Recording in Progress

This class is being recorded

Please turn off your video and/or video if you do not wish to be recorded

CMSC436: Programming Handheld Systems

The BroadcastReceiver Class

Today’s Topics

The BroadcastReceiver Class

Registering for events

Broadcasting events

Processing events

BroadcastReceiver

Base class for components that receive and react to events

BroadcastReceiver

BroadcastReceivers register to receive events in which they are interested

BroadcastReceiver

When Events occur at runtime they are represented as Intents

Those Intents are then broadcast to the system

BroadcastReceiver

Android routes the Intents to BroadcastReceivers that have registered to receive them

BroadcastReceivers receive the Intent via a call to onReceive()

Typical Use Case

Register BroadcastReceivers to receive specific events

When event occurs, broadcast an Intent

Android delivers Intent to registered recipients by calling their onReceive() method

Event handled in onReceive()

Registering for Intents

BroadcastReceivers can register in two ways

- Statically, in AndroidManifest.XML
- Dynamically, by calling a registerReceiver() method

Static Registration

Put <receiver> and <intent-filter> tags in AndroidManifest.xml

<Receiver> Tag Format

```
<receiver
        android:enabled=["true" | "false"]
        android:exported=["true" | "false"]
        android:icon="drawable resource"
        android:label="string resource"
        android:name="string"
        android:permission="string"
        android:process="string" >
    . . .
</receiver>
```

Intent Filter

Specify <intent-filter> tag within a <receiver>

See lecture on Intent class

Static Registration

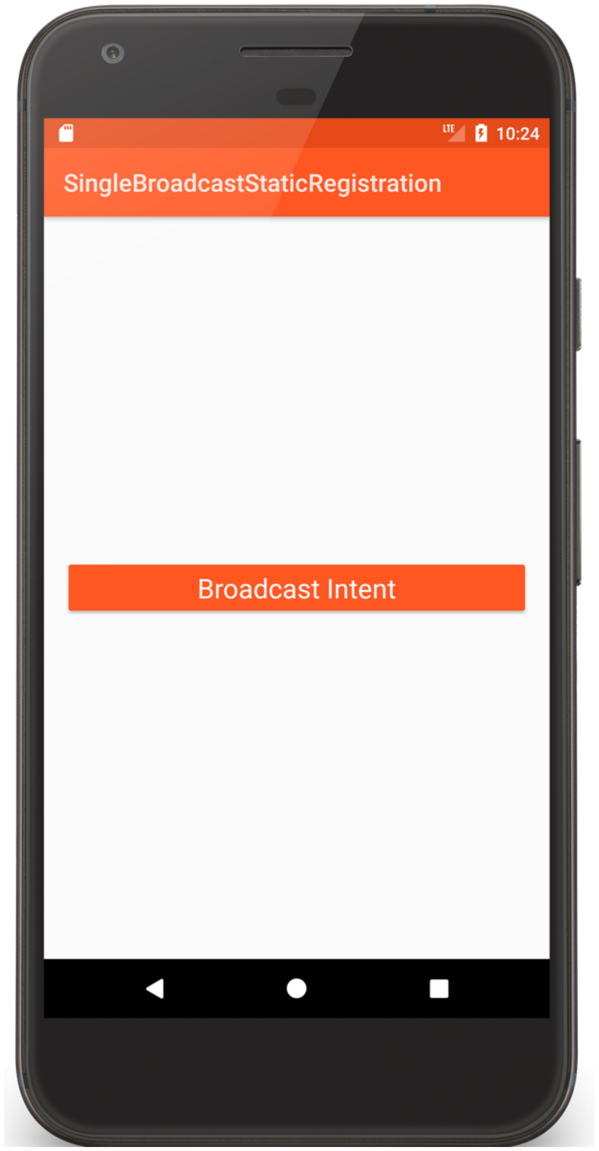
Receivers can be registered in AndroidManifest.xml

