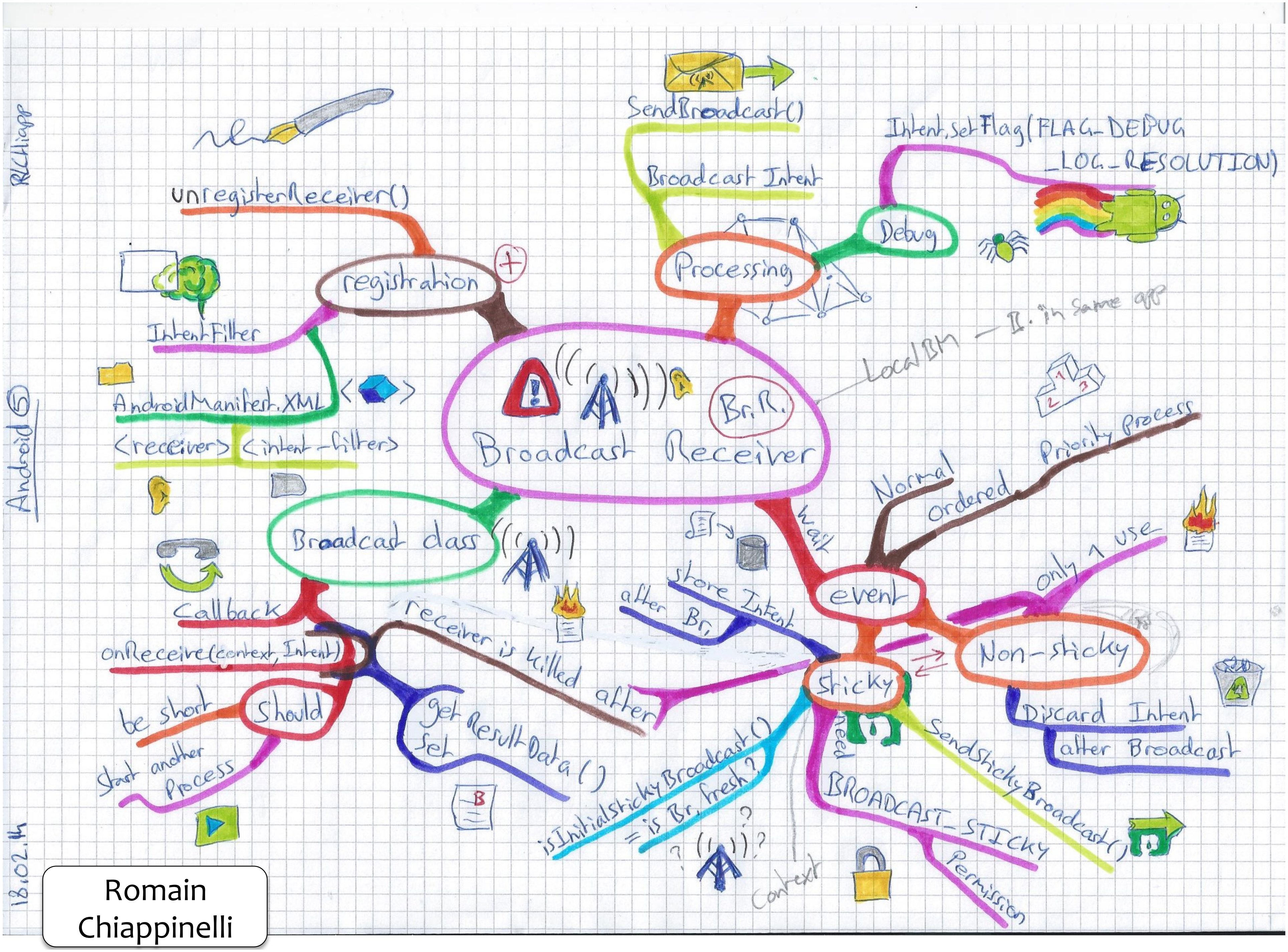


Romain Chiappinelli

Programming Handheld Systems

Adam Porter

The BroadcastReceiver Class

Today's Topics

The Broadcast Class

Registration

Broadcast

Processing

BroadcastReceiver

Base class for components that receive and react to events

BroadcastReceiver

BroadcastReceivers register to receive events in which they are interested

Broadcast Receiver

When Events occur they are represented as Intents

