Grid Monitor:
Avoiding an Unscheduled Space Walk

Objectives

- Read table data from files
- Create and manipulate two-dimensional arrays

Background

Celebrated engineer, M. Scott, has invented a revolutionary new solar array, which is theoretically capable of fully powering a Constitution-class starship operating within eight light-minutes of a G-type main-sequence star. The problem is that mysterious interactions between adjacent cells in prototypes have caused... well... some catastrophic explosions in (former) labs that would certainly result in hull breach if fitted onto a ship. It is believed that any cell with a value that differs from the average of its neighboring cells by more than 50% is at risk of exploding. You have been tasked with building a system to monitor array levels for these dangerous imbalances.

In this very small 3x3 example array, these are the last reported levels just before one of the cells in row 0 went nuclear:

2.0	10.0	7.0
4.0	5.0	8.0
5.0	6.0	9.0

If you sum the values of the surrounding cells for each cell, you get this table:

18.0	24.0	32.0
16.0	28.0	29.0
20.0	25.0	32.0

These sums were calculated as follows:

- For each cell of the array:
  - Sum the values of the neighboring four cells:
    - the cell above
    - the cell below
    - the cell to the left
    - the cell to the right
  - For cells on the border, which don't have four neighboring cells:
    - Replace the values of the missing cells with the value of the cell itself

For example:

Sum for the middle cell (1,1) =
= value above (0,1) + value below (2,1) + value left (0,1) + value right (1,2)

= 10.0 + 6.0 + 4.0 + 8.0
= 28.0
Sum for the cell in the upper lefthand corner (0,0) =
= its own value + below (1,0) + its own value + value right (0,1)
= 2.0 + 4.0 + 2.0 + 10.0
= 18.0

To calculate the average of the surrounding values, divide each of the sums by 4:

4.5	6.0	8.0
4.0	7.0	7.25
5.0	6.25	8.0

A cell may explode if its value deviates from its surrounding average by more than 50%. To calculate the maximum deltas, divide each of the averages by 2:

2.25	3.0	4.0
2.0	3.5	3.625
2.5	3.125	4.0

Therefore, the safe range of operation for each of these cells is the average of the surrounding cells plus/minus these delta values:

2.25 - 6.75	3.0 - 9.0	4.0 - 12.0
2.0 - 6.0	3.5 - 10.5	3.625 - 10.875
2.5 - 7.5	3.125 - 9.375	4.0 - 12.0

Comparing the original grid values to the safe operation range of each cell, we can identify which cells are in danger of exploding and hopefully shut them down before the crew and cargo of the fitted ship are abruptly vented into space:

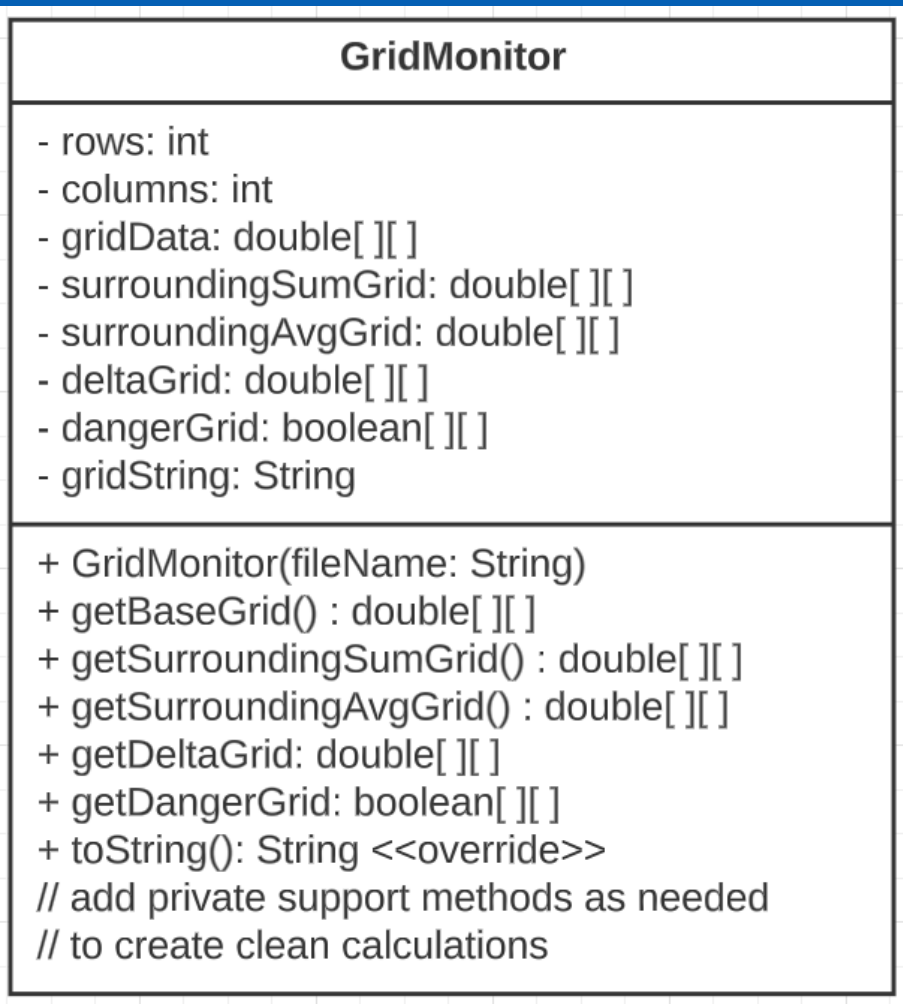
true	true	false
false	false	false
false	false	false