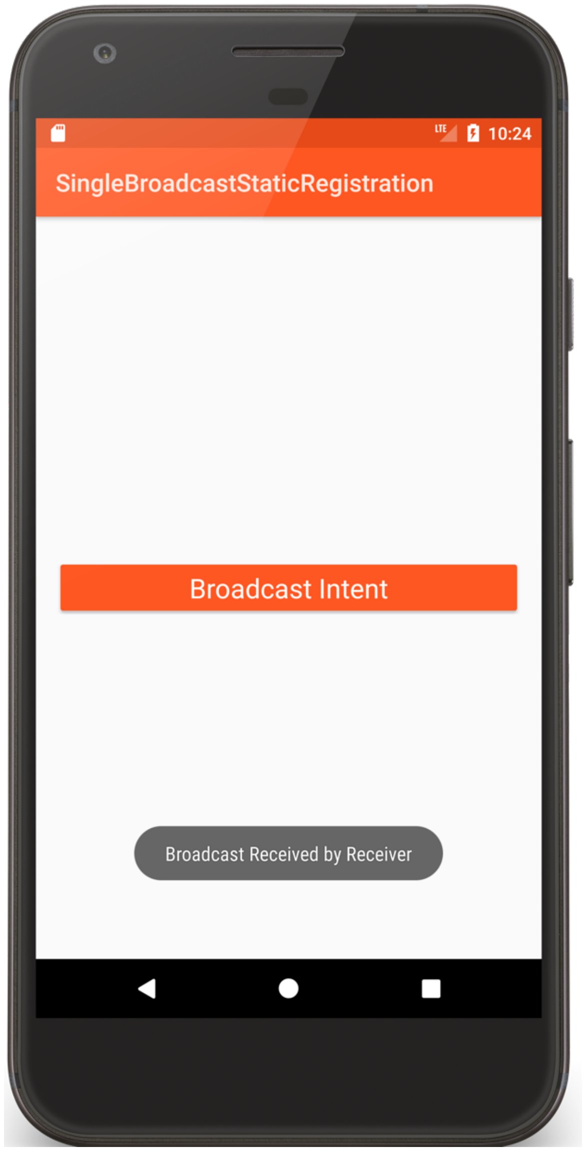
Will be woken to receive broadcasts, if needed

In API 26+, statically registered receivers cannot receive most implicit intents

See: https://developer.android.com/guide/components/broadcast-exceptions.html

BcastRec
SinBcast
StatReg




10:24
SingleBroadcastStaticRegistration
Broadcast Intent
Broadcast Received by Receiver

AndroidManifest.xml

```xml
<receiver
    android:name=".Receiver"
    android:exported="false"
    android:permission="android.permission.VIBRATE">
      <intent-filter>
        <action android:name="course.examples.broadcastreceiver.
                             singlebroadcaststaticregistration.SHOW_TOAST" />
      </intent-filter>
</receiver>
```

SimpleBroadcastActivity.kt

```
class SimpleBroadcastActivity : Activity() {
   …
   private const val CUSTOM_INTENT= "course.examples.broadcastreceiver.
                              singlebroadcaststaticregistration.SHOW_TOAST”
   …
fun onClick(view: View) {
    Log.i(TAG, "Broadcast sent")
    val intent = Intent(CUSTOM_INTENT)
    intent.setPackage("course.examples.broadcastreceiver.
                                   singlebroadcaststaticregistration")
    sendBroadcast(intent, Manifest.permission.VIBRATE)
    }
}
```

