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

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?

It is a class not an interface thus cant be implemented or used by a class that already has a superclass. setChanged() is protected so you cant call it unless youre in the class subclassed to it

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.)

The observable object and the args argument are passed to update in the push version of the observer pattern

* 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.

The observable object and null are passed to update using the pull version of notify observers

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

Marks the observable object as having been changed so now hasChanged will return true

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

Observers will not be notified if setChanged is not called first

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.