**Lab Manual for Application for Mobile Devices**

**Lab No. 4**

# **Basic Event Handling**

Objectives

The purpose of this lab is to familiarize with basics of event handling

**LAB # 04**

**Basic Event Handling**

## **Introduction**

In android application we have two important files *Layout* files or .xml file and *Activity* or .java files. Xml files are used to design a UI for our app whereas .java file is used to write functionality or logics. By default, we have MainActivity.java files under the Java folder in project directory and activity\_main.xml file under layout folder of res directory. To clearly understand open android studio, create new project and apply following Examples:

### **Buttons**

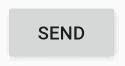
A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.



Depending on whether you want a button with text, an icon, or both, you can create the button in your layout in three ways:

1. With text, using the Button class:

<**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="send"** />



1. With an icon, using the ImageButton class:

<**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:src="@drawable/icon"  
 android:contentDescription="@string/icon"** />



1. With text and an icon, using the Button class with the android:drawableRight attribute:

<**Button  
 android:layout\_width="116dp"  
 android:layout\_height="wrap\_content"  
 android:adjustViewBounds="true"  
 android:backgroundTint="#ffffff"  
 android:drawableRight="@drawable/next\_icon"  
 android:text="Next"** />



### **Responding to Click Events**

When the user clicks a button, the Button object receives an on-click event.

To define the click event handler for a button, add the android:onClick attribute to the <Button> element in your XML layout. The value for this attribute must be the name of the method you want to call in response to a click event. The Activity hosting the layout must then implement the corresponding method.

For example, here's a layout with a button using android:onClick:

**public void** sendMessage(View view) {  
 *// Do something in response to button click*}

The method you declare in the [android:onClick](https://developer.android.com/reference/android/R.attr.html" \l "onClick) attribute must have a signature exactly as shown above. Specifically, the method must:

* Be public
* Return void
* Define a [View](https://developer.android.com/reference/android/view/View.html) as its only parameter (this will be the [View](https://developer.android.com/reference/android/view/View.html) that was clicked)

**EXAMPLE 01:**

**.xml Code:**

<**Button  
 android:id="@+id/button\_send"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="send"  
 android:onClick="sendMessage"**/>

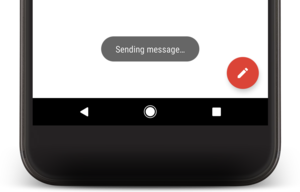
**.java Code:**

**package** com.example.basiceventhandling;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }   
   
 **public void** sendMessage(View view) {  
 *// Do something in response to button click* }  
   
}

**Apply toast on button click event:**

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

For example, clicking Send on an email triggers a "Sending message..." toast, as shown in the following screen capture:



First, instantiate a Toast object with one of the makeText() methods. This method takes three parameters: the application Context, the text message, and the duration for the toast. It returns a properly initialized Toast object. You can display the toast notification with show(), as shown in the following example:

**.Java File:**

**import** android.widget.Toast;

**public void** sendMessage(View view) {

Toast T = Toast.*makeText*(**this**,**"You just ToastButton"** , Toast.***LENGTH\_SHORT***);

T.show();  
}

**Positioning your Toast**

A standard toast notification appears near the bottom of the screen, centered horizontally. You can change this position with the setGravity(int, int, int) method. This accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

For example, if you decide that the toast should appear in the top-left corner, you can set the gravity like this:

T.setGravity(0,0,0);

### **Using an OnClickListener:**

You can also declare the click event handler programmatically rather than in an XML layout. This might be necessary if you instantiate the Button at runtime or you need to declare the click behavior in a Fragment subclass.

