Homework 5 - Encapsulation

Due Apr 22, 2019 by 5pm **Points** 0

Before you begin, please make sure you are familiar with the guidelines discussed on the Expectations on Homework page and the "Homework Assignments" section of the CS 2102 FAQ Page.

Assignment Goals

- To be able to write clean, encapsulated code
- To be able to use the Java API documentation to learn how to use a library class (GregorianCalendar)

Reminders

- **Please use the default package in your Java project.** There will be a penalty for using a named package.
- Please include a Javadoc statemnt above each method. There will be a penalty for forgetting your Javadoc statements.
- In your test cases, please use descriptive names for your test case methods and/or comments above each test case explaining what the method tests. This will help the TAs and SAs in grading. There will be a penalty for forgetting this step.

Overview

Our goal this week is to write clean, encapsulated code. You should aim 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 produce the right answers--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 the data--are part of the what you are being asked to figure out.

Although the assignment is described in two parts, you'll only be submitting the program for Part 2. However, we suggest you work through Part 1 as an initial implementation, then make the modifications described in Part 2.

Part 1: The Weather Monitoring Tool

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

- `averageTempForMonth`, which takes a month (designated by a number such as 1 for January, 2 for February, etc) and a year and produces the average temperature over all days that month.*
- `totalRainfallForMonth`, which takes a month (designated by a number such as 1 for January, 2 for February, etc) and a year and produces the total rainfall over all days that month.*

The weather data you are tracking is initially gathered from a weather sensor. The sensor produces `Reading`s containing the `Time` of the reading (hour and minute, both ints), the temperature in degrees Fahrenheit (use a double), and the amount of rainfall since the last reading (use a double). Because the volume of readings is so high, your weather monitor will store daily weather reports that each have all of the readings for that particular day. A `DailyWeatherReport` contains the date (use the Java class `GregorianCalendar`, see description below) and two LinkedLists: one for the temperature readings and one for the rainfall for that 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. For Part 1 of this assignment, the WeatherMonitor's daily reports should be stored in a LinkedList. (You may assume that a daily report for the provided date does not already exist.)

(* We won't actually require you to provide examples of daily weather reports for every single day in a month. Your calculations of `averageTempForMonth` and `totalRainfallForMonth` should produce the averages over all days in the month for which there are daily weather reports.)

Gregorian Calendar class

The date field for a daily weather report should be of type `GregorianCalendar`. This is a type available in the java.util library. Look at the **Java API (https://docs.oracle.com/javase/8/docs/api/index.html?overview-summary.html)** to learn how to use `GregorianCalendar`. Here are a couple of hints to get you started:

When constructing a date for a daily weather report, use this version of the `GregorianCalendar` constructor:

```
    public GregorianCalendar (int year, int month, int dayOfMonth)
```

To extract individual pieces of information from objects of type `GregorianCalendar`, use the `get()` method (inherited from the abstract Java class `Calendar`). Here's an example of how to extract the month:

```
GregorianCalendar date = new GregorianCalendar(2016, 11, 17);
int month = date.get(GregorianCalendar.MONTH);  // value of month will be 11
```

Part 2: Protecting and Encapsulating the Data

This set of exercises asks you to modify your code from Part 1 as needed to satisfy certain goals. Do not turn in separate code for these exercises; just modify your existing code from Part 1 as needed to meet these goals, then turn in the final (modified) version of your code.

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 overall `WeatherMonitor` class.

A `WeatherMonitor` constructor should take an interface type as its argument, and any fields that hold a `DailyWeatherReport` object should have an interface type.

A rogue hacker wants to modify the weather data in order to create a false panic about a pending natural disaster. The hacker has access to a `WeatherMonitor` object. Edit your code as needed to guarantee that the hacker cannot change any of the daily weather reports or their readings. For `DailyWeatherReport`, please do not write methods that simply return the lists of temperatures and/or rainfall for that day.

Something to think about: in Part 1 you were asked to define two methods, `averageTempForMonth` and `totalRainfallForMonth`. Presumably, the code you developed for these two methods will have some similarities. We've talked about why duplicating code is undesirable. Were you able to come up with a way to eliminate the duplicate code? For those of you who took CS 1101/1102, would you have handled the task of eliminating the duplicate code differently in Racket?

Grading

Homework 5 Grading Rubric (https://canvas.wpi.edu/courses/12951/files/1898805/download?download_frd=1)

Due to the nature of this assignment, there is no CompileCheck program to test your code against. Your programs will not be graded against an auto-tester. When grading your programs, we will be looking for

- programs that follow the principles of encapsulation
- a set of test cases to show that your code behaves as specified above
- the following classes (at minimum) with appropriate fields and methods

- `DailyWeatherReport`
- `Reading`
- `Time`
- `WeatherMonitor`
- at least one interface, used appropriately

The three methods in the `WeatherMonitor` class (`addDailyReport`, `avgTempForMonth`, `totalRainfallForMonth`), must conform to the signatures described above.

Programs must compile in order to receive credit. Code that is commented out will not be graded.

What to Turn In

Submit (via **InstructAssist** **(https://ia.wpi.edu/cs2102/)**) a single zip file (not tar, rar, 7zip, etc) containing all of your .java files that contain your classes, interfaces, and examples for this assignment. The name of the project in InstructAssist is **Homework 5**. Do not submit the .class files.