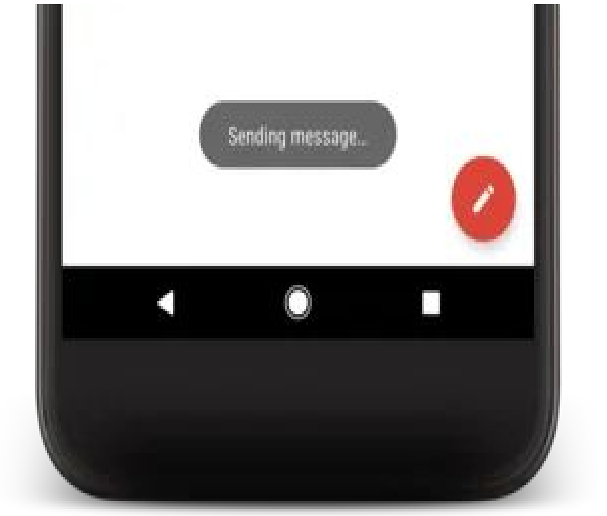
Toast in Android

•A toast provides simple feedback about an operation in a small popup.
•It is used to display short and temporary messages in android apps.
• 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.



Features of Toast

- It is an Android widget that is used to show a message for a short duration of time.
- It disappears after a short time.
- It doesn't block the Activity or Fragment when it runs.
- It can be used to give feedback to the user regarding any operations, like form submission etc.

A Toast can be created using the android.widget.Toast class, which extends the java.lang.Object class.

Methods of Toast class

public static Toast makeText(Context context, CharSequence text, int duration)

> This method makes the Toast widget with the specified text and for the specified duration.

public void show() : This method shows the Toast.

public void setMargin(float horizontal, float vertical) : This method can be used to set horizontal and vertical margin

Constants of Toast class

public static final int LENGTH_LONG : This can be used to display the Toast for a longer duration.

***public static final int LENGTH_SHORT* :** This can be used to display the Toast for a short duration.

The constant **LENGTH_LONG** sets a display duration of 3.5 sec while the constant **LENGTH_SHORT** sets a display duration of 2 sec for the Toast.

Steps

1. Make an object of the Toast class : ***Toast t = new Toast(this);***
2. Call ***makeText(Context c, CharSequence text, int duration)*** method which needs three parameters.
    - ***Context c:*** It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc. We can get this Context object by using the method getApplicationContext()
    - ***CharSequence text:*** This is the message which is shown in the toast. It can be any text.
    - **int duration:** This is the time duration for which you want your message to appear on the screen. There are two values: Toast.LENGTH_SHORT and Toast.LENGTH_LONG

setOnClickListener:-

One of the most usable methods in android is **setOnClickListener** method which helps us to link a listener with certain attributes.