Those Intents are then broadcast to the system

Broadcast Receiver

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

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> format

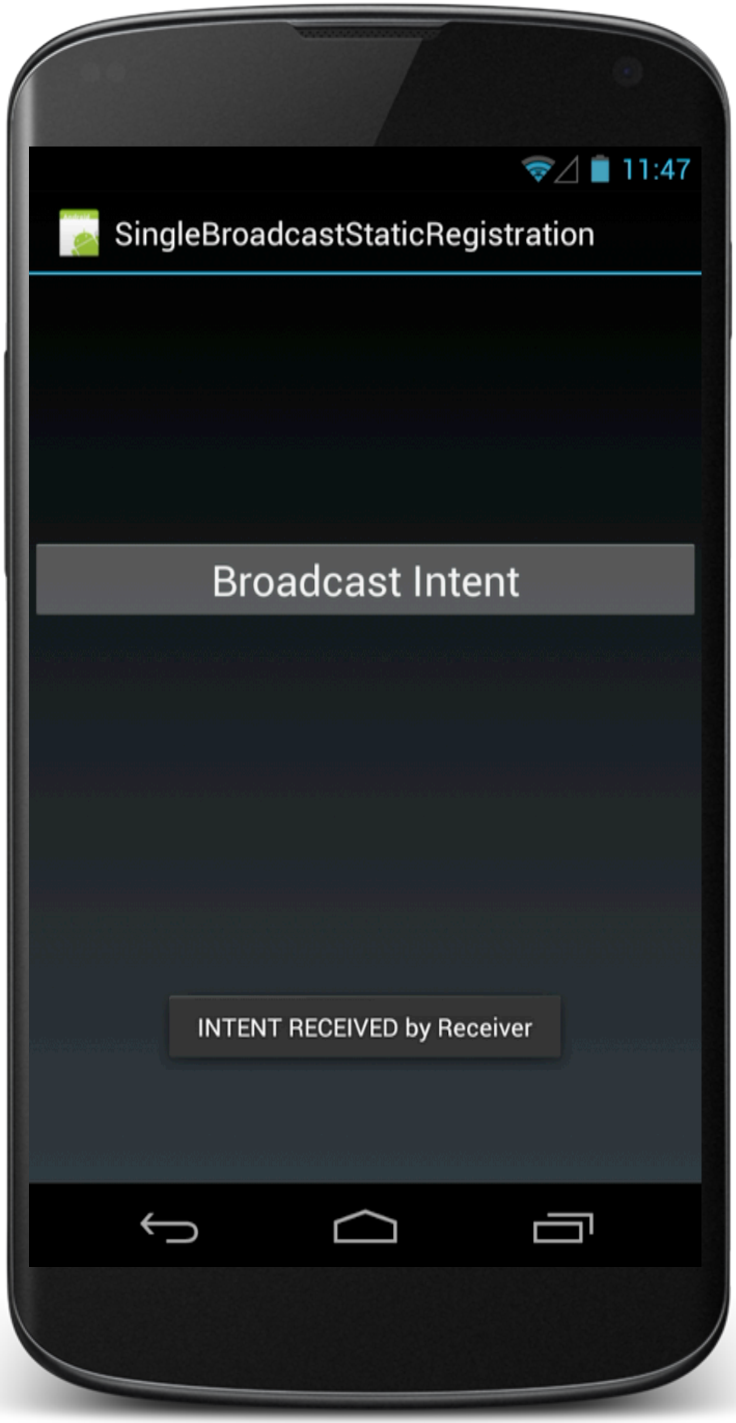
```
<receiver
        android:enabled=["true" | "false"]
        android:exported=["true" | "false"]
        android:icon="drawable resource"
        android:label="string resource"
        android:name="string"
        android:permission="string"
