UNIT III: Designing Your User Interface with Views

3.1 Text View, Button, Image Button, Edit Text.

3.2 CheckBox, Toggle Button, Radio Button, and Radio Group.

3.3 Views, Progress Bar View, AutoCompleteTextView, Image View, Image Switcher.

3.4 TimePicker View, Date Picker View.

3.5 Using List Views to Display Long Lists-List View, Using the Spinner.

3.1 Text View, Button, Image Button, Edit Text.

1 Text View

Text View in Android is one of the basic and important UI elements. This plays a very important role in the UI experience and depends on how the information is displayed to the user.

Attributes of Text View:

1. **id:** id is an attribute used to uniquely identify a text view. Below is the example code in which we set the id of a text view.

2. **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.

3. **text:** text attribute is used to set the text in a text view. We can set the text in xml as well as in the java class.

4. **textColor:** textColor attribute is used to set the text color of a text view. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

5. **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).

6. **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.

```
<TextView
   android:id="@+id/simpleTextView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="AbhiAndroid"
   android:layout_centerInParent="true"
   android:textSize="40sp"
   android:textStyle="bold|italic"/>
```

2 Button

Button represents a push button. A Push buttons can be clicked, or pressed by the user to perform an action.

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.**

Attributes of Button in Android:

1. **id:** id is an attribute used to uniquely identify a text Button. Below is the example code in which we set the id of a Button.
2. **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.
3. **text**: text attribute is used to set the text in a Button. We can set the text in xml as well as in the java class.
4. **textColor:** textColor attribute is used to set the text color of a Button. Color value is in the form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.
5. **textSize:** textSize attribute is used to set the size of the text on Button. We can set the text size in sp(scale independent pixel) or dp(density pixel).
6. **background:** background attribute is used to set the background of a Button. We can set a color or a drawable in the background of a Button.
7. **padding:** padding attribute is used to set the padding from left, right, top or bottom. In above example code of background we also set the 10dp padding from all the side’s of button.

```
<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:drawableRight="@drawable/ic_launcher"/>
```

3 Image Button

ImageButton is a button with an image that can be pressed or clicked by the users

Attributes of ImageButton:

1. **id:** id is an attribute used to uniquely identify a image button. Below is the example code in which we set the id of a image button.

2. **src:** src is an attribute used to set a source file of image or you can say image in your image button to make your layout look attractive.

3. **background:** background attribute is used to set the background of an image button. We can set a color or a drawable in the background of a Button.

4. **padding:** padding attribute is used to set the padding from left, right, top or bottom of the ImageButton.

5. **paddingRight :** set the padding from the right side of the image button**.**

6. **paddingLeft :** set the padding from the left side of the image button**.**

7. **paddingTop :** set the padding from the top side of the image button**.**

8. **paddingBottom :** set the padding from the bottom side of the image button**.**
9. **padding :** set the padding from the all side's of the image button**.**

```
<ImageButton
   android:id="@+id/simpleImageButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:background="#000"
   android:src="@drawable/home"
   android:padding="30dp"/>
```

4 Edit Text

EditText is a standard entry widget in android apps

Attributes of EditText:

1. **id:** id is an attribute used to uniquely identify a text EditText. Below is the example code in which we set the id of a edit text.

2. **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.

3. **hint:** hint is an attribute used to set the hint i.e. what you want user to enter in this edit text. Whenever user start to type in edit text the hint will automatically disappear.

4. **textColorHint:** textColorHint is an attribute used to set the color of displayed hint.

```
<EditText
   android:id="@+id/simpleEditText"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:hint="Enter Your Name Here"
   android:textColorHint="#0f0"/>
```

3.2 CheckBox, Toggle Button, Radio Button, and Radio Group.

1 CheckBox

CheckBox is a type of two state button either unchecked or checked in Android.

Attributes of CheckBox:

1.id: id is an attribute used to uniquely identify a check box. Below we set the id of a image button.

2. checked: checked is an attribute of check box used to set the current state of a check box. The value should be true or false where true shows the checked state and false shows unchecked state of a check box. The default value of checked attribute is false. We can also set the current state programmatically.

3. gravity: The gravity attribute is an optional attribute which is used to control the alignment of the text in CheckBox like left, right, center, top, bottom, center_vertical, center_horizontal etc.

4. text: text attribute is used to set the text in a check box. We can set the text in xml as well as in the java class.

5. textColor: textColor attribute is used to set the text color of a check box.

