The `setChanged()` method is used to signify that the state has changed and that `notifyObservers()`, when it is called, should update its observers. If `notifyObservers()` is called without first calling `setChanged()`, the observers will NOT be notified. Let’s take a look behind the scenes of Observable to see how this works:

```
setChanged() {
    changed = true
}

notifyObservers(Object arg) {
    if (changed) {
        for every observer on the list {
            call update (this, arg)
        }
        changed = false
    }
}

notifyObservers() {
    notifyObservers(null)
}
```

Why is this necessary? The `setChanged()` method is meant to give you more flexibility in how you update observers by allowing you to optimize the notifications. For example, in our weather station, imagine if our measurements were so sensitive that the temperature readings were constantly fluctuating by a few tenths of a degree. That might cause the `WeatherData` object to send out notifications constantly. Instead, we might want to send out notifications only if the temperature changes more than half a degree and we could call `setChanged()` only after that happened.

You might not use this functionality very often, but it’s there if you need it. In either case, you need to call `setChanged()` for notifications to work. If this functionality is something that is useful to you, you may also want to use the `clearChanged()` method, which sets the changed state back to false, and the `hasChanged()` method, which tells you the current state of the changed flag.