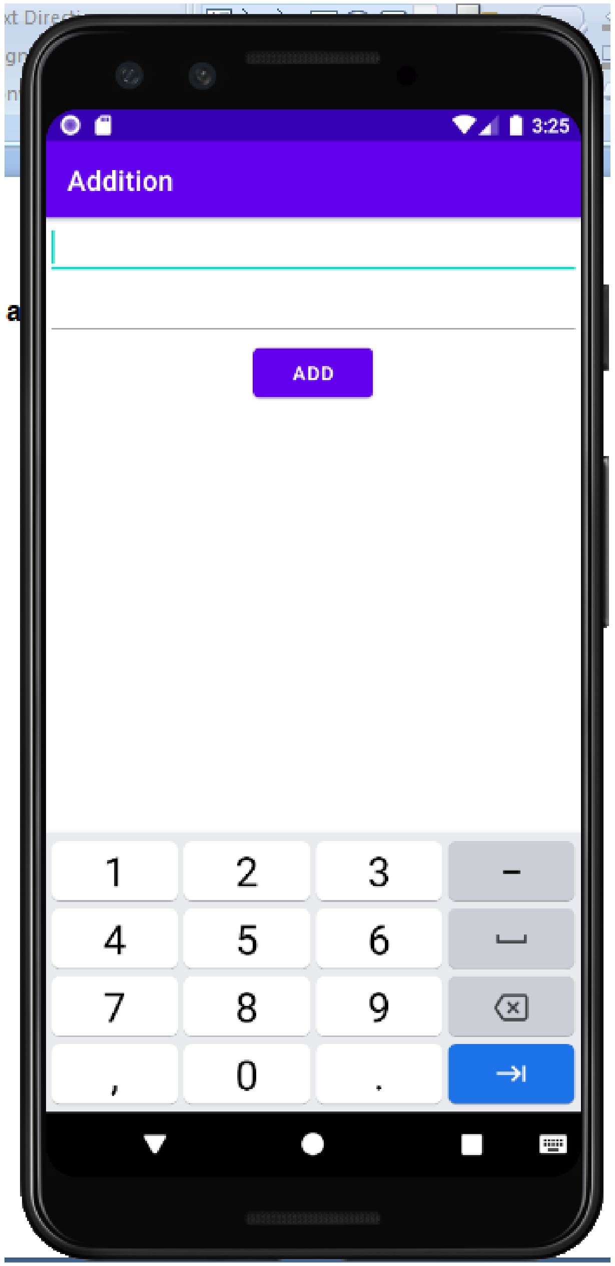
setOnClickListener is a method in Android basically used with buttons, image buttons etc. You can initiate this method easily like,

public void setOnClickListener(View.OnClickListner)

While invoking this method a callback function will run. One can also create a class for more than one listener, so this can lead you to code reusability.

After making the class you can implement **android.view.View.OnClickListener{}** method which gives you an override method inherited from super class called **onClick(View v){}** in which you can easily implement your code.

Add two numbers and display result in Toast Message



Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/Num1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="number" />

    <EditText
        android:id="@+id/Num2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="number" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Add" />

