Module 1 Lab -- Weather

Due Sep 12 by 11:59pm **Points** 100 **Available** after Sep 7 at 12am

> To practice implementing and testing a class.

Weather

Often when you go outside, it feels much warmer or colder than predicted. This is not a conspiracy. It is simply that the predicted temperature is not always what the real world *feels like*. Most weather channels and meteorology apps have started using this term to give folks a better understanding of how they will actually feel out in their environment. They do this by collecting readings from *Stevenson Screen* shelters.

A Stevenson Screen is a standard shelter that protects meteorological instruments which measure things like temperature, dew point, wind speed, rain, etc. These readings are used to calculate the different temperatures that appear in your weather report. For instance:

- **Dew point** is the temperature below which water droplets (or dew) begin to form. The relationship between dew point (D), temperature in degrees Celsius (T), and relative humidity in percentage between 0 and 100 (R) is expressed in the following formula:

  $$D = T - \frac{100-R}{5}$$

- **Heat index** is a measure of how hot it feels when relative humidity is factored in with the actual temperature. It is calculated in a number of different ways, but for our example, consider the formula for heat index (HI):

  $$HI = c_1 + c_2T + c_3R + c_4TR + c_5T^2 + c_6R^2 + c_7T^2R + c_8TR^2 + c_9T^2R^2$$

  where T is the temperature in degrees Celsius, R is the relative humidity in percent, and the coefficients are: c_1 = -8.78469475556, c_2 = 1.61139411, c_3 = 2.33854883889, c_4 = -0.14611605, c_5 = -0.012308094, c_6 = -0.0164248277778, c_7 = 0.002211732, c_8 = 0.00072546, and c_9 = -0.000003582.

- **Wind chill** is related to heat index and is used when the real-feel temperature is lower than the actual temperature. There is some variation in how it is calculated depending on where you are, but here in the United States, it is calculated with this formula:

  $$WC = 35.74 + 0.6215T - 35.75v^{+0.16} + 0.4275Tv^{+0.16}$$

  where WC is the wind chill based on the air temperature in degrees Fahrenheit (T) and the wind speed in miles per hour (v).

What to do

Package: `weather`

Design and implement a class called `WeatherReading` that represents a single reading of a weather station in a Stevenson Screen. Your constructor takes four parameters: the air temperature in Celsius, the dew point temperature in Celsius which cannot be greater than the air temperature, the non-negative wind speed in miles per hour, and the non-negative total rain received in the last 24 hours in millimeters. It should throw an `IllegalArgumentException` for invalid values.

Methods that get different values from an object are called *accessor methods* or *getters* because they are named starting with "get". Your implementation should include each of the following:

- `getTemperature`
- `getDewPoint`
- `getWindSpeed`
- `getTotalRain`
- `getRelativeHumidity`
- `getHeatIndex`
- `getWindChill`

Do not forget that every well-written class in Java should also override the `toString` method. Your implementation should create an output like this one:

```
Reading: T = 23, D = 12, v = 3, rain = 12
```

Testing

Whenever you write a class, you should also write tests for that class that prove not only that your code CAN work but that it WILL ALWAYS work. Your goal here is to write tests to achieve as close to 100% coverage as possible. But even more importantly, your tests should be sufficient to ***convince someone else that your code works correctly***.

What to Submit

1. Create a zip file that directly contains only your src/ and test/ folders. When you unzip the file, you must see only these two folders.
2. Log on to the **Handins server** (https://handins.ccs.neu.edu/)
3. Navigate to this lab and submit the zip file.
4. Wait for a few minutes for the grader's feedback to appear, and take action if needed.

Grading Criteria

This assignment will be assessed via the automatic testing available on the Handins server. You should ensure that your code does not have any code style violations and that it passes all of the test cases. This is worth 1% of your final grade.

Additionally, your code will be assessed via peer evaluation. Each student will be required to review one other student's code for an additional 0.5% of their final grade. This grade will be based on the quality of the review **you give**, not the review you receive. During the peer review, you will evaluate the given code based on three criteria:

1. Design and organization of classes, interfaces, and methods including
   - How appropriately you captured the data and operations in the implementation
   - Whether code looks well-structured and clean (not unnecessarily complicating things or using unwieldy logic)
   - How well your design and implementation embraces various design principles that you have learned so far
2. Understandability and quality of documentation including
   - Whether the documentation enhances the understandability of the code rather than just repeating how something is implemented
   - Whether it is standalone, meaning whether the documentation is sufficient for use without reading the implementation
3. Quality and coverage of tests including
   - Whether tests are modular -- do you write one big, monolithic test method or does your test class have separate methods for each test

  - Whether tests are written in such a way that the reader knows what is being tested
  - Whether the tests convince the reader that if all tests pass then the code it is testing works correctly