Introduction

Hello folks! In the first assignment, you have set up your own Java coding environment and tried to create, compile and run a simple Java program demo. In assignment 2, you will start to do some real coding that covers most contents in lecture 2, 3 and 4 (including elementary programming, selections, mathematical operations, character, strings, loops, and methods). You are also required to come up with your own tests to make sure your code has the correct behavior as described in this writeup.
Note: You should complete this assignment individually.

Part 1: Get started

1. Make sure there is no problem with your Java coding environment. If there is any, review assignment 1 or come to the office hour before you start this assignment.
2. Review lecture 2 - 4 and/or read chapter 2 - 6 of Introduction to Java Programming and Data Structures.
3. Download the starter code.
   If you work with your local machine, download it from Piazza -> Resources -> Homework -> Assignment2.java.
   Start terminal/command line and navigate to the directory that contains Assignment2.java.
   If you decide to use Linux Cloud, use the following command to copy the starter code to a new directory called HW2:

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

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

   $ java Assignment2
   Failed test.
Part 2: Implement three methods

In class Assignment2 of the starter code, 5 methods are already declared: sphereVolume, factorial, drawTriangle, unitTests and main. Your task in this part is to implement the first three methods. But before you implement your methods, make sure to take a look at the constants on the top. All error messages are declared as constant String in Assignment2 class for you. They all start with the keywords “private final static”. Use them to ensure that your code always gives correct output, and you don’t need to worry about misspelling. Also note that \( \pi \) is declared above error messages as \( \pi = 3.1415 \). Use this \( \pi \) in your calculation.

Note: We need to set the input upper bound because we don’t want the calculated result to exceed the maximum value that can be stored in a double or int variable without losing precision. Especially for functions like factorial, the result gets large very fast, so for simplicity and consistency, the range of all numerical inputs are set to \([0, 10]\) in this assignment.

1. sphereVolume:
   This method takes a double variable, radius, as input and should calculate the volume of a sphere with that radius. Then the method should return a string of that volume with 2 digits of accuracy to the right of the decimal point. If the input radius is negative or is larger than 10, the method should return an error message string: “Invalid input: out of range \([0, 10]\)”. Sample:
   sphereVolume(4.2) returns-> “310.33"
   sphereVolume(-1.2) returns-> "Invalid input: out of range [0, 10]"

2. factorial:
   This method takes an int variable, xArg, as input and should calculate the factorial of input, or \((xArg)!\). Similar to sphereVolume, this method should return the result as a string and handle out-of-range inputs (again, the valid range is \([0, 10]\)) by returning the same error message.
   Sample:
   factorial(3) returns-> "6"
   factorial(-1) returns-> "Invalid input: out of range [0, 10]"
3. **drawTriangle**:

This method takes two inputs: a string `singleChar` and an int variable `height`. First, `drawTriangle` needs to validate both inputs in order. If `singleChar` is not a single character, return the error message string: “Invalid input: not a single character”; then if `height` is not in range [0, 10], return the out-of-range error message. If validation passes, construct and return a string that uses character `singleChar` to draw a solid isosceles triangle of which the number of rows equals to `height`. The first row should have one character and the next row should always have 2 more characters than the previous row. For example: when `singleChar` is *, and `height` is 4, the return string should be

```
   *
  ***
 ****
******
```

(\n is the newline character). When this string is printed out, it should look like this:

![Example output for drawTriangle](image)

* **Important**: Notice that in the correct result the newline character “\n” is added right after the last “*”. In other words, space character should only appear before “*”, NOT after “*”.

**Part 3: Test the correctness of three methods**

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

In the starter code, one test case for `sphereVolume` is already given to 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 function 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. To get full credit, for each method, create at least two test cases that cover different situations. (Hint: make sure your methods return correct messages when the input is invalid).
The `unitTests` method should return true only when all the test cases are passed, otherwise return false.

**Part 4: Complete main**

After completing part 3, compile and run Assignment2. You should see the message “All unit tests passed”. If not, it’s very likely that you have bugs in your code. Read part 2 and part 3 instructions carefully while inspecting your code to fix bugs (we call process “debug”).

The main method is the method that will be called when running program Assignment2. In the starter code, the main method looks as below:

```java
public static void main(String[] args) {
    // Perform unitTests first
    if(unitTests()) {
        System.out.println("All unit tests passed.\n");
    } else {
        System.out.println("Failed test.\n");
        return;
    }
}
```

The code to run `unitTests` and print prompt according to the testing result is already given to you. Don’t change any code above the comment “Start the user-machine interaction below”. Below that comment, your last task is to implement a ask-answer interaction functionality via command line.

First, your program should print the prompt “Which method do you want to call?” 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 “sphereVolume”, the program should print another prompt “Please enter radius”. It will then read the user input number as a double variable and call method `sphereVolume` with that variable as the argument `radius` . Finally the program will print out the result.

2. If the input method name is “factorial”, the program should print another prompt “Please enter an integer”. It will then read the user input number as an int variable and call method `factorial` with that variable as the argument `xArg`. Finally the program will print out the result.
3. If the input method name is “drawTriangle”, the program should print the prompt “Please enter a character”. It will then read the user input as a String variable. Then the program should print another prompt “Please enter height” and read the user input as an int variable. Then call method drawTriangle with argument singleChar and height. Finally the program will print out the result.

4. If the input is “end”, return from the main method, which will terminate the execution of the program.

5. If the input is invalid (e.g. when input method name is misspelled), print error message “Invalid method, only options are: sphereVolume, factorial and drawTriangle”.

After performing one of the above, your program should print the prompt “Which method do you want to call?” 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 reproduce this output with your program
Part 5: Coding style

When coding in Java, there are several style rules that people usually follow to make the code clean and readable. In this course, you are asked to follow rules specified in link below:

https://cseweb.ucsd.edu/classes/fa20/cse8B-a/styleguide.html

Read the coding style guide carefully and refine your code for this and all future assignments.
Submission

Very important! Plz follow the instructions below carefully and make the exact submission format. This is important since we will use scripts to grade so if you don't follow the same submission format you probably will receive a zero.

1. Go to Gradescope 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. Just the file solely. Make sure the name of the file is correct.
3. You can resubmit unlimited 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 the use of the instructional team. You won't see the result of the autograder. As long as you uploaded your file you're good to go.