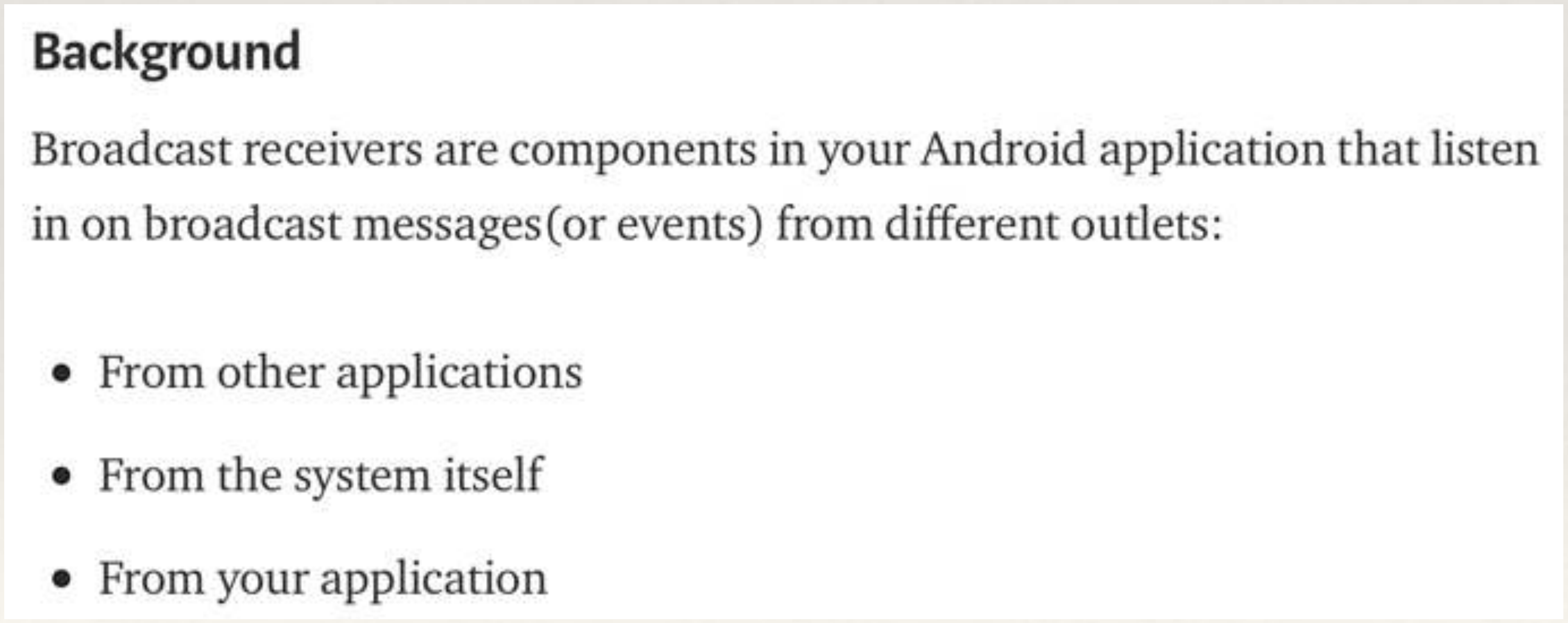
Introduction to Mobile Programming

Android Programming

Chapter 5

Broadcasts Overview

1. Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the publish-subscribe design pattern. These broadcasts are sent when an event of interest occurs.
    1. System Broadcasts
    2. Custom Broadcasts

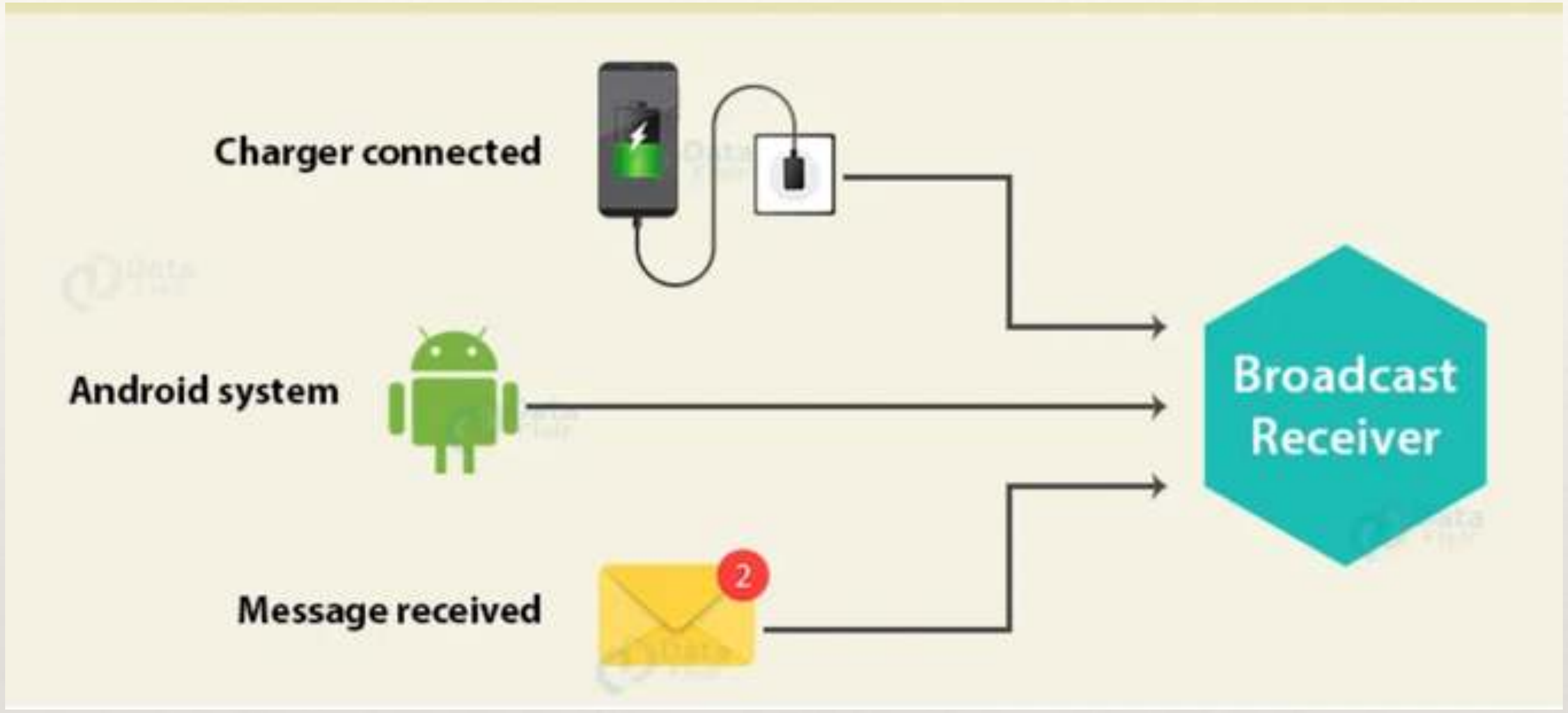


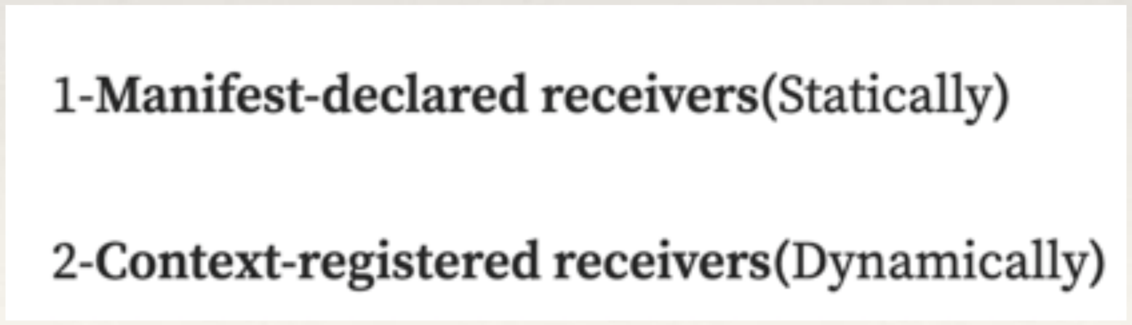
Background

Broadcast receivers are components in your Android application that listen in on broadcast messages(or events) from different outlets:

- From other applications
- From the system itself
- From your application

Broadcast Receivers




1-**Manifest-declared receivers**(Statically)

2-**Context-registered receivers**(Dynamically)

System Broadcasts

219 DIFFERENT SYSTEM BROADCASTS

1. BATTERY_CHANGED
2. BATTERY_LOW
3. BOOT_COMPLETED android.intent.action.BOOT_COMPLETED
4. ACTION_POWER_CONNECTED
5. AIRPLANE_MODE
6. SCREEN_OFF
7. SCREEN_ON
8. SMS_RECEIVED android.provider.Telephony.SMS_RECEIVED

