Software Workshop

Exercises 2

Marks available: 100

Date issued: 10/10/2017

Deadline: 1700, 18/10/2017

Feedback: 1700, 20/10/2017

Submission instructions

Submit your mark by the deadline above. The accepted submission formats are a zip archive or a jar archive. Your archive should contain only Java source files. Do not submit Java class files. Any questions for which you submit a class file will be given a mark of zero. Please do not submit Eclipse (or other IDE) projects.

Your work must follow all of the conventions you have been told about in the lectures relating to layout, naming, magic numbers, etc. Submissions that do not meet those standards will lose marks.

You must also consider validation. Any validation you are asked to do you must do, but you must also consider yourself any reasonable validation that you should do. Programs with poorly validated data will lose marks.

Consider any testing you could do to check that your programs work as expected. For this set of exercises, you could do this manually by writing a 'test plan'. This does not need to be extensive. There are lots of online resources about test plans.

You may use libraries from the standard Java distribution but not third-party libraries **unless otherwise stated**.

Unless otherwise stated, string case is not important.

Exercises with an 'x' appended to the name are extension activities. You may find these difficult at first. They have no marks attached to them but you should attempt them as soon as you feel able.

Exercise 1 [10 marks]

Write a program to read a series of strings from the keyboard, each string submitted by the user pressing the 'Enter' key. When the user types 'quit', the program should output the number of strings that were input and then output the actual strings in the order that they were entered.

Exercise 2 [10 marks]

Write a program to read a series of strings from the keyboard, each one submitted by the user pressing the 'Enter' key. When the user types 'quit', the program should output the strings in ascending length order and then in descending length order. The program should also state which the longest string is (not just the length of that string).

Exercise 3 [25 marks]

In a program, create the following two-dimensional array. This is an array of **single digit numbers**. Do not ask the user to input the array. The array is declared in the code.

0145
3797
1821

Print out the array as shown above, i.e. do not not put spaces or commas, etc. in the output.

Then prompt the user for a number written in the format "xy" with no space in between. This number represents an index in the array. The value of x is the row number and the value of y is the column number.

Output the array element given by the index that was entered. For example, if the use enters "23" it should output '1'. Now set the value of the element addressed by the index to zero and print out the array again. In this example, the array would now look like this:

0145
3797
1820

Continue to prompt the user for an index until they enter 'quit' to quit.

Exercise 4 [25 marks]

Write a program that prompts the user to enter three strings, each followed by the 'Enter' key. Each of the three strings represents a *row* on a tic-tac-toe board. A tic-tac-toe board is a 3 x 3 square. Each string will have three characters, not separated by spaces. Each of the three characters will either be an 'x' or an 'o' (lower case). If the input is not *well formed*, i.e. it is not in the format just described, output an error message to that effect.

Store the entered characters in a two-dimensional array that represents the tic-tac-toe board.

Print out the tic-tac-toe board in the following format (with an example configuration shown):

xox
xxo
xoo

Next print out which of the characters 'x' or 'o' has won, if either of them has. A character has won if any row or any column or any diagonal contains only that character. If it is a draw, state that.

Assumptions: you can assume that a full board will be entered and that only one winning character and one winning line is contained in the input.

Exercise 4x [0 marks]

Expand your solution to Exercise 4 to a full tic-tac-toe game. Create a two-player game where each player selects their character at the keyboard and then, for each turn, enters the coordinates in the board where they want to play. After each turn the program prints the latest state of the board, and whether a player has won yet. Then create an 'AI' character that plays against a single player. The 'AI' character can have a very simple strategy such as simply playing in the next available space.

Exercise 5 [30 marks]

You must implement the algorithm for this question yourself. Do not use sorting methods or other related utilities from the Java libraries or third-party libraries.

Write a command line program that accepts two lists of numbers. The two lists will be entered in the following format (this is an example):

"12 1 14 -3 7" "45 2 16 -6"

Each number in each list is separated by a space.

The program should merge the two lists into a single ordered list. The resulting single list is ordered in ascending order. The program should output the resulting ordered list.

Marking Scheme

The general marking scheme is set out below.

Fail
0 marks: No attempt or little attempt to answer the question.
1 marks: An attempt that relates to the question.

Pass
2 marks: A mostly working solution that shows understanding of the problem and solution.
3 marks: A fully working solution that shows understanding of the problem and solution.
4 marks: A good, clean, model solution.
5 marks: A high quality solution based upon solid design principles.