UNIT III: BUILDING USER INTERFACES

4.1 Text controls
4.2 Button controls
4.3 ProgressBar View
4.4 TimePicker View, DatePicker View
4.5 WebView
4.6 Toast notifications

4.1 Text controls

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.

android:id

This is the ID which uniquely identifies the control.

android:layout_width

This is the layout width which can set width of textView control.

android:layout_height

This is the layout height which can set height of textView control.

android:text

Text to display.

android:textColor

Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

android:textSize

Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).

android:textStyle

Style (bold, italic, bolditalic) for the text. You can use or more of the following values separated by '|'.

- normal - 0
- bold - 1
- italic - 2

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.

```
<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:textSize="50dp"/>
```

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.

android:autoText

If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.

android:id

This is the ID which uniquely identifies the control.

android:layout_width

This is the layout width which can set width of textView control.

android:layout_height

This is the layout height which can set height of textView control.

android:hint

Hint text to display when the text is empty.

android:textColorHint

Color of the hint text. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

```
<EditText

    android:id="@+id/text_id"

    android:layout_width="300dp"

    android:layout_height="200dp"

    android:hint="Enter Name"

    android:textColorHint="@android:color/holo_blue_dark"
```

```
    android:textSize="50dp"

android:textStyle="Bold"/>
```

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”
/>

String[] countries={"India","Australia","West indies","indonesia","Indiana",
                "South Africa","England","Bangladesh","Srilanka","singapore"};

ArrayAdapter<String> adapter=new
ArrayAdapter<String>(this,android.R.layout.simple_dropdown_item_1line,coun
tries);
     AutoCompleteTextView
textView=(AutoCompleteTextView)findViewById(R.id.txtcountries);

     textView.setThreshold(3); //will start working from first character

     textView.setAdapter(adapter);
```

4.2 Button controls

Button

A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.

android:id

This is the ID which uniquely identifies the control.

android:layout_width

This is the layout width which can set width of textView control.

android:layout_height

This is the layout height which can set height of textView control.

android:text

Text to display.

android:textColor

Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

android:textSize

Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).

android:textStyle

Style (bold, italic, bolditalic) for the text. You can use or more of the following values separated by '|'.

- normal - 0
- bold - 1
- italic - 2

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.

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

```
    android:layout_width="300dp"
    android:layout_height="200dp"
    android:text="Save"
    android:textColor="@android:color/holo_blue_dark"
   android:textSize="50dp"/>
```

ImageButton

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.

android:adjustViewBounds

Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.

android:src

This sets a drawable as the content of this ImageView.

android:visibility

This controls the initial visibility of the view.

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/imageButton"
    android:src="@drawable/abc"/>
```

ToggleButton

A ToggleButton displays checked/unchecked states as a button. It is basically an on/off button with a light indicator.

android:textOff

This is the text for the button when it is not checked.

android:textOn

This is the text for the button when it is checked.

android:background

This is a drawable to use as the background.

android:visibility

This controls the initial visibility of the view.

```
<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="On"
    android:id="@+id/toggleButton"
    android:checked="true"/>
```

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.

android:id

This is the ID which uniquely identifies the control.

android:layout_width

This is the layout width which can set width of checkbox control.

android:layout_height

This is the layout height which can set height of checkbox control.

android:text

Text to display.

android:textColor

Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

android:textSize

Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).

```
<CheckBox
    android:id="@+id/checkBox2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Do you like android "
    android:checked="false"/>
```

RadioGroup

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.

android:id

This is the ID which uniquely identifies the control.

android:layout_width

This is the layout width which can set width of checkbox control.

android:layout_height

This is the layout height which can set height of checkbox control.

android:background

This is a drawable to use as the background.

android:visibility

This controls the initial visibility of the view.

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="90dp"
    android:id="@+id/radioGroup"/>
```

RadioButton

A RadioButton has two states: either checked or unchecked.This allows the user to select one option from a set.

android:id

This is the ID which uniquely identifies the control.

android:layout_width

This is the layout width which can set width of checkbox control.

android:layout_height

This is the layout height which can set height of checkbox control.

android:background

This is a drawable to use as the background.

android:visibility

This controls the initial visibility of the view.

android:text

Text to display.

android:textColor

Text color. May be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

android:textSize

Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).