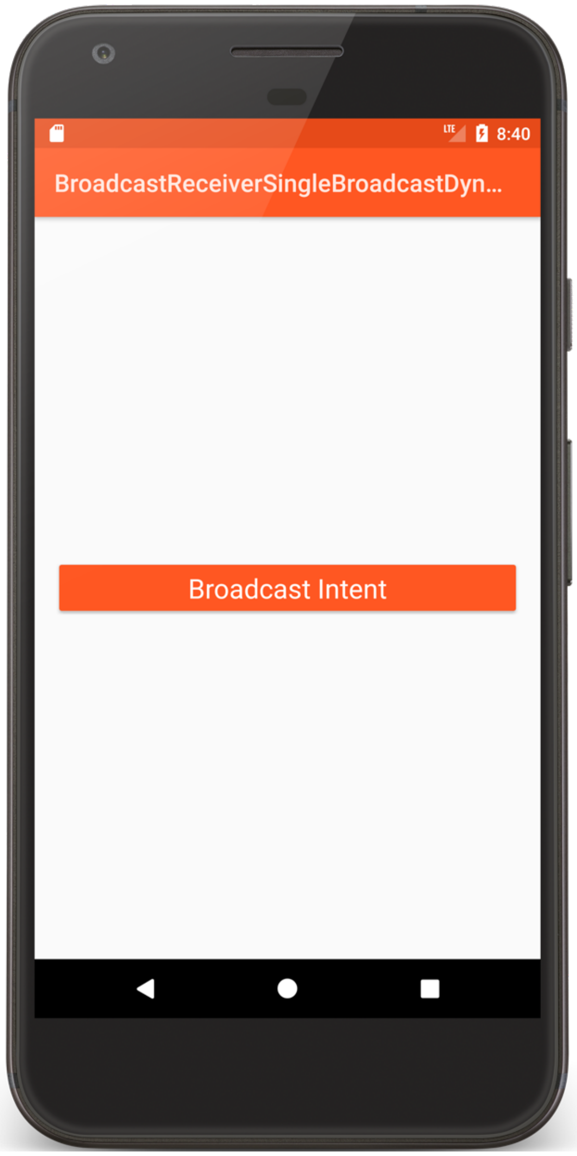
Receiver.kt

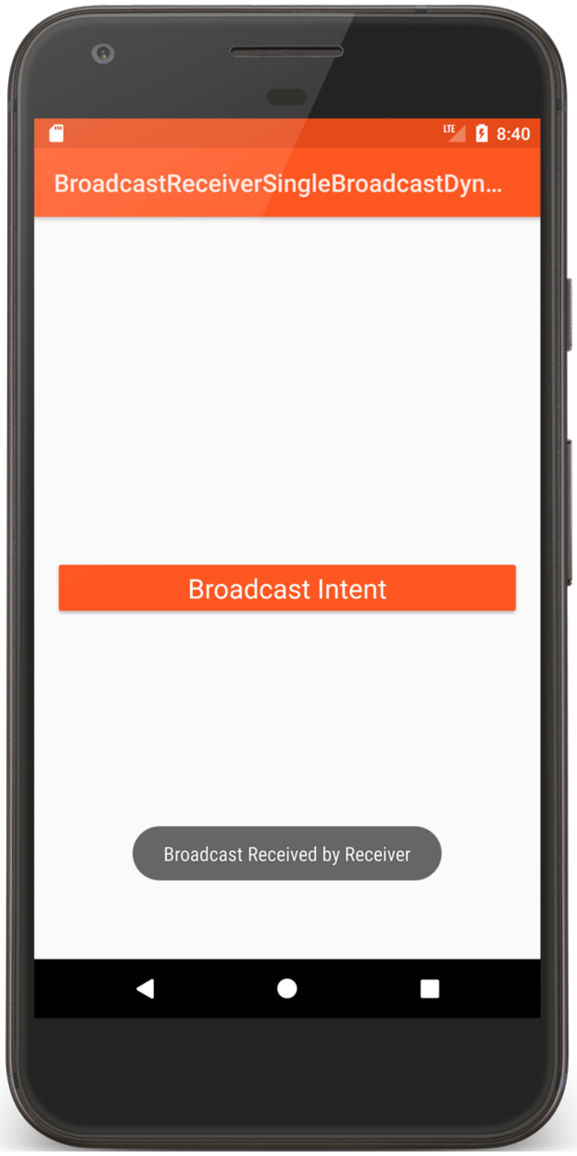
```kotlin
class Receiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
      Log.i(TAG, "Broadcast Received")
      val vibrator = context
            .getSystemService(Context.VIBRATOR_SERVICE) as Vibrator
      vibrator.vibrate(VibrationEffect.createOneShot(500,
                                VibrationEffect.DEFAULT_AMPLITUDE))
      Toast.makeText(context, "Broadcast Received by Receiver",
                                          Toast.LENGTH_LONG).show()
    }
}
```

Dynamic Registration

- Create an IntentFilter
- Create a BroadcastReceiver
- Register BroadcastReceiver using registerReceiver()
  - LocalBroadcastManager
  - Context
- Call unRegisterReceiver() to unregister BroadcastReceiver