To declare the event handler programmatically, create an View.OnClickListener object and assign it to the button by calling setOnClickListener(View.OnClickListener). For example:

**.xml Code:**

<**Button  
 android:id="@+id/button\_text1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@color/colorAccent"  
 android:text="My Button 1"  
 android:textColor="#ffffff"  
 android:textStyle="bold"  
 android:textSize="15sp"**

/>

**.java Code:**

**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
**import** android.os.Bundle;  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 **private** Button **getButtonText**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **getButtonText** = (Button) findViewById(R.id.***button\_text1***);  
  
 **getButtonText**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Toast.*makeText*(MainActivity.**this**, **"Result: "** + **getButtonText**.getText().toString(), Toast.***LENGTH\_LONG***).show();  
 }  
 });  
  
 }  
  
}

## **Time Boxing**

|  |  |  |
| --- | --- | --- |
| Activity Name | Activity Time | Total Time |
| Login Systems + Setting up android studio Environment | 3 mints + 5 mints | 8 mints |
| Walk through Theory & Tasks | 60 mints | 60 mints |
| Implement Tasks | 80 mints | 80 mints |
| Evaluation Time | 30 mints | 30 mints |
|  | Total Duration | 178 mints |

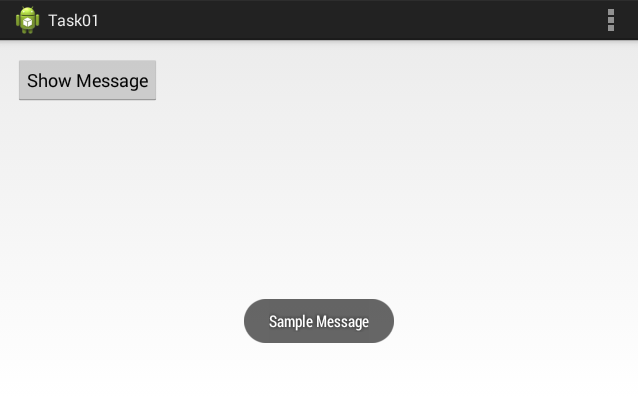
## **Objectives/Outcomes**

The purpose of this lab is to familiarize with basic of Android studio IDE

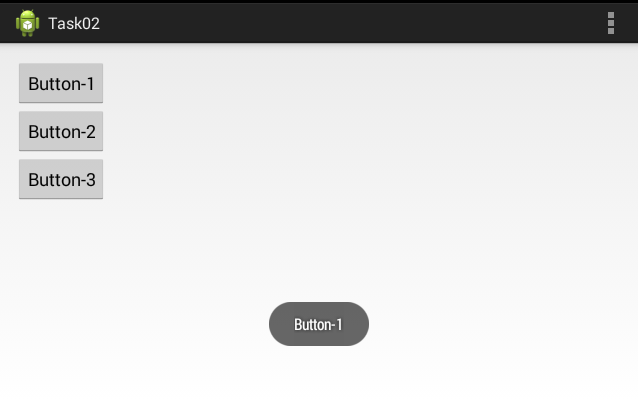
• Understanding Linear Layout

## **Lab Tasks/Practical Work**

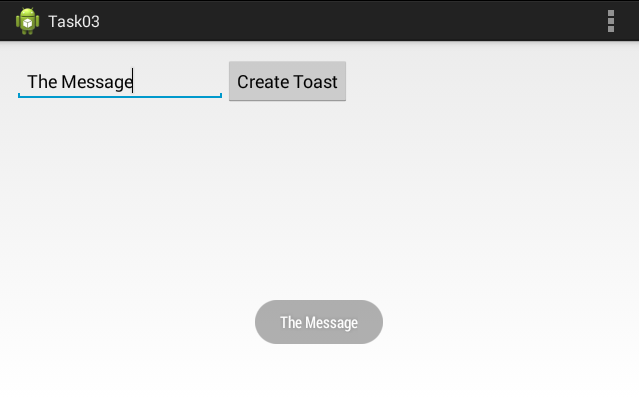
1. Create an application which can display a Toast message by pressing a button.



1. Create an application having three buttons. Bind those buttons with the same callback method. On pressing any button, identify which button was pressed.



1. Create an application which takes a string message from user and create a toast of it on pressing button. Your GUI must look as follows:



1. Create a calculator application for two integers. The output must look as follows

