CSCI 3302 Programming Assignment 04 (100 Points)

Due: Oct 17, 11:59 PM

OBJECTIVES:

- Demonstrate Java programming proficiency in the use of double arrays.
- Demonstrate reading/writing from a file in Java.
- Demonstrate Java programming proficiency for using stacks to solve problems.

ASSIGNMENT ASSISTANCE:

- This homework assignment is due before the date and time specified above.
- This assignment is restricted to individual effort. As per our syllabus, the use of AI is prohibited. You may not receive help from any person except the instructor or the AARC (help from the AARC must be well documented!).
- Any resource used (other than Dr. Becnel or the course text) must be documented in the code (as comments) detailing the source and describing exactly what was learned and how that information was used. Submissions will be severely penalized if copied in part or whole from any source.
- If you need help, visit your instructor during his posted office hours. If your schedule cannot accommodate any of these times, then email your instructor to schedule a different time.

PROBLEM DESCRIPTION:

1. In this assignment, you will use Stacks and double arrays to solve the following problem: People are seated in a movie theatre. One person becomes ill and infects the people sitting around them. These people, in turn, infect the people adjacent to them. The goal is to find the number of infected individuals and note the status of all seats in the theatre.

2. Problem Representation: The seats and people are represented by 1's and 0's in a matrix/table configuration. For example,

	0	1	2	3	4	5
0	1	1	0	0	0	0
1	0	1	1	0	1	1
2	0	0	0	0	1	1
3	1	1	0	0	0	1

Seat locations are denoted using rows and columns. The above has rows 0, 1, 2, 3 and columns 0, 1, 2, 3, 4, 5. Seat occupancy is denoted with 0 and 1:

- A 1 denotes that someone is in the seat (occupied).
- A 0 denotes that no one is in the seat (vacant).

Continuing this example, let us assume the person at the seat (1,2) gets infected.

	0	1	2	3	4	5
0	1	1	0	0	0	0
1	0	1	X	0	1	1
2	0	0	0	0	1	1
3	1	1	0	0	0	1

This person infects the person at (1,1).

	0	1	2	3	4	5
0	1	1	0	0	0	0
1	0	X	X	0	1	1
2	0	0	0	0	1	1
3	1	1	0	0	0	1

The person at (1,1) infects the people at (0,0) and (0,1).

	0	1	2	3	4	5
0	X	X	0	0	0	0
1	0	X	X	0	1	1
2	0	0	0	0	1	1
3	1	1	0	0	0	1

Thus 4 total people are infected, which are represented in red above.

3. To solve this problem create a file called `Seat`.java with the following:
    a. Enumerable: The class should have a private enumerable called `Status` with three options: `VACANT, OCCUPIED, INFECTED`.
        - `VACANT` – no one is sitting in the seat
        - `OCCUPIED` – someone is sitting in the seat and not currently infected
        - `INFECTED` – someone is sitting in the seat and is infected

        If you are unfamiliar with enumerable, here is a guide: https://www.baeldung.com/a-guide-to-java-enums
    b. Fields
        - `row` – an integer representing the row location of the seat
        - `column` – an integer representing the column location of the seat
        - `status` – a field of type `Status` (the enumerable) to hold the status of the seat.

        The fields should be `private` and the row/column should be made immutable with the `final` keyword.
    c. Constructor: One public constructor that takes arguments `row`, `column` (as integers), and a boolean `vacant`. The constructor sets the `row` and `column` fields based on the arguments and sets the `status` to `VACANT` or `OCCUPIED` based on the boolean value.

d. Methods:
   - `getRow/getColumn` – return an integer representing the row/column of the seat
   - `isVacant` – returns true if the status is `VACANT`, false otherwise
   - `isInfected` – returns true if the status is `INFECTED`, false otherwise
   - `isOccupied` – returns true if the status is `OCCUPIED`, false otherwise
   - `infect` – a method that sets the status of the seat to infected if the seat is occupied. If the seat is vacant, nothing happens.

e. You may add helper methods and a `toString`(), but these are not considered when grading.

4. Create another file called `Contagion`.java with the following fields and methods:
   a. Fields:
      - `seats` – a `private` double array of type `Seat` (from the previous step)

   b. Constructor: One public constructor that takes as input a `String` called `infile`. The constructor reads from the file to create the double array of `seats`. The format of the file is
      - First line: number of rows and numbers of columns
      - The remainder of the file is 0 and 1, with 1 representing an occupied seat and 0 a vacant seat.
      - The example above is provided for you in the file seating_ex.txt. The contents are as follows:

```
4   6

1    1      0      0      0      0

0    1      1      0      1      1

0    0      0      0      1      1

1    1      0      0      0      1
```

      - Wrap the code in a try/catch and print a message in the catch. Do not throw any exceptions.

   c. Methods:
      - `infect` – this public method takes two integer arguments. The first is `row` and the second is `column` representing the seat location. If the seat is vacant, then nothing happens. If the seat at the given position is occupied, the person in the seat becomes infected and infects those around them (a total of 8 possible adjacent seats), who in turn infect other adjacent individuals.

        This method must use the built-in Java `Stack` class.
        https://docs.oracle.com/javase/8/docs/api/java/util/Stack.html

Hint: Create an empty stack and put the infected seat in the stack. As long as the stack is not empty acquire an infected seat from the stack and consider their neighbors. When a neighbor becomes infected, add them to the stack.

- `isInfected` – this public method takes two integer arguments. The first is `row` and the second is `column` representing the seat location. This method returns a boolean: `true` if the seat has an infected individual and `false` otherwise.

- `countInfections` – this public takes no arguments and returns an integer representing the number of infected individuals in the seats.

- `write` – the public method takes a `String` called `outfile` and writes the status of the seats to the file. If a seat is vacant, a 0 is written to the file. If the seat is occupied by an uninfected individual, a 1 is written to the file. If an infected individual is in the seat, then an X is written to the file. In the example

	0	1	2	3	4	5
0	X	X	0	0	0	0
1	0	X	X	0	1	1
2	0	0	0	0	1	1
3	1	1	0	0	0	1

the contents of the file would be

```
X X 0 0 0 0
0 X X 0 1 1
0 0 0 0 1 1
1 1 0 0 0 1
```

5. You are not responsible for exception handling other than those required to read/write from files.

6. A simple test file is provided for you. However, you are responsible for extensively testing your program.

7. Your program should work in the GitHub codespace (Linux environment) and locally (Windows environment).

SUBMISSION:

- Review the rubric below to ensure you have met all the requirements.

- Commit electronic copy of `Seat.java, Contagion.java` to GitHub. Upload a backup copy to D2L. You may have testing files in your repository, but they will not be considered for grading.

EVALUATION

Remember to consult Program Requirements.docx

Automatic Deductions:	
Late/Not Submitted	-100
Code not submitted to GitHub	-30
Code does not run/compile	-50
Stack not used to complete the assignment	-50
Earn Points for the following:	
Code has a comment header with name, section, date	5 pts
Code organization, structure, and indention are appropriate (SHFT + ALT + F in VS Code)	5 pts
Enumerable used correctly in Seat	5 pts
Fields and Constructor correct in Seat	5 pts
Methods correct in Seat	10 pts
Fields correct in Contagion	5 pts
Constructor in Contagion sets fields correctly	10 pts
isInfected method in Contagion	5 pts
countInfections method in Contagion	20 pts
write method in Contagion	10 pts
infect method in Contagion (must use Stack)	20 pts