Your superior officer has defined an interface, called `GridMonitorInterface`, which specifies methods that will be required of the monitoring system. He has also started a test class called `GridMonitorTest` that will confirm that these methods operate as required. He has chosen, however, to delegate actual implementation and testing of the monitoring class to you, one of his dwindling supply of red shirt subordinates, while he takes a few months of accrued leave.

To complete the grid-monitor system, you will create a class called `GridMonitor` that implements `GridMonitorInterface`. Use driver class `GridMonitorTest` to test your `GridMonitor` and be reasonably sure that it works, since your life could depend on it.

Give it all you've got, Engineer!

UML



Tasks

1. Download the following files:
   - GridMonitorInterface.java - the interface that must be implemented by your `GridMonitor` class.
   - Sample files - sample input files in the format that your `GridMonitor` must be able to read.
     - First line - row and column dimensions, in that order, of the two-dimensional array as integer values
     - Subsequent lines - a single row of double-valued levels from the grid
   - GridMonitorTest.java - a test class that confirms correct operation of `GridMonitor` methods for a limited set of use cases.
2. Write the `GridMonitor` class, which must implement `GridMonitorInterface` and include a constructor with the following signature:

   public GridMonitor(String filename) throws FileNotFoundException

   The constructor should attempt to open and read the specified plain-text file containing current grid levels using one or more `Scanner`s. The `Scanner` class will throw

a `FileNotFoundException` if the given file name is not valid or the file cannot be read. Do *not* catch this exception. Let it go.

3. Test your `GridMonitor` class with the provided `GridMonitorTest` driver class and the sample files. Look at the situations being tested and consider if there are other input files with dimensions or values you think should also be tested before you would get on a ship outfitted with your GridMonitor.
4. Document your project:
    - Edit the plain-text README file, following the format described in the README.txt file.
    - Comment your code appropriately; assume someone may be looking at your code and not know what you are trying to accomplish.

Documentation: README

Update the plain text file called README and write your name and class section on the top.

1. Fill out the **Overview** section
   Concisely explain what the program does. If this exceeds a couple of sentences, you are going too far. I do not want you to just cut and paste, but paraphrase what is stated in the project specification.
2. Fill out the **Compiling and Using** section
   This section should tell the user how to compile and run your code. It is also appropriate to instruct the user how to use your code. Does your program require user input? If so, what does your user need to know about it to use it as quickly as possible?
3. Fill out the **Discussion** section
   Think about and answer the following reflection questions as completely as you can. You may not have an “earth-shattering” reflection response to each one but you should be as thoughtful as you can.
   Reflections...
    - What problems did you have? What went well?
    - What process did you go through to create the program?
    - What did you have to research and learn that was new?
    - What kinds of errors did you get? How did you fix them?
    - What parts of the project did you find challenging?
    - Is there anything that finally "clicked" for you in the process of working on this code?
    - Is there anything that you would change about the lab?
    - Can you apply what you learned in this lab to future projects?
4. Fill out the **Testing** section
   You are expected to test your projects before submitting them for final grading. Pretend that your instructor is your manager or customer at work. How did you ensure that you are delivering a working solution to the requirements?

Grading

Points will be awarded according to the following breakdown:

Tasks	Points
Documentation: README file, comments	10
`GridMonitor` functionality	30
Quality - code formatting, naming conventions, encapsulation, etc.	10

Required Files

Submit the following files:

- README.txt
- GridMonitor.java
- GridMonitorTest.java
- GridMonitorInterface.java.