BcastRec
SinBcast
DynReg





Broadcast Received by Receiver

SingleBroadcastActivity.kt

```
class SingleBroadcastActivity : Activity() {
  companion object {
    private const val CUSTOM_INTENT ="course.examples.broadcastreceiver.
                             singlebroadcastdynamicregistration.SHOW_TOAST"
    }
    private val intentFilter = IntentFilter(CUSTOM_INTENT)
    private val receiver = Receiver()
    private lateinit var mBroadcastMgr: LocalBroadcastManager

    public override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)

      mBroadcastMgr = LocalBroadcastManager.getInstance(applicationContext)
      setContentView(R.layout.main)
    }
```

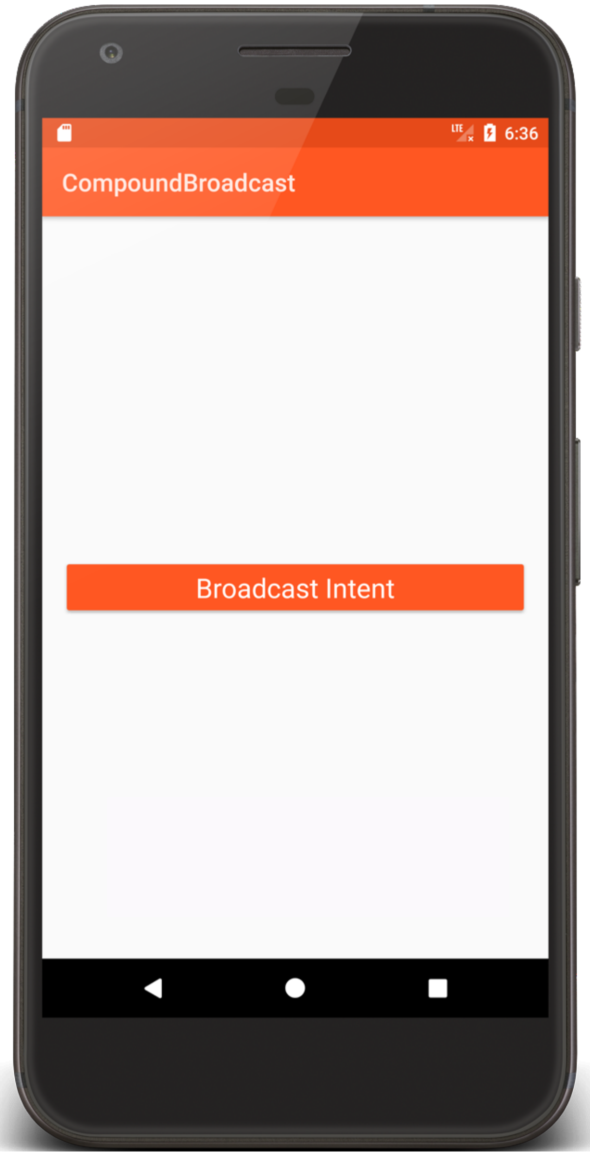
SingleBroadcastActivity.kt

```
    // Called when Button is clicked
    fun onClick(v: View) {
        mBroadcastMgr.sendBroadcast(
          Intent(CUSTOM_INTENT).setFlags(Intent.FLAG_DEBUG_LOG_RESOLUTION))
    }

    override fun onStart() {
        super.onStart()
        mBroadcastMgr.registerReceiver(receiver, intentFilter)
    }

    override fun onStop() {
        mBroadcastMgr.unregisterReceiver(receiver)
        super.onStop()
    }
}
```

BcastRec
CompBcast



CompoundBroadcast.kt

```
…
 private const val CUSTOM_INTENT = "course.examples.broadcastreceiver.
                                   compoundbroadcast.SHOW_TOAST"
 private val mIntentFilter = IntentFilter(CUSTOM_INTENT)
…
fun onClick(v: View) {
    val intent = Intent(CUSTOM_INTENT).setPackage(packageName)
                      .setFlags(Intent.FLAG_DEBUG_LOG_RESOLUTION)
    sendBroadcast(intent, Manifest.permission.VIBRATE)
}
```

CompoundBroadcast.kt

```
    override fun onStart() {
        super.onStart()
        registerReceiver(mReceiver1, mIntentFilter)
    }

    override fun onStop() {
        unregisterReceiver(mReceiver1)
        super.onStop()
    }
}
```

