<u>Homework 03 – Classroom 1331</u>

Problem Description

Hello! This HW assignment focuses on **lessons 7 and 8**, but we encourage you to use methods (lesson 9) if you feel comfortable with them. Please make sure to read all parts of this document carefully.

You have been hired as a software developer to help create a way for professors to keep track of where their students are seated in a classroom!

Solution Description

Create a file `Classroom.java` that allows the professor to see where all their students are seated. We have provided you with an initial seating arrangement of the classroom (five by five grid). Create a single Java class called `Classroom` with a main method header. Do **NOT** instantiate a Scanner object outside of the main method.

`2D Array Chart`

The following String values must be stored within a 2D array called **classroom.** Create the 2D array with the values from the following chart. Each cell visually represents a seat. X represents that no student is initially seated in that seat, use **null** to store the seats without students.

- Do NOT assign the value of the 2D array outside of main.
- Assign it inside of the main method at the beginning of the method.
    - Every time main gets run the value must be set again to the initial configuration. The autograder may run the main method multiple times in a single execution.

<u>Provided 2D Array/seating arrangement</u>

x	Amanda	x	x	Karishma
Ignacio	x	Kelly	x	x
x	x	x	Lucas	x
x	x	Shreya	x	Makiyah
x	Rita	x	x	Gautam

Name your program `Classroom`, and it should work as follows:

Program Description

- Part 1: The program must be able to receive command line arguments, which correspond to students that have arrived late to the class. You must add those students to the 2D array before doing other operations.
    - Each string in the command line arguments will represent a student and a location, with the format "[name] [row] [column]"

        - You may assume the name doesn't have any spaces and that row/column are valid integers and that the seat is currently empty. Students that arrive late will choose different seats.
        - All students (from initial configuration and arrived late) have unique names.
    - The command line arguments may contain 0 or more students.
- Part 2: The program will do the following operations until the user decides to finish the program.
    - Prompt and ask the user if they want to find a student: "Would you like to find a student? [Y]es or [N]o
        - If the answer is exactly "N" or "No" (case INSENSITIVE), move to part 3. All other responses are considered as yes
    - Otherwise, ask for the name of the student with the sentence: "Please provide the name of the student you would like to find." (case SENSITIVE)
        - If the student doesn't exist, print "This student is not in the classroom!" and begin part 2 again
        - Ask the user if they want to move the student or indicate the student has left with the following sentence: "Would you like to move the student or indicate that they have left? [M]ove or [L]eft"
        - If they input "M" (case INSENSITIVE) follow the instructions in **"Moving a student"** and repeat part 2
        - If they input "L" (case INSENSITIVE) follow the instructions in **"Student leaves the classroom"** and repeat part 2
- Part 3: Print "Final configuration of the classroom:" and print the 2d array with the format that follows. If no student is located at a seat, put a single space in that location.

  Example of printed 2D array (as initial configuration)

```
| |Amanda| | |Karishma|
|Ignacio| |Kelly| | |
| | | |Lucas| |
| | |Shreya| |Makiyah|
| |Rita| | |Gautam|
```

    - Print "Thanks for using our software to track your class!" and then exit the program. Do NOT use System.exit.

Moving a student:

- Ask for a new location with the following sentence: "Please select a location to move this student to."
- If the location is not valid (whether any of the elements isn't an integer or if it's not a valid location in the array): print "Please enter a valid location." and wait for a new location
- If the location is valid but a student is present, print "This is a valid seat location, but student [name] is seated there! Please select a different location." and wait for a new location

    - You may assume that a student will always move to a location different from their current location.
- Otherwise, move the student from one location in the 2D array to the new one and print: "[Student Name] has been successfully moved!"

Student leaves the classroom:

- Remove the student from the 2D array and print, "Student [name], located at [row] [column], left the classroom!"

Example Output #1 - User input is bolded.

(Your program should look **exactly** like this with the exact spacings and lines! If there is any deviation, you will l**ose points**)

args = [] (empty array)

```
Would you like to find a student? [Y]es or [N]o
Y

Please provide the name of the student you would like to find.
Karishma

Would you like to move the student or indicate that they have left? [M]ove
or [L]eft
M

Please select a location to move this student to.
0 5

Please enter a valid location.
0 4

This is a valid seat location, but student Karishma is seated there! Please
select a different location.
0 3

Karishma has been successfully moved!

Would you like to find a student? [Y]es or [N]o
N

Final configuration of the classroom:
```

```
| |Amanda| |Karishma| |
|Ignacio| |Kelly| | |
| | | |Lucas| |
| | |Shreya| |Makiyah|
| |Rita| | |Gautam|

Thanks for using our software to track your class!
```

Example Output #2 - User input is bolded.

(Your program should look **exactly** like this with the exact spacings and lines! If there is any deviation, you will **lose points**)

args = [“Gon 2 0”, “Killua 1 4”]

```
Would you like to find a student? [Y]es or [N]o
Y

Please provide the name of the student you would like to find.
Ignacio

Would you like to move the student or indicate that they have left? [M]ove
or [L]eft
m

Please select a location to move this student to.
0 -3

Please enter a valid location.
0 4.5

Please enter a valid location.
hi bye

Please enter a valid location.
0 0

Ignacio has been successfully moved!

Would you like to find a student? [Y]es or [N]o
Y
```

```
Please provide the name of the student you would like to find.
lucas

This student is not in the classroom!

Would you like to find a student? [Y]es or [N]o
Y

Please provide the name of the student you would like to find.
Lucas

Would you like to move the student or indicate that they have left? [M]ove
or [L]eft
L

Student Lucas, located at 2 3, left the classroom!

Would you like to find a student? [Y]es or [N]o
N

Final configuration of the classroom:

|Ignacio|Amanda| | |Karishma|
| | |Kelly| |Killua|
|Gon| | | | |
| | |Shreya| |Makiyah|
| |Rita| | |Gautam|

Thanks for using our software to track your class!
```

Rubric

[100] Classroom.java

- [20] Correctly updates array
- [10] Starts and exits based on Y/N
- [15] Correctly prompts the user at all locations
- [5] User correctly adds students
- [10] User correctly drops students
- [10] User correctly moves students
- [10] Correct display of array

- [20] Correct program behavior and output

Allowed Imports

To prevent trivialization of the assignment, you are **only allowed to import:**

- `java.util.Scanner`

The Checkstyle cap for this homework assignment is 5. Up to 5 points can be lost from Checkstyle errors.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`

Collaboration

Collaboration Statement

To ensure that you acknowledge a collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one .java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Allowed Collaboration

When completing homeworks for CS1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homeworks
- Parts of the homework you are unsure of and need more explanation
- Online resources that helped you find a solution
- Key course concepts and Java language features used in your solution

You may **not** discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

Examples of approved/disapproved collaboration:

- **approved**: "Hey, I'm really confused on how we are supposed to implement this part of the homework. What strategies/resources did you use to solve it?"

- **disapproved**: "Hey, it's 10:40 on Thursday... Can I see your code? I won't copy it directly I promise"

In addition to the above rules, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due.

Turn-In Procedure

Submission

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

- Classroom.java

Make sure you see the message stating "HW03 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

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: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds 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.

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.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for a note containing all official clarifications