        android:process="string" >
    . . .
</receiver>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

Intent Filter

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

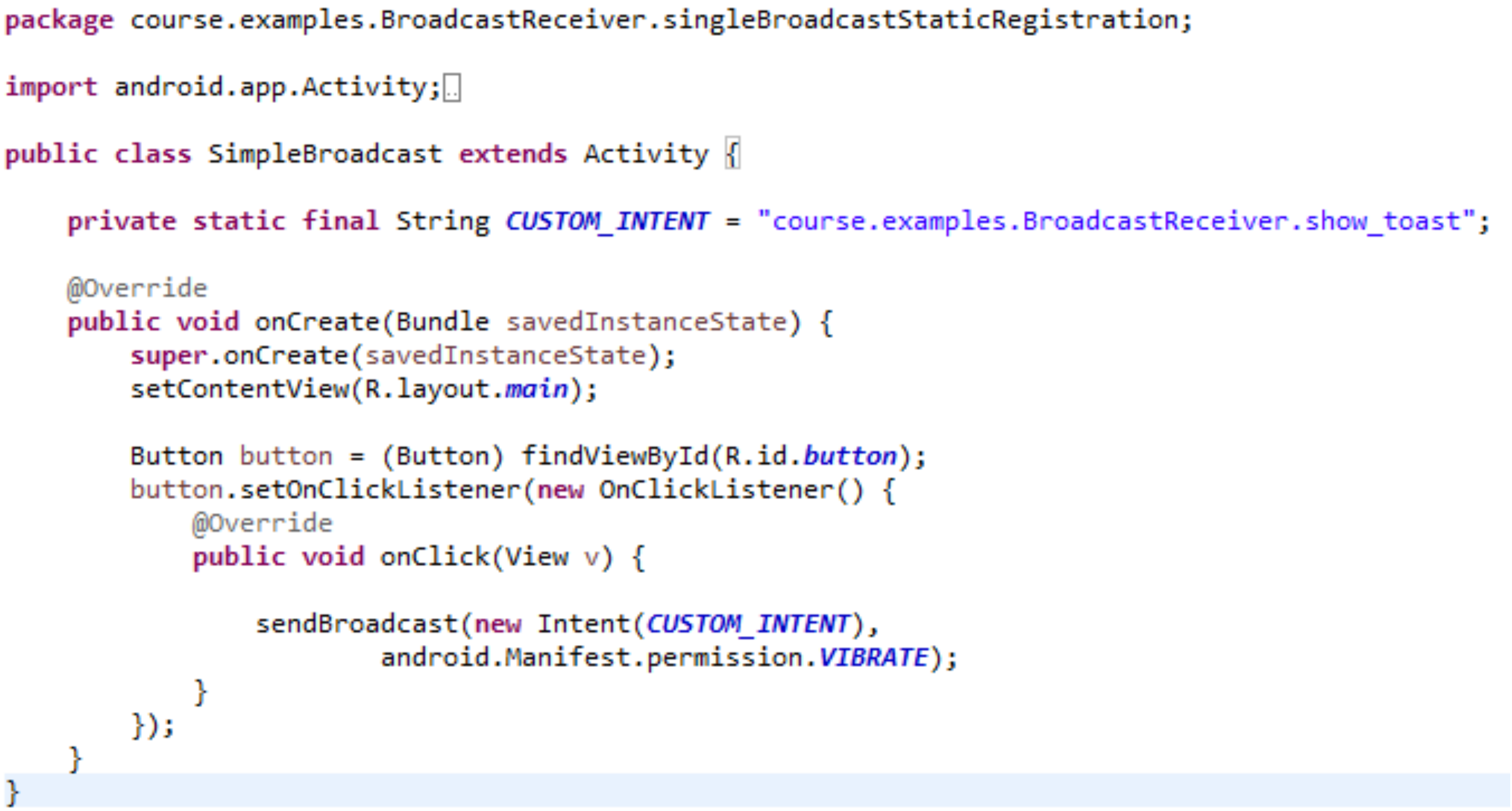
Static Registration

Receivers are registered with the system at boot time or when their application package is added at runtime



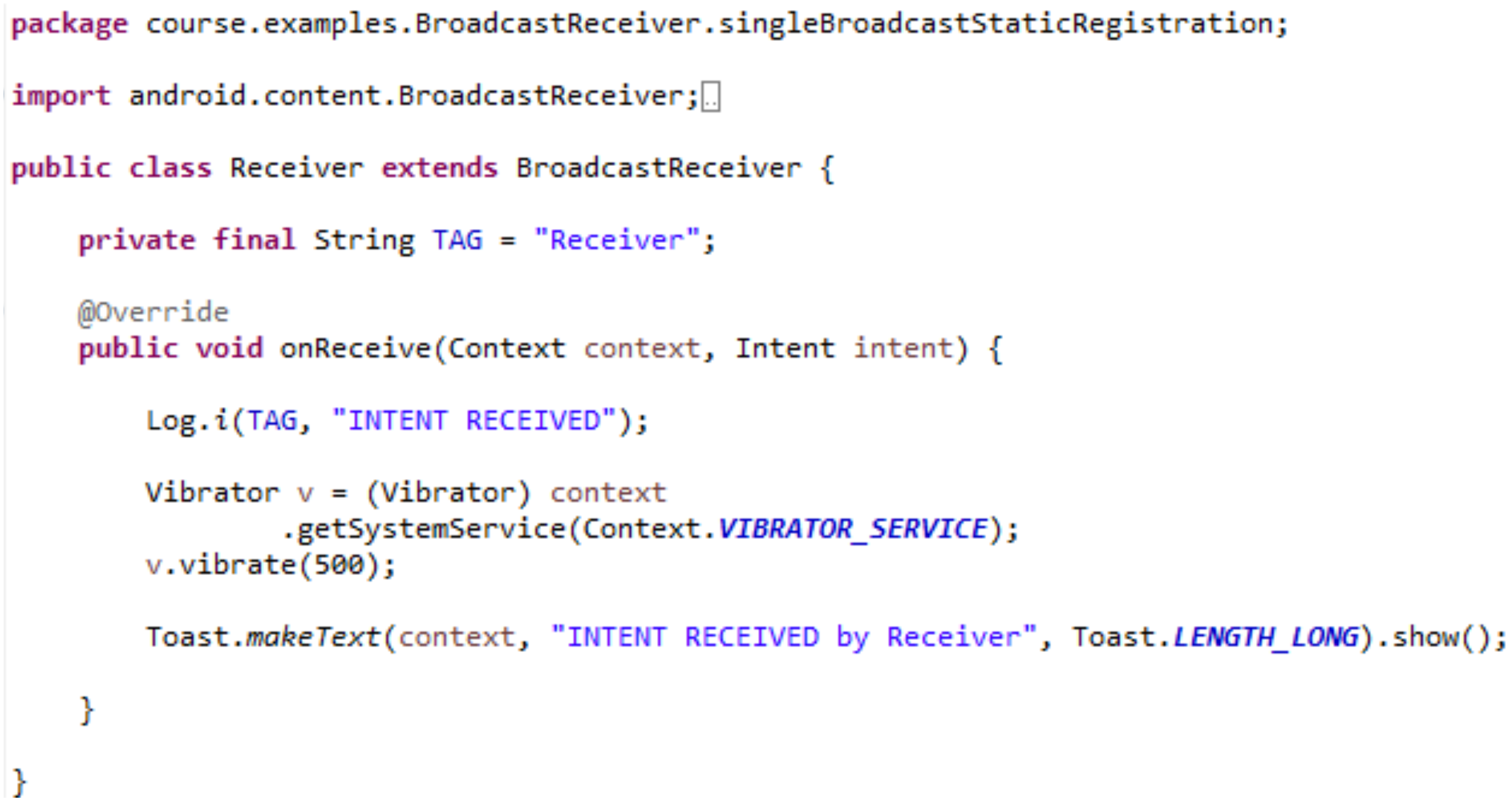
INTENT RECEIVED by Receiver

BcastRecSinBcastStatReg



