**Assignment 3: Programming with Lists, Encapsulation**

**Due:** Tuesday, April 7 at 11:59pm via [Turnin](https://turnin.cs.wpi.edu/) . myWPI questionnaire due 45 minutes later (closing at 12:45am). Once you access the questionnaire, you will have only 30 minutes to submit your answers.

Our goals this week are to practice programming with lists (if you choose) and writing clean encapsulated code. Your goal is to produce code that is well organized, uses helper methods and interfaces where appropriate, and follows good object-oriented design techniques. We will pay particular attention to these issues in grading your work.

This assignment is designed to help you think about where data and computations belong in a well-designed object-oriented program. Methods that produces the right answer, but aren’t structured well, won’t earn you many points. Figuring out where to put the various pieces of data, and what methods you need to create to work with them, are part of the what you are being asked to figure out.

**NOTE:** Although the problem descriptions refer to lists, you are welcome to use whatever data structure you wish for managing your weather data. Decide whether you want practice with lists or another data structure and program accordingly.

**Where’s the Advanced Option?:** Encapsulation is a sufficiently important course topic that there is no separate advanced assignment this week. If you are looking for more of a challenge, try to eliminate the common code across the low and high temperature computations.

**1****The Weather Monitoring Tool**

Your overall goal is to provide a program that reports weather trends. To keep things simple, we are interested in two trends: average daily high temperature during a particular month, and average daily low temperature during a particular month. To this end you need to create a WeatherMonitor class with (at least these) two methods:

* averageHighForMonth, which takes a month (designated by a number such as 1 for January, 2 for February, etc) and a year and produces the average high temperature over all days that month.
* averageLowForMonth, which takes a month (designated by a number such as 1 for January, 2 for February, etc) and a year and produces the average low temperature over all days that month.

The weather data you are tracking is initially gathered from a weather sensor. The sensor produces *readings* containing the time of the reading (hour and minutes) and the temperature in degrees Fahrenheit. Because the volume of readings is so high, your weather monitor will store only daily weather reports. A *daily weather report* contains the date (month, day, year) and two temperature readings: one for the lowest temperature of the day and one for the highest temperature of the day.

* To manage the daily weather data, your WeatherMonitor must also provide a method addDailyReport that consumes a date and a list of readings (nominally for that date) and stores a daily report for the given date (computing the high and low temperature readings from the given list of readings for that date).

**2****Protecting and Encapsulating the Data**

This set of questions asks you to modify your code as needed to satisfy certain goals, then to give us brief descriptions of how you did so. Put your descriptions in a text file named *questions.txt*. You do not turn in separate code for these questions: just modify your existing code as needed to meet these goals.

1. Thinking ahead, you know that the weather monitor program should be able to support different data structures for the daily weather reports. Edit your code as needed to encapsulate the type of the daily weather reports from the overallWeatherMonitor class. In *questions.txt*, explain what specifically in your code would achieves this goal. As part of your answer, explain where (which lines/expressions) someone would have to change in order to use a spanky newweatherGlob data structure instead of your current LinkedList implementation. (Note: you are NOT being asked to implement weatherGlob. Just assume such a class exists and tell us how someone would plug it into your code.)
2. The weather watchdog group that uses your Weather Monitor has an activist staff member who wants to find evidence of global warming, and is willing to alter weather data slightly to help his cause. The activist has access to aWeatherMonitor object. Edit your code as needed to guarantee that the staff member cannot change any of the daily weather reports or their readings. In*questions.txt*, explain what specifically in your code achieves this goal.
3. Look at the code you have now: are there any aspects of it that you would like to clean up, but don’t yet know how in Java? Are you satisfied with your code from a design standpoint?

**3****Formatting and What to Turn In**

Starting with this assignment, we will follow two aspects of standard Java programming practice:

1. Create a separate file for every class or interface. The name of the file should match the name of the class or interface. For example, the Examples class is inExamples.java.
2. Bundle your code into a *package* for submission. To create a Java package, you put all of your java files in the same directory (say "hwk3") and you put the line

package hwk3;

at the top of each .java file. Java convention says to start package names with lowercase letters (so they don’t clash with class or interface names, which conventionally start with an uppercase letter). Zip up and submit your package directory (in this case, "hwk3").