**No: 1 Anusha Ananthan**

**Date: 11/05/2021 MCA S4**

**Rollno : 10**

# AIM

Illustrate Basic Building blocks – Activities, Services, Broadcast Receivers and Content providers with suitable examples

## Basic components of an Android application

## The basic components of an Android application are:

* + - 1. **Activities**
      2. Services

### 1.Activities

An activity is a class that is considered as an entry point for users that represents a single screen. A messenger application might have an activity that shows a new notification, another activity which reads messages and another which composes a new message.

Each activity is independent of one another.

To implement an activity, extend the Activity class in your subclass:

public class MainActivity extends Activity {

//code

}

## Android Activity Lifecycle

An activity can have four states, which are :

1. Running
2. Paused
3. Resumed
4. Stopped

The above are the four states that Android activity can achieve during its whole lifecycle.

#### 1. Running State

An activity is in the **running** state if it’s shown in the foreground of the users’ screen. Activity is in the running state when the user is interacting with it.

#### 2. Paused State

When an activity is not in the focus but is still alive for the user, it’s in a **paused** state. The activity comes in this state when some other activity comes in with a higher position in the window.

#### 3. Resumed State

It is when an activity goes from the paused state to the foreground that is an **active** state.

#### 4. Stopped State

When an activity is no longer in the activity stack and **not visible** to the users.

### Android Activity Methods

Android activities go through four states during their entire lifecycle. These activities have callback methods() to describe each activity in each of the four stages. These methods need to be overridden by the implementing subclass. In Android, we have the following 7 callback methods that activity uses to go through the four states:

1. onCreate()
2. onStart()
3. onPause()
4. onRestart()
5. onResume()
6. onStop()
7. onDestroy()

We’ll understand these in the following:

#### 1. onCreate()

The Android oncreate() method is called at the very start when an activity is created. An activity is created as soon as an application is opened. This method is used in order to create an Activity.

**Syntax:**

@Override protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

...

}

#### 2. onStart()

The Android onstart() method is invoked as soon as the activity becomes visible to the users. This method is to start an activity. The activity comes on the forescreen of the users when this method is invoked.

**Syntax:**

@Override protected void onStart()

{

super.onStart();

...

}

#### 3. onPause()

The Android onPause() method is invoked when the activity doesn’t receive any user input and goes on hold. In the pause state, the activity is partially visible to the user. This is done when the user presses the back or home buttons. Once an activity is in the pause state, it can be followed by either **onResume()** or**onStopped()** callback method.

**Syntax:**

@Override protected void onPause()

{

super.onPause();

...

}

#### 4. onRestart()

The Android onRestart() method is invoked when activity is about to start from the stop state. This method is to restart an activity that had been active some time back. When an activity restarts, it starts working from where it was paused.

**Syntax:**

@Override protected void onRestart()

{

super.onRestart();

...

}

#### 5. onResume()

The Android onResume() method is invoked when the user starts interacting with the user. This callback method is followed by **onPause()**. Most of the functionalities of an application are implemented using **onResume()**.

**Syntax:**

@Override protected void onResume()

{

super.onResume();

...

}

#### 6. onStop()

The Android onStop() method is invoked when the activity is no longer visible to the user. The reason for this state is either activity is getting destroyed or another existing activity comes back to **resume** state.

**Syntax:**

@Override protected void onStop()

{

super.onStop();

...

}

#### 7. onDestroy()

The Android onDestroy() is the method that is called when an activity finishes and the user stops using it. It is the final callback method received by activity, as after this it is destroyed.

**Syntax:**

@Override protected void onDestroy()

