SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

UI ELEMENTS AND EVENTS

Course: **Mobile Application Development**

Unit : II – Building Blocks of Mobile Apps - I

Class / Semester: II MCA / III Semester

SESSION OBJECTIVES

- ❑ Understand the significance of UI elements for user applications
- ❑ Implement various UI Elements into layout for Android App Development
- ❑ Apply Events on UI elements using event listeners for user's action

UI ELEMENTS & EVENTS

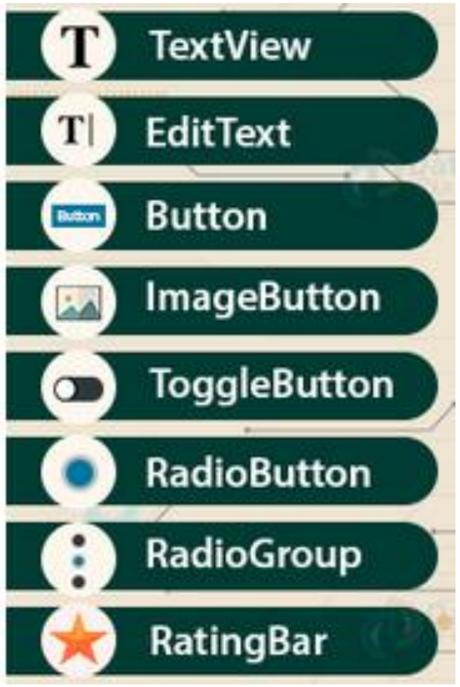
- ❑ Android provides views to handle data and user interaction for a functional UI
- ❑ Common are **TextView** and **Button**
- ❑ User interaction may happen in multiple forms – data entry in an editable view, display of textual/visual data
- ❑ Triggering of an event based on user actions such as click of a button, drag of a list, or multi touch gestures
- ❑ User interaction with a view triggers an event associated with it, Such events handled by a developer to provide required functionalities in views

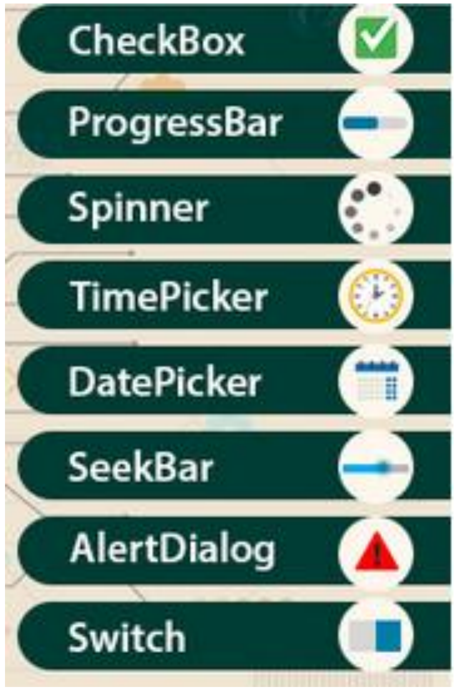
UI ELEMENTS

- UI elements are declared inside layout file and Layout editor provides a mechanism to drag and drop elements into the layout
- Equivalent XML file generated automatically
- UI elements have attributes

Vocabulary	Description
android:id	Used to identify UI element. Like android:id= "@+id/id_name"
Android:layout_width android:layout_height	Width of an element, it may be one of the following • *wrap_content, Match_parent, fill_parent, Hardcoded values*
android:gravity	Sets the alignment of components inside an UI element. • *Center, left, right, center_horizontal, and center_vertical. We can also compound values, e.g., top|left*

UI ELEMENTS


TextView
EditText
Button
ImageButton
ToggleButton
RadioButton
RadioGroup
RatingBar


CheckBox
ProgressBar
Spinner
TimePicker
DatePicker
SeekBar
AlertDialog
Switch

EVENT HANDLING PARADIGM

- ❑ Each view in Android is an *event source* and it generates an *event object* for user action
- ❑ This event object is passed on to an *event listener*, provided this view has registered for that event listener
- ❑ Event listener is a specialized interface designed to listen to the generated event, and respond to it through callback methods, also referred to as *event handlers*

