**Lab Manual for Application for Mobile Devices**

**Lab No. 6**

# **Basic UI Controls**

Objectives

The purpose of this lab is to familiarize with basics of UI Controls in android

**LAB # 06**

**Basic UI Controls**

## Introduction

There are number of UI controls provided by Android that allow you to build the graphical user interface for your app.

|  |  |
| --- | --- |
| **Sr.No.** | **UI Control & Description** |
| 1 | [TextView](https://www.tutorialspoint.com/android/android_textview_control.htm)  This control is used to display text to the user. |
| 2 | [EditText](https://www.tutorialspoint.com/android/android_edittext_control.htm)  EditText is a predefined subclass of TextView that includes rich editing capabilities. |
| 3 | [AutoCompleteTextView](https://www.tutorialspoint.com/android/android_autocompletetextview_control.htm)  The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing. |
| 4 | [Button](https://www.tutorialspoint.com/android/android_button_control.htm)  A push-button that can be pressed, or clicked, by the user to perform an action. |
| 5 | [ImageButton](https://www.tutorialspoint.com/android/android_imagebutton_control.htm)  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. |
| 6 | [CheckBox](https://www.tutorialspoint.com/android/android_checkbox_control.htm)  An on/off switch that can be toggled by the user. You should use check box when presenting users with a group of selectable options that are not mutually exclusive. |
| 7 | [ToggleButton](https://www.tutorialspoint.com/android/android_togglebutton_control.htm)  An on/off button with a light indicator. |
| 8 | [RadioButton](https://www.tutorialspoint.com/android/android_radiobutton_control.htm)  The RadioButton has two states: either checked or unchecked. |
| 9 | [RadioGroup](https://www.tutorialspoint.com/android/android_radiogroup_control.htm)  A RadioGroup is used to group together one or more RadioButtons. |
| 10 | [ProgressBar](https://www.tutorialspoint.com/android/android_progressbar.htm)  The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background. |
| 11 | [Spinner](https://www.tutorialspoint.com/android/android_spinner_control.htm)  A drop-down list that allows users to select one value from a set. |
| 12 | [TimePicker](https://www.tutorialspoint.com/android/android_timepicker_control.htm)  The TimePicker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode. |
| 13 | [DatePicker](https://www.tutorialspoint.com/android/android_datepicker_control.htm)  The DatePicker view enables users to select a date of the day. |

Create UI Controls

Input controls are the interactive components in your app's user interface. Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, check box, zoom buttons, toggle buttons, and many more.

We can create it in two ways:

XML file:

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= "http://schemas.android.com/apk/res/android"

<TextView

//attributes to describe it

/>

</LinearLayout>

Activity file:

In this, we declare it using the setText() method as follows:

setContentView(R.layout.activity\_main);

LinearLayout linearlayout\_name = (LinearLayout)findViewById(R.id.LinearLayout);

TextView textview\_name = new TextView(this);

textveiw\_name.setText(“Hello I am Text View”);

linearLayout.addView(textView);

There are various attributes to describe the TextView some of them are named below:

Android: id – it is a unique id for the control.

Android: width – It displays the exact width of the TextView.

Android: height – It displays the exact height of the TextView.

Android:textColor – It set the color of the text.

Android: gravity – It is to align the TextView.

There are some more attributes the above are the major ones.

2. EditText

EditText is a user interface control that allows the users to enter some text.

We can create it in two ways:

XML file:

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<EditText

//attributes

>

</LinearLayout>

Activity file:

In activity, we declare it using the getText() method as follows:

setContentView(R.layout.activity\_main);

LinearLayout linearlayout\_name = (LinearLayout) findViewById (R.id.LinearLayout) ;

EditText edittext\_name = **new** EditText(**this**);

edittext\_name.setHint(“Hello I am EditText”);

linearLayout.addView(edittext\_name);

### **3. Button**

This is a UI that is used to perform some action as soon as the user clicks on it.