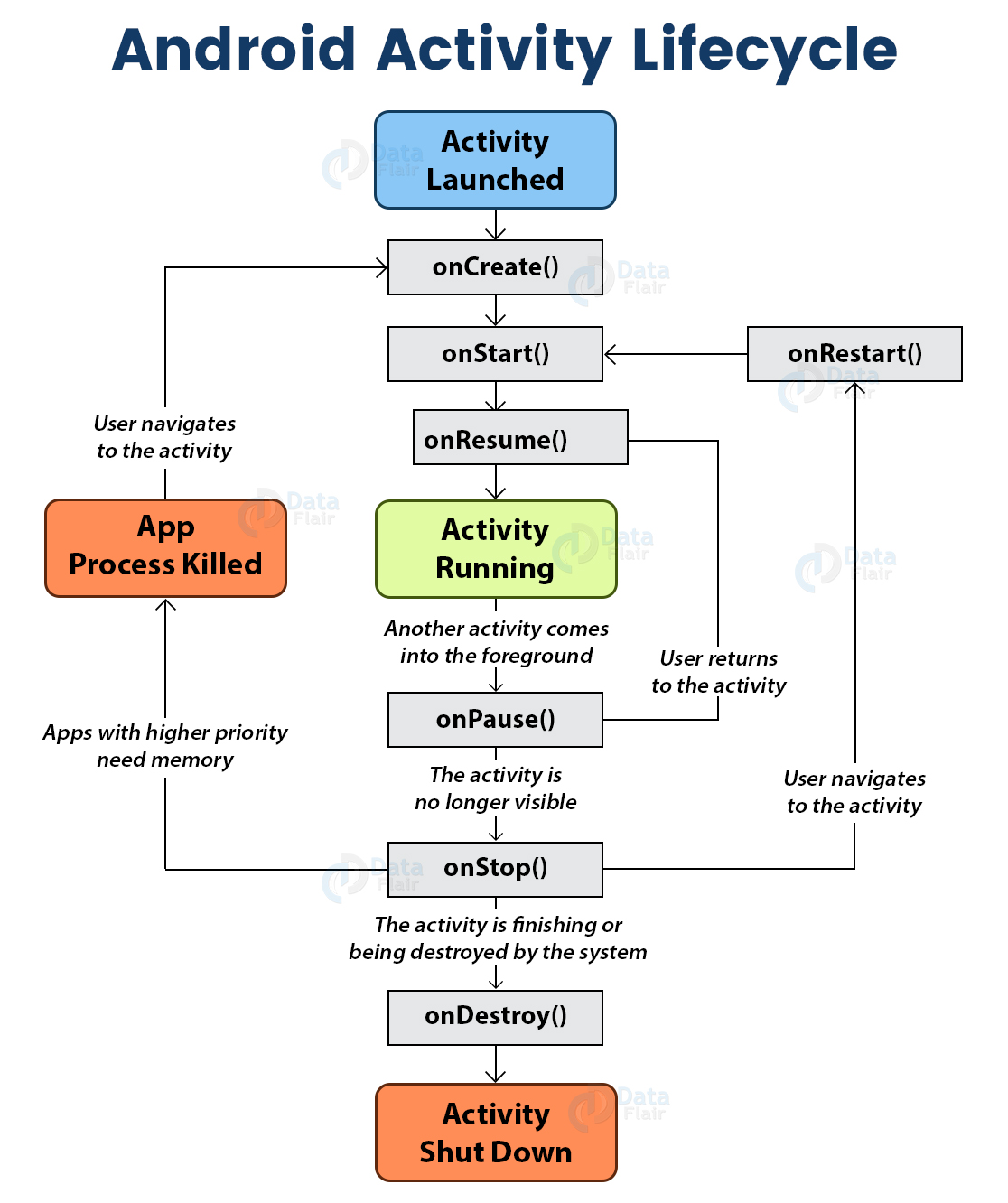
{

super.onDestroy();

...