AndroidManifest.xml

```
<receiver
    android:name=".Receiver3"
    android:exported="false">
      <intent-filter>
        <action android:name="course.examples.broadcastreceiver.
                                                   compoundbroadcast.SHOW_TOAST" />
            </intent-filter>
</receiver>

<receiver
    android:name=".Receiver2"
    android:exported="false">
      <intent-filter>
        <action android:name="course.examples.broadcastreceiver.
                                                   compoundbroadcast.SHOW_TOAST" />
      </intent-filter>
</receiver>
```

Event Broadcast

- Multiple broadcast methods supported
- Normal vs. Ordered
  - Normal: processing order undefined
  - Ordered: sequential processing in priority order

Some Debugging Tips

Log extra Intent resolution information

Intent.setFlag(FLAG_DEBUG_LOG_RESOLUTION)

List registered BroadcastReceivers

Dynamically registered

```
% adb shell dumpsys  activity b
```

Statically registered

```
% adb shell dumpsys  package
```

Event Delivery

- Intents are delivered to BroadcastReceiver by calling onReceive(Context, Intent)
  - The Context in which the receiver is running
  - The Intent that was broadcast

Event Handling in onReceive()

Hosting process has high priority while onReceive() is executing

onReceive() runs on the main Thread

So onReceive() should be short-lived

Event Handling in onReceive()

Note: If event handling is lengthy, consider starting a Service, rather than performing complete operation in onReceive()

Will cover the Service class later in the course

Event Handling in onReceive()

BroadcastReceiver is not considered valid once onReceive() returns

Normally, BroadcastReceivers can’t start asynchronous operations

- e.g., showing a Dialog, starting an Activity via startActivityForResult()
- Why not?

Ordered Broadcasts

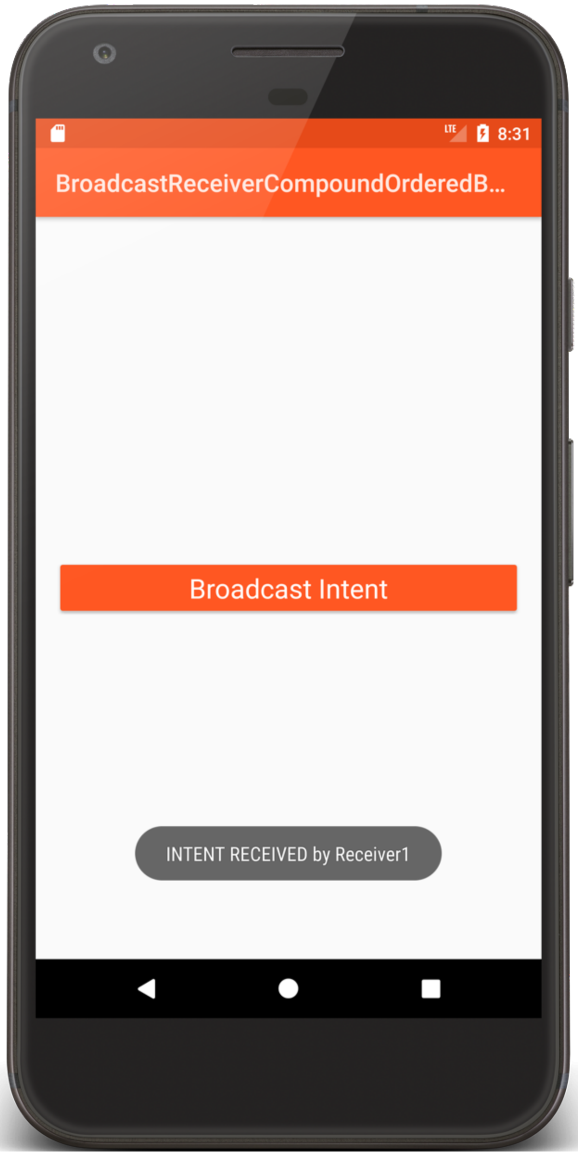
```
// send Intent to BroadcastReceivers in priority order

void sendOrderedBroadcast (Intent intent,  String receiverPermission)

// send Intent to BroadcastReceivers in priority order. Includes multiple
// parameters for greater control

void sendOrderedBroadcast (Intent intent,
                           String receiverPermission,
                           BroadcastReceiver resultReceiver,
                           Handler scheduler,
                           int initialCode,
                           String initialData,
                           Bundle initialExtras)
```