We can create it in two ways:

**XML file:**

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<Button

//attributes

/>

</LinearLayout>

**Activity file:**

In activity, we declare it programmatically as below:

setContentView(R.layout.activity\_main);

LinearLayout linearlayout\_name = (LinearLayout)findViewById(R.id.LinearLayout);

Button btn\_name = **new** Button(**this**);

btn\_name.setText(“Hello I am Button”);

linearLayout.addView(btn\_name);

### **4. ImageButton**

It is the same as a Button but it’s used to display an image on the button to perform an Action. In this, we need to give the source of the image so that the system can load it.

We can create it in two ways:

**XML file:**

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<ImageButton

//other attributes...

android:src= “@drawable/add\_icon”/>

</LinearLayout>

**Activity file:**

In the activity file, we declare it programmatically as below:

setContentView(R.layout.activity\_main);

LinearLayout linearlayout\_name = (LinearLayout)findViewById(R.id.LinearLayout);

ImageButton btn\_name = **new** Button(**this**);

btn\_name.setImageResource(R.drawable.add\_icon);

linearLayout.addView(btn\_name);

### **5. ToggleButton**

The toggle button displays the ON/OFF states of a button with a light indicator.

We can create it in two ways:

**XML file:**

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<ToggleButton

//attributes

android:checked="true"

android:textOff="OFF"

android:textOn="ON"/>

</LinearLayout>

**Activity file:**

In the activity file we declare it programmatically as below:

setContentView(R.layout.activity\_main);

LinearLayout linearlayout\_name = (LinearLayout)findViewById(R.id.LinearLayout);

ToggleButton tb\_name = **new** ToggleButton(**this**);

tb\_name.setTextOff("OFF");

tb\_name.setTextOn("ON");

tb\_name.setChecked(**true**);

linearLayout.addView(btn\_name);

### **6. RadioButton**

Radio button in Android is the one that has only two possible states, that are either checked or unchecked. Initially, it is in the unchecked state, once it’s checked it can’t be unchecked.

We can create it in two ways:

**XML file:**

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<RadioButton

android:text="Radio button"

android:checked="true"/>

</LinearLayout>

**Activity file:**

In the activity file, we declare it programmatically as below:

setContentView(R.layout.activity\_main);

LinearLayout linearlayout\_name = (LinearLayout)findViewById(R.id.LinearLayout);

RadioButton btn\_name = **new** RadioButton(**this**);

btn\_name.setText("Hello");

btn\_name.setChecked(**true**);

linearLayout.addView(btn\_name);

### **7. RadioGroup**

It’s a group of Radio buttons that are alike. In this, only one of all the buttons can be chosen.

We can create it in two ways:

**XML file:**

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<RadioGroup android:orientation="vertical">

<RadioButton android:text="Radio Button 1"/>

<RadioButton android:text="Radio Button 2"/>

<RadioButton android:text="Radio Button 3"/>

</RadioGroup>

</LinearLayout>

**Activity file:**

In the activity file, we declare it programmatically as below:

RadioButton rdb1,rdb2,rdb3;

@override

protected **void** onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

rdb1 = (RadioButton)findViewById(R.id.rdb\_1);

rdb2 = (RadioButton)findViewById(R.id.rdb\_2);

rdb3 = (RadioButton)findViewById(R.id.rdb\_3);

Button btn = (Button)findViewById(R.id.getBtn);

}

Above is the example to declare it, to make it functional, we would override and define the methods that are required.

### **8. CheckBox**

A CheckBox is the UI control that has two states that are either checked or unchecked. If we have a group of CheckBox, we can select as many as we want, unlike RadioGroup.

We can create it in two ways:

**XML file:**

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android= ”http://schemas.android.com/apk/res/android”>

<CheckBox

android:checked="true"

android:text="CheckBox"

// other attributes

/>

</LinearLayout>

**Activity file:**

In activity, we declare it programmatically as below:

setContentView(R.layout.activity\_main);

LinearLayout linearlayout\_name = (LinearLayout)findViewById(R.id.LinearLayout);

CheckBox cb\_name = **new** CheckBox(**this**);

cb\_name.setText("DataFlair");

cb\_name.setChecked(**true**);

layout.addView(cb\_name);

### **9. ProgressBar**

In Android, we have a progress bar that shows the progress of some action that is happening like pasting a file to some location. A progress bar can be in two modes:

* **Determinate Mode:**

In this, the progress is shown with the percent of action completed. Also, the time to be taken is already determined.