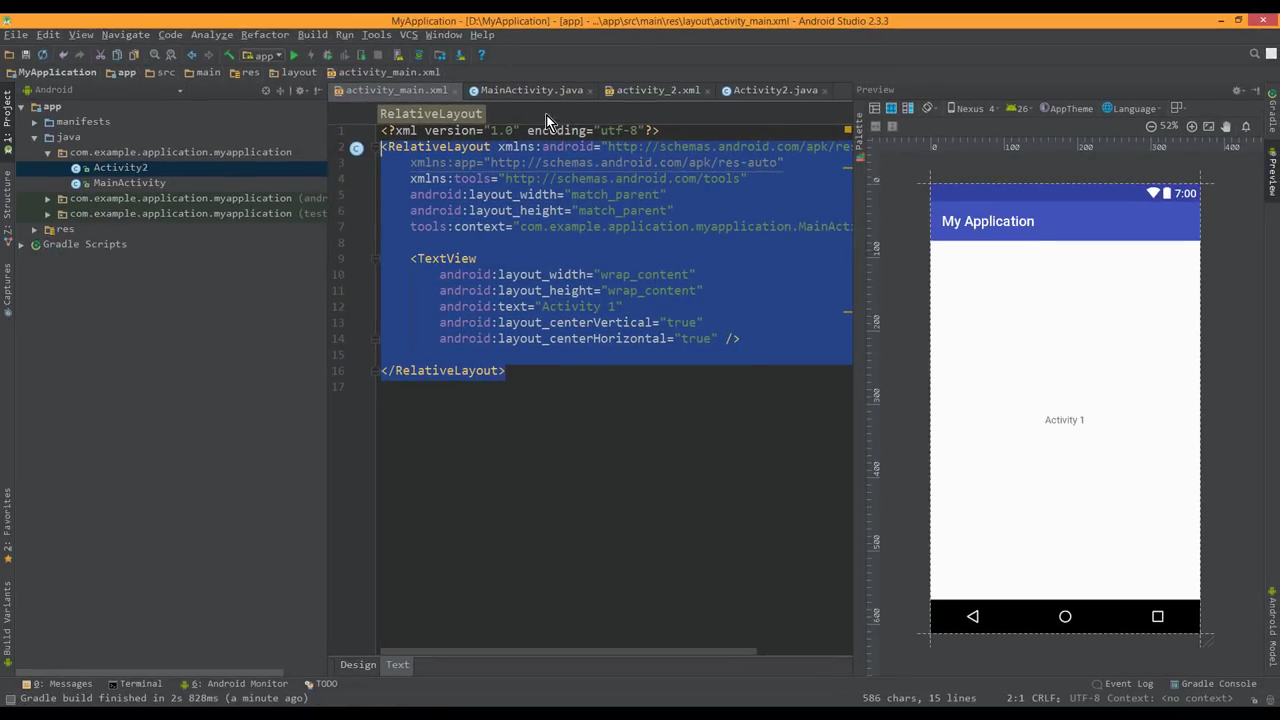
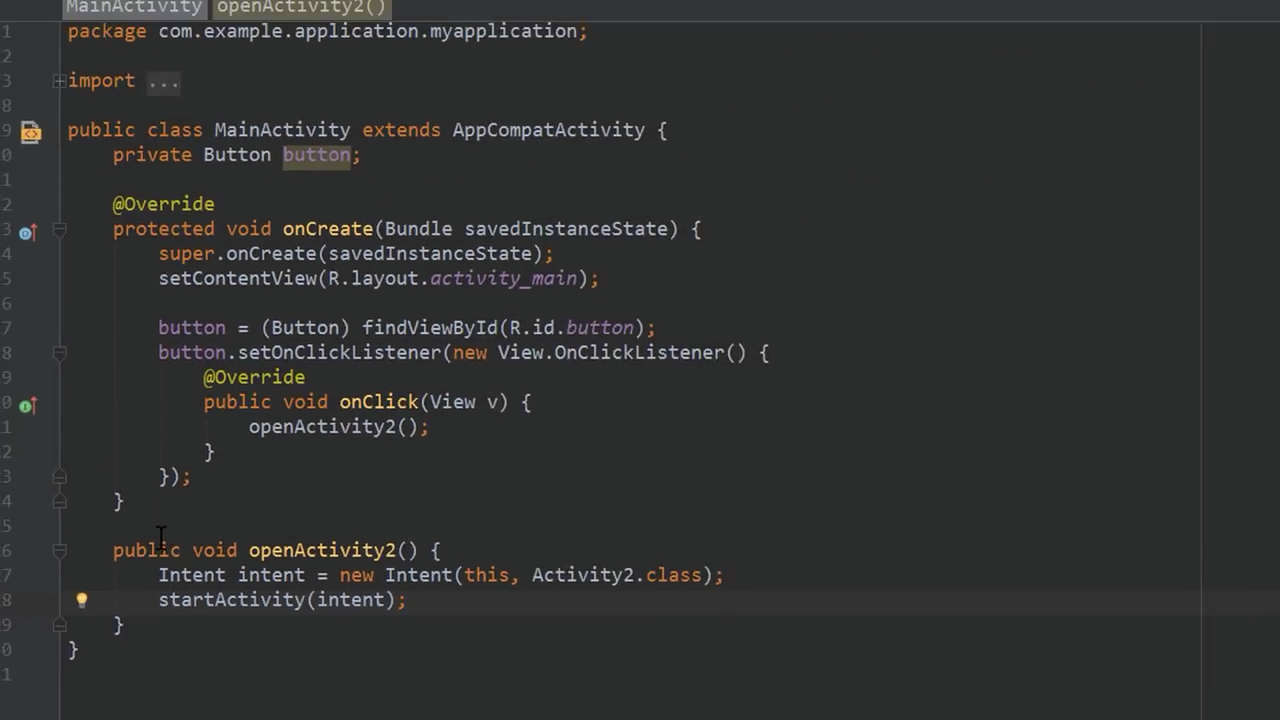
}