BcastRec
CompOrd
Bcast



INTENT RECEIVED by Receiver1

AndroidManifest.xml

```
<receiver
  android:name=".Receiver2"
  android:exported="false">
  <intent-filter android:priority="1">
    <action android:name="course.examples.BroadcastReceiver.
                                    compoundorderedbroadcast.SHOW_TOAST" />
        </intent-filter>
    </receiver>
<receiver
  android:name=".Receiver3"
  android:exported="false">
  <intent-filter android:priority="10">
    <action android:name="course.examples.BroadcastReceiver.
                                    compoundorderedbroadcast.SHOW_TOAST" />
        </intent-filter>
</receiver>
```

CompoundOrderedBroadcast.kt

```kotlin
fun onClick(v: View) {
    sendOrderedBroadcast(
        Intent(CUSTOM_INTENT).setPackage(packageName).setFlags(
            Intent.FLAG_DEBUG_LOG_RESOLUTION),
                android.Manifest.permission.VIBRATE)
}
override fun onStart() {
    super.onStart()
    val intentFilter = IntentFilter(CUSTOM_INTENT)
    intentFilter.priority = 3
    registerReceiver(mReceiver, intentFilter)
}
override fun onStop() {
    unregisterReceiver(mReceiver)
    super.onStop()
}
```

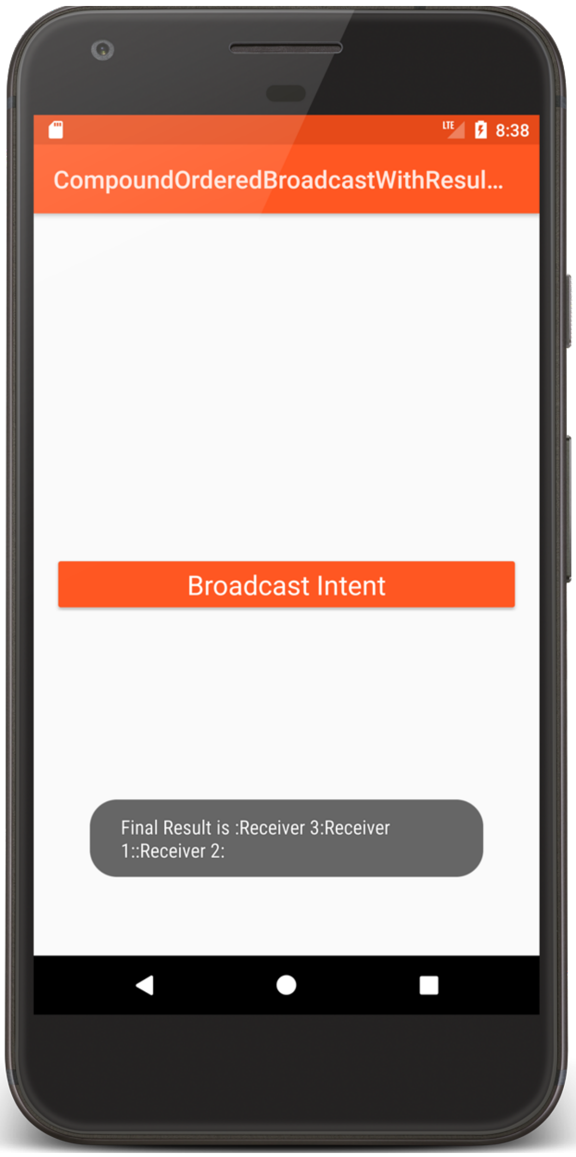
Receiver1.kt

```
class Receiver1 : BroadcastReceiver() {
    …
    override fun onReceive(context: Context, intent: Intent) {

        Log.i(TAG, "INTENT RECEIVED")

        if (isOrderedBroadcast) {
            Log.i(TAG, "Calling abortBroadcast()")
            abortBroadcast()
        }
    }
}
```

BcastRecCompOrd
BcastWithResRec



```
Final Result is :Receiver 3:Receiver
1::Receiver 2:
```

CompoundOrderedBroadcastWithResultReceiver.kt

```
fun onClick(v: View) {
  sendOrderedBroadcast(Intent(CUSTOM_INTENT).setPackage(packageName),
        null, object : BroadcastReceiver() {
            override fun onReceive(context: Context, intent: Intent) {
                Toast.makeText(context, "Final Result is $resultData",
                    Toast.LENGTH_LONG).show()
            }
        }, null, 0, null, null)
}
```

