Software Workshop

Exercises 1 (Formative)

Marks available: 100*

Date issued: 26/09/2017

Deadline: 1700, 04/10/2017

Feedback: 1700, 06/10/2017

Submission instructions

Submit your mark by the deadline above. The accepted submission formats are a zip archive or a jar archive. Create a new project for each exercise and then submit a single archive containing all of your projects for this set of exercises. Please do not submit Eclipse (or other IDE) projects.

---

*This set of exercises constitutes a formative assignment. That means you must submit it but it will not contribute to your final module mark. You will get feedback on your work.

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 that prompts the user for a positive integer and then outputs whether that number was odd or even. If a number lower than 1 is entered the program should issue a suitable error message.

Exercise 2 [10 marks]

Write a program that prompts the user for their name. Names can be entered as either "first-name last-name" or "first-name middle-name last-name" (with spaces between each part of the name; users will not enter quote marks). Names will be entered on one line. The program outputs each part of the name on a separate line. The program must **not** prompt the user for which name format they will be using and should **not** print out a blank line in the case where a middle name is not entered.

Exercise 3 [10 marks]

Write a program that prompts the user for two strings and outputs if they are anagrams of each other. Use only the methods available in the `String` class. Do not use other Java library utilities.

Exercise 4 [20 marks]

Write a program that prompts the user for a mathematical expression in the form:

<operand> <operator> <operand>

For example, they might enter 2 + 3 or 4 * 5 or 10 % 2.

There will be a single space between each of the terms of the expression.

You can assume that all numbers entered are positive integers and that only the five operators '+', '-', '*', '/' and '%' will be entered and that all expressions are *well formed*.

The program should output the result of evaluating the expression. Division by zero should be disallowed.

Exercise 5 [20 marks]

Write a program that prompts the user for a sequence of positive integers. The user should press enter after each integer has been input. If the user inputs -1 that ends the sequence. Once the user has entered -1, the program should output:

- The lowest value entered.
- The largest value entered.
- The number of values entered.
- The sum of the values entered.
- The mean of the values entered (using floating point arithmetic).

Exercise 6 [30 marks]

Write a command line program that accepts a password as an argument and outputs whether the password is of the required 'strength'. A password of the required strength must:

- Be at least 8 characters in length.
- Contain at least one upper case letter.
- Contain at least one digit.
- Contain no spaces.

You will need to investigate how some of these checks can be done. For the last three requirements, you could use a loop and check each character to see if it meets the requirement. You could also use regular expressions if you are familiar with them, or wish to learn them.

The program should output if the password is strong enough or not. If it is not strong enough, the program should print generic information giving the reasons. For example, "Your password does not contain a digit." or "Your password is not long enough."

Exercise 6x [0 marks]

Expand the previous program so that the user can also provide their username and date of birth (in a format of your choosing) as command arguments. The password check should then disallow any passwords that contain the username (and, optionally, versions of it) and details from the date of birth. If the user attempts to use the program incorrectly (for example, if the date of birth appears to be in the wrong format), print usage information. If the user runs the program with only the word `help` as a parameter, the program should print usage information.

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.