Homework 1 - Converter

Problem Description

After completing the previous homework, you should be familiar with creating, compiling, and running a basic Java program. So far we have only asked you to perform some simple calculations and output the results by printing them to the terminal. In this homework, you will go a step further by gathering input from a person using your program, performing calculations with that input, and then outputting a formatted version of the result by printing it to the terminal. Additionally, this homework requires you to do some basic debugging on erroneous code that we have provided with this document and to submit some information to help us get to know you better.

Make sure you read the entire document! There are some things even at the very end that you will not want to miss for this and future assignments!

Solution Description

TemperatureConverter

A formula for converting temperature in Kelvin to degrees Fahrenheit is as follows:

```
Fahrenheit = Kelvin * (9/5) - 459.67
```

Your goal for this assignment is to write one file, `TemperatureConverter.java`, which prompts the user for a temperature in Kelvin, converts the temperature to degrees Fahrenheit, and then outputs the results by printing it to the terminal. Note that the temperature provided by the user may or may not have a decimal value (e.g. 42 and 3.14159 are both valid inputs). The converted result that you output should be rounded to two decimal places (e.g. 2 -> 2.00, 28.1231 -> 28.12). You may assume that the user will enter a valid number when prompted for a temperature.

Below is an example of the way a user would interact with your program. Be sure to **follow the syntax exactly**. Note that each execution ends with a newline character (after Fahrenheit), and that the temperature in Kelvin is identical to the input when it is printed back to the console.

```
$ java TemperatureConverter
Enter a temperature in Kelvin: 291.2359
291.2359 Kelvin is 64.55 degrees Fahrenheit
$ java TemperatureConverter
Enter a temperature in Kelvin: 278.706
278.706 Kelvin is 42.00 degrees Fahrenheit
```

Learning to debug

With this document we have provided three Java programs, `Bad1.java`, `Bad2.java`, and `Bad3.java`, each of which contains a single error. Attempt to compile and run each program and observe the resulting error message. In a file named `errors.txt`, record the error message for each of the bad Java programs, state whether the error occurs at compile time or runtime, and write a single sentence for each bad program which describes the reason for that error. Then fix whatever is wrong with the programs so that you are able to compile and run them.

You will submit the fixed versions of `Bad1.java`, `Bad2.java`, and `Bad3.java`.

Tell us who you are!

As TAs, we're here to help you be successful in this course. A good first step is for us to learn a little more about you and to put names to the faces in our section. Along with your submission, please attach a picture of yourself (optional) and a text file `me.txt` containing a few sentences about yourself (e.g. your major, how long you have been at Georgia Tech, a personal interest, etc.).

Allowed Imports

To prevent trivialization of the assignment, you are only allowed to import the following classes:

- java.util.Scanner
- java.lang.Math

If you would like to import anything else, ask on Piazza.

Feature Restrictions

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

- var (the reserved keyword)

Rubric

[60] **TemperatureConverter.java**

- [20] Program is runnable (main method)
- [10] Correctly prompts for user input
- [10] Correct output format
- [20] Correct conversion and printed result
  - [10] Basic cases (see example in Solution Description)
  - [10] Complex cases

[30] **Bad#.java files and errors.txt**

- [15] Describes each error within errors.txt
- [15] Corrects each java file

[10] **me.txt**

- [10] Includes a few sentences about themselves

We reserve the right to adjust the rubric, but this is typically only done for correcting mistakes.

Collaboration

Collaboration Statement

To ensure that you acknowledge a 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].

Allowed Collaboration

When completing homeworks for CS1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homeworks
- Parts of the homework you are unsure of and need more explanation
- Online resources that helped you find a solution

- Key course concepts and Java language features used in your solution

You may **not** discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

Examples of approved/disapproved collaboration:

- **approved**: "Hey, I'm really confused on how we are supposed to implement this part of the homework. What strategies/resources did you use to solve it?"
- **disapproved**: "Yo it's 10:40 on Thursday... Can I see your code? I won't copy it directly I promise"

In addition to the above rules, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due.

Turn-In Procedure

Submission

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

- TemperatureConverter.java
- Bad1.java
- Bad2.java
- Bad3.java
- errors.txt
- me.txt
- me.png (optional)

Make sure you see the message stating "HW## 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. 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 and run your code; you can do that locally on your machine.

Other portions of your assignment are also autograded once the submission deadline has passed. The autograders used are often dependent on specific output formats, so **make sure that your submission passes all test cases marked "FORMAT:"**.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Ensure you pass all "FORMAT:" tests

- Read the “Allowed Imports” and “Restricted Features” to avoid losing points
- Check on Piazza for a note containing all official clarifications