Signal System

Lets learn about broadcast receivers and how to broadcast a messages from anywhere to our GUI thread
What is Broadcast receiver?

- Android component
- Broadcast messages from other applications or from the system itself

```java
public class MyConnectivityBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
    }
}
```
Device Connectivity

- Determine if a device is connected:

```java
ConnectivityManager cm =
    (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
boolean isConnected = activeNetwork != null &&
    activeNetwork.isConnectedOrConnecting();

boolean isWiFi = activeNetwork.getType() == ConnectivityManager.TYPE_WIFI;
```

- Type of connection:
public class MyConnectivityBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        ConnectivityManager cm =
            (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();

        boolean isConnected =
            activeNetwork != null && activeNetwork.isConnectedOrConnecting();

        Log.d("MY_TAG", "Is connected: " + isConnected);
    }
}
Broadcast example (manifest):

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

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <receiver android:name=".receivers.MyConnectivityBroadcastReceiver">
        <intent-filter>
            <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
        </intent-filter>
    </receiver>

</application>
```
### (Some) More triggers

<table>
<thead>
<tr>
<th>Event Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>android.intent.action.BATTERY_CHANGED</td>
<td>Sticky broadcast containing the charging state, level, and other information about the battery.</td>
</tr>
<tr>
<td>android.intent.action.BATTERY_LOW</td>
<td>Indicates low battery condition on the device.</td>
</tr>
<tr>
<td>android.intent.action.BATTERY_OKAY</td>
<td>Indicates the battery is now okay after being low.</td>
</tr>
<tr>
<td>android.intent.action.BOOT_COMPLETED</td>
<td>This is broadcast once, after the system has finished booting.</td>
</tr>
<tr>
<td>android.intent.action.BUG_REPORT</td>
<td>Show activity for reporting a bug.</td>
</tr>
<tr>
<td>android.intent.action.CALL</td>
<td>Perform a call to someone specified by the data.</td>
</tr>
<tr>
<td>android.intent.action.CALL_BUTTON</td>
<td>The user pressed the &quot;call&quot; button to go to the dialer or other appropriate UI for placing a call.</td>
</tr>
<tr>
<td>android.intent.action.DATE_CHANGED</td>
<td>The date has changed.</td>
</tr>
<tr>
<td>android.intent.action.REBOOT</td>
<td>Have the device reboot.</td>
</tr>
</tbody>
</table>
The ‘onReceive’ function should only trigger action and exit. If you spend too much time in ‘onReceive’, Android will terminate the application.
Why do we need signal system?

- Broadcast about an event
- Get information to GUI thread
- Get information to specific place

Signals examples:

- Got push notifications
- Connectivity changed
- Got data from server side
Creating a signal system – Step 1

- Create a action interface

```java
public interface IUIUpdateInterface {
    void onDataChange(Enums.UIActions action, boolean bSuccess, Intent data);
}
```
Creating a signal system – Step 2

- Declare the actions

```java
public class Constants {
    public enum UIActions {
        uiNetworkStatusChanged
    }
}
```
Create a signal registration system

```java
public class SignalSystem {
    private static SignalSystem ref;
    private static Context ctx;

    private static ArrayList<IUIUpdateInterface> mUIUpdateInterfaces;

    public static void initialize(Context context) {
        if (ref == null) {
            ctx = context;
            ref = new SignalSystem();
        }
    }

    public static SignalSystem getInstance() {
        return ref;
    }

    private SignalSystem() {
        mUIUpdateInterfaces = new ArrayList<IUIUpdateInterface>();
    }

    public void registerUIUpdateChange(IUIUpdateInterface aInterface) {
        if (!mUIUpdateInterfaces.contains(aInterface)) {
            mUIUpdateInterfaces.add(aInterface);
        }
    }
}
```
Create a signal fire function

```java
public void fireUpdateChange(final Constants.UIActions uiAction, final boolean bSuccess, final Intent data) {
    if (Looper.myLooper() == Looper.getMainLooper()) {
        // We are on the main GUI thread
        int index;
        int iSize;

        iSize = mUIUpdateInterfaces.size();

        for (index = 0; index < iSize; index++) {
            ((IUIUpdateInterface)mUIUpdateInterfaces.get(index)).onDataChange(uiAction, bSuccess, data);
        }
    } else {
        // We are not on the GUI thread, need to switch context
        // Get a handler that can be used to post to the main thread
        Handler mainHandler = new Handler(ctx.getMainLooper());

        Runnable myRunnable = new Runnable() {
            @Override
            public void run() {
                fireUpdateChange(uiAction, bSuccess, data);
            }
        };

        mainHandler.post(myRunnable);
    }
}
```
Creating a signal system – Step 5

Register to get signals

```java
public class MainActivity extends ActionBarActivity implements IUIUpdateInterface {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        SignalSystem.getInstance().registerUIUpdateChange(this);
    }

    @Override
    protected void onStop() {
        super.onStop();
        SignalSystem.getInstance().unRegisterUIUpdateChange(this);
    }

    @Override
    public void onDataChange(Constants.UIActions action, boolean bSuccess, Intent data) {
        // Something has happened
    }
}
```