Receiving Broadcasts / Manifest-Declared Registers

- If you declare a broadcast receiver in your manifest, the system launches your app (if the app is not already running) when the broadcast is sent
- If your app targets API level 26 or higher, you cannot use the manifest to declare a receiver for *implicit* broadcasts (broadcasts that do not target your app specifically)

```
<receiver android:name=".MyBroadcastReceiver"  android:exported="true">
   <intent-filter>
      <action android:name="android.intent.action.BOOT_COMPLETED"/>
      <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
   </intent-filter>
</receiver>
```

```
public class MyBroadcastReceiver extends BroadcastReceiver {
      private static final String TAG = "MyBroadcastReceiver";
      @Override
      public void onReceive(Context context, Intent intent) {
         StringBuilder sb = new StringBuilder();
         sb.append("Action: " + intent.getAction() + "\n");
         sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");
         String log = sb.toString();
         Log.d(TAG, log);
         Toast.makeText(context, log, Toast.LENGTH_LONG).show();
      }
   }
```

Receiving Broadcasts / Context-Registered Receiver

```
BroadcastReceiver br = new MyBroadcastReceiver();
```

```
IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
    filter.addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED);
    this.registerReceiver(br, filter);
```

```
unregisterReceiver(android.content.BroadcastReceiver)
```

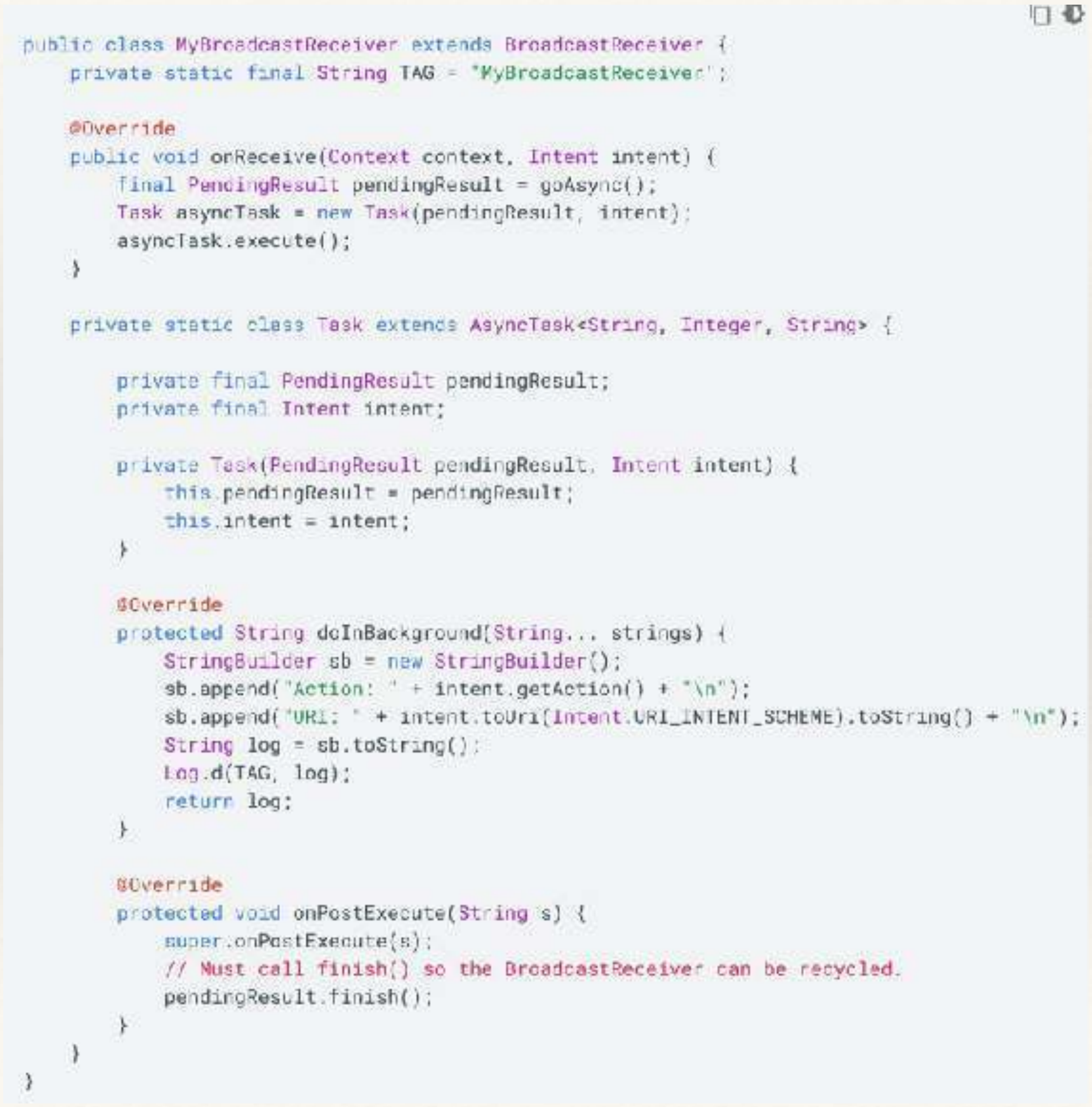
- Context-registered receivers receive broadcasts as long as their registering context is valid. For an example, if you register within an Activity context, you receive broadcasts as long as the activity is not destroyed. If you register with the Application context, you receive broadcasts as long as the app is running.

`onCreate(Bundle)` `onDestroy()`

`onResume()`, `onPause()`

`onSaveInstanceState(Bundle)`

Effects on Process State



```
public class MyBroadcastReceiver extends BroadcastReceiver {
    private static final String TAG = "MyBroadcastReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        final PendingResult pendingResult = goAsync();
        Task asyncTask = new Task(pendingResult, intent);
        asyncTask.execute();
    }

    private static class Task extends AsyncTask<String, Integer, String> {

        private final PendingResult pendingResult;
        private final Intent intent;

        private Task(PendingResult pendingResult, Intent intent) {
            this.pendingResult = pendingResult;
            this.intent = intent;
        }

        @Override
        protected String doInBackground(String... strings) {
            StringBuilder sb = new StringBuilder();
            sb.append("Action: " + intent.getAction() + "\n");
            sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");
            String log = sb.toString();
            Log.d(TAG, log);
            return log;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            // Must call finish() so the BroadcastReceiver can be recycled.
            pendingResult.finish();
        }
    }
}
```

Sending Broadcasts

1. **sendOrderedBroadcast**

   This method sends broadcasts to one receiver at a time. As each receiver executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the android:priority attribute of the matching intent-filter; receivers with the same priority will be run in an arbitrary order.

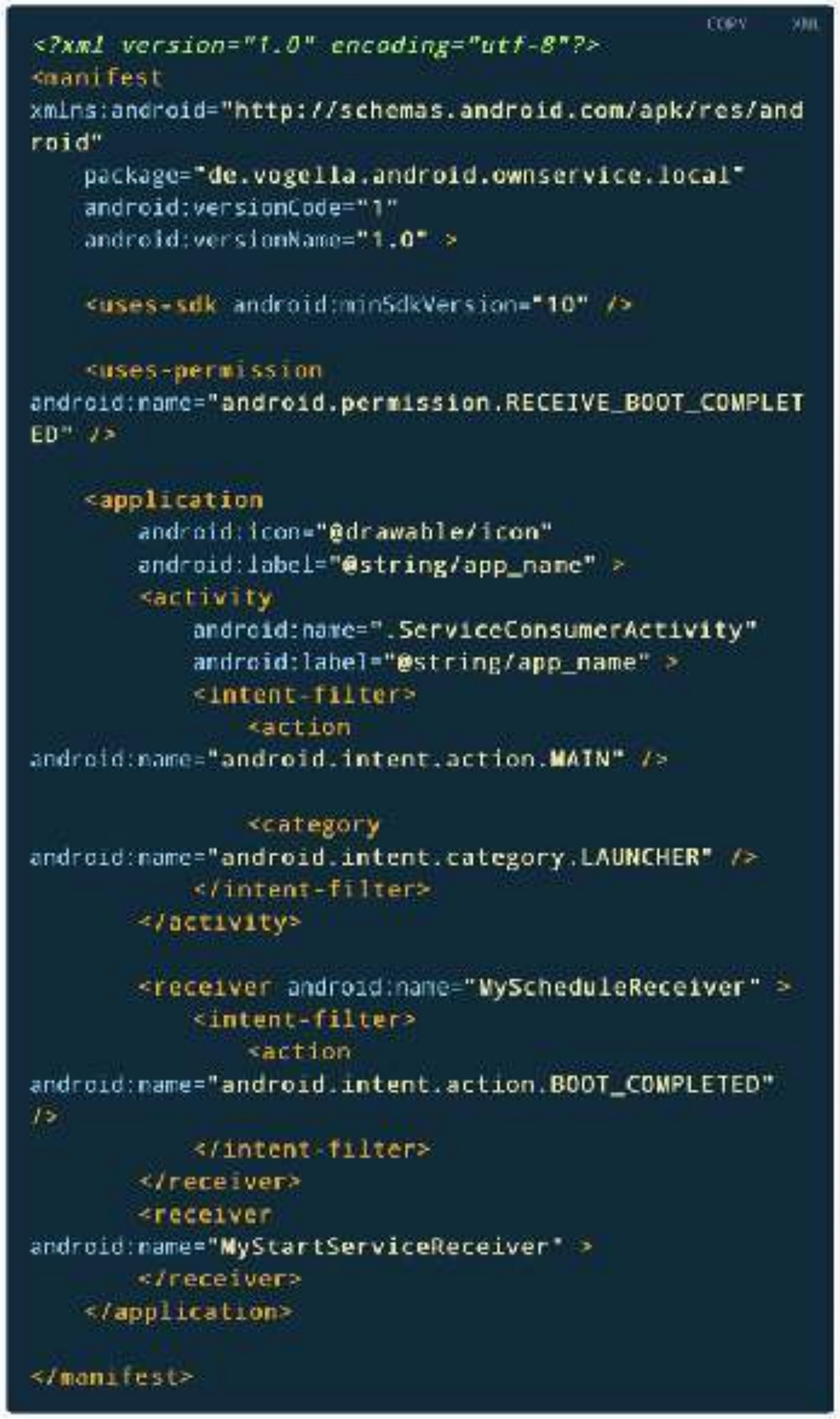
2. **sendBroadcast**

   This method sends broadcasts to all receivers in an undefined order. This is called a Normal Broadcast. This is more efficient, but means that receivers cannot read results from other receivers, propagate data received from the broadcast, or abort the broadcast.

