CSE 8B: Introduction to Programming and Computational Problem Solving - 2

Assignment 2

Selections, Loops, and Strings in Java

Due: Wednesday, October 12, 11:59 PM

Learning goals:

- Write Java code that includes:
  - Selections
  - Mathematical operations
  - String and character parsing
  - Loops
- Write your own test cases to test the correctness of your methods

NOTE: This assignment should be completed INDIVIDUALLY. Pair programming is NOT allowed for this assignment.

Coding Style (10 points)

For this programming assignment, we will be enforcing the CSE 8B Coding Style Guidelines. These guidelines can also be found on Canvas. Please ensure to have *COMPLETE* file headers, class headers, and method headers, to use descriptive variable names and proper indentation, and to avoid using magic numbers.

Part 0: Getting started with the starter code (0 points)

1. Make sure there is no problem with your Java software development environment. If there is any, then review Assignment 1, or come to the office/lab hours before you start Assignment 2.
2. First, navigate to the `cse8b` folder that you have created in Assignment 1 and create a new folder titled `assignment2`
3. Download the starter code.
   You can download the starter code from Piazza → Resources → Homework → `Assignment2.java`

The starter code should only be one file called `Assignment2.java`. Place the starter code within the `assignment2` folder that you have just created

4. Compile and run the starter code, and you should expect the following output:

```
darrenyeung@Darrens-MacBook-Pro assignment2 % javac Assignment2.java
darrenyeung@Darrens-MacBook-Pro assignment2 % java Assignment2

triangularPrismVolume Output 1: 0.00

FAILED: triangularPrismVolume 1
ERROR: Failed test.

darrenyeung@Darrens-MacBook-Pro assignment2 %
```

Although we have not covered Java methods in lecture yet, you can think of Java methods as a math function. A method takes in zero or more parameters (input) and returns a value (output). The starter code is set up so you need not know anything about the details of creating a method from scratch. Your job is to just fill in the starter code.

Part 1: Implement three methods (75 points)

In class `Assignment2` of the starter code, 5 methods are already declared for you: `triangularPrismVolume`, `geometricSum`, `createHourglass`, `unitTests`, and `main`. **For this part of the assignment, your task is to implement the first three methods (`triangularPrismVolume`, `geometricSum`, and `createHourglass`).**

Before you implement your methods, please take a look at the constants at the top of the class. All of the error messages as well as method-specific variables are already defined for you with the keywords `private final static`. **IMPORTANT: You should use these constants when developing your methods to ensure that your code will always give the correct output. You technically don't have to but using them will help you remain consistent.**

`triangularPrismVolume` (25 points):

This method takes two `double` variables, `triangleSideLength` and `height` as input, and calculates the volume of an equilateral triangular prism with that side length and height. If the input side length or height is negative or is greater than 15, then the method should return an error message string: `"INVALID INPUT: out of range [0, 15]"`. Otherwise, the method should return a string of that volume with *2 digits of accuracy* to the right of the decimal point. The starter code already checks whether `triangleSideLength` and `height` is in range.

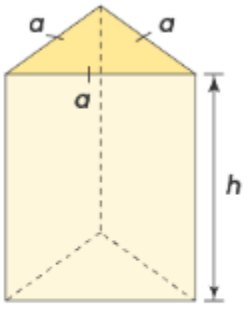
Rounding to within 2 digits of accuracy is also already completed. **Your job is to calculate the volume and assign the value to the variable** `volume`.

As a side note, an easy way to round the volume to 2 digits of accuracy is to use `String.format()`. To round to 2 digits, the format string should be `"%.2f"`. Here is an example of how `String.format()` works:

```
String myStr = String.format("The value is: %.2f", 12.478223)
System.out.println(myStr);
// This will output: "The value is: 12.48"
```

IMPORTANT: **We are only concerned with equilateral triangular prisms in this assignment**. So `triangleSideLength` will be the same on all three sides.

The volume of an equilateral triangular prism is shown below. **We have defined a constant for you that can be used to square a number.**



$$V = \frac{\sqrt{3}}{4} \times a^2 \times h$$

Sample:

`triangularPrismVolume(3.5, 10)` → `"53.04"`

`geometricSum` (25 points):

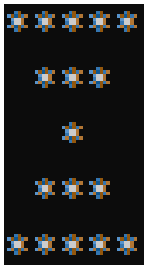
This method takes an `integer` variable, `n`, as the input, and the method should calculate the sum of a geometric series. A [geometric series] a sequence of numbers of the following form:

$a,\ ar,\ ar^2,\ ar^3,\ \ldots$. In this problem, we want you to calculate the sum of a geometric series with coefficient `a = 1` and ratio `r = 3/2` up to `n` terms. More specifically, $1 + \frac{3}{2} + \frac{9}{4} + \ \ldots + (\frac{3}{2})^{n-1}$ Similar to `triangularPrismVolume` if the input value is less than 1 or is greater than 20, then the method should return an error message string: `"INVALID INPUT: out of range [1, 20]"`. Otherwise, the method should return a string of that sum of reciprocals with *5 digits of accuracy* to the right of the decimal point. The starter code already checks whether `n` is in range. Rounding to within 5 digits of accuracy is also already completed. **Your job is to calculate the sum and assign the value to the variable `result`.**

Sample:
`geometricSum(3) → "4.75000"`

`createHourglass` (25 points):

This method takes two inputs: a character `ch` and an integer `height`. **You can assume that `ch` will always be a valid character.** If `height` is less than 1 or greater than 50, then the method should return an error message string: `"INVALID INPUT: out of range [1, 50]"`. If `height` is an even integer, then the method should return the error message `"INVALID INPUT: n must be odd"`. Otherwise, construct and return a String that uses the character `ch` to create an hourglass with the number of rows equal to `height`. More specifically, each half of the hourglass will have (`height-1)/2` rows. For example, when `ch` is `'*'`, and `height` is `5`, then the result string should be: `"*****\n *** \n  *  \n *** \n*****\n"` (where `\n` is the new line character). When this string is printed out, it should look like this:



NOTE: Notice how each row has one extra space in the **beginning and end** until you reach the middle of the hourglass which is just a single `ch` which then starts reversing. Since `height = 5`, there are a total of 5 rows. Thus, you can imagine if `height = 1`, then the string would just be a single `'*'` with a new line appended.

