Broadcast?

Android broadcast is used to announce an event such as time-zone changes, data-connection status, incoming SMS messages, or phone calls. And we can use this idea to add functionality to our programs.

Using Intents to Broadcast Events

As a system-level message-passing mechanism, Intents are capable of sending structured messages across process boundaries.

Intents can also be used to broadcast messages anonymously between components via the `sendBroadcast` method. You can implement Broadcast Receivers to listen for, and respond to, these broadcast Intents within your applications.

Broadcast Intents are used to notify listeners of system or application events, extending the event-driven programming model between applications.

Listening for Broadcasts with Broadcast Receivers

Broadcast Receivers are used to listen for broadcast Intents. For a Broadcast Receiver to be enabled it needs to be registered, either in code or within the application manifest. (What difference does it make?) When registering a Broadcast Receiver you must use an Intent Filter to specify which Intents it is listening for.
Registering a Broadcast Receiver in XML

Broadcast Receivers registered this way are always active, and will receive broadcast Intents even when the application has been killed or hasn’t been started.

```xml
<receiver android:name=".MyReciver" android:enabled="true">
  <intent-filter android:priority="1000">
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
  </intent-filter>
</receiver>
```

Registering Broadcast Receivers in Code

You can also register Broadcast Receivers in code. A receiver registered programmatically will respond to broadcast Intents only when the application component it is registered within is running.

```java
IntentFilter filter = new IntentFilter();
filter.addAction("android.provider.Telephony.SMS_RECEIVED");
filter.setPriority(1000);
receiver = new MyReciver();
registerReceiver(receiver, filter);
```

Native Android Broadcast Actions

The following list introduces some of the native actions exposed as constants in the Intent class; these actions are used primarily to track device status changes.

➤ ACTION_BOOT_COMPLETED Fired once when the device has completed its startup sequence. An application requires the RECEIVE_BOOT_COMPLETED permission to receive this broadcast.

➤ ACTION_DATE_CHANGED and ACTION_TIME_CHANGED These actions are broadcast if the date or time on the device is manually changed.

Other examples can be found in page 162
Introducing adapters and ListViews

Sometimes we need to create a list—which is common in mobile world. To do so we need to use ListView and populate it using ArrayAdapters.
This is simple:

- **First we must have an array source, normal JAVA array 😊.**

  ```java
  final ArrayList<String> mySourceArray = new ArrayList<String>();
  mySourceArray.add("Ahmed");
  mySourceArray.add("Osama");
  ```

- **Second we should create an ArrayAdapter and pass the source array to it.**

  ```java
  int LayoutID = android.R.layout.simple_list_item_1;
  final ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,LayoutID,mySourceArray);
  ```

- **At last we have to make the List view point at the adapter.**

  ```java
  ListView myDestinationListView = (ListView) findViewById(R.id.mylistview);
  myDestinationListView.setAdapter(adapter);
  ```

- **And we are done 😊.**

  if we need to add something we add it to the source array and notify the list its source data array has been changed

  ```java
  mySourceArray.add("Ma7moud");
  adapter.notifyDataSetChanged();
  ```

---

**Lab Work 2**

- Implement to do list program.

---

**Homework**

- Later