BROADCAST RECEIVER

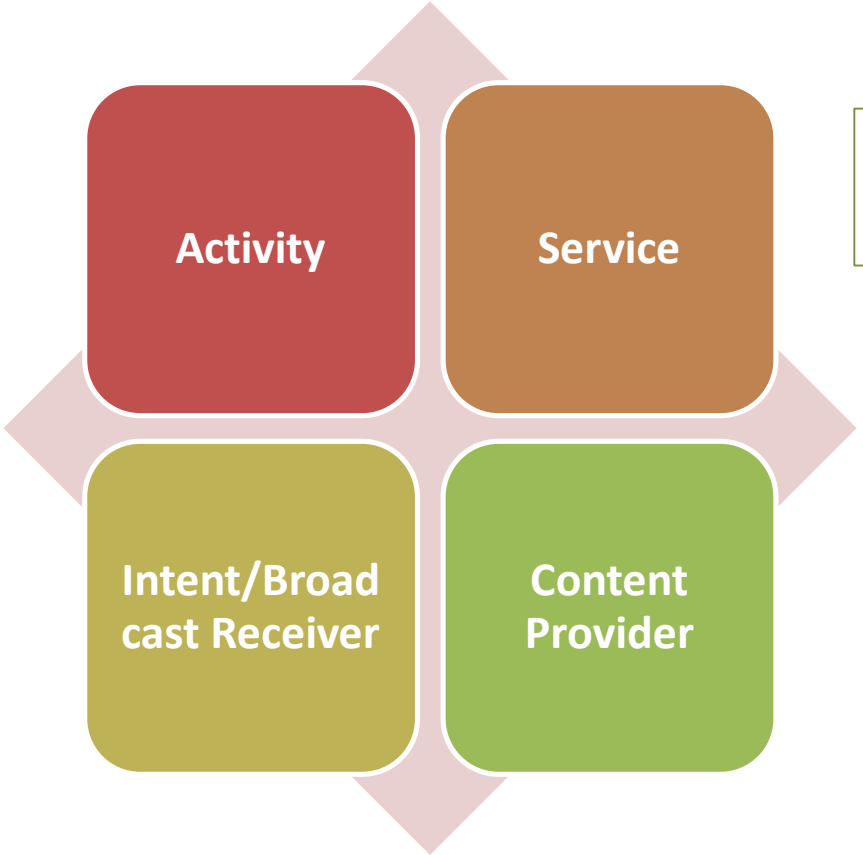
Course: Mobile Application Development

Unit : III – Building Blocks of Mobile Apps - II

Class / Semester: II MCA / III Semester

Building Blocks of App

- User Interface
- Represents a single screen
- Contain one/more views
- Extends **Activity** Class



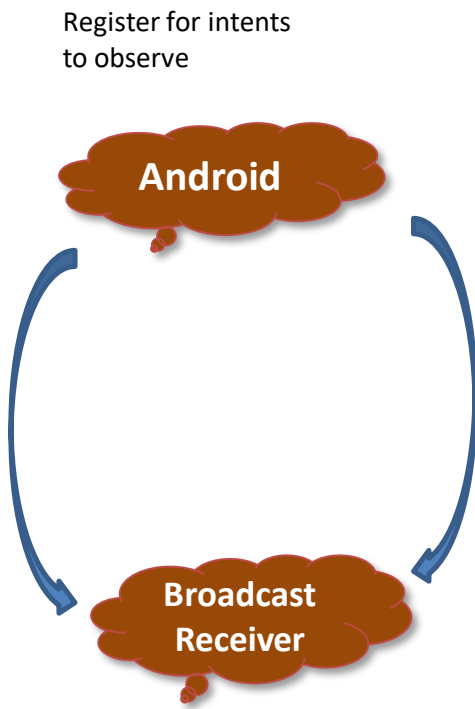
- No User Interface
- Runs in background
- Extends **Service** class

- Receives and reacts to broadcast intents
- No UI, but can start activity
- Extends BroadCastReceiver class

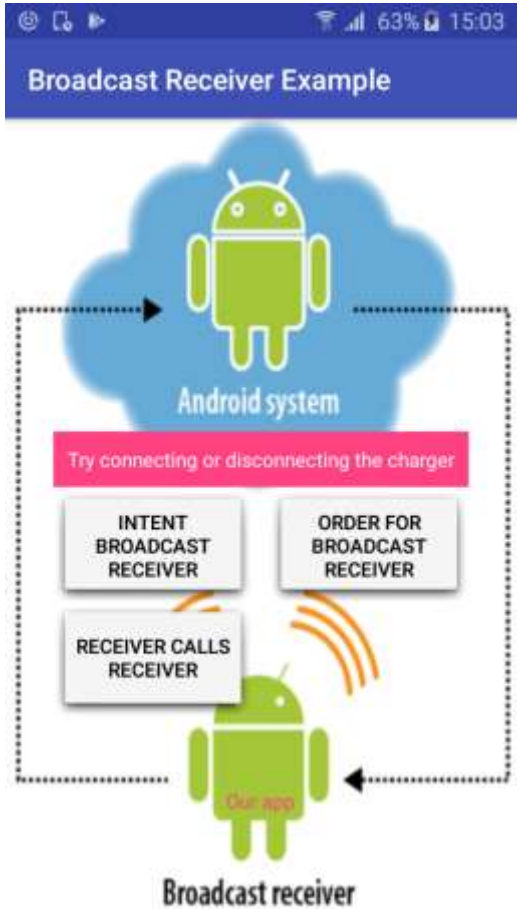
- Make app data available to other apps
- Data stored in SQLite Db
- Extends **ContentProvider** class

