TCSS 143 A Winter 2019, Project 2

Given the problem statement below, complete the following:

- ✓ A class for CrosswordPuzzle ADT that meets all requirements listed in this description. In order to receive credit, your code MUST (80 pts):
  - compile and run in jGrasp
  - use a CrosswordPuzzle class with a 2d array of characters representing the crossword grid and a 1d array of Strings representing crossword words
  - and be compatible with the instructor-provided driver
- ✓ Coding style and comments
  - Coding style (meaningful identifiers, indentation, etc.) and the header with your name and course info at the top of the file (10 pts)
  - Comments (10 points)
    - Javadocs for every method and class
    - Inline comments explaining code logic

You are NOT allowed to help one another with this program or use somebody else's code – check the rules listed on the syllabus. However, you may seek help from CSS mentors or lab/lecture instructors.

Problem statement

In this program, you are to design and implement a program that processes text files representing crossword puzzles and calculates various statistics. Crossword users typically use such statistics to determine the difficulty level of a puzzle.

It is mandatory that you create a CrosswordPuzzle class in order to solve the problem, with a 2d array of of characters representing the crossword grid and a 1d array of Strings representing all the words in the puzzle. Your CrosswordPuzzle class needs to be compatible with `CrosswordDriver.java` - provided in `Canvas`. The set of CrosswordPuzzle commands your application must support is:

Command	Meaning
`CrosswordPuzzle(x)`	Creates a 1d array of strings of some default size and a 2d array of characters with x rows
`readFile(inputfile)`	Reads the crossword puzzle lines from a file and populates a 2d array of characters
`extractWords()`	Extracts all horizontal and vertical words from a 2d array (a word must have at least 2 characters) – this method uses 2 private method helpers: extractHorizontalWords() and extractVerticalWords() to populate the 1d array of Strings
`displayWords()`	Writes all the words from a 1d array of words to console
`displayBoard(outputStream)`	Writes the 2d array of characters to the outputStream; a blank character follows each original character from the 2d array
`rotate90()`	Rotates the matrix 90 degrees
`displayStats()`	Displays crossword stats to console, as shown in the sample run
`countBlackSquares()`	Counts and returns the number of black squares in the crossword puzzle
`averageWordLength()`	Calculates and returns an average word length

Additional Notes

- The first line of the input file (e.g. *crosswords1.txt*) contains the number of rows for the crossword puzzle
- The input file is consistent with the idea that each row in the crossword puzzle has the same number of characters (including letters and black spaces)
- You may assume that all entries will be valid – no input validation required
- Your class is to support proper encapsulation, i.e. client code should not be able to modify any class or object fields directly but only through provided methods.
- Test your code by modifying the provided driver and using print statements and the debugger.

Specifications

Name your class file `CrosswordPuzzle.java` and include your name, course and section numbers, and project number inside your file as a comment. If your code is not fully functional, you need to provide appropriate comments in the `CrosswordPuzzle.java` file and comment out the commands that do not work correctly.

Use appropriate comments, coding style, and meaningful identifiers throughout this program.

Your code should be organized in the following manner:

- All fields are declared before all constructors.
- All constructors are declared before all other methods.
- All static fields are declared before all non-static fields.
- All static methods are declared before all non-static methods.
- Among the same type of entity (constructors, instance methods, static methods, instance fields, static fields), the order for information hiding modifiers should be public, then private.

Extra Credit

You may earn extra credit of up to 10 points for this assignment. Extra credit is given at the discretion of the instructor and graded more strictly than the regular parts of the program. Extra credit can only add to the existing functionality and cannot modify any of the required methods.

Add other interesting functionality (e.g. based on a list of words, generate your own crossword puzzle, determine puzzle difficulty by comparing the words in the puzzle to the list of words along with their rankings – file *wordlist.csv* with ranking provided in *Canvas,* calculate difficulty level by assigning point values to letters – like in Scrabble – and return the sum

(1 point) - A, E, I, O, U, L, N, S, T, R
(2 points) - D, G
(3 points) - B, C, M, P
(4 points) - F, H, V, W, Y
(5 points) - K
(8 points) - J, X
(10 points) - Q, Z)

If you complete extra credit, you must create your own `CrosswordDriverEC.java` that includes code showcasing your code's capabilities.

Sample run

```
How many files would you like to process?
▶▶ 1
   Enter the name of the file:
▶▶ crosswords1.txt
   Original puzzle:
```

```
B A A # # N C #

U # L # # O H M

S O L I D # I #

# # O # O # L #

# # W I T H I N

Extracted words:
BAA
NC
OHM
SOLID
WITHIN
BUS
ALLOW
DOT
NO
CHILI

Rotated puzzle:
B U S # #

A # O # #

A L L O W

# # I # I

# # D O T

N O # # H

C H I L I

# M # # N

number of words: 10
number of black squares: 15
and their percentage of the grid: 37.50
shortest word: 2
longest word: 6
average word length: 3.70
```

Project Submission

On or before the due date, use the link posted in *Canvas* next to *project 2* to submit your `CrosswordPuzzle.java`, `CrosswordDriver.java`, and if you complete extra credit, `CrosswordDriverEC.java` with necessary input files.