is used to set whether the spinner of the date picker in shown or not.
  - In this method you have to set a Boolean value either true or false.
  - True indicates spinner is shown, false value indicates spinner is not shown.
- **getDayOfMonth(), getMonth(), getYear():**
- **getFirstDayOfWeek():**
  - Used to get the first day of the week. This method returnsan integer value.

Attributes of DatePicker

- **id:**
- **datePickerMode:**
  - Used to set the Date Picker in mode either spinner or calendar
- **background:**
- **padding:**