FA21 CSE 8B Homework 2: Loops and Methods in Java

Due Date: Wednesday, October 13th, 11:59 pm PDT

Learning goals:

- Write Java code to build classes that include:
  - Selections
  - Mathematical operations
  - String and character parsing
  - Loops
  - Methods
  - Scanner
- 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 coding environment. If there is any, then review Assignment 1, or come to the office/lab hours before you start Assignment 2.
2. Download the starter code.
   If you are working on your local machine, then you can download the starter code from Piazza → Resources → Homework → `Assignment2.java`
   If you are working via UCSD Linux Cloud through your CSE 8B account, then use the commands below to copy the starter code to a new directory called `HW2`:

```
$ cd ~
$ mkdir HW2
$ cp ../public/assignments/PA2/Assignment2.java ./HW2
```

   The starter code should only be one file called `Assignment2.java`.

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

```
PS C:\Users\Benson\Documents\GitHub\FA21_CSE8B_PA2> javac Assignment2.java
PS C:\Users\Benson\Documents\GitHub\FA21_CSE8B_PA2> java Assignment2
FAILED: tetrahedronVolume 1
ERROR: Failed test.
```

Part 1: Implement three methods (50 points)

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

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.**

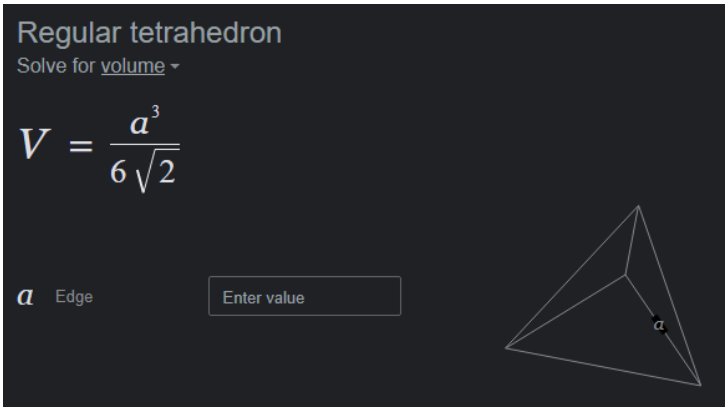
`tetrahedronVolume` (15 points):

This method takes a double variable, `edgeLength`, as input, and the method should calculate the volume of a tetrahedron with that edge length. If the input edge length 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.

HINT: 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"
```

The volume of a tetrahedron is shown below. **We have defined a constant for you that represents the denominator $6\sqrt{2}$ and another constant that can be used to cube a number.**



Sample:

`tetrahedronVolume(12.5)` → `"230.18"`

`tetrahedronVolume(20)` → `"INVALID INPUT: out of range [0, 15]"`

`reciprocalSum` (15 points):

This method takes an integer variable, `value`, as the input, and the method should calculate the sum of reciprocals, or $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ \ldots + \frac{1}{n}$ where `n` = `value`. Similar to `tetrahedronVolume`, 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 *4 digits of accuracy* to the right of the decimal point.

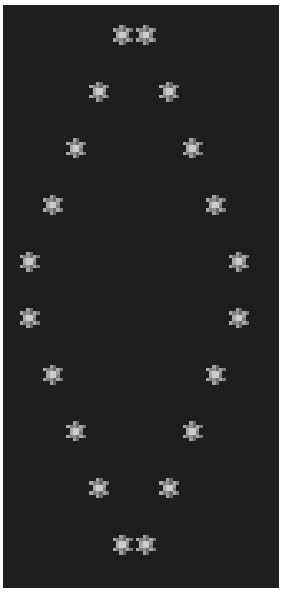
Sample:

`reciprocalSum(12)` → `"3.1032"`

`reciprocalSum(0)` → `"INVALID INPUT: out of range [1, 20]"`

`drawDiamond` (20 points):

This method takes two inputs: a character `ch` and an integer `height`. **You can assume that `ch` will always be a valid character.** However, you will need to validate `height` to ensure that it is between 1 and 10 (inclusive). If it is not between 1 and 10 (inclusive), then return the error message: `"INVALID INPUT: out of range [1, 10]"`. Otherwise, construct and return a String that uses the character `ch` to draw a hollow diamond of which half the number of rows is equal to `height`. For example: when `ch` is `'*'`, and height is `5`, then the result string should be: `"  **\n  *  *\n  *  *\n *  *\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 always has 2 of `ch`, spaced apart depending on `height`. Since `height = 5`, there are 10 total rows. Thus, you can imagine if `height = 1`, then it would be 2 rows with just two characters next to each other.

IMPORTANT: Notice that, in the result string, the newline character `'\n'` is added right after the last `'*'` in each row, in other words, there should be NO spaces after the last `'*'` in each row.

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 create at least four total test cases that cover different situations (including the ones we have provided).** In other words, you will need to create two more tests for `tetrahedronVolume`, three more tests for `reciprocalSum`, and three more tests for `drawDiamond`.

Part 3: Complete `main` (30 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("All unit tests passed.\n");
    } else {
        System.out.println("ERROR: Failed test.\n");
        return;
    }

    // Start the user-machine interaction
    // TODO: Initialize Scanner object

    // Continuously loop until the user inputs "end"
    while (true) {
        // TODO: Complete user-machine interaction here

        break;
    }

    // TODO: Don't forget to close the scanner at the end of the program
}
```