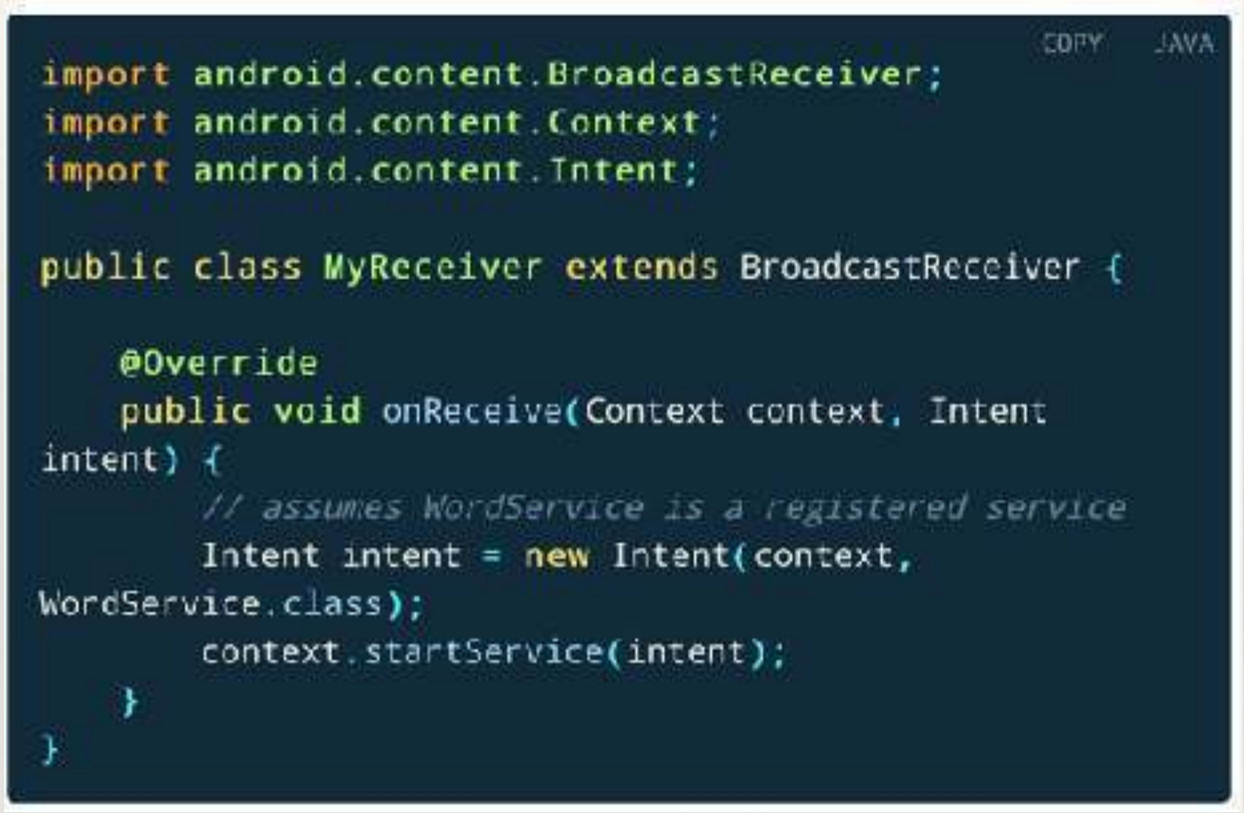
3. **LocalBroadcastManager. sendBroadcast**

   This method sends broadcasts to receivers that are in the same app as the sender. If you don't need to send broadcasts across apps, use local broadcasts. The implementation is much more efficient (no interprocess communication needed) and you don't need to worry about any security issues related to other apps being able to receive or send your broadcasts.

Starting Services from a Receiver

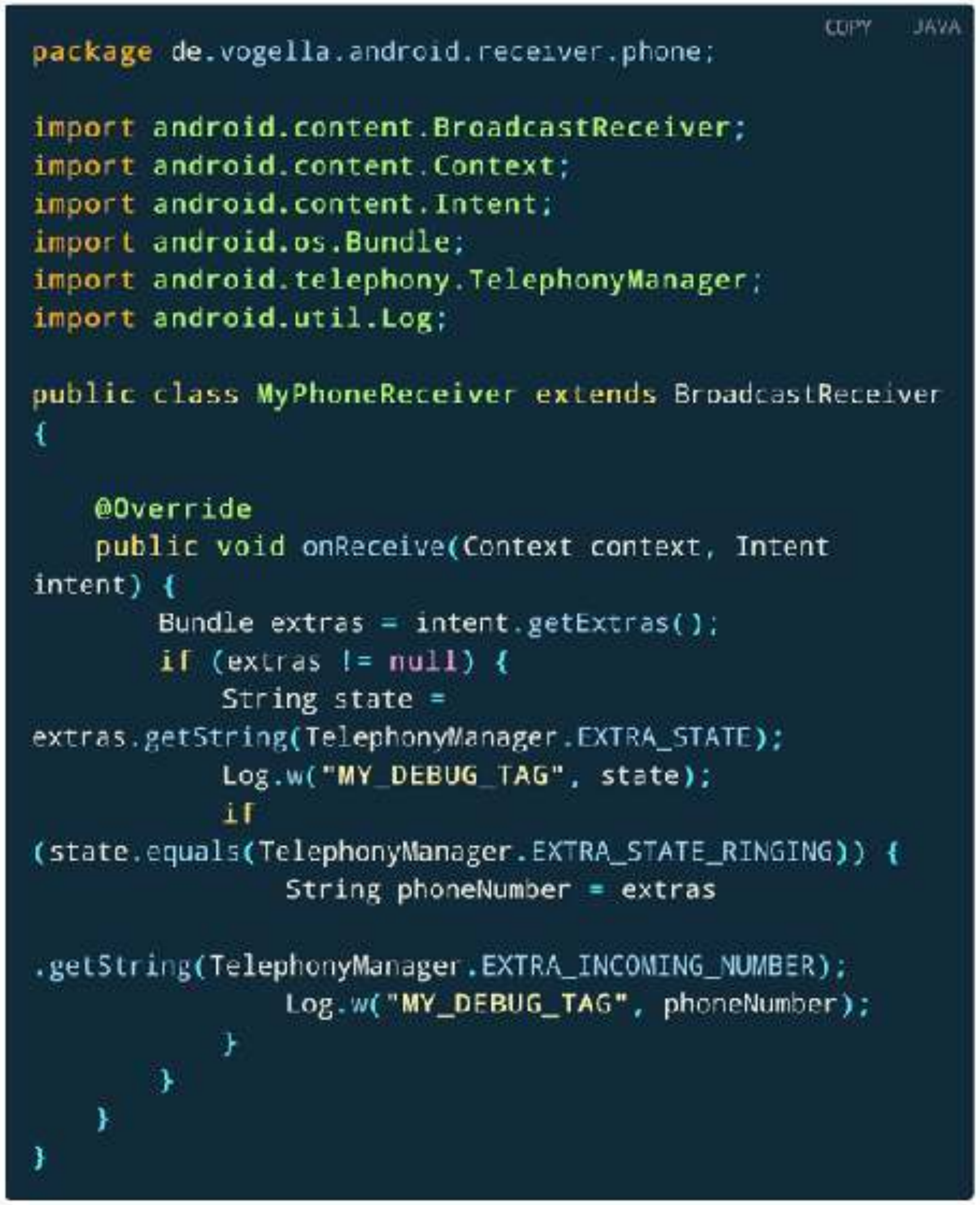

COPY XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    package="de.vogella.android.ownservice.local"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLET
ED" />

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".ServiceConsumerActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver android:name="MyScheduleReceiver" >
            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED"
/>
            </intent-filter>
        </receiver>
        <receiver
android:name="MyStartServiceReceiver" >
        </receiver>
    </application>