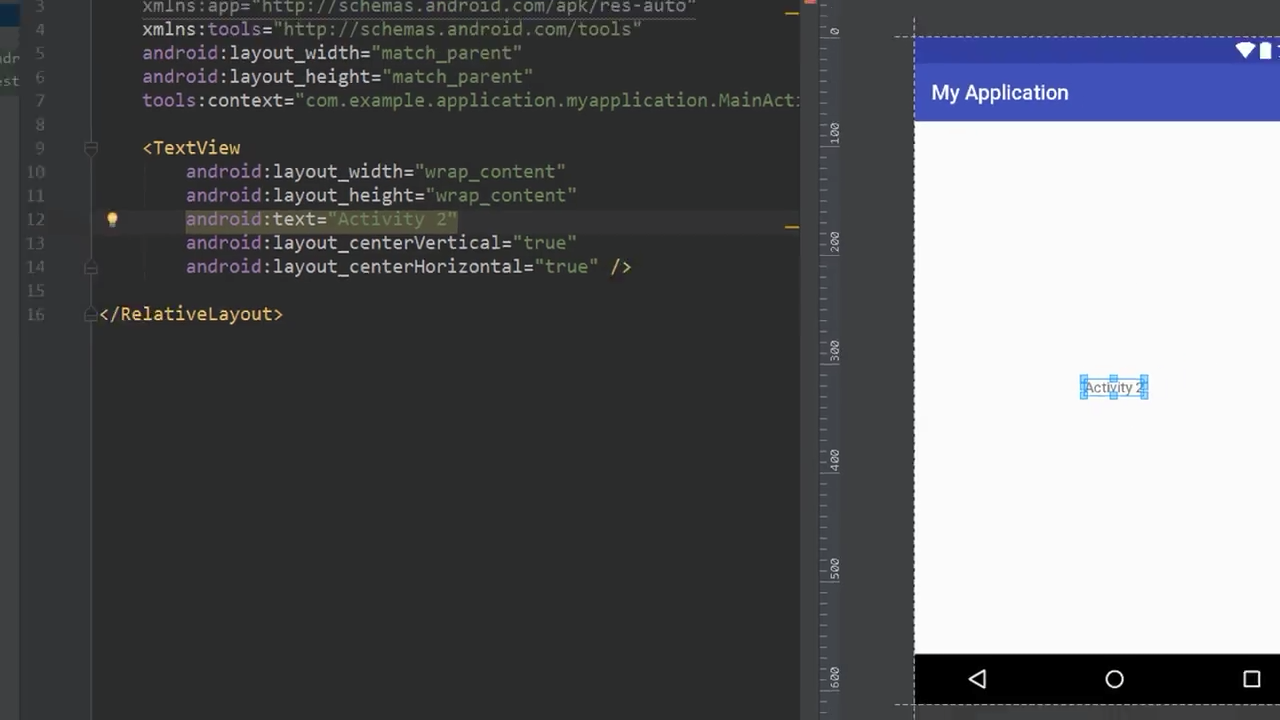
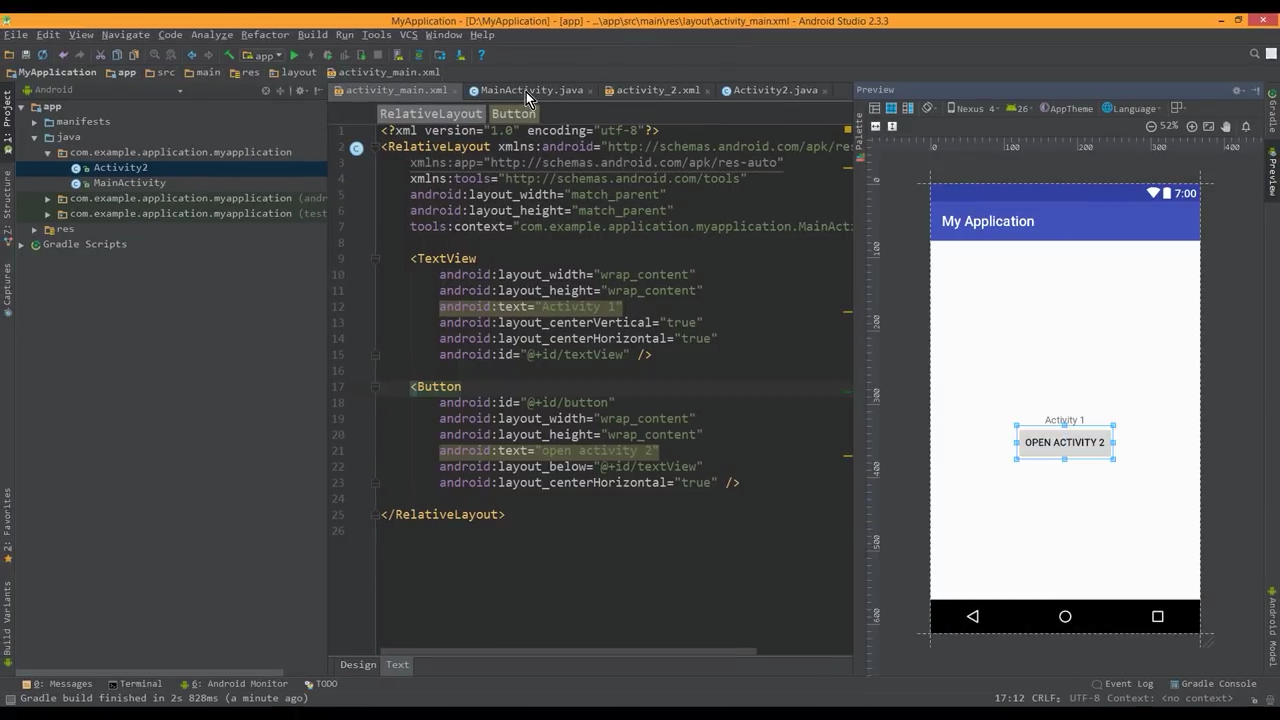
Understand the flow of Activity through the four states using the seven methods from the diagram:

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2020/01/android-activity-lifecycle.jpg)

***An example program : Using button OPENACTIVITY2 go through activity1 to activity2***

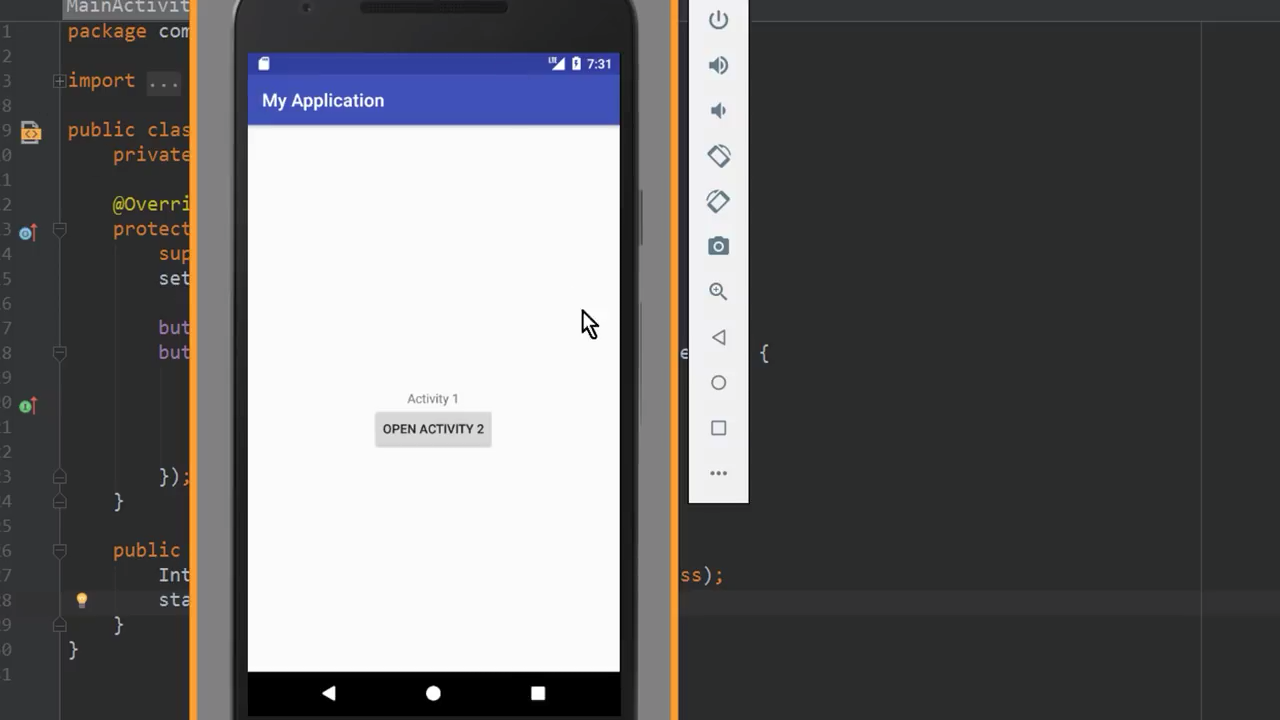
***10 Sec***

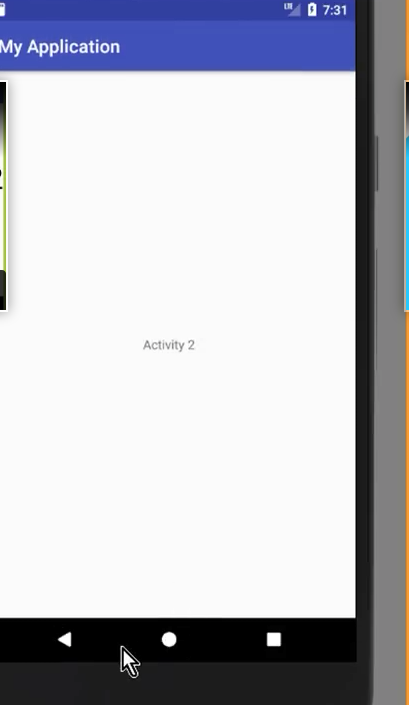
 

By running the program output as following :

By clicking button OPENACTIVITY2 from one activity1 to activity2





### 2. Services

A service is a component that runs in the background, it acts as an invisible worker of our application. It keeps updating data sources and activities. It also**broadcasts intents** and performs tasks when applications are not active. An example of service is we can surf the internet or use any other application while listening to music.

To execute services, extend the Services class in your sub-class:

public class MyService extends Services {

//code

}

### 3. Content Providers

Content Provider is a component that allows applications to share data among multiple applications. It hides the details of the database and can be used to read and write private data of the application which is not shared.

**For example –** you can consider looking for contact details in contact list. Or You might want photos from the gallery which are also provided by Content Provider.

To implement this, extend ContentProvider in your subclass:

public class Provider\_Name extendsContentProvider {

//code

}

### 4. Broadcast Receiver

Broadcast Receiver is a component that responds to broadcast messages from another application or the same system. It can also deliver broadcasts to applications that are not running. **For example –** notify the user that the battery is low. Android developers can use broadcast messages in the application or outside the normal flow.

To implement this, extend BroadcastReceiver to your receiver:

public class Broadcast\_Name extendsBroadcastReceiver {

//code

}