```
package course.examples.BroadcastReceiver.singleBroadcastStaticRegistration;

import android.app.Activity;

public class SimpleBroadcast extends Activity {

    private static final String CUSTOM_INTENT = "course.examples.BroadcastReceiver.show_toast";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {

                sendBroadcast(new Intent(CUSTOM_INTENT),
                        android.Manifest.permission.VIBRATE);
            }
        });
    }
}
```

BcastRecSinBcastStatReg

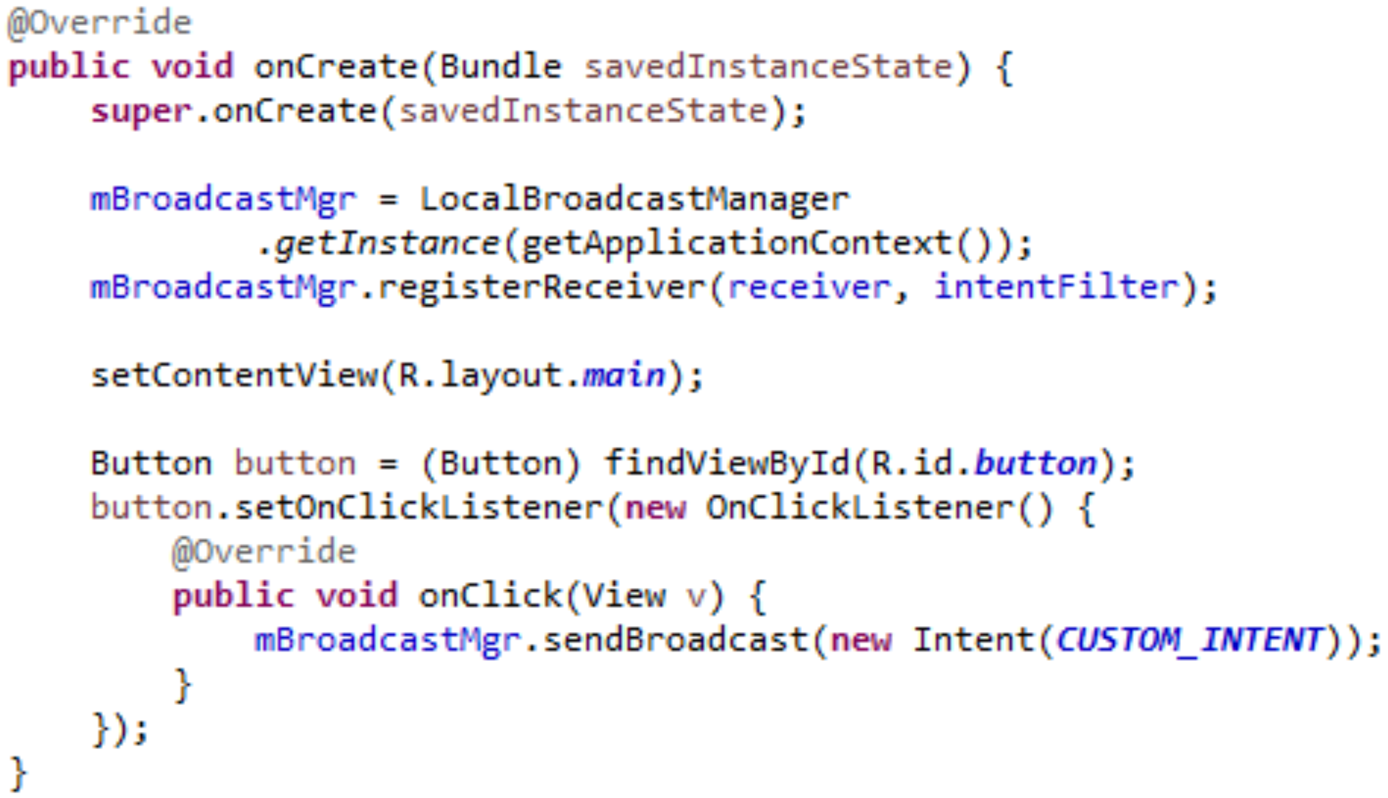


```
package course.examples.BroadcastReceiver.singleBroadcastStaticRegistration;

import android.content.BroadcastReceiver;

public class Receiver extends BroadcastReceiver {

    private final String TAG = "Receiver";

    @Override
    public void onReceive(Context context, Intent intent) {

        Log.i(TAG, "INTENT RECEIVED");

        Vibrator v = (Vibrator) context
                .getSystemService(Context.VIBRATOR_SERVICE);
        v.vibrate(500);

        Toast.makeText(context, "INTENT 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

BcastRecSinBcastDynReg



```java
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    mBroadcastMgr = LocalBroadcastManager
            .getInstance(getApplicationContext());
    mBroadcastMgr.registerReceiver(receiver, intentFilter);

    setContentView(R.layout.main);

    Button button = (Button) findViewById(R.id.button);
    button.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            mBroadcastMgr.sendBroadcast(new Intent(CUSTOM_INTENT));
        }
    });
}
```

Event Broadcast

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

Event Broadcast

- Sticky vs. Non-Sticky
  - Sticky: Store Intent after initial broadcast
  - Non-Sticky: Discard Intent after initial broadcast
- With or without receiver permissions

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 delivered by calling onReceive(), passing in:

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

Event Handling in onReceive()