```
<RadioButton
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="JAVA"
    android:id="@+id/radioButton"
    android:textSize="25dp"
    android:textColor="@android:color/holo_red_light"
    android:checked="false"
    android:layout_gravity="center_horizontal" />
```

4.3 ProgressBar

ProgressBar 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.

To add a progress bar to a layout (xml) file, you can use the <ProgressBar> element. By default, a progress bar is a spinning wheel (an indeterminate indicator). To change to a horizontal progress bar, apply the progress bar's horizontal style.

ProgressBar code:

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

Horizontal ProgressBar code:

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

Methods of Progress Bar

1. getMax() – returns the maximum value of progress bar

We can get the maximum value of the progress bar in java class. This method returns a integer value.
Ex. int maxValue=simpleProgressBar.getMax();

2. getProgress() – returns current progress value

We can get the current progress value from a progress bar in java class. This method also returns a integer value.
Ex. int progressValue=simpleProgressBar.getProgress();

Attributes of Progress Bar

1. id: id is an attribute used to uniquely identify a Progress bar.

2. max: max is an attribute used in android to define maximum value of the progress can take. It must be an integer value like 100, 200 etc.

3. progress: progress is an attribute used in android to define the default progress value between 0 and max. It must be an integer value.

4. progressDrawable: progress drawable is an attribute used in Android to set the custom drawable for the progress mode.

5. background: background attribute is used to set the background of a Progress bar. We can set a color or a drawable in the background of a Progress bar.

6. Indeterminate: indeterminate attribute is used in Android to enable the indeterminate mode. 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. In this mode the actual working will not be shown.

Horizontal Progress Bar

Vertical Progress Bar

Indeterminate Progress Bar

4.4 TimePicker View, DatePicker 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.

setHour(Integer hour): *From api level 23 we have to use **setHour(Integer hour)***. In this method there is only one parameter of integer type which is used to set the value for hours.

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

setMinute(Integer minute): *From api level 23 we have to use **setMinute(Integer minute)***. In this method there is only one parameter of integer type which set the value for minutes.

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

*From api level 23 you have to use **getHour().*** This method returns an integer value.

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.

6. 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.

- **paddingRight:** set the padding from the right side of the time picker**.**
- **paddingLeft:** set the padding from the left side of the time picker**.**
- **paddingTop:** set the padding from the top side of the time picker**.**
- **paddingBottom:** set the padding from the bottom side of the time picker**.**
- **Padding:** set the padding from the all side's of the 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"/>
```

4.5 WebView

WebView is a view that display web pages inside your application. You can also specify HTML string and can show it inside your application using WebView.

WebView makes turns your application to a web application.

Activity to access the Internet and load the web pages in a WebView, we must add the internet permissions to our Android Manifest file (Manifest.xml).

```
<uses-permission android:name="android.permission.INTERNET" />

<WebView
android:id="@+id/simpleWebView"
android:layout_width="fill_parent"
android:layout_height="fill_parent" />
```

loadUrl() – Load a web page in our WebView.

loadUrl(String url)

This function is used to load a web page in a web view of our application. In this method we specify the url of the web page that should be loaded in a web view.

```
// initiate a web view
WebView simpleWebView=(WebView) findViewById(R.id.simpleWebView);
// specify the url of the web page in loadUrl function
simpleWebView.loadUrl("https://abhiandroid.com/ui/");
```

loadData() – Load Static Html Data on WebView

This method is used to load the static HTML string in a web view. loadData() function takes html string data, mime-type and encoding param as three parameters.

```
WebView webView = (WebView) findViewById(R.id.simpleWebView);

// static html string data
String customHtml = "<html><body><h1>Hello, AbhiAndroid</h1>" +
    "<h1>Heading 1</h1><h2>Heading 2</h2><h3>Heading 3</h3>" +
    "<p>This is a sample paragraph of static HTML In Web view</p>" +
    "</body></html>";

// load static html data on a web view
webView.loadData(customHtml, "text/html", "UTF-8");
```

Load Remote URL on WebView using WebViewClient:

WebViewClient help us to monitor event in a WebView. You have to Override the shouldOverrideUrlLoading() method. This method allow us to perform our own action when a particular url is selected. Once you are ready with the WebViewClient, you can set the WebViewClient in your WebView using the setWebViewClient() method.

```
// set web view client
simpleWebView.setWebViewClient(new MyWebViewClient());
// string url which you have to load into a web view
String url = "https://www.google.com”
simpleWebView.getSettings().setJavaScriptEnabled(true);
simpleWebView.loadUrl(url); // load the url on the web view
}

// custom web view client class who extends WebViewClient
private class MyWebViewClient extends WebViewClient {
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
view.loadUrl(url); // load the url
return true;
}
```

4.6 Toast notifications

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).

Important Methods Of Toast:

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.

Constants of Toast:

Below is the constants of Toast that are used for setting the duration for the Toast.

1. 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.

2. 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. This method is display the text which we create using makeText() method of Toast.

3. setGravity(int,int,int):

This method is used to set the gravity for the Toast. This method 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. If we use makeText() method and then we want to change the text value for the Toast then we use this method.

```
Toast toast = Toast.makeText(getApplicationContext(), "Simple Toast In Android", Toast.LENGTH_LONG);

// initiate the Toast with context, message and duration for the Toast

toast.setGravity(Gravity.TOP | Gravity.LEFT, 0, 0);     // set gravity for the Toast.

toast.setText("Changed Toast Text"); // set the text for the Toast

toast.show(); // display the Toast


 LayoutInflater li = getLayoutInflater();

     //Getting the View object as defined in the customtoast.xml file

View layout = li.inflate(R.layout.customtoast,(ViewGroup) findViewById(R.id.custom_toast_layout));


     //Creating the Toast object

     Toast toast = new Toast (getApplicationContext());

     toast.setDuration(Toast.LENGTH_SHORT);

     toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);

     toast.setView(layout);//setting the view of custom toast layout

     toast.show();
```

Thank You