Vocabulary	Description
Event source	The view on which user performs an action. **Ex. Button**
Event object	The object generated by the view. **Ex. Click**
Event object	The object generated by the view. **Ex. OnClickListener**
Event handler	The callback method that responds to the event. **Ex. onClick**

EVENT LISTENERS

Event Listeners	Event listener Description
onClick()	Occurs when user clicks on an item on the screen in touch
onLongClick()	Occurs when a user clicks on an item or screen for more than 1 second.
onFocusChange()	It occurs when a user navigates away from an item that was on focus.
onKey()	It occurs when a user focuses and clicks on an item.
onTouch()	It occurs when a user touches a particular range of an item with gestures or simple touch or tap.
onCreateContextMenu()	This event occurs when a Context Menu is built.
onMenuItemClick()	It occurs when a user clicks or selects an item from a menu

EVENT HANDLERS

Event Handler	Event Handler Description
onKeyUp()	The system invokes this method when a new key event occurs.
onKeyDown()	The system invokes this method when a key down event occurs.
onTrackballEvent()	The system invokes this method when a trackball motion event occurs.
onTouchEvent()	The system invokes this method when some touch event occurs.
onFocusChange()	The system invokes this method when an item gets into focus or loses focus.

EVENT REGISTRATION

- Event Registration is the process in which Event Listeners are registered with Event Handlers
- It can be done in the following three ways
  1. Register event listeners is by directly mentioning them in activity_main.xml
  2. Register event listeners by using Activity class that implements a listener interface
  3. By using an Anonymous class

```
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
Button btn; TextView tView;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
btn = (Button)findViewById(R.id.btnClick);
tView = (TextView)findViewById(R.id.txtResult);
btn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
TextView txtView = (TextView) findViewById(R.id.textView);
txtView.setText("You've Clicked \n The Event has taken place");
txtView.setTextSize(25);
txtView.setGravity(Gravity.CENTER);
txtView.setTextColor(R.color.colorAccent);
}
});
}
```

UI ELEMENTS EXAMPLES

- For example, **Button** element is created with <Button tag in layout file

```
<Button
 android:id="@+id/clickButton"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Click Me"/>
```

- **EditText** is used to accept user input. Declaring this component is as follows

```
<EditText
 android:id="@+id/editText1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
/>
```

- Accessing EditText by using *findViewById(int)* method

```
EditText editText1=(EditText)findViewById(R.id.editText1);
```

- To get the text entered by the user, We use the getText() method

```
String enteredText=editText1.getText().toString();
```

EditText EVENT

- We set listeners for various actions on **EditText** . One of the common events is onFocus

Event Object	Event Listener	Event Handler
FocusChange	OnFocusChangeListener	onFocusChange()

- Implementation method

```
editText1.setOnFocusChangeListener(new OnFocusChangeListener() {

     @Override
      public void onFocusChange(View arg0, boolean arg1) {

        Log.i("Focus changed event", "The focus on the edit text
              has been changed");
       }
});
```

CHECKBOX

- It is created with <CheckBox> tag

```
<CheckBox
 android:id="@+id/checkBox1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="CheckBox"/>
```

- It can be accessed like

```
CheckBox checkBox=(CheckBox)findViewById(R.id.checkBox1);
```

- And common event **CheckedChange** can be implemented by

Event Object	Event Listener	Event Handler
CheckedChange	OnCheckedChangeListener	onCheckedChange()

```
checkBox.setOnCheckedChangeListener(new
OnCheckedChangeListener() {

     @Override
      public void onCheckedChanged(CompoundButton arg0, boolean
                              arg1) {
```