onReceive() runs on the main Thread, so it should be short-lived

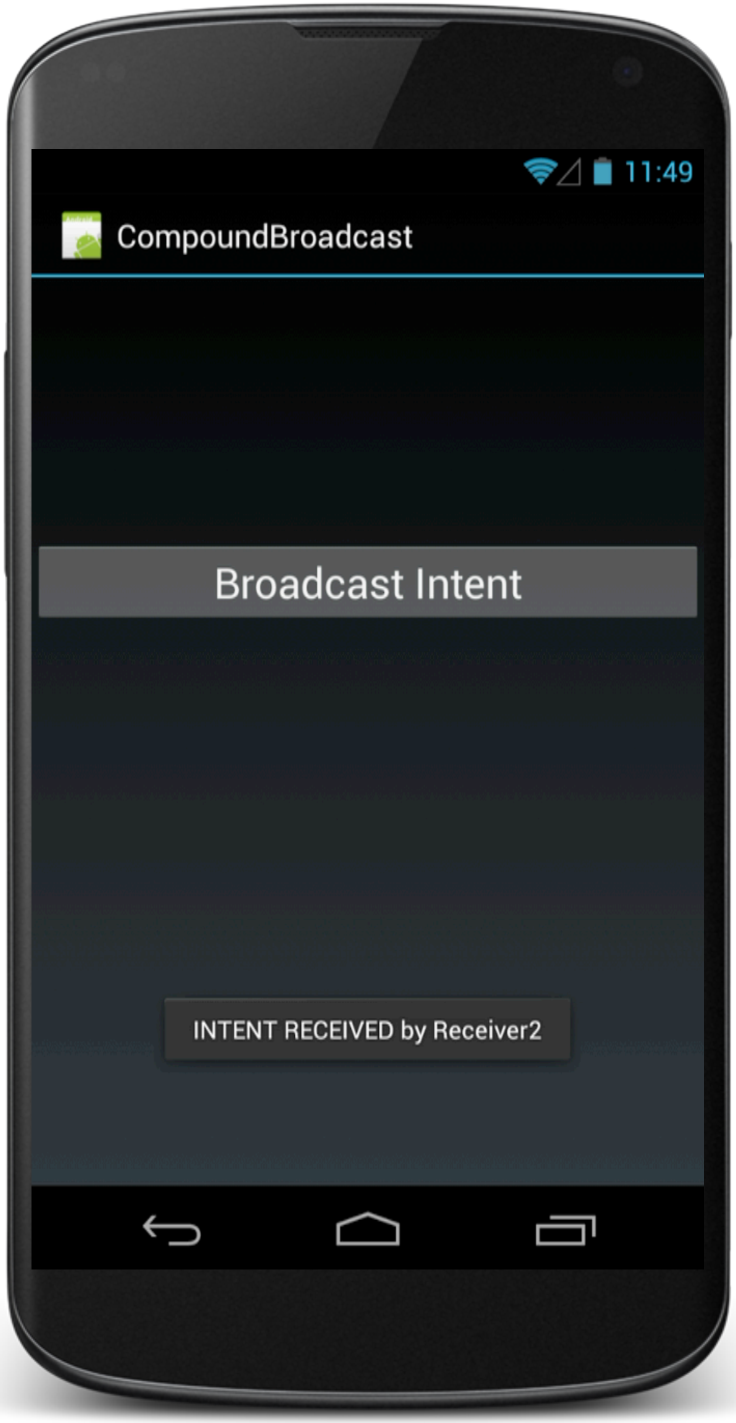
If event handling is lengthy, consider starting a Service, rather than performing complete operation in onReceive()

Event Handling in onReceive()

- Receiver is not considered valid once onReceive() returns

- Normally BroadcastReceivers can't start asynchronous operations

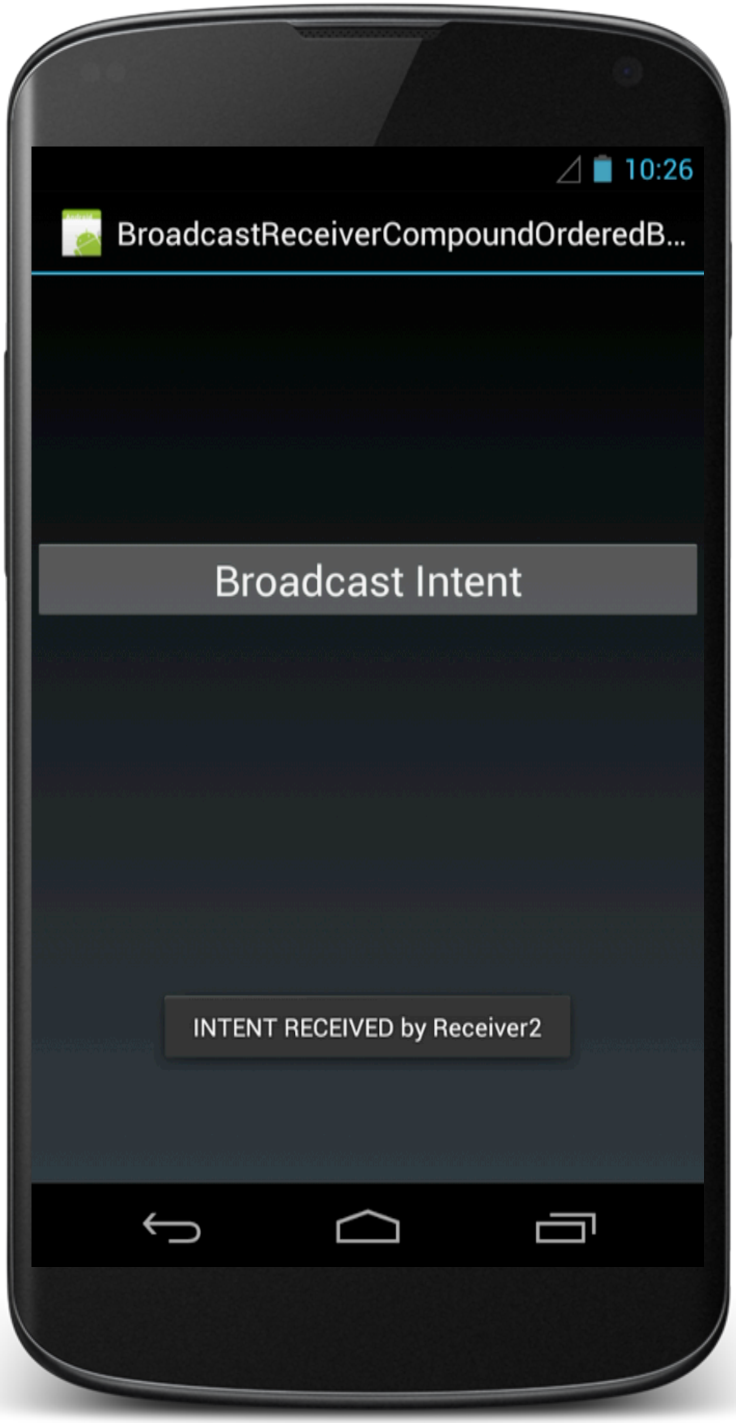
  - e.g., showing a dialog, starting an Activity via startActivityForResult()