* **Indeterminate Mode:**

In this, there is no idea of when the task would be completed, therefore, it functions continuously.

We can create it in two ways:

**XML file:**

For this, we declare it in the layout tag as follows:

<Linear Layout xmls:android=”http://schemas.android.com/apk/res/android”>

<ProgressBar

// attributes, here we define the speed, layout, id, etc.

/>

</LinearLayout>

**Activity file:**

In activity, we declare it programmatically as below:

setContentView(R.layout.activity\_main);

pgsBar = (ProgressBar)findViewById(R.id.pBar);

txtView = (TextView)findViewById(R.id.tView);

Button btn = (Button)findViewById(R.id.btnShow);

Here we need to set the sleep time and override the onClick() and onCreate() methods

The above were some of the very important UI controls, now we will also read about some more UI controls :

### **1. Spinner**

The Spinner in Android is a User Interface that is used to select a particular option from a list of given options. Spinner in Android is the same as dropdown in [HTML](https://html.com/). It provides us with a faster way to select an option of our choice. When we click on the down arrow, it shows a list of values from which we can select one. By default, the first value would be shown as the currently selected Value.

### **2. TimePicker**

Time picker is a UI component that works as an intermediate to select a time of the day. The time chosen by it is shown either in 24 hrs format or in 12hrs format using AM and PM.

It gives a virtual Clock/watch to select it. This virtual clock makes it easy to choose the time.

### **3. DatePicker**

Like we have time picker, we have a date picker as UI control too. In this, the System shows a virtual calendar to the users to choose the day.

This enables the user to choose a particular date using either a calendar or a dropdown. These both are made to make it easier for the user to pick up a date and a time.

### **4. SeekBar**

In Android, Seekbar is an extended Progress bar. A seekbar comes with a pointer that is draggable throughout the bar either in left or right. This pointer helps to set the progress as well. This helps the user to choose a particular range of values.

### **5. RatingBar**

A rating bar in Android is an extended version of a seekbar. It is used to give the rating by touching it. In the rating bar, a user can rate at a scale of 5 with a difference of 0.5.

Its rating is in Stars. The user needs to tap/click the stars.

### **6. AlertDialog**

Alert Dialog Box is a UI that gives the users an Alert or Warning of something. It appears on the screen in a small window. Once it comes, the user needs to decide or choose an option that it shows.

For example, when you enter the wrong password for email id.

### **7. Switch**

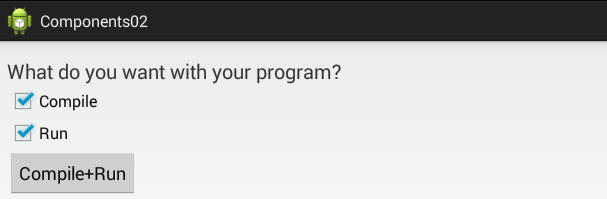
In Android, a switch is a two-state UI element that holds either ON or OFF state. ON generally means Yes and OFF means No. By default, a switch is in the OFF state. A user can change its state many times.

### **8. AutoCompleteTextView**

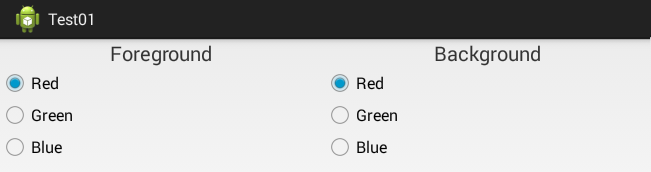
AutoCompleteTextView is an extension of EditText. In this UI element, the user is provided with a few suggestions of some values/texts. The value can be chosen by the user while filling AutoCompleteTextView.

**Tasks**

1. Taking composite choice from the user using checkboxes.



1. ColorChanger using RadioButton



1. Show the choices in TextView

