Homework 02 - Temperature Converter

Authors: Jack Kelly, Vince Li, Andrew Chafos, Shishir Bhat

Problem Description

The company that has hired you has a problem. Half of their thermostats display in degrees Celsius and the other half display in degrees Fahrenheit. All of their thermostats display in integers. Unfortunately there is no way to change the unit of a given thermometer. To remedy this problem your company has asked you to write a simple Java program that will allow your co-workers to easily convert between the two units. Your company has provided you with a Java class `Converter.java`, so all you will have to do is fill in the provided methods.

Solution Description

For this assignment you have been provided `Converter.java`. This class has several methods where only a method header has been written. To complete this assignment you should correctly fill in the body of each method as specified here and in the provided java file.

The methods you will have to fill in are listed here.

`fahrenheitToCelsius(int temp)`

This method takes in a temperature in Fahrenheit and should return that temperature in Celsius as a double.

`celsiusToFahrenheit(int temp)`

This method takes in a temperature in Celsius and should return that temperature in Fahrenheit as a double.

`printFahrenheitConversionTable(int lower, int upper)`

This method prints out a table of Fahrenheit temperatures from the specified lower bound (inclusive) to the specified upper bound (inclusive) and that temperature converted to Celsius. Round to three decimal places.

Ex) if we ran printFahrenheitConversionTable(32, 33) we should get what's printed below. Please make sure to use this format exactly or you will not get credit.

Fahrenheit: 32 -> Celsius: 0.000
Fahrenheit: 33 -> Celsius: 0.556

Note: This method should be completed using a FOR loop. **You MAY NOT use a while loop anywhere in this method.** Doing so will yield no credit

`printCelsiusConversionTable(int lower, int upper)`

This method prints out a table of Celsius temperatures from the specified lower bound (inclusive) to the specified upper bound (inclusive) and that temperature converted to Fahrenheit. Round to three decimal places.

Ex) if we ran printCelsiusConversionTable(32, 33) we should get what's printed below. Please make sure to use this format exactly or you will not get credit.

Celsius: 32 -> Fahrenheit: 89.600
Celsius: 33 -> Fahrenheit: 91.400

Note: This method should be completed using a WHILE loop. **You MAY NOT use a for loop anywhere in this method.** Doing so will yield no credit.

`celsiusWarmer(int celsius, int fahrenheit)`

This method takes in two temperatures, the first in Celsius and the second in Fahrenheit, it should return true if the first temperature is warmer than the second.

Note: **Do NOT use a ternary expression.** Doing so will yield no credit.

`fahrenheitWarmer(int fahrenheit, int celsius)`

This method takes in two temperatures, the first in Fahrenheit and the second in Celsius, it should return true if the first temperature is warmer than the second.

Note: you should use a ternary expression to complete this method. **Do NOT use an if statement.** Doing so will yield no credit.

Testing your code

If you want to test out your own code before submitting, we've provided some sample cases in the main method. If you run `java Converter` after compiling your code, this will be the output from the provided cases.

```
fahrenheitToCelsius: 1.1111111111111112
celsiusToFahrenheit: 39.2
printFahrenheitConversionTable:
Fahrenheit: 32 -> Celsius: 0.000
Fahrenheit: 33 -> Celsius: 0.556
Fahrenheit: 34 -> Celsius: 1.111
Fahrenheit: 35 -> Celsius: 1.667
```

Feel free to create your own tests in the `main` method! Try to write tests for the remaining methods in the file!

Rubric

- [15] celsisusToFahrenheit
- [15] fahrenheitToCelsius
- [20] printFahrenheitConversionTable
- [20] printCelsiusConversionTable
- [15] fahrenheitWarmer
- [15] celsiusWarmer

Allowed Imports

To prevent trivialization of the assignment, you are *not* allowed to import any classes or packages.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- var (the reserved keyword)
- System.exit

Checkstyle

For this assignment, we will be enforcing style guidelines with Checkstyle, a program we use to check Java style. Checkstyle can be downloaded here.

To run Checkstyle, put the `jar` file in the same folder as your homework files and run

```
java -jar checkstyle-6.2.2.jar *.java
```

The Checkstyle cap for this assignment is **5 points**. This means that up to five points can be lost from Checkstyle errors.

Collaboration

Collaboration Statement

To ensure that you acknowledge collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Recall that comments are special lines in Java that begin with `//`.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `Converter.java`

Make sure you see the message stating "HW02 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for all official clarifications