Broadcast Receiver

- It is an Android component which allows you to register for system or application events
- Simply respond to broadcast events from other apps or from the Android OS . For example, events like phone booting, low battery, charger connected
- Many broadcasts originate from system
- Application can also originate broadcasts, by creating a status bar notification to alert user when a broadcast event occurs
- It is a gateway to other components and it is intended to do minimal amount of work
- An intent used to send broadcasts to other applications, called broadcast intents: it may be system events or application events



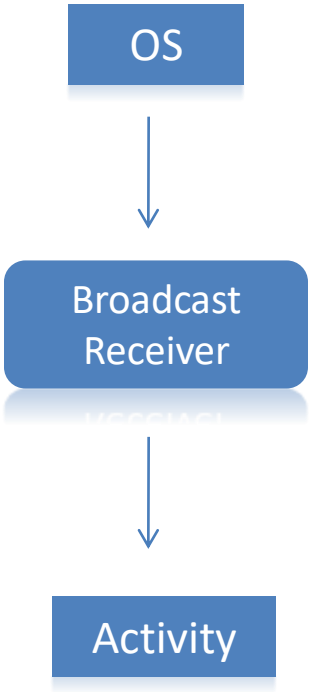
Broadcast Receiver Example



Connecting / Disconnecting charger

Broadcast Receiver

- Broadcast Receiver's job is to **pass a notification to the user,** in case a specific event occurs
- Each event creates a new Broadcast Receiver object and it runs on the *main* thread of the app, and after run, it is ready for garbage collection
- Android mandates a Broadcast Receiver to complete its execution within 10s
- There are two ways to register Broadcast Receiver
  - **Static:** Use <receiver> tag in your AndroidManifest.xml file
  - **Dynamic:** Use Context.registerReceiver () method to dynamically register an instance

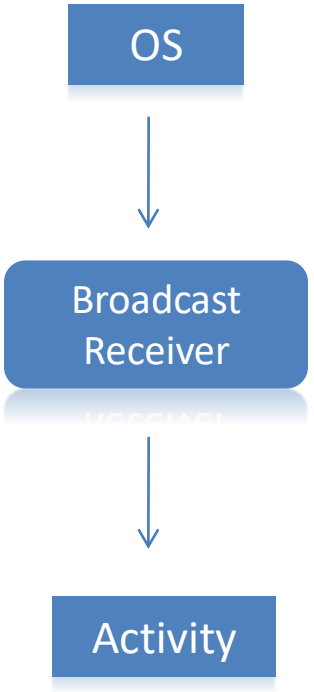


Broadcast Receiver

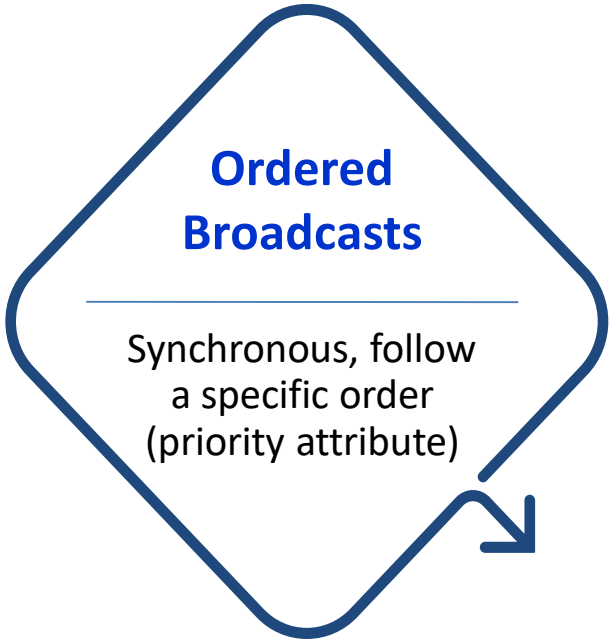
- BroadcastReceiver class containing code to that receive broadcast events and handle requests
- To register BroadcastReceiver to the application, declare in manifest file

