**Workout 2020-08-26 – The Observer Pattern Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Put your answers to all but the last question in this Word document. Make sure your name is at the top of this Word document. Write the program for that last question. Place your Word document in the root directory of the program. Zip up the program directory and submit the zip file as your submission.

1. What does the textbook say is the “dark side” of the Java Observer Pattern API?

***The textbook says the dark side of the Java Observer API because Observable is a class not an Interface. Also, Observable contains a protected method meaning you HAVE to implement the method upon inheritance. Therefore, the design principle that favors composition over inheritance is broken.***

1. In Java API implementation of the Observer pattern:
   1. What is passed via the notifyObservers method using the push version of the Observer pattern? Dig a little deeper and find what is passed in the call to update. (Update is called from inside notifyObservers.)

***notifyObservers() is overloaded with 2 methods. In the push version of the observer pattern notifyObservers(Object arg) takes and object argument to push out to its respective observers.***

* 1. What is passed via the notifyObservers method in the data object using the pull version of the Observer pattern? Dig a little deeper and find what is passed in the call to update.

**In the pull version of the observer pattern the overloaded notifyObserver() method isn’t passed any variables. HOWEVER, within notifyObservers()… notifyObservers(null) is called. Hence why the observer must pull.**

1. setChanged() in the Java Observer API
   1. What does it do?

***Setchanged() simply sets a Boolean value. If an observed subject changes then set Boolean value changed to true.***

* 1. What happens if setChanged() is NOT called before calling notifyObservers?

***Observers won’t be notified because the ‘changed’ variable will be false. Therefore, notifyObservers() will return before update() is called.***

1. For your programming exercise, implement a new display named ArchiveDisplay; however, do this in the context of the Java Observer Pattern API. Weather State program using the Java Observer Pattern API: <https://github.com/bethrobson/Head-First-Design-Patterns/tree/master/src/headfirst/designpatterns/observer/weatherobservable>

The book’s program using “pull” instead of “push” in this version. Use the “pull” process in your code as well.

Write a new observer for the book’s observer pattern example code named ArchiveDisplay. When notified by the WeatherData class, ArchiveDisplay appends the current temperature to an ArrayList of temperatures (declared in this class and instantiated/initialized in the ArchiveDisplay constructor). (Note: nothing is displayed when update is called, just log the current temperature.)

In addition to a constructor and the update method, ArchiveDisplay has a method named display that displays out all the temperatures archived thus far. Display the temperatures as a simple table with a heading “Temperatures” and one temperature per line.

Finally, modify the WeatherStation class (main) so that the HeatIndexDisplay (included in the textbook’s code) is instantiated, is subscribed, and displays its output when notified. Also, instantiate an ArchiveDisplay, and at the end of the program, call its display method to display all the temperature values.