RADIOGROUP ELEMENT

- It is created with <RadioGroup> tag

```
<RadioGroup
 android:id=“@+id/radioGroup1”
 android:layout_width=“wrap_content”
 android:layout_height=“wrap_content”>

 <RadioButton
  android:id=“@+id/radio0”
  android:layout_width=“wrap_content”
  android:layout_height=“wrap_content”
  android:checked=“true”
  android:text=“RadioButton”/>

 <RadioButton
  android:id=“@+id/radio1”
  android:layout_width=“wrap_content”
  android:layout_height=“wrap_content”
  android:text=“RadioButton”/>
 </RadioGroup>
```

- *Event is same as like checkedbox element*

- It can be accessed like

RadioGroup radioGroup=(RadioGroup)findViewById(R.id.radioGroup1);

LISTVIEW

- **ListView** to show a scrollable list of items on the screen wherein each item is selectable
-
```
<ListView
 android:id=“@+id/listView1”
 android:layout_width=“match_parent”
 android:layout_height=“wrap_content”>
</ListView>
```

- It can be populated in two ways: either at compile time through a string array resource or programmatically at runtime

```
<ListView
 android:id=“@+id/listView1”
 android:layout_width=“match_parent”
 android:layout_height=“wrap_content”
 android:entries=“@array/nations”>
</ListView>
```

During Compile time

```
<string-array name=“nations”>
 <item>India</item>
 <item>Malaysia </item>
 <item>Singapore</item>
 <item>Thailand</item>
</string-array>
```

During Runtime

Event Object	Event Listener	Event Handler
ItemClick	OnItemClickListener	onItemClick()

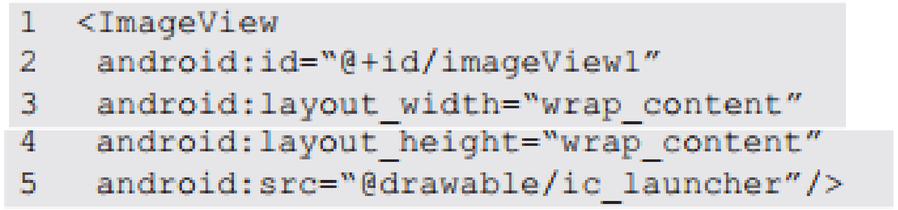
LISTVIEW

- User clicks on an item in the ListView is a common event which will be implemented by

```
listView.setOnItemClickListener(new OnItemClickListener() {
@Override
 public void onItemClick(AdapterView<?> arg0, View arg1, int
                          arg2, long arg3) {
     Log.i("Item Click event on ListView", "The position of
             the item clicked in the ListView is "+arg2);
 }
});
```

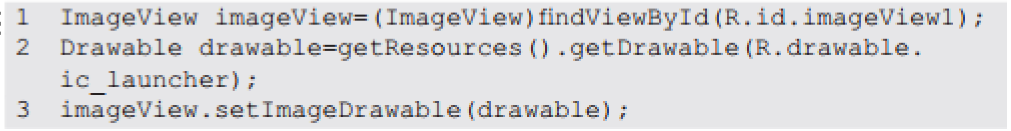
IMAGEVIEW

- *ImageView* is a container for image resources



```
<ImageView
 android:id="@+id/imageView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@drawable/ic_launcher"/>
```

