**Workout 2020-08-26 – The Observer Pattern Name: Sydney Magee**

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 “dark side” of the Java Observer Pattern API according to the textbook is Observable is defined as a class rather than an interface. The class Observable does not implement an interface either. Due to the fact that Observable is a class, you must subclass it. You cannot create your own implementation because there is not an Observable interface. The Observable class protects important methods and because of this, you must subclass in order to use the Observable API.**

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.)

**Using the push version of the Observer pattern, you can pass the data as a data object to the notifyObservers(arg) method. The update method passes an Observable object and the data object argument within the notifyObservers method. This successfully pushes notifications to 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.

**Using the pull version of the Observer Pattern, you do not pass a data object with the notifyObservers() call. The Observers will use getTemperature(), getHumidity(), and getPressure() to pull the information that it desires from the subject.**

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

**setChanged() is responsible for signifying that the state has changed in your object. You must call setChanged() prior to notifyObservers() because otherwise the observers will not be notified.**

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

notifyObservers does not execute without setChanged() being called first.

**If setChanged() is not called before calling notifyObservers will not function nor notify anyone. The setChanged() method gives you more flexibility in updating observers and notification preferences. For example if the temperature changes constantly, we can use setChanged() to limit the amount of notification that notifyObservers() send out by restricting the notifications to be sent out only if the temperature changes drastically.**

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 Archive Display, and at the end of the program, call its displayArchive method to display all the temperature values.

create an arraylist object