The code to run `unitTests` is already given to you. **Do NOT change any code above the comment "`// Start the user-machine interaction`".**

Below that comment, your final task is to implement an *ask-answer* interaction functionality via command line.

First, your program should print the prompt "`Which method do you want to call? Type \"end\" to stop the program.`" and wait for the user to enter feedback. After the method name is entered, it will read the input method name via Scanner. Then:

1. If the input method name is "`tetrahedronVolume`", the program should print another prompt "`Please enter an edge length (as a double).`". It will then read the user input number as a double variable and call method `tetrahedronVolume` with that variable as the argument `edgeLength`. Finally, the program will print out the result.
2. If the input method name is "`reciprocalSum`", the program should print another prompt "`Please enter an integer.`". It will then read the user input number as an integer variable and call method `reciprocalSum` with that variable as the argument `value`. Finally, the program will print out the result.
3. If the input method name is "`drawDiamond`", the program should print the prompt "`Please enter a character.`". It will then read the user input as a character

variable. As noted before, you can assume that the character variable will always be a valid character. Then, the program should print another prompt `"Please enter a height (as an integer)."` and read the user input as an integer variable. Then, call method `drawDiamond` with argument `ch` and `height`. Finally, the program will print out the result.

4. If the input is `"end"`, return from the main method by breaking out of the infinite loop, which will terminate the execution of the program.
5. If the input is invalid (*e.g.* when input method name is misspelled), print the error message: `"Invalid method - options are: tetrahedronVolume, reciprocalSum, and drawDiamond."`

After performing one of the above, your program should print the prompt `"Which method do you want to call? Type \"end\" to stop the program."` again and repeat the whole process. The repetition should never stop until `"end"` is entered or `Ctrl + C` is pressed to forcibly stop the program.

Example: you should be able to *exactly* reproduce this output below with your program.

```
PS C:\Users\Benson\Downloads> javac Assignment2.java
PS C:\Users\Benson\Downloads> java Assignment2
All unit tests passed.

Which method do you want to call? Type "end" to stop the program.
foo
Invalid method - options are: tetrahedronVolume, reciprocalSum, and drawDiamond.

Which method do you want to call? Type "end" to stop the program.
tetrahedronVolume
Please enter an edge length (as a double).
3.4
4.63

Which method do you want to call? Type "end" to stop the program.
reciprocalSum
Please enter an integer.
14
3.2516

Which method do you want to call? Type "end" to stop the program.
reciprocalSum
Please enter an integer.
30
INVALID INPUT: out of range [1, 20]

Which method do you want to call? Type "end" to stop the program.
drawDiamond
Please enter a character.
^
Please enter a height (as an integer).
3
  ^^
 ^  ^
^    ^
^    ^
 ^  ^
  ^^


Which method do you want to call? Type "end" to stop the program.
end
PS C:\Users\Benson\Downloads>
```

NOTE: Notice that there is an empty line after each method output. It is very important that your program also prints this empty line!

Submission

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

1. Go to Gradescope via Canvas and click on `PA2`.
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 one tester. 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!**

Submit Programming Assignment

Upload all files for your submission

SUBMISSION METHOD

Upload

Add files via Drag & Drop or Browse Files.

NAME	SIZE	PROGRESS
Assignment2.java	**7.7** KB	

SUBMITTING FOR

Benson Budiman

Upload Cancel