</manifest>
```

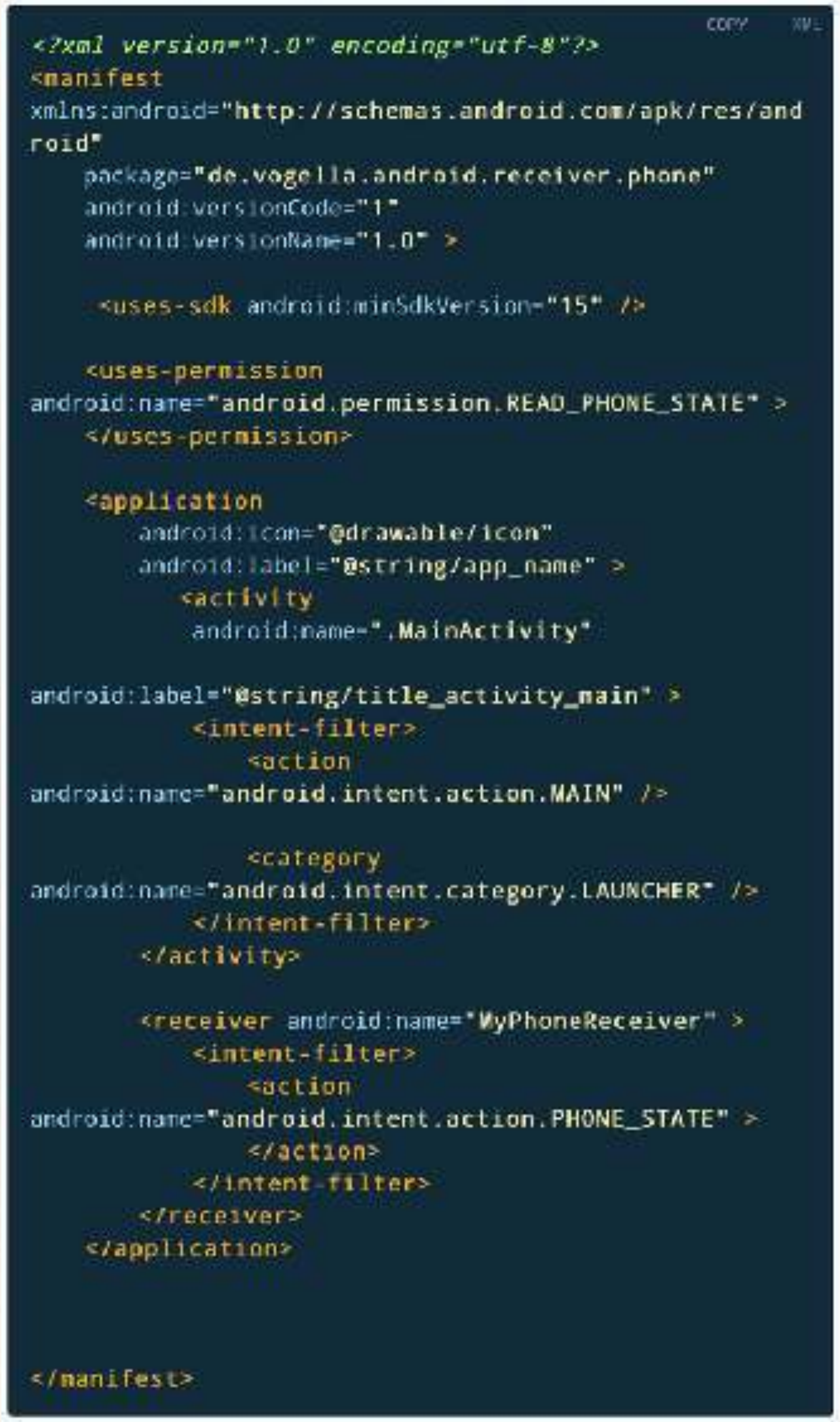

COPY JAVA

```java
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent
intent) {
        // assumes WordService is a registered service
        Intent intent = new Intent(context,
WordService.class);
        context.startService(intent);
    }
}
```

Register a Receiver for Incoming Phone Calls

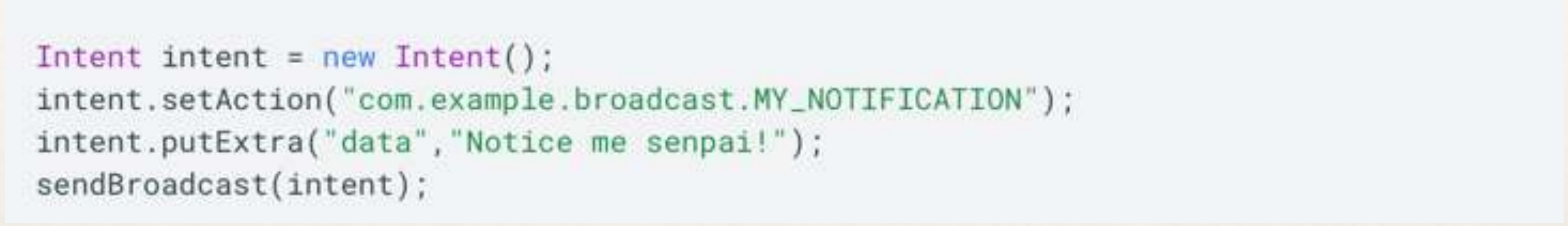


```java
package de.vogella.android.receiver.phone;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;

public class MyPhoneReceiver extends BroadcastReceiver
{

    @Override
    public void onReceive(Context context, Intent
intent) {
        Bundle extras = intent.getExtras();
        if (extras != null) {
            String state =
extras.getString(TelephonyManager.EXTRA_STATE);
            Log.w("MY_DEBUG_TAG", state);
            if
(state.equals(TelephonyManager.EXTRA_STATE_RINGING)) {
                String phoneNumber = extras

.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
                Log.w("MY_DEBUG_TAG", phoneNumber);
            }
        }
    }
}
```

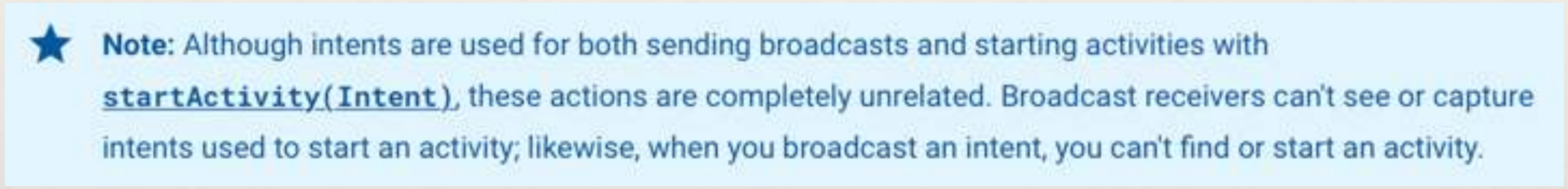


```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    package="de.vogella.android.receiver.phone"
    android:versionCode="1"
    android:versionName="1.0" >

     <uses-sdk android:minSdkVersion="15" />

    <uses-permission
android:name="android.permission.READ_PHONE_STATE" >
    </uses-permission>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
           <activity
            android:name=".MainActivity"

android:label="@string/title_activity_main" >
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver android:name="MyPhoneReceiver" >
            <intent-filter>
                <action
android:name="android.intent.action.PHONE_STATE" >
                </action>
            </intent-filter>
        </receiver>
    </application>


</manifest>
```

Sending Broadcast



```
Intent intent = new Intent();
intent.setAction("com.example.broadcast.MY_NOTIFICATION");
intent.putExtra("data","Notice me senpai!");
sendBroadcast(intent);
```



★ **Note:** Although intents are used for both sending broadcasts and starting activities with `startActivity(Intent)`, these actions are completely unrelated. Broadcast receivers can't see or capture intents used to start an activity; likewise, when you broadcast an intent, you can't find or start an activity.

Restricting Broadcasts with Permissions