6. textSize: textSize attribute is used to set the size of text of a check box. We can set the text size in sp(scale independent pixel) or dp(density pixel).

```
<CheckBox
   android:id="@+id/simpleCheckBox"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Text size Attribute Of Check Box"
   android:textColor="#fff"
   android:textSize="20sp"
   android:textStyle="bold|italic"
   android:checked="true"
   android:background="#000" />
```

2 Toggle Button

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.

Attributes of ToggleButton:

1. **id:** id is an attribute used to uniquely identify a toggle button.
2. **checked:** checked is an attribute of toggle button used to set the current state of a toggle button. The value should be true or false where true shows the checked state and false shows unchecked state of a toggle button. The default value of checked attribute is false. We can also set the current state programmatically.
3. **textOn And textOff:** textOn attribute is used to set the text when toggle button is in checked/on state. We can set the textOn in XML as well as in the java class.
4. **textColor:** textColor attribute is used to set the text color of a toggle button.

```
<ToggleButton
  android:id="@+id/simpleToggleButton"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:checked="true"
  android:textOff="Off State"
  android:textOn="On State"
  android:layout_centerHorizontal="true"
  android:textColor="#000"
  android:drawableTop="@drawable/ic_launcher" />
```

3 Radio Button, and Radio Group.

RadioButton are mainly used together in a RadioGroup. In RadioGroup checking the one radio button out of several radio button added in it will automatically unchecked all the others. It means at one time we can checked only one radio button from a group of radio buttons which belong to same radio group.

The most common use of radio button is in Quiz Android App code. RadioButon is a two state button that can be checked or unchecked. If a radio button is unchecked then a user can check it by simply clicking on it.

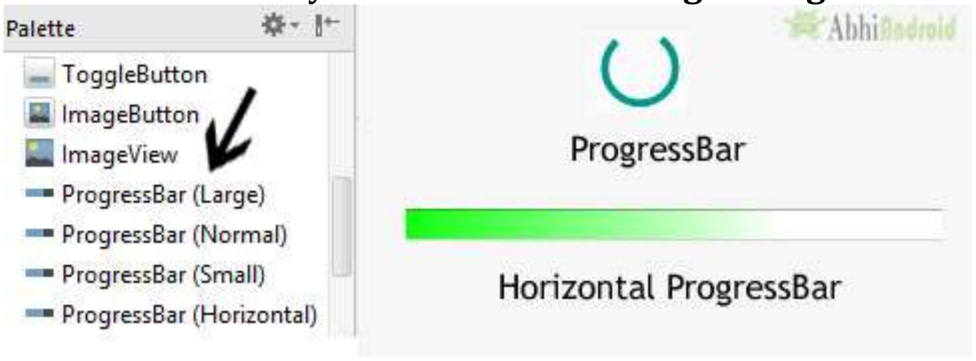
Attributes of RadioButton In Android:

1. **id:** id is an attribute used to uniquely identify a radio button.
2. **checked:** checked attribute in radio button is used to set the current state of a radio button. We can set it either true or false where true shows the checked state and false shows unchecked state of a radio button.
3. **text:** text attribute is used to set the text in a radio button.
4. **gravity:** The gravity attribute is an optional attribute which is used to control the alignment of text like left, right, center, top, bottom, center_vertical, center_horizontal etc.
5. **textColor:** textColor attribute is used to set the text color of a radio button.

3.3 Progress Bar View, AutoCompleteTextView, Image View, Image Switcher, Using the Spinner View.

1 Progress Bar View

Progress Bar is used to display the status of work being done like analyzing status of work or downloading a file etc.
By default a progress bar will be displayed as a spinning wheel but If we want it to be displayed as a horizontal bar then we need to use style attribute as horizontal. It mainly use the **"android.widget.ProgressBar"** class.



```
<ProgressBar
android:id="@+id/simpleProgressBar"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

```
<ProgressBar
android:id="@+id/simpleProgressBar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
style="@style/Widget.AppCompat.ProgressBar.Horizontal"/>
```

2 AutoCompleteTextView

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. The list of suggestions is displayed in drop down menu. The user can choose an item from there to replace the content of edit box with.
android:completionHint

This defines the hint displayed in the drop down menu.

