EMBEDDED SYSTEMS PROGRAMMING 2014-15

Android Broadcast Receivers

APP COMPONENTS

- **Activity**: a single screen with a user interface
- **Broadcast receiver**: responds to system-wide broadcast events. No user interface
- **Service**: performs (in the background) long-running operations (e.g., music playback). No user interface
- **Content provider**

BROADCAST RECEIVERS (1/3)

- Respond to system-wide broadcast announcements
- Handled via the **`BroadcastReceiver`** abstract class, plus the `Intent` class (used to send/receive broadcasts)
- A broadcast receiver must be registered either
  - statically, through the **`<receiver>`** tag in `AndroidManifest.xml`, or
  - dynamically, by invoking the **`registerReceiver(BroadcastReceiver receiver, IntentFilter filter)`** method of the `Context` class

BROADCAST RECEIVERS (2/3)

- Many broadcasts originate **from the system** —
  for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured

- **Apps can also broadcast intents to other components or other apps** —
  for example, to let such parties know that some data has been downloaded and is available for them to use

SOME SYSTEM ACTIONS (1/3)

- `Intent.ACTION_AIRPLANE_MODE_CHANGED`
  The user has switched the phone into or out of "airplane mode"
- `Intent.ACTION_CONFIGURATION_CHANGED`
  Device configuration (orientation, locale, etc) has changed
- `Intent.ACTION_DATE_CHANGED`, `Intent.ACTION_TIME_CHANGED`
  The date/time has changed
- `Intent.ACTION_INPUT_METHOD_CHANGED`
  An input method has been changed
- `Intent.ACTION_LOCALE_CHANGED`
  The current device's locale has changed
- `Intent.ACTION_PACKAGE_CHANGED`
  An existing application package has been changed
  (e.g. a component has been enabled or disabled)

SOME SYSTEM ACTIONS (2/3)

- `Intent.ACTION_BOOT_COMPLETED`
  Broadcast once after the system has finished booting
- `Intent.ACTION_CAMERA_BUTTON`
  The camera button was pressed
- `Intent.ACTION_DEVICE_STORAGE_LOW`
  `Intent.ACTION_DEVICE_STORAGE_OK`
  Indicates low memory condition on the device begins / no longer exists
- `Intent.ACTION_SCREEN_OFF`
  `Intent.ACTION_SCREEN_ON`
  The device has gone to / exits from non-interactive mode
- Battery-related and power-related actions defined in the Intent class (already discussed)

SOME SYSTEM ACTIONS (3/3)

- **`Camera.ACTION_NEW_PICTURE`**
  `Camera.ACTION_NEW_VIDEO`
  A new picture/video has been taken by the camera,
  and it has been added to the media store
- **`AudioManager.ACTION_AUDIO_BECOMING_NOISY`**
  Audio is about to become "noisy" due to a change in audio outputs (e.g., a wired headset has been unplugged)
- **`ConnectivityManager.CONNECTIVITY_ACTION`**
  A change in network connectivity has occurred:
  a default connection has either been established or lost

USING A BROADCAST RECEIVER

1. Implement the receiver as a subclass of `BroadcastReceiver`
2. Register the receiver
3. When a matching intent is broadcast, the **`onReceive(Context context, Intent intent)`** method of the receiver is invoked even if the receiver is contained in a stopped process
4. When `onReceive()` returns, the receiver object is no longer active, and the process may be stopped

BROADCAST RECEIVERS (3/3)

- A `BroadcastReceiver` object is only valid for the duration of the call to `onReceive()`
- `onReceive()` is given **10 seconds to complete execution**: after that, the receiver is considered “blocked” and it may be killed
- Consequently, a broadcast receiver cannot perform asynchronous or long-running operations, even binding to a service (however, it can invoke `startService()`)
- A broadcast receiver cannot display a user interface (however, it may create a status bar notification)

EXAMPLE (1/3)

- Implementing a broadcast receiver

```
public class MyReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        String action = intent.getAction();
        Log.i(TAG, "Received broadcast action: " + action);

        // Perform some useful work here
        // (after having further examined the intent, if necessary)

        …

    }
}
```

EXAMPLE (2/3)

- Registering `MyReceiver` in the manifest: the app receives all intents since the device is started

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="it.unipd.dei.es1011.brtest"
          android:versionCode="1"
          android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <receiver android:name=".MyReceiver" android:enabled="true">
          <intent-filter>
              <action android:name="android.intent.action.PHONE_STATE"></action>
          </intent-filter>
      </receiver>

…

    </application>
    <uses-sdk android:minSdkVersion="8" />

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

EXAMPLE (3/3)

- Registering `MyReceiver` dynamically from within an activity: the app receives intents only when the activity is in the foreground

```
…

private receiverInstance = new MyReceiver();

@Override
public void onResume()
{
    super.onResume();
    // Register the instance of MyReceiver
    this.registerReceiver(receiverInstance,
                          new IntentFilter(TelephonyManager.ACTION_PHONE_STATE_CHANGED);
}

@Override
protected void onPause()
{
    // Unregister since the activity is not visible.
    // Do not unregister in onSaveInstanceState()!
    this.unregisterReceiver(receiverInstance);
    super.onPause();
}

…
```

SENDING BROADCAST INTENTS

- Broadcast intents can be sent by invoking the `Context.sendBroadcast(Intent intent)` method
- Call returns immediately while the intent is distributed to all interested (i.e., previously registered) broadcast receivers
- No results are propagated from receivers
- Both system-defined and custom actions can be sent. However, remember that **some system-defined actions are protected** and can be sent only by the system itself

LOCALBROADCASTMANAGER CLASS

- Helper class to broadcast intents only to local objects within your process
- Obtain an instance by invoking the static method
  `LocalBroadcastManager.getInstance(Context context)`
- No IPC: more efficient than sending a global broadcast
- Broadcast data do not leave the app:
  no need to worry about leaking private data
- Other apps cannot send broadcasts to locally-registered objects:
  no need to worry about security holes that such apps can exploit

EXAMPLE (1/2)

- Dynamically registering `MyReceiver` with `LocalBroadcastManager`: only local broadcasts will be received

```
…

private receiverInstance = new MyReceiver();

@Override
public void onResume()
{
    super.onResume();
    // Register the instance of MyReceiver
    LocalBroadcastManager.getInstance(this).registerReceiver(receiverInstance,
                                 new IntentFilter(TelephonyManager.ACTION_PHONE_STATE_CHANGED);
}

@Override
protected void onPause()
{
    // Unregister since the activity is not visible.
    // Do not unregister in onSaveInstanceState()!
    LocalBroadcastManager.getInstance(this).unregisterReceiver(receiverInstance);
    super.onPause();
}

…
```

EXAMPLE (2/2)

- Sending a broadcast intent with `LocalBroadcastManager`: the broadcast will be limited to registered, local objects

```
private void sendAction()
{
    Intent intent = new Intent("foo-event");

    // Add some extra data
    intent.putExtra("message", "data");

    LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
}
```