```
sendBroadcast(new Intent("com.example.NOTIFY"),
              Manifest.permission.SEND_SMS);
```

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

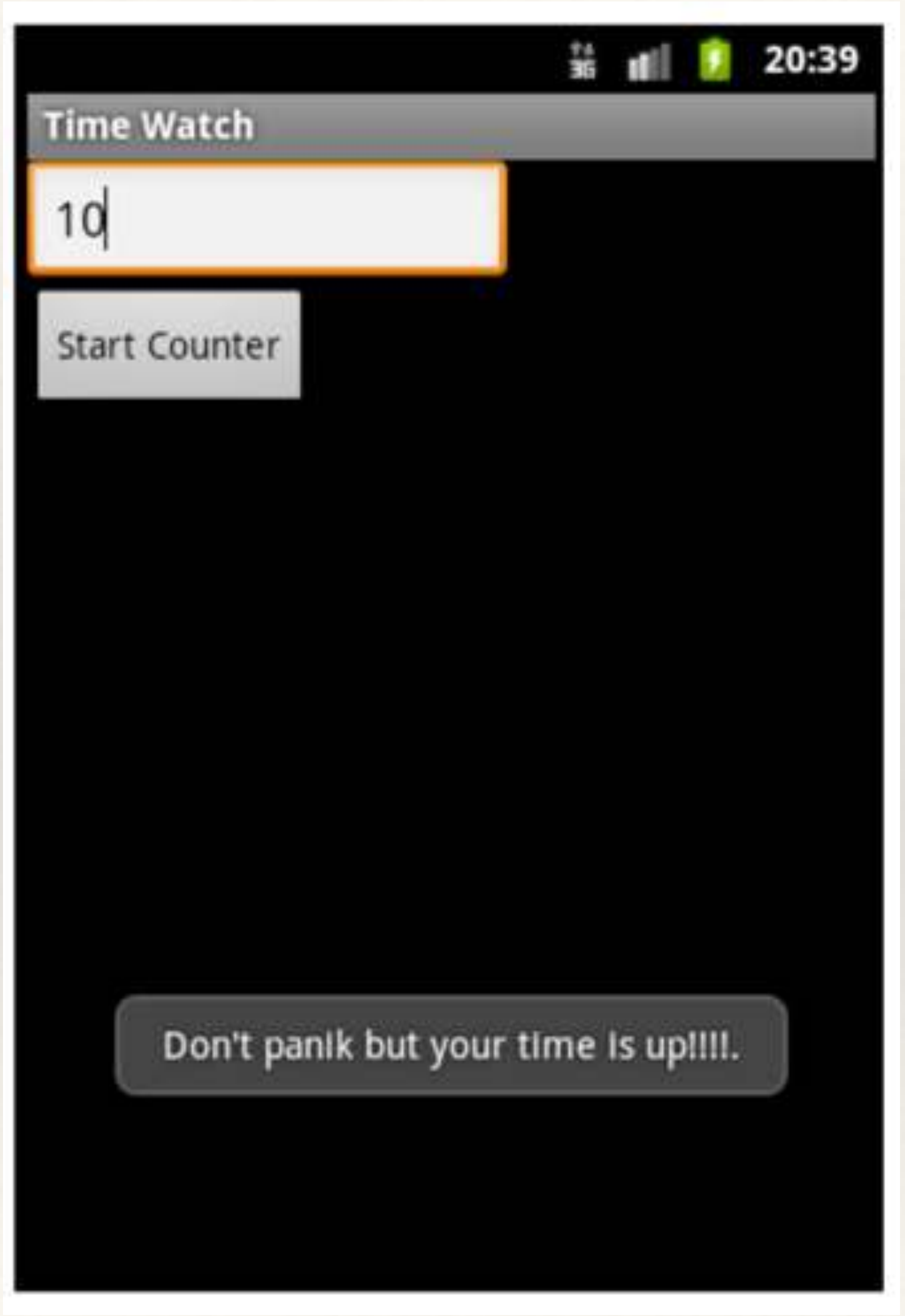
Sending with permissions

```
<receiver android:name=".MyBroadcastReceiver"
          android:permission="android.permission.SEND_SMS">
    <intent-filter>
        <action android:name="android.intent.action.AIRPLANE_MODE"/>
    </intent-filter>
</receiver>
```

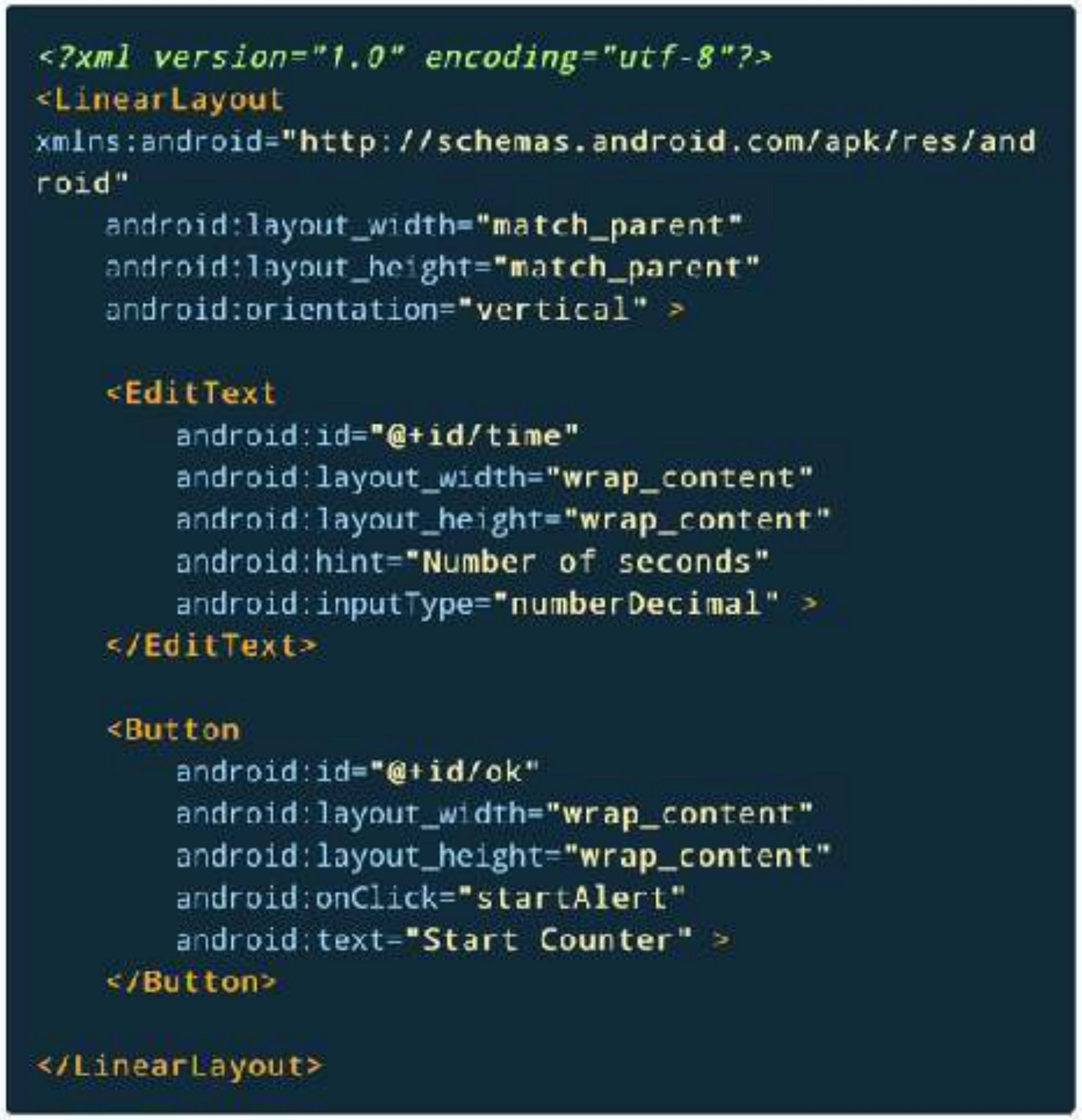
Receiving with permissions

