Using Java's built-in Observer Pattern

So far we’ve rolled our own code for the Observer Pattern, but Java has built-in support in several of its APIs. The most general is the Observable interface and the Observable class in the java.util package. These are quite similar to our Subject and Observer interface, but give you a lot of functionality out of the box. You can also implement either a push or pull style of update to your observers, as you will see.

To get a high level feel for java.util.Observer and java.util.Observable, check out this reworked OO design for the WeatherStation:

The Observable class keeps track of all your observers and notifies them for you.

Observable is a CLASS not an interface, so WeatherData extends Observable.

This doesn’t look familiar! Hold tight, we’ll get to this in a sec...

Here’s our Subject, which we can now also call the Observable. We don’t need the register(), remove() and notifyObservers() methods anymore; we inherit that behavior from the superclass.

With Java’s built-in support, all you have to do is extend Observable and tell it when to notify the Observers. The API does the rest for you.

This should look familiar. In fact, it’s exactly the same as our previous class diagram!

We left out the DisplayElement interface, but all the displays still implement it too.

There will be a few changes to make to the update() method in the concrete Observers, but basically it’s the same idea... we have a common Observer interface, with an update() method that’s called by the Subject.