- An Image can also be rendered in the ImageView programmatically using the s

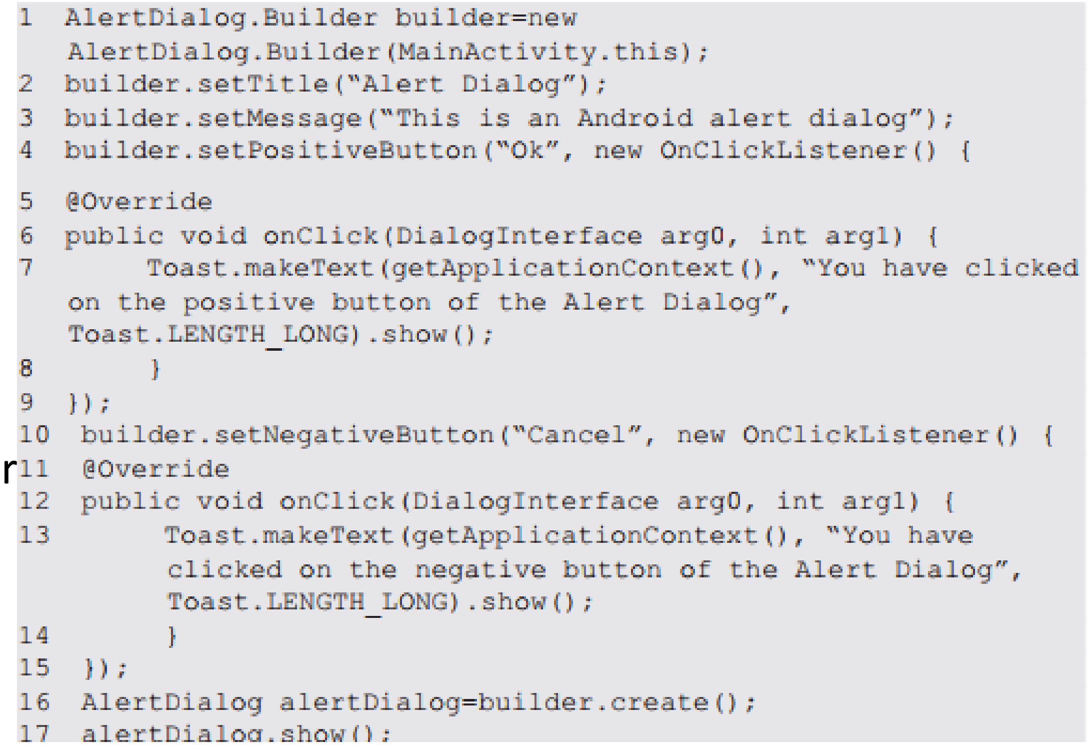


```
ImageView imageView=(ImageView)findViewById(R.id.imageView1);
Drawable drawable=getResources().getDrawable(R.drawable.
ic_launcher);
imageView.setImageDrawable(drawable);
```

-

DIALOG

- Dialog is a modal window displayed on the current Activity
- There are different type of dialog boxes like AlertDialog, ProgressDialog, TimePickerDialog, and DatePickerDialog
- AlertDialog is created using a Builder class



```
AlertDialog.Builder builder=new
 AlertDialog.Builder(MainActivity.this);
builder.setTitle(“Alert Dialog”);
builder.setMessage(“This is an Android alert dialog”);
builder.setPositiveButton(“Ok”, new OnClickListener() {

@Override
public void onClick(DialogInterface arg0, int arg1) {
     Toast.makeText(getApplicationContext(), “You have clicked
 on the positive button of the Alert Dialog”,
 Toast.LENGTH_LONG).show();
     }
});
 builder.setNegativeButton(“Cancel”, new OnClickListener() {
 @Override
 public void onClick(DialogInterface arg0, int arg1) {
      Toast.makeText(getApplicationContext(), “You have
      clicked on the negative button of the Alert Dialog”,
      Toast.LENGTH_LONG).show();
      }
 });
 AlertDialog alertDialog=builder.create();
 alertDialog.show();
```

DIALOG ELEMENT

- **Builder** class is an inner class of AlertDialog that is used to set the layout and behavior of the dialog
- It allows us to configure the title, message, and buttons of the AlertDialog
- Builder provides the setPositiveButton() and setNegativeButton() methods to configure two buttons of dialog boxg the Activity class
- Toast class is a widget used to show unobtrusive messages to the user
- To create a Toast message, we have to use the makeText() method that returns a Toast object. It accepts the following parameters – context, text to be displayed, and duration for which the Toast has to be displayed
- Toast is displayed to the user using the show()method

REFERENCES

- Anubhav Pradhan, Anil V Deshpande, "Composing Mobile Apps using Android", Wiley Edition, 2014
- https://www.tutorialspoint.com/android/android_application_components.htm
- https://www.javatpoint.com/android-core-building-blocks