```
IntentFilter filter = new IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED);
registerReceiver(receiver, filter, Manifest.permission.SEND_SMS, null );
```

Broadcast Receiver Exercise

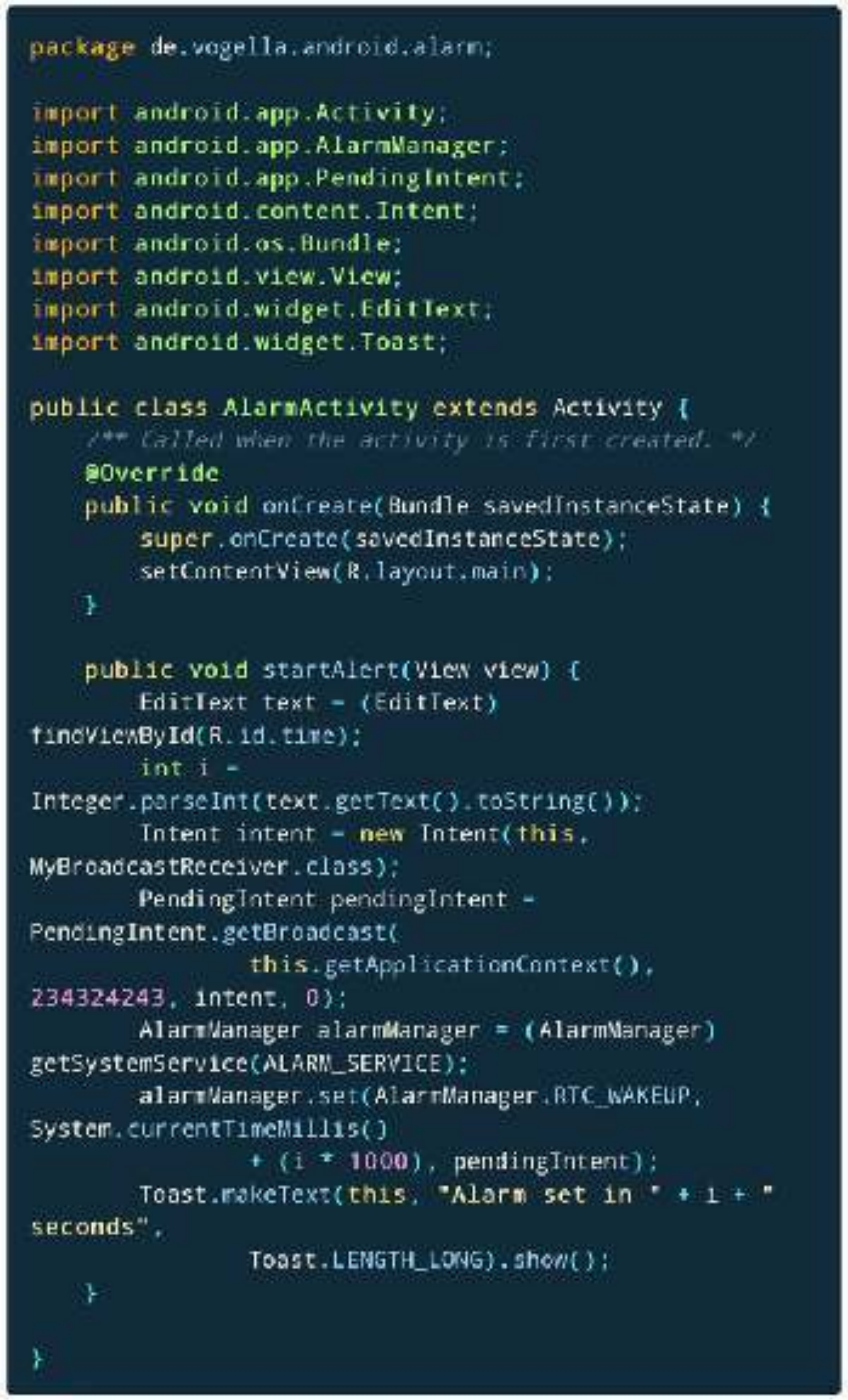


Don't panik but your time is up!!!!.



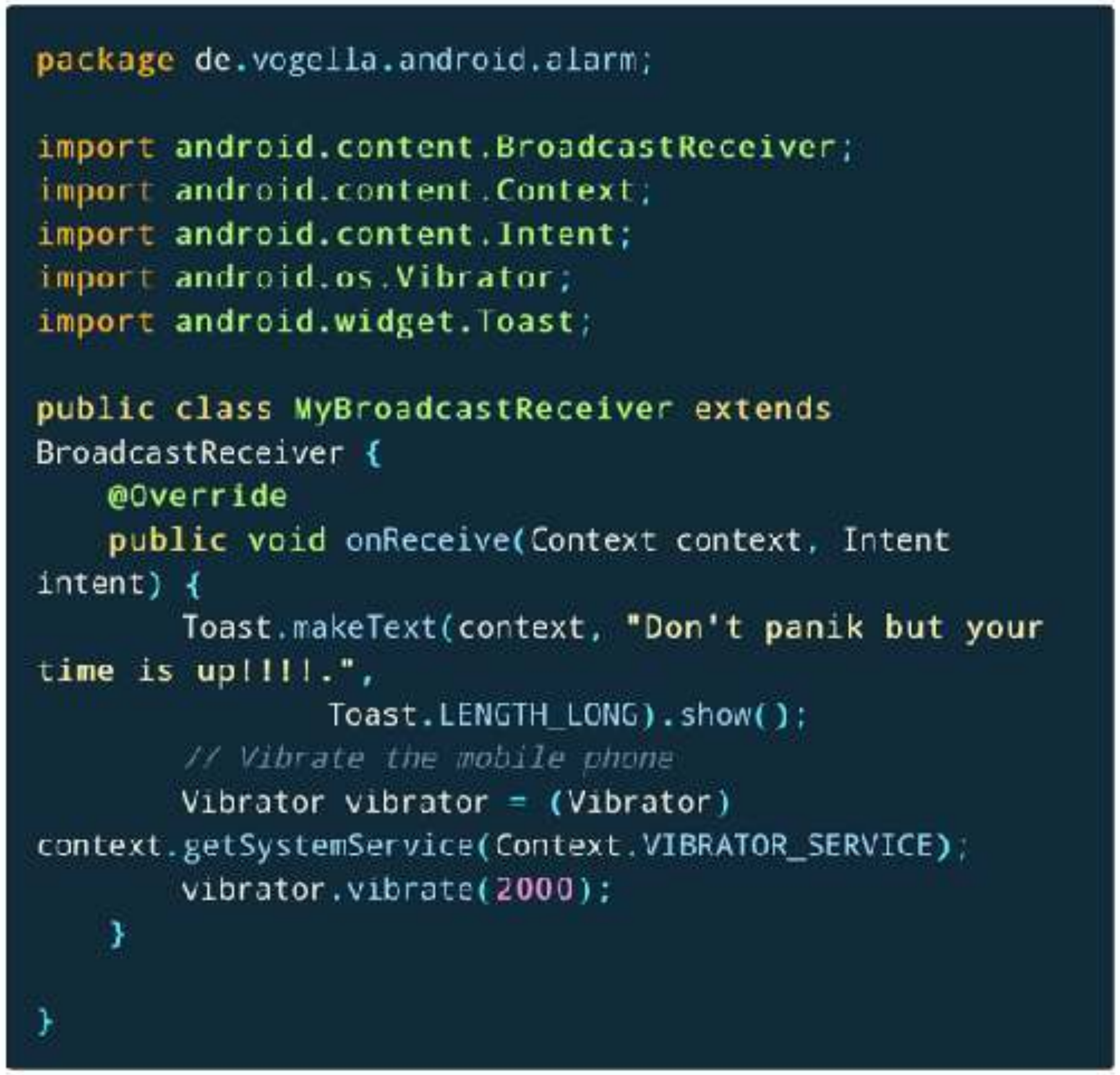
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <EditText
        android:id="@+id/time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Number of seconds"
        android:inputType="numberDecimal" >
    </EditText>

    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="startAlert"
        android:text="Start Counter" >
    </Button>

</LinearLayout>
```

Broadcast Receiver Exercise

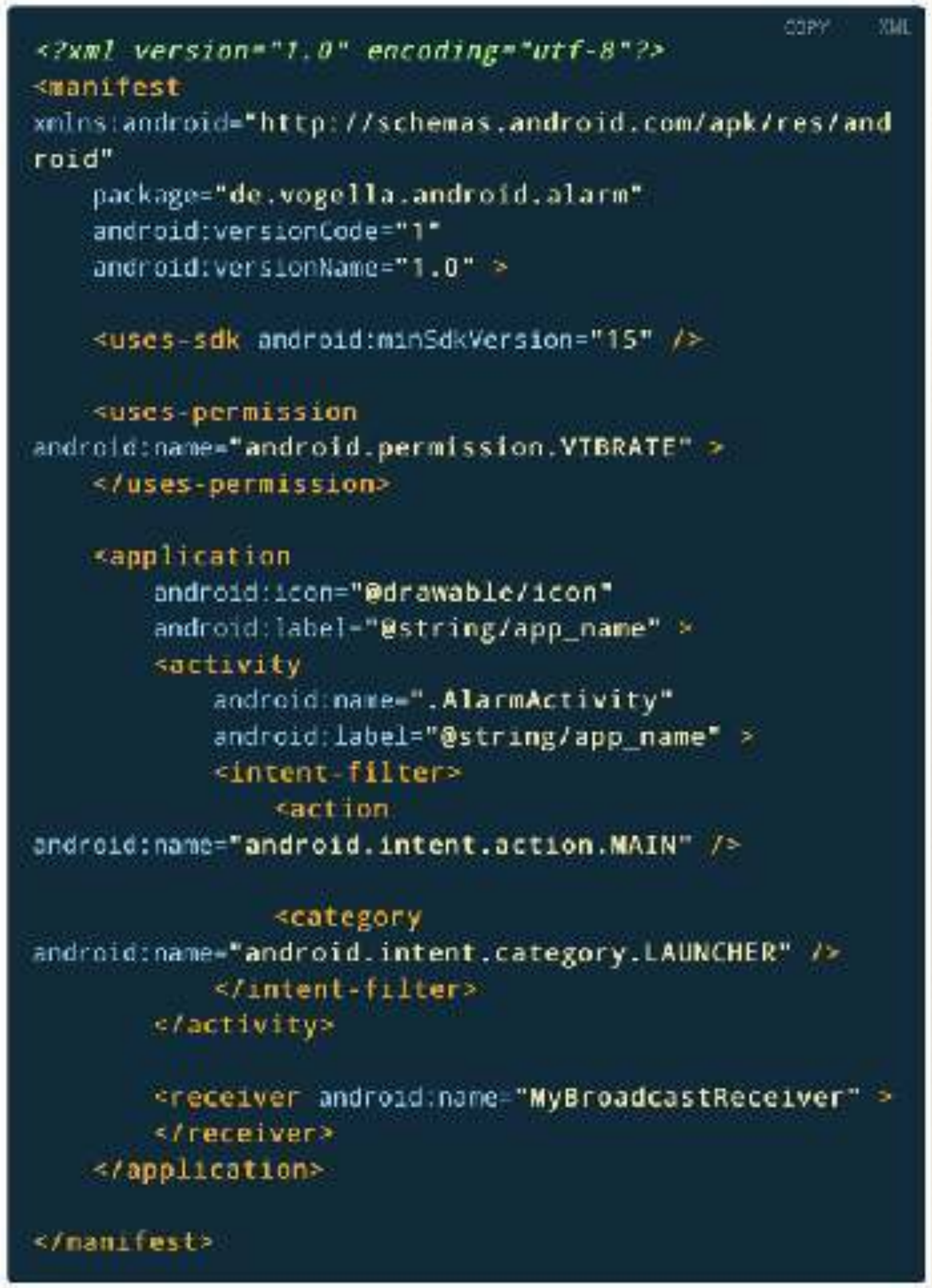


```
package de.vogella.android.alarm;

import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class AlarmActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void startAlert(View view) {
        EditText text = (EditText)
findViewById(R.id.time);
        int i =
Integer.parseInt(text.getText().toString());
        Intent intent = new Intent(this,
MyBroadcastReceiver.class);
        PendingIntent pendingIntent =
PendingIntent.getBroadcast(
                this.getApplicationContext(),
234324243, intent, 0);
        AlarmManager alarmManager = (AlarmManager)
getSystemService(ALARM_SERVICE);
        alarmManager.set(AlarmManager.RTC_WAKEUP,
System.currentTimeMillis()
                + (i * 1000), pendingIntent);
        Toast.makeText(this, "Alarm set in " + i + "
seconds",
                Toast.LENGTH_LONG).show();
    }

}
```