INTENT RECEIVED by Receiver2

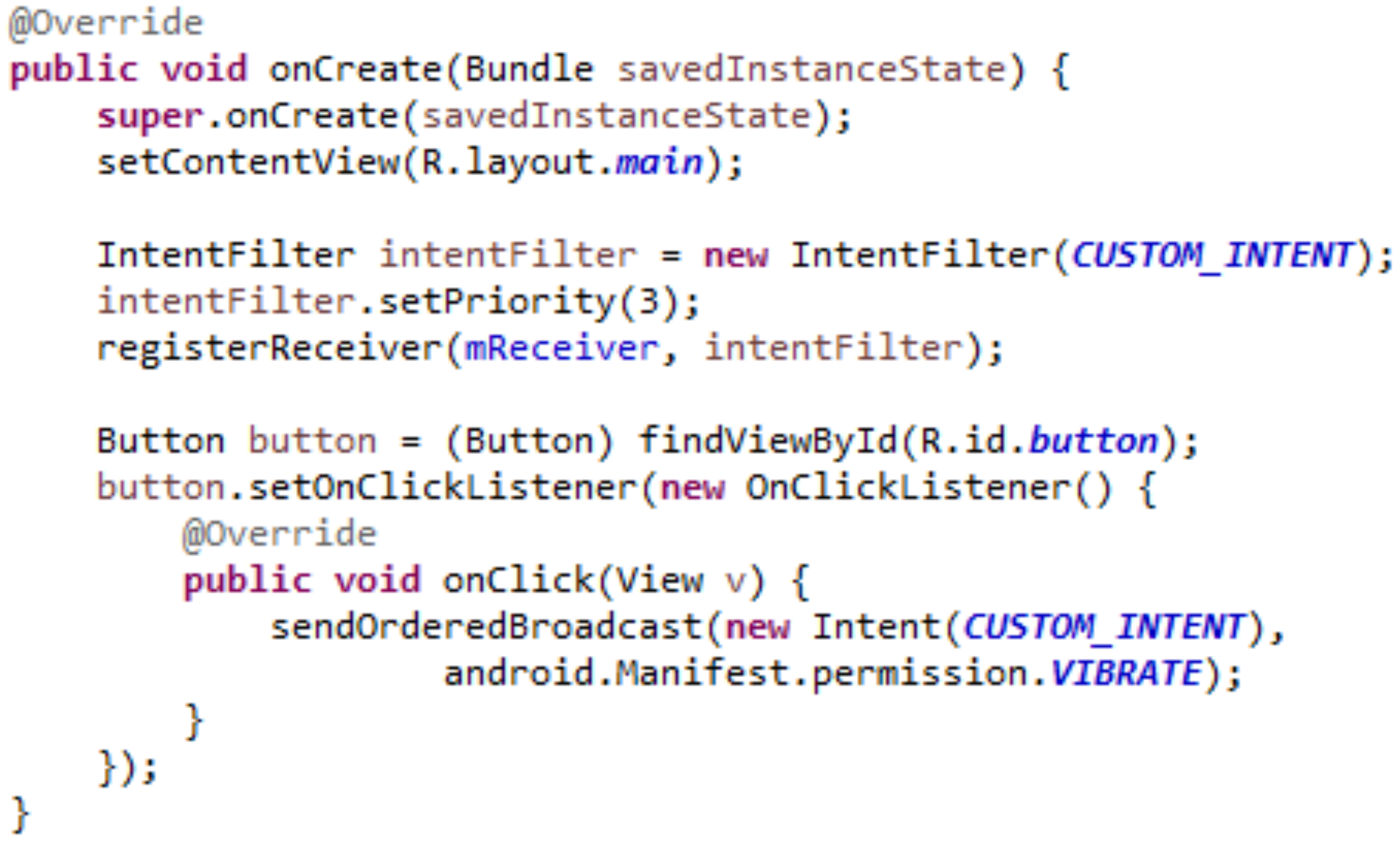
Ordered Broadcasts

```
// 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)
```

