SWE 437 Assignment 2
Fall 2025
Maintenance (Evolution) work—new UI

Due 19-September

Extra features were once considered desirable. We now recognize that “free” features are rarely free. Any increase in generality that does not contribute to reliability, modularity, maintainability, and robustness should be suspect. — Boris Beizer

---

All assignments are due **BEFORE** the due date and time (**11:59PM** September 19th). You may collaborate in teams of 3 as specified on the syllabus or work by yourself. No other team sizes will be considered. You may also request help and advice from your classmates on Piazza. If someone else does this assignment for you, that is an honor violation.

We will use a *compute numeric averages* program for several assignments this semester. For the first step, you will replace its current user interface (UI) with a new UI. It is currently a web application, but you DO NOT NEED TO KNOW HOW TO PROGRAM WEB APPLICATIONS. You will keep the behavioral parts of the software (**the *backend***) the same but replace the UI (**the *frontend***) to be **a stand-alone command line interface on your computer.**

You should make **minimal** changes to the application, i.e., any line, line break, comment, conditional, variable, etc. should NOT be changed unless you need the change for an elegant solution. Please also ensure that you read this specification in its **entirety** before you start and turn this assignment in. The main reason why students lost points on this assignment in the past was because they did NOT fully read this webpage.

Very important: You **do not need** to understand how to program web apps (the program is a Java servlet). Your challenge is to separate the behavioral code from the UI code, design a new UI, and implement the new UI.

The program is a single Java file, computeAverage.java.

Goal of this assignment: Instead of interacting with users through a web app, add the functionality to provide user input through command line arguments. See below for some examples of what your program should output.
You may edit the code in any IDE you wish to use but you should ensure that your submission creates the output exactly as detailed (i.e., when the content of '$' lines are inputted into the terminal with your program, you get back the output of the lines without '$').
You should plan to make minimal changes to the given Java file for it to compile and support the command line interface.

```
$ ls
Assignment2.java
$ javac Assignment2.java


$ java Assignment2 -ln "1 2 3" -sm 2
The Median is: 2.0

$ java Assignment2 -ln "1;2;3" --statistics-mode 2
The Median is: 2.0

$ java Assignment2 -ln "1 2 3" -sm 4
The Mean is: 2.0
The Median is: 2.0
The Mode is: 1 2 3
```

In fact, your program should print the following help message and support all arguments described in the help message.

```
$ java Assignment2
java Assignment2
   Arguments
       -ln, --list-of-numbers
              List of numbers to compute statistics with

       -sm, --statistics-mode
              Statistics to apply to the list of numbers. Options are:
              1. Calculate Mean
              2. Calculate Median
              3. Calculate Mode
              4. Print All Computed Averages

        -h, --help
              Prints this message
```

Note that

- if both -ln and -sm (or their extended versions) are provided, then calculate the statistics according to -sm using the numbers from -ln.
- all other cases should simply output the help message.

Your task is to implement the described functionality in a way that matches the instructor's solution. Think carefully about the different edge cases your solution may have trouble with. Part of your job as a software tester is to identify the edge cases in a given program when you are performing software testing.

To get points for this assignment, you will need to either (1) have your solution match the instructor's solution in all cases that we test for, or (2) you need to document what your program's outputs are and why they make sense for all cases that we test for in the PDF that you turn in. That being said, the more effort you spend on accomplishing (1), the more likely you will be at getting full credit for the assignment. Accomplishing (2) can help give you partial credit but is unlikely to get you full credit.

To help you cover edge cases, you are allowed to email the TA's and I to ask for the outputs of up to **10** inputs.

- The title of your email should be "[SWE437] Assignment 2 inputs".
- You must decide who your group members are before your email and CC all of your group members in the email. If you are working alone, please also state that in your request.
- You may **NOT** change the members of your group after you are part of an email request. That is, please make sure your group members or you working alone is finalized before any edge case request is sent.
- Only one email should be sent for every group (i.e., if you work in a group of 3, you only get 10 inputs and not 30 inputs).