Broadcast Receiver Exercise

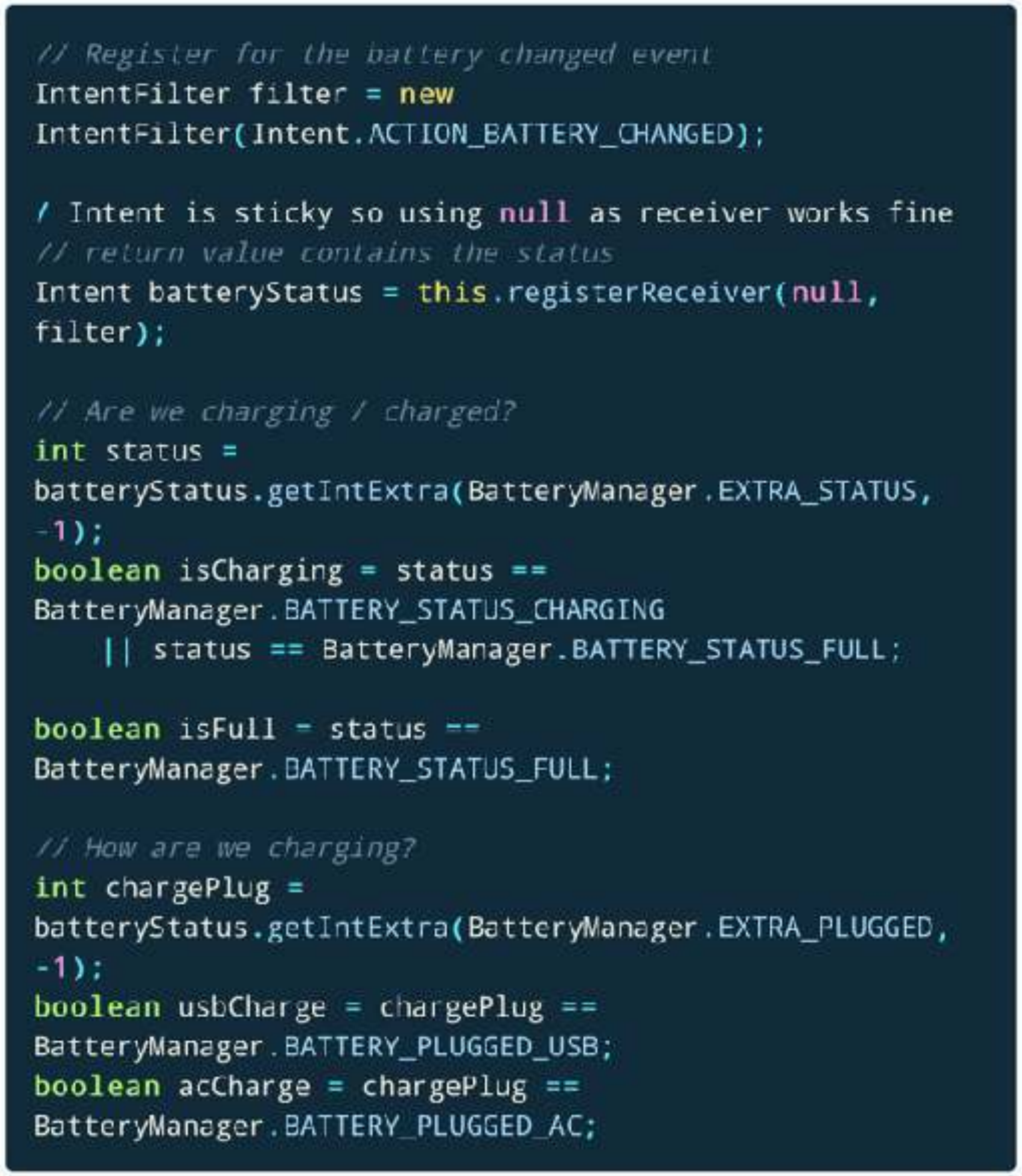


```java
package de.vogella.android.alarm;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Vibrator;
import android.widget.Toast;

public class MyBroadcastReceiver extends
BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent
intent) {
        Toast.makeText(context, "Don't panik but your
time is up!!!!.",
                Toast.LENGTH_LONG).show();
        // Vibrate the mobile phone
        Vibrator vibrator = (Vibrator)
context.getSystemService(Context.VIBRATOR_SERVICE);
        vibrator.vibrate(2000);
    }

}
```



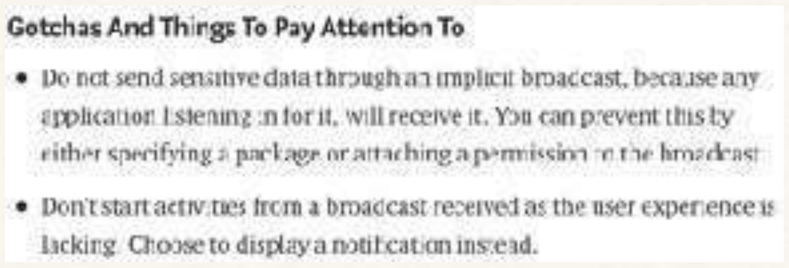
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    package="de.vogella.android.alarm"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="15" />

    <uses-permission
android:name="android.permission.VTBRATE" >
    </uses-permission>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".AlarmActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver android:name="MyBroadcastReceiver" >
        </receiver>
    </application>

</manifest>
```

Sticky (Broadcast) Intents



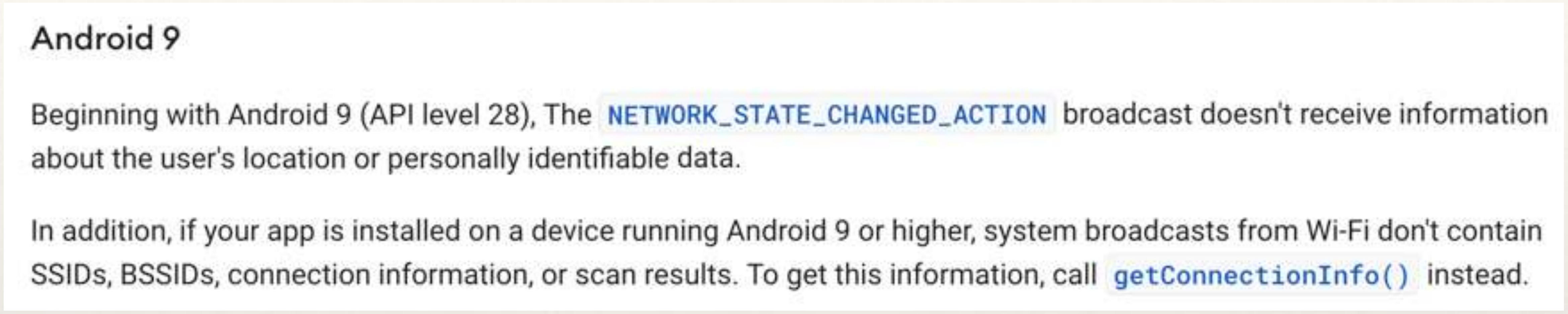
```
// Register for the battery changed event
IntentFilter filter = new
IntentFilter(Intent.ACTION_BATTERY_CHANGED);

/ Intent is sticky so using null as receiver works fine
// return value contains the status
Intent batteryStatus = this.registerReceiver(null,
filter);

// Are we charging / charged?
int status =
batteryStatus.getIntExtra(BatteryManager.EXTRA_STATUS,
-1);
boolean isCharging = status ==
BatteryManager.BATTERY_STATUS_CHARGING
    || status == BatteryManager.BATTERY_STATUS_FULL;

boolean isFull = status ==
BatteryManager.BATTERY_STATUS_FULL;