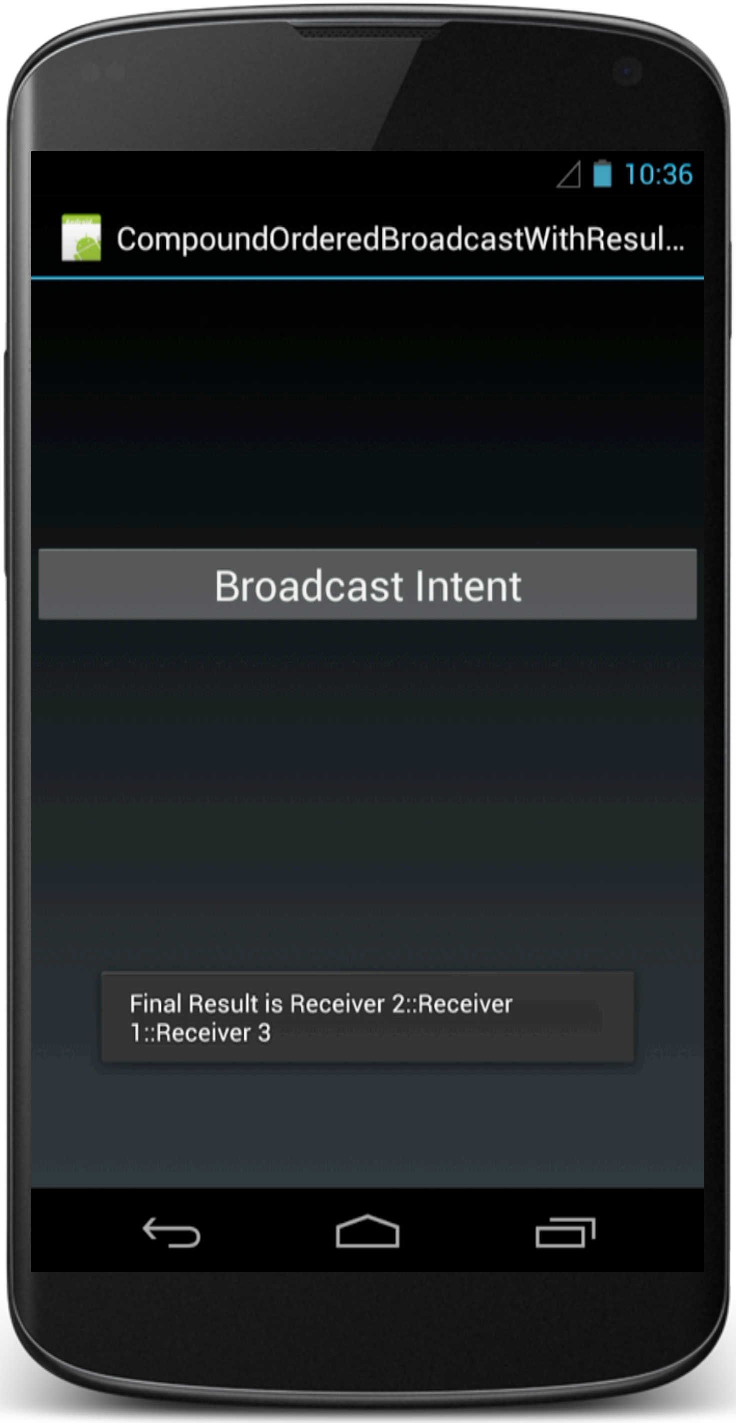


INTENT RECEIVED by Receiver2

BcastRecCompOrdBcast

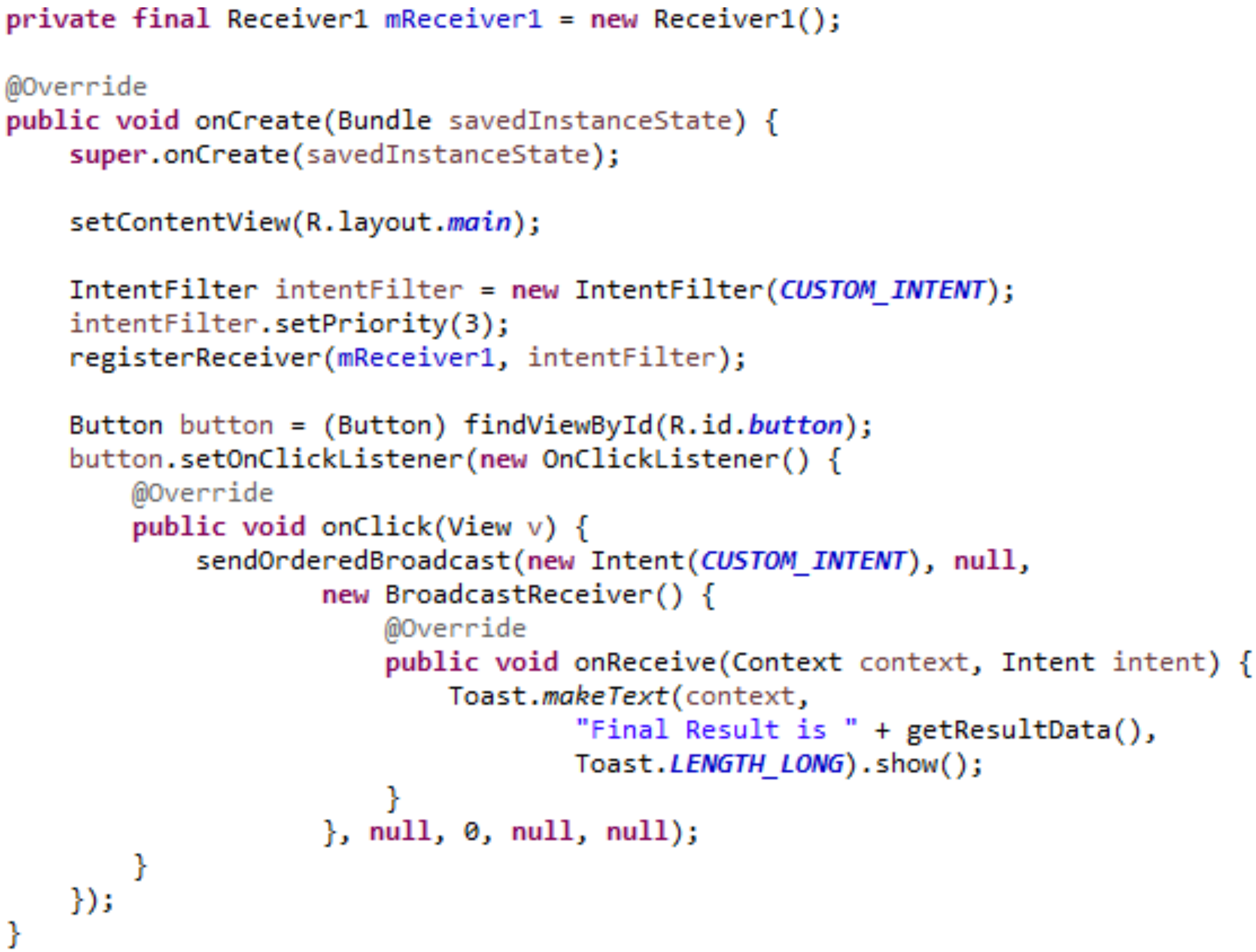


```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    IntentFilter intentFilter = new IntentFilter(CUSTOM_INTENT);
    intentFilter.setPriority(3);
    registerReceiver(mReceiver, intentFilter);

    Button button = (Button) findViewById(R.id.button);
    button.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            sendOrderedBroadcast(new Intent(CUSTOM_INTENT),
                    android.Manifest.permission.VIBRATE);
        }
    });
}
```



