Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code:	CM1210
Module Title:	Object Oriented Java Programming
Lecturer:	Neetesh Saxena and Surya Thottam Valappil
Assessment Title:	Data Structures and Algorithms in Java
Assessment Number:	2
Date Set:	25th March 2022
Submission Date and Time:	13th May 2022 at 9:30am
Return Date:	10th June 2022

This assignment is worth 50% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Your submission must include the official Coursework Submission Cover sheet, which can be found here: https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf

Submission Instructions

You must submit to Learning Central three files, the name of which must commence with your user name, as indicated in the table below:

Description		*Type*	*Name*
Cover sheet	**Compulsory**	One PDF (.pdf) file	[student number].pdf e.g., c1234567.pdf
ONE ZIP file (and no more than one) of all the source code written	**Compulsory**	One ZIP (.zip) archive	[student number]_CW2.zip e.g., c1234567_CW2.zip
ONE PDF file (and no more than one) which contains a written justification for your design of the program and screen shots showing an example of the output of each application.	**Compulsory**	One PDF (.pdf) file	[student number]_CW2.pdf e.g., c1234567_CW2.pdf

- Ensure that your student number is included as a comment at the top of each Java file that makes up your submission.
- Any code submitted will be run on a system equivalent to those available in the Windows laboratory and must be submitted as stipulated in the instructions above.
- Any deviation from the submission instructions above (including the number and types of files submitted) will result in a reduction in marks for that question part of 20%.
- Multiple attempts on the coursework submission is permitted via Learning Central but only the last attempt will be marked.
- You are reminded of the need to comply with Cardiff University's Student Guide to Academic Integrity. If you use external resources (even Oracle guidance), they need to be properly referenced, eg. as in-line comments in your code. If unfair practice is suspected, it will be necessary to invoke the University's Unfair Practice procedure.

Staff reserve the right to invite students to a meeting to discuss coursework submissions

Assignment

[**Total: 100 marks**]

All code must be written by you, although you can use the lecture notes (and lab exercises), textbooks, and the Oracle Java website for guidance.

(1) Stop words are high-frequency words that are removed from the representation of natural language data. Write a method deleteStopwords(input, stopwords) that deletes a list of words stopwords from some text input. It is up to you whether your method expects input to be a referring to the source of the text to be processed, or a String or ArrayList of words; similarly, for the stopwords parameter. However, you should try to optimise the data structure used. Your method should return an ArrayList containing the non-stop words identified. Your method should work successfully with the contents of the input file **Input.txt** and the stop words listed in the file **stopwords.txt** (both files are available on Learning Central).

[**15 marks**]

(Functionality: 8, Design: 4, Ease of use: 2, Presentation: 1)

(2) Implement the **insertion sort** algorithm to sort the words obtained from Question (1) in alphabetical order (the pseudocode for this method is available in the lecture notes). The Java method for insertion sort should be named insertionSort(listofWords).

[**15 marks**]

(Functionality: 7, Design: 4, Ease of use: 2, Presentation: 2)

(3) Implement the **merge sort** algorithm to sort the words obtained from Question (1) above in alphabetical order (the pseudocode for this method is available in the lecture notes). The Java method for merge sort should be named mergeSort(listofWords).

[**20 marks**]

(Functionality: 10, Design: 5, Ease of use: 3, Presentation: 2)

(4) Write a Java method to measure the performance of the insertion sort and merge sort Java methods from Questions (2) and (3), respectively, by:

- Measuring time that is needed to sort the first 100 of the words, first 200 of the words, and first 500 of the words by each of the two algorithms.
- Counting the moves and/or swaps that occur while sorting elements.

(Before attempting this exercise, you should work through the Algorithms lab exercises, available on Learning Central. The techniques used there will help you to work out how to approach this part of the coursework, in particular there are examples on how to time algorithms and count the moves and swaps.)

