Software Development

Mini Assignment 1
Due: September 13, 2018 at 23:55 on MyCourses

This mini assignment practices the software techniques covered in lecture #2 and some things you saw from COMP 250. Primarily this will be a Java programming assignment that explores the concept of writing well designed software as presented in class. You may be aware of other software techniques. Please do not use them. Just focus on what you have seen from Lecture #2 and supporting ideas from COMP 250, like Big Oh and how to program in Java. Lecture #2 introduced the idea that writing well designed software includes the following: optimality in speed (Big Oh), optimality in memory (min bytes), simplicity of solution, correctness, robustness (hard to make it crash), easy to read code, and commenting as documentation.

Write a well-designed software solution for the following problem:

An application uses a 2D character matrix as a map. The map has a max size, maxRow and maxWidth. The map can be populated with a fixed set of characters: '~' for water, 'G' for grass, and '#' for tree. No other characters are permitted. The application begins by asking the user for the size of the map. In other words, the integer values for maxRow and maxWidth. These values must be greater than 0. The matrix is instantiated at runtime, and it is initialized with the water character. Validate whether the matrix was successfully instantiated. After this, the program, in a loop, asks the user for a row and column coordinate, and a character. The program validates the inputs displaying an error message if anything is incorrect, and then prompts the user to provide another input tuple (ie. row, column, character). If the inputted values are correct then the input character is placed into the map at the specified coordinate, overwriting whatever was there. The program then asks the user if they would like to input another character. The application allows the user to input as many characters as they want. If the user does not want to enter more characters, then the program displays the final ASCII map to the console and terminates. The 2D character matrix is within a class called Map (Map.java). The user interface loop will be in the main() method in a class called MapMain (MapMain.java).

The output of the program is as follows:

```
PROMPT $ ./java MapMain.class
Welcome to Map!
Please input the maximum number of rows: 0
A row must be greater than 0.
Please input the maximum number of rows: 10
Please input the maximum number of columns: 5
Map has been created.
Please add an object to the map (~ for water, G for grass, # for tree)
Row: 12
```

```
Invalid row! It must be between 0 and 9.
Row: 9
Column: -1
Invalid column! It must be between 0 and 4.
Column: 3
Character: X
Invalid character! It must be either ~ or G or #.
Character: W
Invalid character! It must be either ~ or G or #.
Character: G
Your G was added to 9,3 in the map.
Would you like to enter another character (Y/N): N
~~~~~

~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
~~~~~
~~~G~
PROMPT $
```

The program has now terminated.

Your program will use three Java source files. One source file will contain all the map code (Map.java). The second file will contain the user interface code (MapMain.java). The last file will contain the testing code (MapTest.java).

MapMain.java and MapTest.java use Map.java. MapMain.java has a main method. MapTest.java also has a main method. They both share the Map.java file. Hence, you are creating two programs. One program, MapMain.java and Map.java is the application described above. The other program, MapTest.java and Map.java, will be used to show the correctness of your Map.java code, automatically and exhaustively.

Do **not** use tools like jUnit and Java Doc. You are just writing simple Java code. You are building things from scratch.

MapMain.java behaves as described above.

Map.java stores the matrix and the maxRow and maxWidth variables as private. It has a constructor and public methods to satisfy the problem description. It has an optimal number of methods. It uses optimal algorithms and data structures. You will need to determine the optimality and correctness of your algorithm and data structure choices. Think carefully.

MapTest.java has a main method that will exhaustively test every method in Map.java. To do this correctly, each method from Map.java will have a corresponding testing method in MapTest.java. For example, if in Map.java we have a method named abc() then in MapTest.java we have a method called abcTest() that exhaustively tests abc(). Each test method will exhaustively test the entire range of possible parameter values for the corresponding method using valid, invalid and boundary cases. The test method outputs to the screen the arguments passed to the method and the value returned by the method. This output is used by the developer to verify the correctness of the method. Make your output look something like this:

```
Test case for method <<method name goes here>> :

Arguments: <<all arguments for call>> Result: <<result from call>>

Arguments: <<all arguments for call>> Result: <<result from call>>

Etc.

Test case for method <<next method name goes here>> :

Arguments: <<all arguments for call>> Result: <<result from call>>

Arguments: <<all arguments for call>> Result: <<result from call>>

Etc.
```

You are creating a well-designed program. I expect to see optimality, pretty code, and comments as documentation.

WHAT TO HAND IN

- MapMain.java
- MapTest.java
- Map.java

HOW IT WILL BE GRADED

For your assignment to be graded your program (a) must run, (b) did not use tools, and (c) followed the assignment instructions. You are doing this assignment on your own.

- +5 - Optimality (Big Oh)
- +5 - Optimality (memory)
- +5 - Simplicity of solution (fewest lines of code + not convoluted code)
- +5 – Correctness (all files) (never wrong, never crashes)
- +5 - Well written code that is easy to read (see lecture 2 for examples + pretty code)
- +5 - Comments as documentation (see lecture 2 for examples + explain what is needed well)