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

BcastRecCompOrdBcastWithResRec



```
private final Receiver1 mReceiver1 = new Receiver1();

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    IntentFilter intentFilter = new IntentFilter(CUSTOM_INTENT);
    intentFilter.setPriority(3);
    registerReceiver(mReceiver1, intentFilter);

    Button button = (Button) findViewById(R.id.button);
    button.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            sendOrderedBroadcast(new Intent(CUSTOM_INTENT), null,
                    new BroadcastReceiver() {
                        @Override
                        public void onReceive(Context context, Intent intent) {
                            Toast.makeText(context,
                                    "Final Result is " + getResultData(),
                                    Toast.LENGTH_LONG).show();
                        }
                    }, null, 0, null, null);
        }
    });
}
```

Sticky Broadcasts

- Sticky Intents are cached by Android
  - New Intents overwrite older Intents they match

Sticky Broadcasts

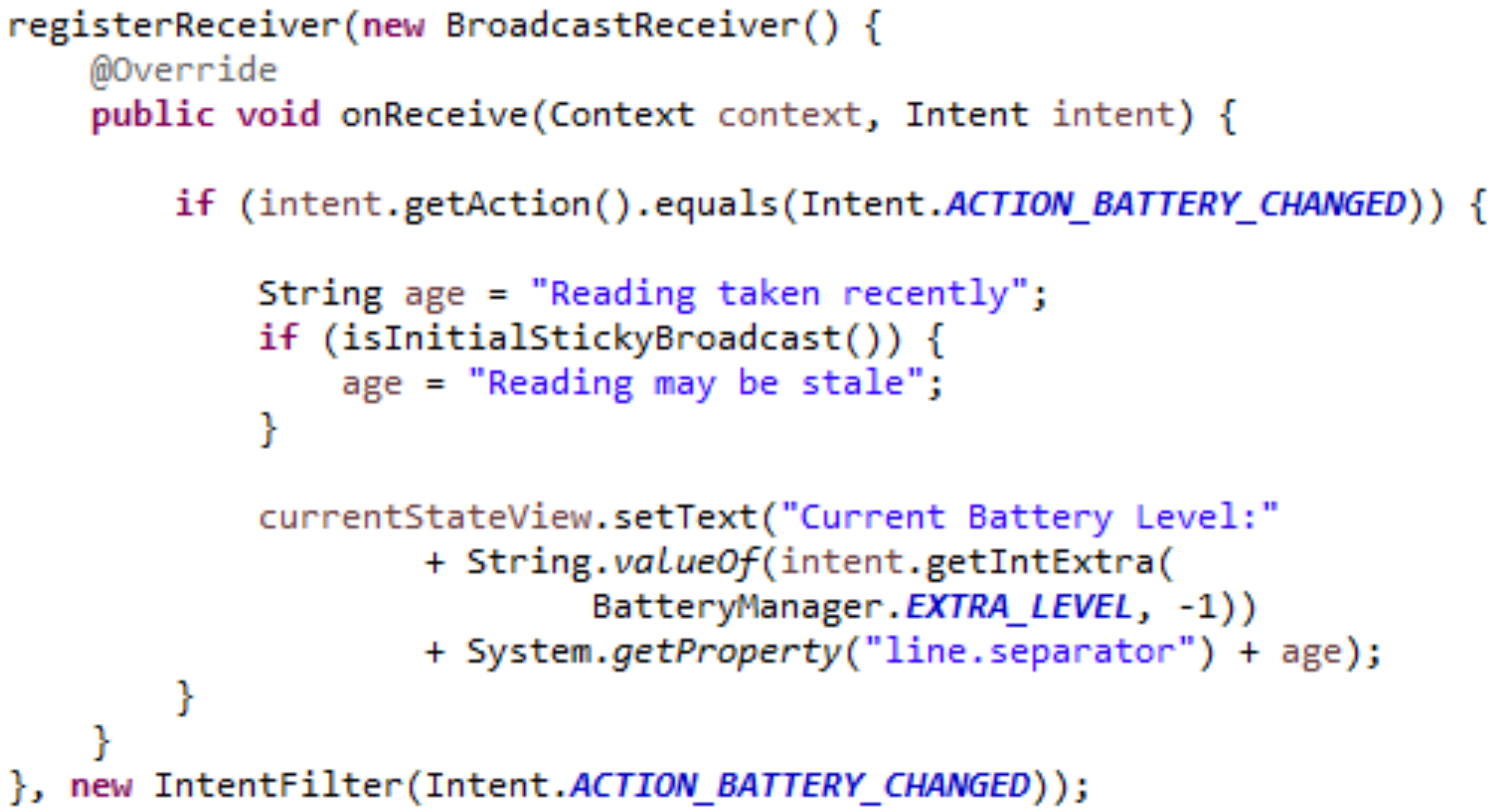
- When BroadcastReceivers are dynamically registered
  - Cached sticky Intents matching the specified IntentFilter are broadcast to the BroadcastReceiver
  - One matching sticky Intent is returned to the caller

Sticky Broadcasts

```
//public abstract class Context ...
// send sticky Intent to interested BroadcastReceivers
void sendStickyBroadcast (Intent intent)

// send sticky Intent to interested BroadcastReceivers in priority order
// sender can provide various parameters for greater control

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

Broadcaster must have BROADCAST_STICKY permission to send sticky Intents

BcastRecStickyInt



```
registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {

        if (intent.getAction().equals(Intent.ACTION_BATTERY_CHANGED)) {

            String age = "Reading taken recently";
            if (isInitialStickyBroadcast()) {
                age = "Reading may be stale";
            }

            currentStateView.setText("Current Battery Level:"
                    + String.valueOf(intent.getIntExtra(
                            BatteryManager.EXTRA_LEVEL, -1))
                    + System.getProperty("line.separator") + age);
        }
    }
}, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
```

Next Time

Threads, AsyncTasks & Handlers