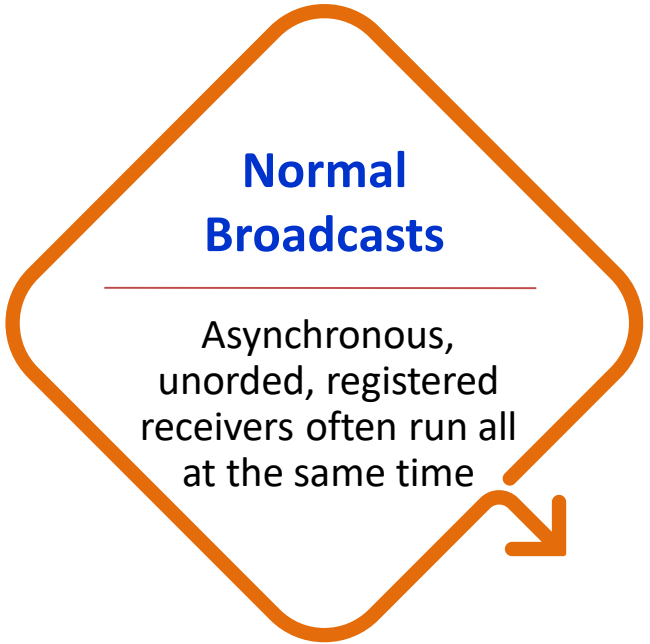
```
<receiver
        android:name="it.package.class"
        android:label="Label" >
</receiver>
```

-



Classes of Broadcast Receiver





System Intents

android.intent.action.BATTERY_CHANGED	battery's charging state, percentage
android.intent.action.BATTERY_LOW	Indicates low battery
android.intent.action.POWER_CONNECTED	power is connected to the device
android.intent.action.POWER_DISCONNECTED	power is disconnected from the device
android.intent.action.BOOT_COMPLETED	shown when the device boots for the first time
android.intent.action.CALL	perform a call to some specific person, according to data
android.intent.action.DATE_CHANGED	date of the device has changed
android.intent.action.REBOOT	device has rebooted
android.intent.action.CONNECTIVITY_CHANGE	network connectivity of device has changed
android.intent.action.BUG_REPORT	reports the bugs if there is any

Types of Broadcast Receiver

Static Registration

- Registration is done in the manifest file, using **<register>** tags

```
<receiver android:name="MyReceiver" >
<intent-filter> <action
android:name="android.net.conn.CONNECTIVITY_CHANGE" /> </intent-
filter> </receiver>
```

Broadcasts work both when the app is active and even if the app is inactive or closed

Implementation (Dynamic)

- It is implemented by extending the **BroadcastReceiver** class, and overriding its only callback method – **onReceive()**
- As soon as a Broadcast Receiver is triggered to respond to an event, the **onReceive()** executed

```
public class MyCustomBroadcastReceiver extends BroadcastReceiver
{
@override
 public void onReceive(Context context, Intent intent)
 {
  Toast.makeText(context, “The BR has been triggered”,
Toast.LENGTH_SHORT).show();
 }
 }
```

Registration

Dynamic Registration

- Registration is done using **Context.registerReceiver()**

```
IntentFilter filter = new IntentFilter();
intentFilter.addAction(getPackageName()+"android.net.conn.CONNECTIVITY_CHANGE");
 MyReceiver myReceiver = new MyReceiver();
registerReceiver(myReceiver, filter);
```

Dynamic Broadcast receivers run only when the app is running

Sending Broadcasts

- We can send a broadcasts in apps using three different ways

Method	Description
sendOrderedBroadcast(Intent, String)	This method is used to send broadcasts to one receiver at a time.
sendBroadcast(Intent)	This method is used to send broadcasts to all receivers in an undefined order.
LoadBroadcastManager.sendBroadcast	This method is used to send broadcasts to receivers that are in the same app as the sender.

Implementation

- Implemented Broadcast Receiver has to be registered in the AndroidManifest.xml file by configuring <intent-filter>

```
<receiver android:name="MyCustomBroadcastReceiver">
 <intent-filter>
<action
android:name="com.mad.broadcastdemo.SIMPLE_BROADCAST"/>
 <category android:name="android.intent.category.DEFAULT"/>
 </intent-filter>
 <receiver/>
```

- Use the sendBroadcast() method from the triggering component (Activity in this case) to send out a broadcast to trigger the Broadcast Receiver

```
Intent intent=new Intent("com.mad.broadcastdemo.SIMPLE_BROADCAST");
sendBroadcast(intent);
```

- A triggering component could be an Activity, a Service, or even another Broadcast Receive

When a Broadcast Receiver is registered in the manifest file, it will always respond to matching broadcasts, and there is no way to disable it

Implementation

if we wish to control enabling or disabling a Broadcast Receiver in an app, we can register and unregister it programmatically

```
protected void onResume()
{
 super.onResume();
 registerReceiver(myCustomBroadcastReceiver, new
IntentFilter("com.mad.broadcastdemo.SIMPLE_BROADCAST"));
 }
 protected void onPause() {
 super.onPause();
 unregisterReceiver(myCustomBroadcastReceiver);
 }
```

Thank you