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:	20-March-2023
Submission Date and Time:	08-May-2023 at 9:30am
Feedback Return Date:	05-June-2023

Extenuating Circumstances submission deadline will be 1 weeks after the submission date above

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.

Extensions to the coursework submission date can ***only*** be requested using the Extenuating Circumstances procedure. Only students with approved extenuating circumstances may use the extenuating circumstances submission deadline. Any coursework submitted after the initial submission deadline without *approved* extenuating circumstances will be treated as late.

More information on the extenuating circumstances procedure can be found on the Intranet: https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/extenuating-circumstances

By submitting this assignment you are accepting the terms of the following declaration:

I hereby declare that my submission (or my contribution to it in the case of group submissions) is all my own work, that it has not previously been submitted for assessment and that I have not knowingly allowed it to be copied by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another writer, as one's own is plagiarism. I also understand that plagiarising another's work or knowingly allowing another student to plagiarise from my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings[1].

[1] https://intranet.cardiff.ac.uk/students/study/exams-and-assessment/academic-integrity/cheating-and-academic-misconduct

Assignment [Total:100 marks]

Section 1 Code: [**Total: 80 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. Full details of how and where you may reuse code is provided in the **Code Reuse** section below.

(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 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).

You should not remove punctuations, rather you will simply map all the words available in the file and match with Input.txt file and remove those words from the file. All remaining words, including duplicates, will be stored in the ArrayList. All words are case insensitive. Also, in the situation where “I” and “have” are in the stopwords.txt file, but “I've” is not in stopwords.txt, you will not have to remove “I've”.

[**15 marks**]
(Functionality: 10, Design: 3, Presentation: 2)

(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: 10, Design: 3, 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).

[**15 marks**]
(Functionality: 10, Design: 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.)

[**15 marks**]
(Functionality: 10, Design: 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) {…}

[**12 marks**]
(Functionality: 8, Design: 2, Presentation: 2)

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

public Object dequeue() {…}

[**8 marks**]
(Functionality: 5, Design: 2, Presentation: 1)

In both methods exceptions should be handled properly. For example what happens when adding an element to a full circular queue? This has been explained in the Lecture, please refer. There will be skeleton code for MyArrayQueue available on Learning central that contains the MyArrayQueue class with the following fully implemented methods: Constructor-MyArrayQueue & MyArrayQueue(int initialCapacity), 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. There will be penalty for making any changes. This class/skelton should only be used for attempting Q5.

Section2 Report [**Total: 20 marks**]

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

Learning Outcomes Assessed

- Implement basic data structures and algorithms
- Analyse and describe the performance of data-structures and algorithms
- Communicating technical details of object oriented software and the data structures and algorithms underpinning it

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 or java.oracle.com or 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

Criteria for assessment for your Code are as follows:

Marks	Functionality	Design	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 their use (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity).	Excellent use of in line comments in the code and excellent readability (means things like placement of curly braces, code indentation, wrapping of long lines, layout of parameter lists, etc.)
2:1 (60-69%)	All required functionality is met, and the code is generally working properly with some minor errors.	Good design with proper use of appropriate types, program control structures and classes from the core API and carefully considers their use (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity).	Good use of in line comments in the code and good code readability (means things like placement of curly braces, code indentation, wrapping of long lines, layout of parameter lists, etc.)
2:2 (50-59%)	Some of the functionality developed with incorrect output and minor errors.	Moderate design with the use of appropriate types, program control structures and classes from the core API and carefully considers their use (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity).	Some in line comments and moderate code readability (means things like placement of curly braces, code indentation, wrapping of long lines, layout of parameter lists, etc.)
3rd /Pass (40-49%)	Some of the functionality developed with incorrect output and minor errors.	Low design with incomplete classes and data structures and their use (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity).	in line comments are missing at requires places and low code readability (means things like placement of curly braces, code indentation, wrapping of long lines, layout of parameter lists, etc.)
Fail (<40%)	Faulty functions with wrong implementation and incorrect output.	Poor design with incomplete and/or incorrect classes and data structures and their use (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity).	Poor/no comments and poor code readability (means things like placement of curly braces, code indentation, wrapping of long lines, layout of parameter lists, etc.)

Criteria for assessment for your writing are as follows:

Mark	assessment Criteria
1st (70-100%)	*An excellent and original discussion is provided. Your points are justified with excellent quality writing and screenshots*
2:1 (60-69%)	*A very good and original discussion is provided. Your points are justified with very goof quality writing and screenshots*
2:2 (50-59%)	*A good discussion is provided, and points are justified with screenshots.*
3rd /Pass (40-49%)	*Some discussion is provided and points are not justified with screenshots or inconsistent screenshots submitted*
Fail (<40%)	*Limited or no discussion and no screenshot evidence submitted*

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on 05-June-2023 via Learning Central.

Submission Instructions

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

Description		*Type*	*Name*
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

Any code submitted will be run on a system equivalent to those in the Windows laboratory OR University provided laptop 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%.

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

Support for assessment

Questions about the assessment can be asked on https://stackoverflow.com/c/comsc/ and tagged with 'CM1210' along with lecturer's name: Neetesh Saxena or Surya Thottam Valappil

Support for the programming elements of the assessment will be available in the lab classes in *[Weeks 8,9, 10 and 11] or in the daily drop-in lab sessions*