</LinearLayout>
```

MainActivity.java

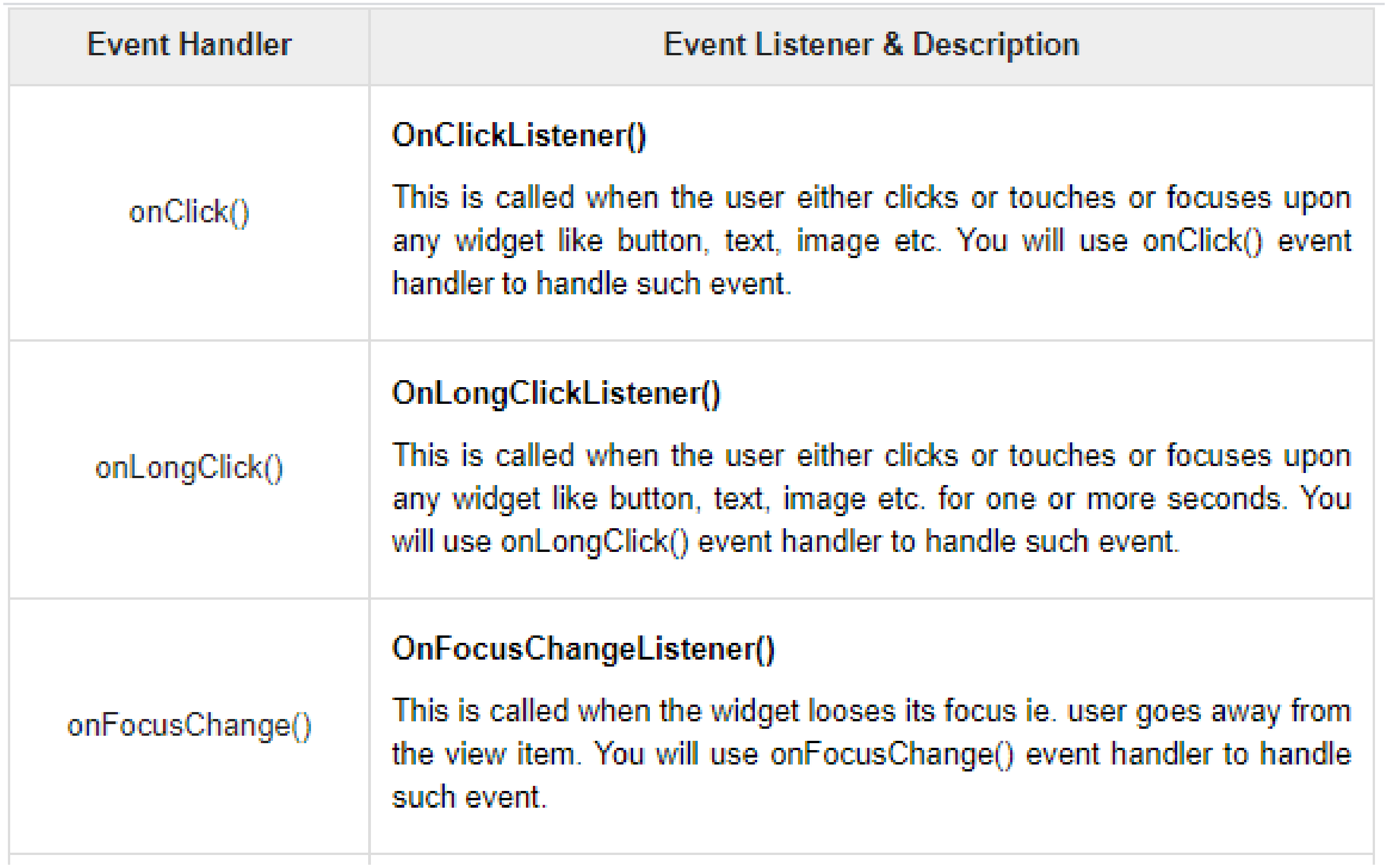
```java
package com.example.addition;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
   // Create instances of controls
    EditText Number1;
    EditText Number2;
    Button Add;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
// Capture our edit texts from layout
        Number1=findViewById(R.id.Num1);
        Number2=findViewById(R.id.Num2);
// Capture our button from layout
        Add=findViewById(R.id.button);
// Register the onClick listener with the implementation
        Add.setOnClickListener(new View.OnClickListener() { // Create an anonymous
implementation of OnClickListener
            @Override
            public void onClick(View v) {
                double num1=Double.parseDouble(Number1.getText().toString());
                double num2=Double.parseDouble(Number2.getText().toString());
                double sum=num1+num2;
Toast.makeText(getApplicationContext(),"Addition:"+Double.toString(sum),
Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

Input Events

- On Android, there's more than one way to intercept the events from a user's interaction with your application.
- When considering events within your user interface, the approach is to capture the events from the specific View object that the user interacts with.
- The View class provides the means to do so.



Event Handler	Event Listener & Description
onClick()	**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.
onLongClick()	**OnLongClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.
onFocusChange()	**OnFocusChangeListener()** This is called when the widget looses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.

onKey()	**OnFocusChangeListener()** This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.
onTouch()	**OnTouchListener()** This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.
onMenuItemClick()	**OnMenuItemClickListener()** This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.
onCreateContextMenu()	**onCreateContextMenuItemListener()** This is called when the context menu is being built(as the result of a sustained "long click)

- **Event Listeners**
  - An event listener is an interface in the View class that contains a single callback method.
  - These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
- **Event Handlers**
  - When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Event Listeners Registration

- Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

Ways to register your event listener for any event

- Using an Anonymous Inner Class
- Activity class implements the Listener interface.
- Using Layout file activity_main.xml to specify event handler directly.