```
  <AutoCompleteTextView
      android:id="@+id/autoCompleteTextView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content”
/>
```

3 Image View

ImageView class is used to display any kind of image resource in the android application either it can be android.graphics.Bitmap

Attributes of ImageView

android:id - To uniquely identify an image view
android:src: src is an attribute used to set a source file or you can say image in your imageview to make your layout attractive.
android:background - To provide a background color to the inserted image
android:padding - To add padding to the image from the left, right, top, or bottom of the view

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

4 Image Switcher

Android image switcher provides an animation over images to transition from one image to another.

Syntax:

```
  <ImageSwitcher
     android:id="@+id/imgSw"
     android:layout_width="match_parent"
     android:layout_height="250dp">
</ImageSwitcher>
```

After that we need to create an instance of ImageSwitcher and use setFactory() method to implement the ViewFactory interface to return the ImageView. We need to add the required animation effects to ImageSwitcher based on our requirements.

3.4 Time Picker View, Date Picker View.

TimePicker

Android TimePicker widget is used to select date. It allows you to select time by hour and minute. You cannot select time by seconds.

The android.widget.TimePicker is the subclass of FrameLayout class.

1. setHour(Integer hour):
This method is used to set the current hours in a time picker.

2. setMinute(Integer minute):
This method is used to set the current minutes in a time picker.

3. getHour():
This method is used to get the current hours from a time picker.

4. getMinute():
This method is used to get the current minutes from a time picker.
getMinute(): From api level 23 we have to use getMinute(). This method returns an integer value.

5. 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.

Attributes of TimePicker:

1. id: id is an attribute used to uniquely identify a time picker.
2. timePickerMode: time picker mode is an attribute of time picker used to set the mode either spinner or clock
3. background: background attribute is used to set the background of a time picker. We can set a color or a drawable image in the background. We can also set the background color programmatically means in java class.
4. padding: padding attribute is used to set the padding from left, right, top or bottom for a time picker.

```
<TimePicker
    android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

DatePicker

DatePicker is a widget used to select a date. It allows to select date by day, month and year in your custom UI (user interface). If we need to show this view as a dialog then we have to use a DatePickerDialog class.

Methods of DatePicker

1. getDayOfMonth():
This method is used to get the selected day of the month from a date picker. This method returns an integer value.

2. getMonth():
This method is used to get the selected month from a date picker. This method returns an integer value.

3. getYear():
This method is used to get the selected year from a date picker. This method returns an integer value.

4. getFirstDayOfWeek():
This method is used to get the first day of the week. This method returns an integer value.

Attributes of DatePicker

1. id: id is an attribute used to uniquely identify a date picker.

2. datePickerMode: This attribute is used to set the Date Picker in mode either spinner or calendar. Default mode is calendar but this mode is not used after api level 21, so from api level 21 you have to set the mode to spinner.

3. padding: padding attribute is used to set the padding from left, right, top or bottom for a date picker.

paddingRight: set the padding from the right side of the date picker.
paddingLeft: set the padding from the left side of the date picker.
paddingTop: set the padding from the top side of the date picker.
paddingBottom: set the padding from the bottom side of the date picker.
Padding: set the padding from the all side's of the date picker.

```
<DatePicker
android:id="@+id/simpleDatePicker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:datePickerMode="spinner"
android:padding="40dp"/>
```

3.5 Using List Views to Display Long Lists-List View

List of scrollable items can be displayed in Android using ListView.

It helps you to displaying the data in the form of a scrollable list. Users can then select any list item by clicking on it.

ListView is default scrollable so we do not need to use scroll View or anything else with ListView.

ListView is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.

```
<ListView
    android:id="@+id/user_list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:dividerHeight="1dp" />
```

Atributes of ListView.

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 specify the height of the divider between list items. This could be in dp(density pixel),sp(scale independent pixel) or px(pixel).

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

android:entries

Specifies the reference to an array resource that will populate the ListView.

In android commonly used adapters are:

1. Array Adapter
2. Base Adapter

Array Adapter:

Whenever you have a list of single items which is backed by an array, you can use ArrayAdapter. For instance, list of phone contacts, countries or names.

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

2. 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 you create your own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying a custom list item. ArrayAdapter is also an implementation of BaseAdapter.

3.6 Using the Spinner.

Spinner provides a quick way to select one value from a set of values. Android spinners are nothing but the drop down-list seen in other programming languages.

In a default state, a spinner shows its currently selected value.

It provides a easy way to select a value from a list of values. spinner is like a combo box of AWT or swing where we can select a particular item from a list of items.

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

To fill the data in a spinner we need to implement an adapter class. A spinner is mainly used to display only text field so we can implement Array Adapter for that.

```
ArrayAdapter aa = new ArrayAdapter(this,
android.R.layout.simple_spinner_item,bankNames);
aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
//Setting the ArrayAdapter data on the Spinner
spin.setAdapter(aa);
```

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, and spinner.

```
@Override
public void onItemSelected(AdapterView<?> arg0, View arg1, int position,long id) {
Toast.makeText(getApplicationContext(), bankNames[position],
Toast.LENGTH_LONG).show();
}
@Override
public void onNothingSelected(AdapterView<?> arg0) {
// TODO Auto-generated method stub
}
```

Thank You