Recevier3.kt

```
class Receiver3 : BroadcastReceiver() {

…
    override fun onReceive(context: Context, intent: Intent) {

        Log.i(TAG, "INTENT RECEIVED by Receiver3")

        val tmp = if (resultData == null) "" else resultData
        resultData = "$tmp:Receiver 3"
    }
}
```

Long-Running Operations

After onReceive() exits, system can kill BroadcastReceiver

Don’t’ start long-running Threads from onReceive()

Options

- Call goAsync()
- Schedule a JobService with JobScheduler. (Will discuss Services later in course)

goAsync()

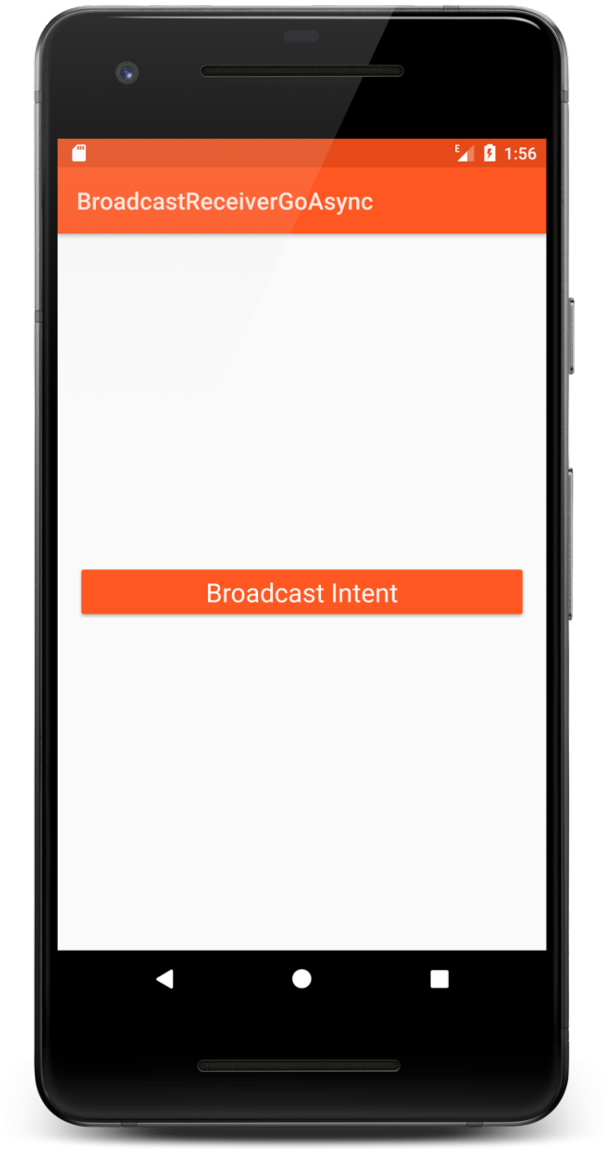
BroadcastReceiver is generally valid only until onReceive() exits

Use goAsync() to allow asynchronous processing from onReceive()

Method returns an object of PendingResult

Receiver considered alive until PendingResult.finish()

BcastRecGoAsync



Receiver.kt

```
override fun onReceive(context: Context, intent: Intent) {
    Log.i(TAG, "Broadcast Received")

    val pendingResult = goAsync()

    GlobalScope.launch(context = Dispatchers.Main) {
        delay(7000)
        Toast.makeText(context,
         "Broadcast Received by Receiver", Toast.LENGTH_LONG).show()
        pendingResult.finish()
    }
}
```

Additional Notes

BroadcastReceiver’s original design has changed to improve security, performance and UX

- Prefer LocalBroadcastManager to Context
- Prefer Context registration over Manifest registration
- Don’t put sensitive info in implicit Intents you broadcast
- Don’t start Activities from onReceive()

Next Time

User Notifications

Example Applications

BcastRecSinBcastStatReg

BcastRecSinBcastDynReg

BcastRecCompBcast

BcastRecCompOrdBcast

BcastRecCompOrdBcastWithResRec

BcastRecGoAsync