[**20 marks**]

(Functionality: 10, Design: 5, Ease of use: 3, Presentation: 2)

(5) You should create two methods for a data structure implementing a *Queue* as a circular array. Your data structure should have the class name MyArrayQueue. The two methods that should be implemented are:

a) Adding element to the queue:

public void enqueue(**Object** theElement) {…}

[**20 marks**]

(Functionality: 15, Design: 2, Ease of use: 2, Presentation: 1)

b) Deleting an element from the queue and return the deleted element:

public Object dequeue() {…}

[**10 marks**]

(Functionality: 6, Design: 2, Ease of use: 1, Presentation: 1)

In both methods errors should be handled properly. For example what happens when adding an element to a full queue?

There will be skeleton code for MyArrayQueue available on Learning central that contains the MyArrayQueue class with the following fully implemented methods: constructor, isEmpty(), getFrontElement(), getRearElement() methods. It also includes signatures of two methods that you should implement: enqueue(Object theElement) & dequeue().

You can reuse any implemented methods in the provided skeleton code. You ARE NOT allowed to change any parts of the implemented methods of the provided code.

You should also submit a pdf file with no more than 800 words that has written justification for your design of the program reflecting on the algorithm and efficiency of the all the codes in your assessment. It should also contain screenshots showing an example of the output of each question.

Learning Outcomes Assessed

- Implement basic data structures and algorithms
- Analyse and describe the performance of data-structures and algorithms

Code Reuse

Your solutions may make use of any classes in the Core Java API. You may also reproduce small pieces of code from:

- The CM1210 course handouts and solutions
- java.oracle.com
- any textbooks

provided:

- The section reproduced does not form the entire solution to a single question
- The source of the code is clearly referenced in your source code
- Your code is commented to demonstrate clearly that you understand how the reproduced code works (i.e., explain why particular types have been selected, why other language features have been used, etc.)

You may **NOT** reproduce code written by any other student or code downloaded from any other website.

If you are in any doubt about whether you may include a piece of code that you have not written yourself, ask the lecturer before submitting.

See “Referencing in code guidance” at Learning Central → COMSC-SCHOOL → Learning Materials → Referencing in code guidance.

Criteria for assessment

Credit will be awarded against the following criteria.

Marks	Functionality	Design	Ease of use	Documentation & Presentation
1st (70-100%)	Fully working code that demonstrates an excellent understanding of the assignment problem using a relevant java approach	Excellent design with proper use of appropriate types, program control structures and classes from the core API and carefully considers the use of most suitable data structures and aiming for optimised algorithms in terms of efficiency	An excellent use of input/output formatting that allows interaction with users and appropriate use of error handling for invalid input	Excellent use of in line comments in the code and proper justification given with clear and appropriate screenshots
2:1 (60-69%)	All required functionality is met, and the code is working probably with some minor errors	A good design with use of classes, types and control structure. The data structure is implemented appropriately.	A good use of input/output formatting with some minor issues in handling invalid input (no error message displayed)	Good use of in line comments in the code and screenshots submitted but justification for the design needs improvement
2:2 (50-59%)	Some of the functionality developed with incorrect output and minor errors	Some classes and types have been implemented with very little focus on efficiency of algorithm	Formatted input/output is implemented but minor issues in handling invalid input (accepts invalid input)	in line comments are not appropriate but screenshots submitted and justification for the design is just acceptable
3rd /Pass (40-49%)	Some of the functionality developed with incorrect output and major errors	Some classes and types have been implemented with no attention to efficiency in algorithm	Formatted input/output is implemented but code crashes for an invalid input	in line comments are not appropriate and no screenshots submitted and justification for the design is not acceptable
Fail (<40%)	Faulty function with wrong implementation and wrong output	Poor design with incomplete classes and data structures used	Poor formatting of input and output; no expected outcome	Poor/no comments, poor/no screenshot submitted, no justification of design

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on 10th June 2022 via Learning Central.