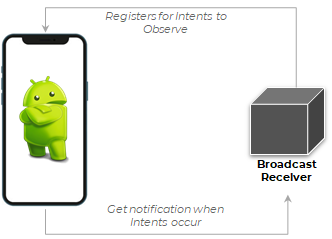
**Lab 25: Android Broadcast Receiver using Kotlin**

# **Introduction**

Simple broadcast messages from other apps or the system itself are responded to by broadcast receivers. Sometimes these signals are referred to as occurrences or intentions. For instance, applications may broadcast to inform other applications that certain data has been downloaded to the device and is ready for use. A broadcast receiver will pounce on this information and take the necessary action.

There are following two important steps to make Broadcast Receiver works for the system broadcasted intents −

* Creating the Broadcast Receiver.
* Registering Broadcast Receiver

****

**Let’s get Started:**

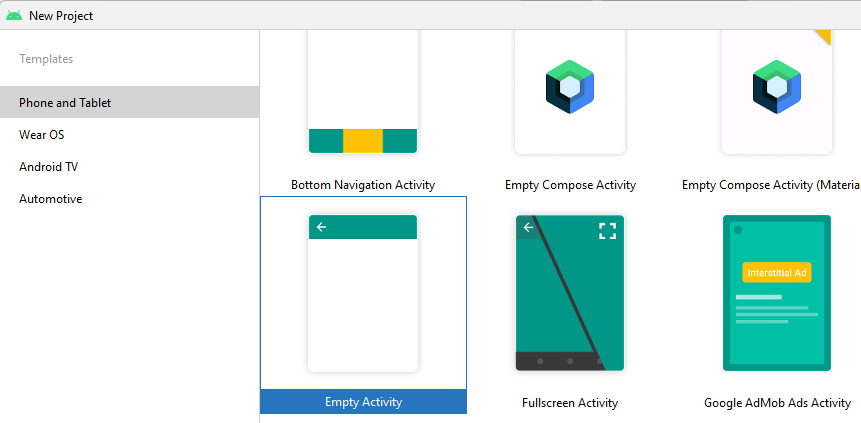
In this experiment we will develop an Android App to demonstrate the use of Android Broadcast Receiver.

**Step 1: Create a New Project in Android Studio as shown below**

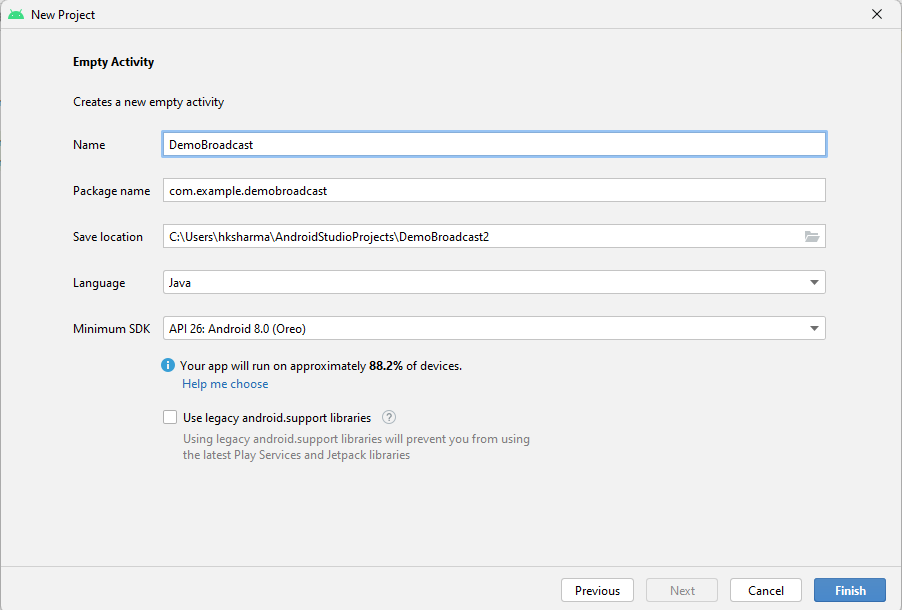
Graphical user interface, text, application

Description automatically generated

**Step 2: Select Empty Activity as shown below**



**Step 3: Provide a Project Name as shown below**

****

**Step 4: Update MainActivity.kt as per the code given below**

**package** com.example.broadcastkotlin  
**import** android.content.Intent  
**import** android.content.IntentFilter  
**import** android.os.Bundle  
**import** androidx.appcompat.app.AppCompatActivity  
  
**class** MainActivity : AppCompatActivity() {  
  
 *// register the receiver in the main activity in order  
 // to receive updates of broadcasts events if they occur* **lateinit var receiver**: AirplaneModeChangeReceiver  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 **receiver** = AirplaneModeChangeReceiver()  
  
 *// Intent Filter is useful to determine which apps wants to receive  
 // which intents,since here we want to respond to change of  
 // airplane mode* IntentFilter(Intent.*ACTION\_AIRPLANE\_MODE\_CHANGED*).*also* **{** *// registering the receiver  
 // it parameter which is passed in registerReceiver() function  
 // is the intent filter that we have just created* registerReceiver(**receiver**, **it**)  
 **}** }  
  
 *// since AirplaneModeChangeReceiver class holds a instance of Context  
 // and that context is actually the activity context in which  
 // the receiver has been created* **override fun** onStop() {  
 **super**.onStop()  
 unregisterReceiver(**receiver**)  
 }  
}

**Step 5: activity\_main.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
</**androidx.constraintlayout.widget.ConstraintLayout**>

**Step 6: AirplaneModeChangeReceiver.kt**

**package** com.example.broadcastkotlin  
**import** android.content.BroadcastReceiver  
**import** android.content.Context  
**import** android.content.Intent  
**import** android.widget.Toast  
  
*// AirplaneModeChangeReceiver class extending BroadcastReceiver class***class** AirplaneModeChangeReceiver : BroadcastReceiver() {  
  
 *// this function will be executed when the user changes his  
 // airplane mode* **override fun** onReceive(context: Context?, intent: Intent?) {  
  
 *// intent contains the information about the broadcast  
 // in our case broadcast is change of airplane mode  
  
 // if getBooleanExtra contains null value,it will directly return back* **val** isAirplaneModeEnabled = intent?.getBooleanExtra(**"state"**, **false**) ?: **return** *// checking whether airplane mode is enabled or not* **if** (isAirplaneModeEnabled) {  
 *// showing the toast message if airplane mode is enabled* Toast.makeText(context, **"Airplane Mode Enabled"**, Toast.*LENGTH\_LONG*).show()  
 } **else** {  
 *// showing the toast message if airplane mode is disabled* Toast.makeText(context, **"Airplane Mode Disabled"**, Toast.*LENGTH\_LONG*).show()  
 }  
 }  
}

Output:

****

**Voila!!** We have successfully completed this lab.