// How are we charging?
int chargePlug =
batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED,
-1);
boolean usbCharge = chargePlug ==
BatteryManager.BATTERY_PLUGGED_USB;
boolean acCharge = chargePlug ==
BatteryManager.BATTERY_PLUGGED_AC;
```

Important Notes



Gotchas And Things To Pay Attention To

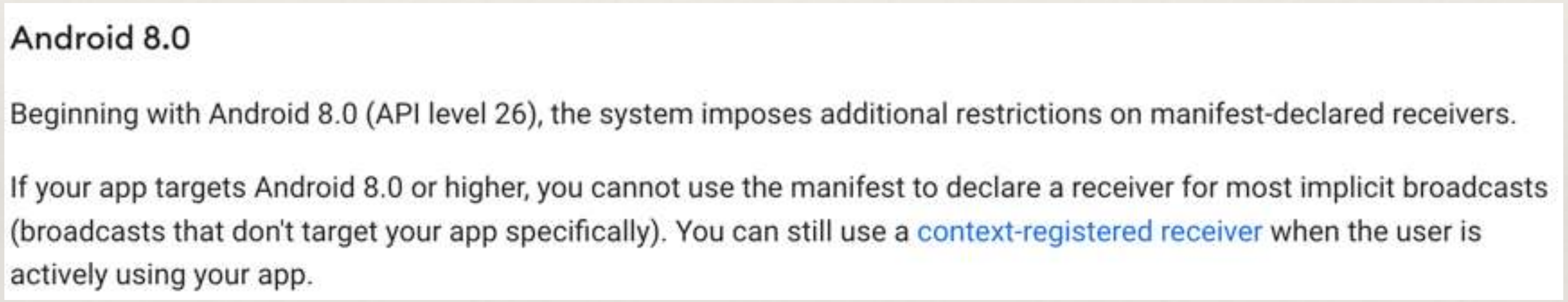
- Do not send sensitive data through an implicit broadcast, because any application listening in for it, will receive it. You can prevent this by either specifying a package or attaching a permission to the broadcast
- Don't start activities from a broadcast received as the user experience is lacking. Choose to display a notification instead.



Android 9

Beginning with Android 9 (API level 28), The `NETWORK_STATE_CHANGED_ACTION` broadcast doesn't receive information about the user's location or personally identifiable data.

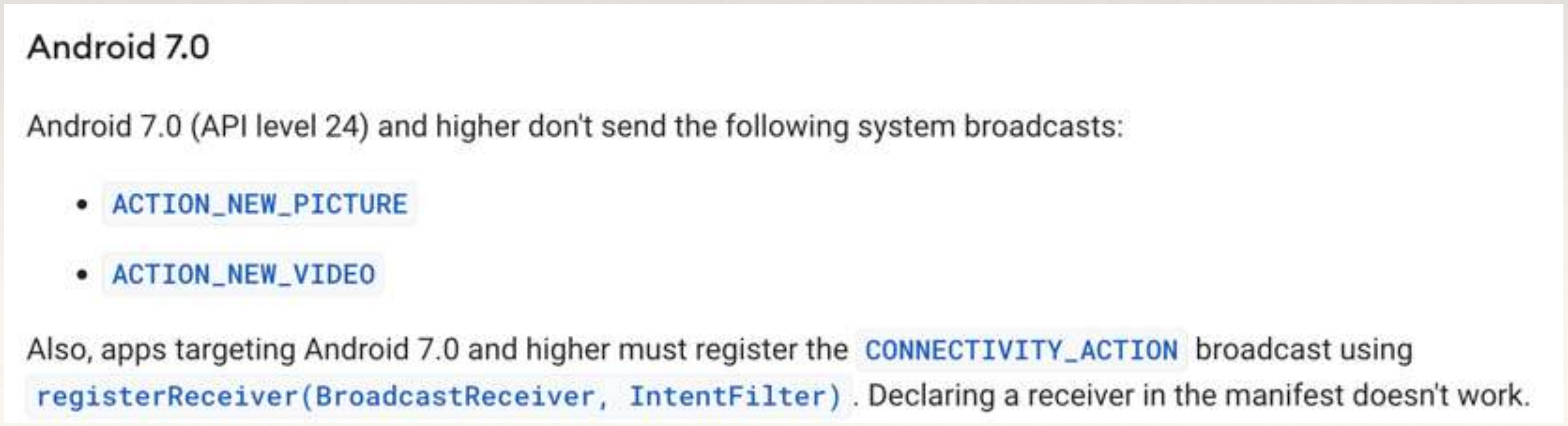
In addition, if your app is installed on a device running Android 9 or higher, system broadcasts from Wi-Fi don't contain SSIDs, BSSIDs, connection information, or scan results. To get this information, call `getConnectionInfo()` instead.



Android 8.0

Beginning with Android 8.0 (API level 26), the system imposes additional restrictions on manifest-declared receivers.

If your app targets Android 8.0 or higher, you cannot use the manifest to declare a receiver for most implicit broadcasts (broadcasts that don't target your app specifically). You can still use a context-registered receiver when the user is actively using your app.



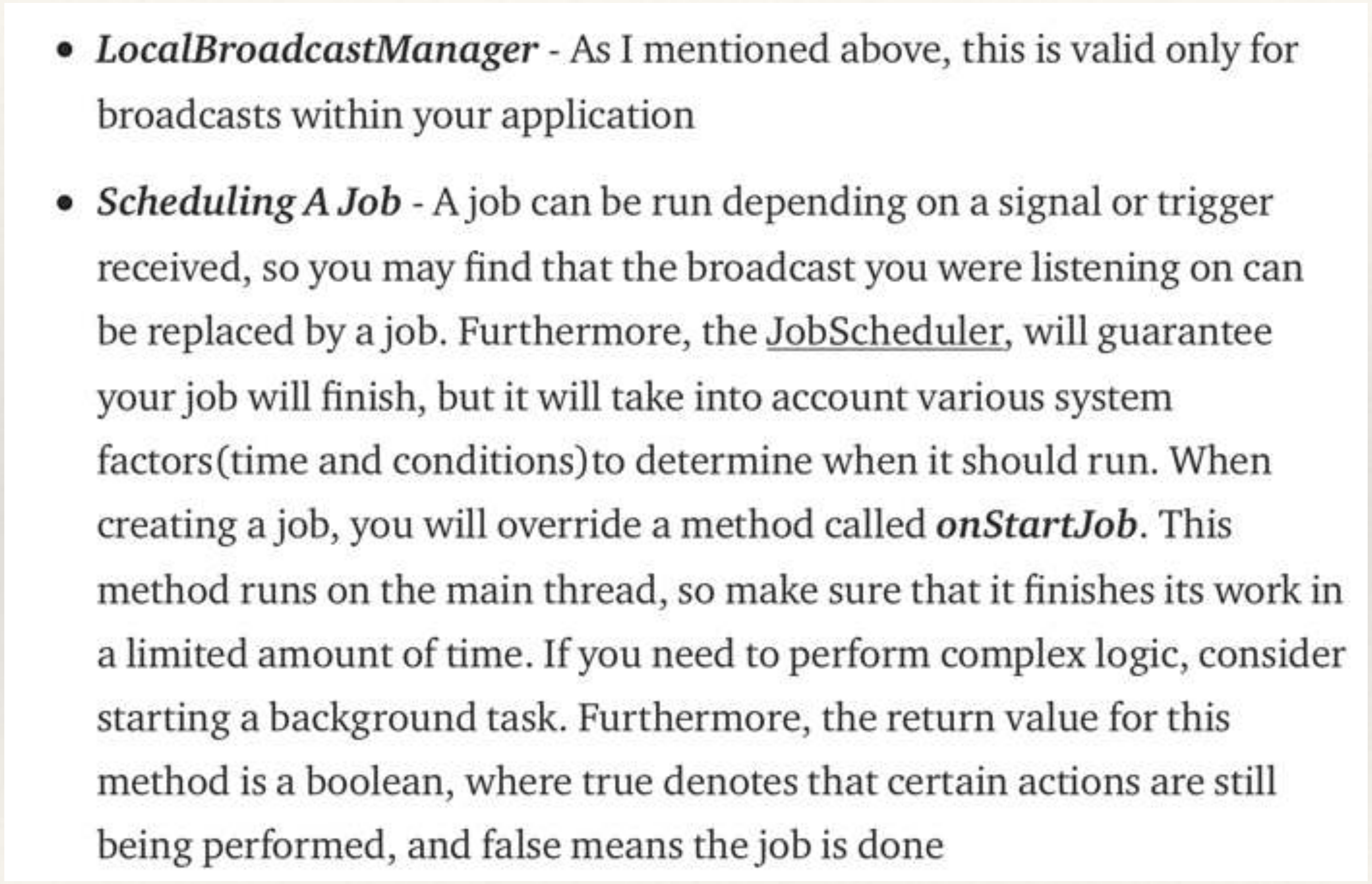
Android 7.0

Android 7.0 (API level 24) and higher don't send the following system broadcasts:

- `ACTION_NEW_PICTURE`
- `ACTION_NEW_VIDEO`

Also, apps targeting Android 7.0 and higher must register the `CONNECTIVITY_ACTION` broadcast using `registerReceiver(BroadcastReceiver, IntentFilter)`. Declaring a receiver in the manifest doesn't work.

Alternatives to Broadcast Receivers



- ***LocalBroadcastManager*** - As I mentioned above, this is valid only for broadcasts within your application
- ***Scheduling A Job*** - A job can be run depending on a signal or trigger received, so you may find that the broadcast you were listening on can be replaced by a job. Furthermore, the JobScheduler, will guarantee your job will finish, but it will take into account various system factors(time and conditions)to determine when it should run. When creating a job, you will override a method called ***onStartJob***. This method runs on the main thread, so make sure that it finishes its work in a limited amount of time. If you need to perform complex logic, consider starting a background task. Furthermore, the return value for this method is a boolean, where true denotes that certain actions are still being performed, and false means the job is done