In your email, please attach one file named **Assignment2-LName1LName2LName3.sh** for the 10 commands you would like us to run and **Assignment2-LName1LName2LName3.log** for what you expect the output to be when you run the script on your machine. **Please look over the Assignment2-LName1LName2LName3.log to make sure that running Assignment2-LName1LName2LName3.sh produces what you expect it to before emailing us!**

We will run `javac Assignment2.java` to compile the instructor solution and then run each command in the file you send on a standard Linux terminal and return the output of each command back to you. You can assume that the instructor solution is already compiled before your file is run. I recommend that you run the script/command yourself and ensure that each of the commands behave as you expect it to **before** sending the email with your own `Assignment2.java`! We will not rerun another file of yours due to incorrect class name (Assignment2.java vs. Assignment3.java vs. computeAverage.java), unicode character, line ending, etc. problems. Your file can have **at most 10 lines** and each line should look like the following:

```
java -cp . Assignment2 -ln "1 2 3" -sm 2 # If this line is given to us, we will run it and tell you the output is "The
java -cp . Assignment2 -ln "1 2 3" -sm 4 # If this line is given to us, we will run it tell you the output is "The Mea
...
```

In your email, please include the following sentence
"I confirm that I have ran the command,

`bash Assignment2-LName1LName2LName3.sh > Assignment2-LName1LName2LName3.log`

The file generated from running the command is the attached Assignment2-LName1LName2LName3.log and I have reviewed the contents of the file."

If you send your requests for outputs by **September 12th 11:59PM**, you are guaranteed a response before the deadline. You may still send requests for outputs after this time, but such requests may not be answered.

Be sure to check the line endings or any other special characters (e.g., double check all your quotation marks!) that may be in the edge cases you send and only include them if you want to purposefully test for them. In the past, students have often included weird line endings in their inputs that consequently got weird outputs that they were not expecting.
Your code will be tested on a standard Linux terminal (e.g., on a Ubuntu machine). You are welcome to test your code on other machines (e.g., Windows Linux Subsystem or MacOS) but your goal is to get it to match the behavior of a stardard Linux terminal.

You should **not** rewrite the program. I chose a small program so you can focus on the learning objectives, not so you could go around the assignment's objectives by writing an entirely new program. If you do that, it will be obvious when we look at your source code and will send a message that you are not serious about learning.

Your goal should be to **make as few changes** to the existing code base as possible. You should **NOT** delete anything from the given file, but you may comment out the minimum amount of code to compile the code and add the needed functionality. As you make the changes, **keep a simple documentation log** of what you do. Note which components and methods you change and which ones you no longer use. You should try to change as little as possible. In fact, if you do this well, you will not need much programming. Summarize the changes in a few words as part of your documentation log. You do not need to go to the level of which variables you create or delete or how you change the control flow. I just want the highlights. The document log should be no more than one page.

Submission

Designate one member of your team to submit your answers. If you are working in a team of three and more than one of you submit answers, we will arbitrarily grade one version and you will lose points for turning multiple versions in. Your answers should contain the following (please don't use zip):

- A PDF named **"Assignment2.pdf"** that has the following short sections:
  1. All three partners' first then last names (exactly as they appear on Blackboard) or just your first then last name if you worked by yourself
  2. The simple documentation log
  3. Any edge cases that you came up with, and what an input and output for the edge case is with your program and **why** it makes sense to handle it this way instead of matching the instructor solution
- A Java file with your modified program named **"Assignment2.java"**

If you work in a group of three, everyone must also submit their own one-paragraph collaboration summary named **"Assignment2.txt"**. You need not share this information with each other. The summary should have all partner names and a summary of what everyone contributed.

- Please also answer the following: If you had $100 to pay youself and the other two members, how much should everyone get for their work on this assignment? How much pay one gets should be dependent on how much they contributed to the answers.

- Only one student submits the assignment answers but all students must submit their own collaboration summary. If you are working alone, you need not submit a collaboration summary.

Start ALL assignment answer files with **“Assignment2-LName1LName2LName3,”** that is, include all partners’ last names. Java file names need not have all partners’ last names but you must include all partners’ last names and **“Assignment2”** in comments at the top of any Java file.

Grading

We will grade on several factors.

- (3 pts) Whether each submission item is included
- (4 pts) Whether the modified software correctly supports the new command line interface
- (3 pts) Clarity and thoughtfulness of the documentation log
- (3 pts) Quality, maintainability, and minimality of the changes you made to the code
- (7 pts) Whether the modified software correctly handles all expected inputs

You receive a 5% bonus for submitting by the stated due date. The bonus will help you if you lose points but it cannot bring your assignment score above 100%.