Programming Exercise 2

Authors: Matthew, Chelsea, Varun, Rachna

Problem Description

The Collatz Conjecture in mathematics is infamous for being an extraordinarily difficult problem - often said to be "completely out of reach of present-day mathematics" to prove. The conjecture is concerned with a sequence of numbers defined as such:

- Start by setting n to any positive integer
- If this term (n) is even, the next term is half of n.
- If this term (n) is odd, the next term is 3n + 1
- Repeat this process to find all the terms in the sequence.

For example, starting with *12* gives you the following sequence:

12 → 6 → 3 → 10 → 5 → 16 → 8 → 4 → 2 → 1

The conjecture states that for any positive integer n, the sequence will always reach 1.

When it comes to definitively proving this conjecture true, it is often said that "Mathematics may not be ready for such problems," but Java is more than ready to *simulate* it! (albeit on a smaller scale).

Your assignment consists of 2 main parts:

- Start with some number and simulate this sequence in code until your integer reaches 1.
- Print out a summary of your sequence (Initial number, steps taken, highest number reached)

Solution Description

1. Create a class called `Collatz`
2. Create the `main` method
3. Inside, create these variables:
    a. Create an **int** called `collatzNum` and assign it any positive integer from 1-100 (Please refer to the PE Clarification Thread on Ed for example output.)
    b. Create an **int** called `numSteps` and assign it 0. The number of "steps" corresponds to the number of times you apply one of the operations.
    c. Create an **int** called `highestNumReached` and assign it the value `collatzNum`
    d. Create an **int** called `initValue` and assign it the value `collatzNum`
4. Print `collatzNum`.
5. Create a `while` loop that terminates when `collatzNum` is equal to 1
6. Within the `while` loop, use an if-else statement to do the following:
    a. If `collatzNum` is even, `collatzNum` should be assigned half of its current value
    b. Otherwise, (if it's odd), `collatzNum` should be three times its value plus one.
7. Check whether the new `collatzNum` is greater than `highestNumReached`. If so, update its value to `collatzNum`
8. Print out `collatzNum` and add 1 to numSteps
9. Close the `while` loop
10. After the `while` loop, use three print statements to output:

©CS1331

a. "Initial value: `[initValue]`"
b. "Number of steps: `[numSteps]`"
c. "Highest number reached: `[highestNumReached]`"

11. Finally, use a switch statement to print **one** of the following statements according to the value of `numSteps`:
    a. *0 steps*: "No steps required"
    b. *1 step*: "Only took one step!"
    c. *2 steps*: "Two steps"
    d. *3 steps*: "Three steps"
    e. *4 steps*: "Four steps"
    f. *All other amounts*: "Wow, `[numSteps]` steps was a lot of steps!" (For example, if 8 steps were taken, "Wow, 8 steps was a lot of steps!" should be printed without the quotes.)

Example Outputs

Please refer to the PE clarification thread on Ed for examples.

HINT: To help debug your code, try inserting print statements in places where variables are changed.

Turn-In Procedure

Submission

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

- `Collatz.java`

Make sure you see the message stating "PE2 submitted successfully". 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 so be sure to **submit every file each time you resubmit**.

Make sure you see the message stating the assignment was submitted successfully. From this point, Gradescope will run a basic autograder on your submission as discussed in the next section. **Any autograder test are provided as a courtesy to help "sanity check" your work and you may not see all the test cases used to grade your work.** You are responsible for thoroughly testing your submission on your own to ensure you have fulfilled the requirements of this assignment. If you have questions about the requirements given, reach out to a TA or Professor via Piazza for clarification.

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 latest submission. **Be sure to submit every file each time you resubmit**.

Gradescope Autograder

If an autograder is enabled for this assignment, you may be able to see the results of a few basic test cases on your code. Typically, tests will correspond 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.

@CS1331

The Gradescope tests serve two main purposes:

- Prevent upload mistakes (e.g. non-compiling code)
- 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.

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`

Import Restrictions

You may not import anything for this homework assignment.

Collaboration

Only discussion of the PE at a conceptual high level is allowed. You can discuss course concepts and HW assignments broadly, that is, at a conceptual level to increase your understanding. If you find yourself dropping to a level where specific Java code is being discussed, that is going **too far**. Those discussions should be reserved for the instructor and TAs. To be clear, you should never exchange code related to an assignment with anyone other than the instructor and TAs, and this exchange should be private.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- It is expected that everyone will follow the Student-Faculty Expectations document, and the Student Code of Conduct. The professor expects a positive, respectful, and engaged academic environment inside the classroom, outside the classroom, in all electronic communications, on all file submissions, and on any document submitted throughout the duration of the course. **No inappropriate language is to be used, and any assignment, deemed by the professor, to contain inappropriate, offensive language or threats will get a zero**. You are to use professionalism in your work. Violations of this conduct policy will be turned over to the Office of Student Integrity for misconduct.

©CS1331