# CS4222 Software Development – Programming Assignment

### Introduction

For this project you are working on a Word Search puzzle and developing a class that will be used to generate puzzles using a two-dimensional grid. A skeletal class for the puzzle is shown below

public class WordSearchPuzzle {

private char[][] puzzle ;

private List<String> puzzleWords ;

public WordSearchPuzzle(List<String> userSpecifiedWords) {

// puzzle generation using user specified words

// The user passes in a list of words to be used

// for generating the word search grid.

}

public WordSearchPuzzle(String wordFile, int wordCount,

int shortest, int longest) {

// puzzle generation using words from a file

// The user supplies the filename. In the file

// the words should appear one per line.

// The wordCount specifies the number of words

// to (randomly) select from the file for use in

// the puzzle.

// shortest and longest specify the shortest

// word length to be used and longest specifies

// the longest word length to be used.

// SO, using the words in the file randomly select

// wordCount words with lengths between shortest

// and longest.

}

// The dimensions of the puzzle grid should be set

// by summing the lengths of the words being used in the

// puzzle and multiplying the sum by 1.5 or 1.75

// or some other (appropriate) scaling factor to

// ensure that the grid will have enough additional

// characters to obscure the puzzle words. Once

// you have calculated how many characters you are

// going to have in the grid you can calculate the

// grid dimensions by getting the square root (rounded up)

// of the character total.

//

// You will also need to add the methods specified below

}

### Methods Required

The class should support/provide the following operations

* The class provides an operation that returns the list of words used in the puzzle. The header is as follows

public List<String> getWordSearchList()

* The class provides an operation that returns the generated grid as a two-dimensional array of characters. The header is as follows

public char[][] getPuzzleAsGrid()

* The class provides an operation that returns the generated grid as a String with newlines (i.e. \n) at the end of each line/row of the grid. The header is as follows

public String getPuzzleAsString()

* The class provides an operation that displays the grid and the list of words used to create it. The header is as follows

public void showWordSearchPuzzle(boolean hide)

If the hide parameter is false the grid display should include details of where each word is located on the grid AND the orientation used to place it on the grid (i.e. left to right horizontally, top down vertically, right to left horizontally, and so on). If the hide parameter is true then only the words placed on the grid should be displayed and the word location information should be suppressed. (See examples below. NOTE: They are **examples** – feel free to design your own grid display layout.)

* The class should include a private method with the following header

private void generateWordSearchPuzzle()

This method is responsible for implementing the creation of the puzzle by coordinating the following operations

* 1. If a list of words has not been supplied by the user then you need to select the words to be used in the puzzle from the dictionary specified by the user. Remember, the number of words to be used and the lengths of the longest and shortest words allowable are passed to the constructor.
  2. Randomly place each word on the grid. Placing a word involves selecting a position (i.e. a row and column pair) and a direction (i.e. horizontal left to right, horizontal right to left, vertical top to bottom, vertical bottom to top). The selected position and direction should have sufficient empty positions to allow the word to be placed (i.e. you cannot overwrite a word already placed). The word can be placed on to the grid by placing each individual character in the appropriate grid position (as determined by the direction).
  3. When all the words have been placed you should fill any remaining grid positions with randomly chosen letters.
  4. It is NOT A REQUIREMENT but you may want to allow interlocking words on the grid. Please note, this makes the project more interesting or challenging or demanding but you might like to consider it anyway. There are NO MARKS for adding an interlocking words feature. In addition, you could introduce “diagonal” word placement (i.e. top-left to bottom right, bottom-left to top-right, and so on).
  5. It is entirely a matter for you as the designer of the solution to use any additional methods YOU feel are necessary to solve the problem.

### Submission Requirements

You have the option of undertaking the project alone or as part of a team. If you wish to form a team please email me by **16h00 Thursday 21 March 2019** with the names and ID numbers of the team members. Teams are restricted to a maximum of four members. All team members are awarded the same score for the project. Management of the team is entirely a matter for the team members.

In addition to the WordSearchPuzzle class you will need to write a Driver program to test the WordSearchPuzzle class. The Driver program should (at least) create several instances of the WordSearchPuzzle class and use the various methods provided by it. You should also try it with a number of word lists and word files.

**NB:** In **EACH AND EVERY** one of the .java files you submit you should include prominent comments in the code that contain the ID NUMBER(s) and NAME(s) of (all of) the author(s).

An electronic copy of ALL of the .java and other files **MUST** be uploaded to your directory for the module stored on [\\srvr\csis](file:///\\srvr\csis). You should create a subdirectory called ASSIGNMENT and place in it a copy of ALL of the files necessary to run your solution.

**NOTE: ONLY .java files and any text files required should be placed in your ASSIGNMENT directory. No exe’s, tar’s or any other file type should be uploaded. It is YOUR responsibility to ensure that the files submitted are the correct versions. The author(s) of a submission may be invited to attend a meeting to explain, justify or describe aspects of the solution. It should NOT be necessary to install additional software to test your submission.**

Submissions must be received on or before **16h00 Thursday 11 April 2019 (GMT).**

**This project accounts for 25% of the overall marks available for the module.**

**Sample Output**

PUZZLE

------

The puzzle with unused positions filled with random characters...

BFNEERGR

RAVNHBJA

ICREAQEK

DEPCMKOO

GNOKMVZV

EGWWEKWZ

PSEFRFSM

TVRLIAND

The puzzle words in alphabetical order...

BRIDGE

FACE

GREEN

HAMMER

NAIL

NECK

POWER

PUZZLE

------

The puzzle with unused positions filled with random characters...

TUBCHMHTYHR

AQESPLANTLR

DSVALUESBIM

ENDKBEXPOEE

XOARGTSONRL

IWNUYIASROM

MTGLRBNSCFG

ZHEERUDIWEX

NIRHEVLBHBF

GNKBBHBLIAB

UGLLUFLEPIZ

The puzzle words in alphabetical order showing

their [row][col] positions and directions...

BEFORE[8][9]U

BERRY[9][4]U

BITE[6][5]U

DANGER[3][2]D

FULL[10][5]L

MIXED[6][0]U

PLANT[1][4]R

POSSIBLE[3][7]D

RHYTHM[0][10]L

RULE[4][3]D

SAND[4][6]D

SNOW[2][1]D

THING[6][1]D

VALUE[2][2]R

WHIP[7][8]D