Once again, just like `triangularPrismVolume` and `geometricSum`, the starter code already checks whether `height` is in range and if it is odd. Your job is to construct the hourglass and assign it to the variable `result`.

IMPORTANT (Will lose points of not followed):

1. There must be the right amount of spaces **before and after** each row. More importantly, a common mistake is if you forget the spaces **after** the stars and print out the string, it may look correct in your terminal but will fail the autograder.

2. Append the newline character `'\n'` after the last space (if any) in every row **including the last row**. Do not append anything else. Any extra characters will fail the autograder.

3. As long as your output matches the format of the result string literal shown in quotation marks above, you should pass.

Part 2: Test the correctness of your three methods (10 points)

Testing is a very important part in programming. In this course, we will get you familiar with unit testing. For this assignment and all future assignments, you will be asked to create your own tests to check whether your code works as expected. **In this part of the assignment, you need to implement your own test cases in the method called `unitTests`.**

In the starter code, several test cases are already implemented for you. You can regard it as an example to implement other cases. The general approach is to come up with different inputs and manually give the expected output, then call the method with that input and compare the result with expected output.

You are encouraged to create as many test cases as you think to be necessary to cover all the edge cases. The `unitTests` method should `return true` only when all the test cases are passed. Otherwise, it should `return false`. **To get full credit for this section, for each method, you should have at least four total test cases that cover different situations (including the ones we have provided).** In other words, you will need to create three more tests for `triangularPrismVolume`, three more tests for `geometricSum`, and three more tests for `createHourglass`.

Part 3: Complete `main` (5 points)

After completing the three methods and creating several unit tests, compile and run `Assignment2`. You should see the message `"All unit tests passed"`. If not, then it is very

likely that you have bugs in your code. Read the previous parts carefully while inspecting your code to fix your bugs. We call this process debugging.

The main method is the method that will be called when running program `Assignment2`. Here is the main method given to you in the starter method:

```
// TODO: Complete the method header and the method body
Run | Debug
public static void main(String[] args) {
    // Perform unitTests first
    if (unitTests()) {
        System.out.println(x: "All unit tests passed.\n");
    } else {
        System.out.println(x: "ERROR: Failed test.\n");
        return;
    }

    // Initilize scanner
    Scanner scanner = new Scanner(System.in);
    System.out.println(PROMPT_MSG_NAME);

    // TODO: Complete machine-user interaction


    // Close scanner
    scanner.close();
}
```

The code to run `unitTests` is already given to you. **Do NOT change any code that is provided to you in main. You should only add more lines of code.**

First, your program should print the prompt `"Please enter your name."` (which is already done in the starter code) and wait for the user to enter feedback.

1. The user will enter their name.
2. After the user enters their name. You should print the prompt `"Please enter your college."` on a new line and wait for the user to enter feedback.
3. The user will enter their college
4. After the user enters their college. You should print the prompt `"Please enter your age."` on a new line and wait for the user to enter feedback.
5. The user will enter their age
6. If the user's age is less than 21, then you should print `"Nice to meet you name! college is amazing. You are not able to drink yet."` on a new line.
7. If the user's age is greater than or equal to 21, then you should print `"Nice to meet you name! college is amazing. You are able to drink."` on a new line instead.

Important: Notice the bolded **`name`** and **`college`** in the string that you are supposed to print in either step 6 or step 7 (depending on the user's age). You **should not** print the actual strings `"name"` or `"college"`. Instead, you must replace it with what the user inputed for those fields when you prompted them for input. An example is shown below.

Example: you should be able to *exactly* reproduce this output below with your program after the unit testing

```
All unit tests passed.

Please enter your name.
Darren
Please enter your college.
Sixth
Please enter your age.
20
Nice to meet you Darren! Sixth is amazing. You are not able to drink yet.

C:\Users\Darren\cse8b\assignment2>
```

To prevent the autograder from failing because of a minor error in the string you printed out, we have relaxed the autograder so it is possible to still pass if you made a minor error in what you printed.

Submission

VERY IMPORTANT: Please follow the instructions below carefully and make the exact submission format.

1. Go to Gradescope via Canvas and click on Assignment 2.
2. Click the DRAG & DROP section and directly select the required file (`Assignment2.java`). Drag & drop is fine. **Please make sure you don't submit a zip. Make sure the filename is correct.**
3. **You can resubmit an unlimited number of times before the due date.** Your score will depend on your final submission, even if your former submissions have a higher score.
4. The autograder is for grading your uploaded files automatically. Make sure your code can compile on Gradescope.

   NOTE: The Gradescope Autograder you see is a minimal autograder. For this particular assignment, it will only show the compilation results and the results of basic

tests. After the assignment deadline, a thorough Autograder will be used to determine the final grade of the assignment. **Thus, to ensure that you would receive full points from the thorough Autograder, it is your job to extensively test your code for correctness via `unitTests`.**

5. Your submission should look like the screenshot below. **If you have any questions, then feel free to post on Piazza!**