Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code: CMT219
Module Title: Algorithms, Data Structures and Programming
Lecturer: Dr Crispin Cooper, Neetesh Saxena
Assessment Title: Algorithms Data Structures and Design Patterns
Assessment Number: 2
Date Set: 23rd March 2023
Submission Date & Time: 11th May 2023 at 9:30am British Summer Time
Return Date: 8th June 2023

Extenuating Circumstances submission deadline will be 1 weeks after the submission date above
Extenuating Circumstances marks and feedback return will be 1 weeks after the feedback return 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

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.

Question 1

Extracting words from a text document is the first step for many text-based applications of artificial intelligence, e.g., detecting abusive tweets on Twitter. This task asks you to write a Java program to extract all valid words from the document **“Input219.txt”** based on a given vocabulary **“google-10000-english-no-swears.txt1”** (you can download both files from the Learning Central). Specifically, if a word token from **“Input219.txt”** matches a word in **“google-10000-english-no-swears.txt”** (case-insensitively), you keep that word, otherwise, you discard it. Your program should create an *ArrayList* containing all valid words from **“Input219.txt”** and print the list to the console.

You are not supposed to remove punctuation, rather you will simply read all the words available in Input219.txt, match with words in **google-10000-english-no-swears.txt1** and output the matched words in order of appearance in the input, including duplicates and attached punctuation. For example, if “for,” appears in Input219.txt and “for” appears in the vocabulary, then the result should include “for,” as an item in the ArrayList. Punctuation that is not adjacent to a space should be treated as part of the word, e.g. “I’ve” should be considered a single word and should only appear if “I’ve” appears in the vocabulary.

All words **must** be stored in ArrayLists.

Write a short report answering the following questions:

1. How does your program work? (max 200 words)
2. How does the use of ArrayLists affect the time complexity of your program?
3. Could the efficiency be improved in any way?

Include screenshots showing an example of the output. Follow the submission instructions.

[30 marks]
(Functionality: 15, Style: 12, Documentation and Presentation: 3)

Question 2

Implement the merge sort (the pseudocode for merge sort is available in the lecture slides) in order to sort the words obtained in question (A) into alphabetical order, i.e., the output of your program will be the sorted words in alphabetical order. For the merge sort algorithm write a method e.g. *mergeSort(...)*, then measure

- the time that is needed to sort the first 100 words, first 200 words, first 300 words, etc. up to all words output by your program in part A. i.e. time separate calls to mergeSort() for 100 words, 200 words, 300 words, etc.

- count the moves/comparisons that occur while sorting the first 100 of the words, first 200 of the words, first 300 of the words, etc. up to all words by the algorithm

(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 the coursework, in particular, there are examples of how to time algorithms and count the moves and swaps).

Write a short report with

1. three graphs, to show how time needed, count of moves, and count of comparisons vary with the number of words to be sorted
2. a short written explanation of how your program works,
3. screenshots showing an example of the output. Follow the submission instructions.

[20 marks]
(Functionality: 12, Style: 6, Documentation and Presentation: 2)

Question 3

Download the ZIP file from Learning Central that contains the source code files of this task: **Product.java, ProductRecommender.java, ChoiceStrategy.java, LeastExpensiveStrategy.java, MostPracticalStrategy.java.**

This Java program creates three new products, each with a cost (measured in £) and practicality (measured on a scale from 1-5). The program then chooses between the products, first by applying the **LeastExpensiveStrategy** and then the **MostPracticalStrategy**.

A) State the purpose of the *Strategy* design pattern. **[5 marks for write-up]**

B) Complete the implementation of the **LeastExpensiveStrategy** class with a *getScore(Product a)* method that returns a suitable score for any product, such that the product with the highest score is the least expensive one. Paste your completed code for this class into your writeup. **[5 marks for code]**

C) Complete the implementation of **MostPracticalStrategy** with a *getScore(Product a)* method that returns a suitable score for any product, such that the

product with the highest score is the most practical one. Paste your completed code for this class into your writeup. **[5 marks for code]**

D) Modify the **getBestProduct ()** method in the **ProductRecommender** class, so that it correctly applies the ScoringStrategy passed to it, in order to generate the following output:

```
Best product according to LeastExpensiveStrategy:
Vauxhall Nova
Best product according to MostPracticalStrategy:
Skoda Octavia
```

Paste your completed code for this method into your writeup. Follow the submission instructions. **[5 marks for code]**

[Total for Question 3: 20 marks]

Learning Outcomes Assessed

This assignment particularly addresses the following module learning outcomes:

- Write code in a given programming paradigm (for example Object-Oriented) using a high-level programming language
- Select and use appropriate algorithms and data structures to provide the best performance in a given scenario
- Critically evaluate design patterns and their applicability to different scenarios
- Select and use appropriate algorithms and data structures to provide the best performance in a given scenario

Code Reuse

Your solutions may make use of any classes in the Core Java API (except where the coursework questions ask for a specific class) You may also reproduce small pieces of code from:

- The CMT219 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 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 it.

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.

Functionality

- To what extent does the program perform the task(s) required by the question
- Formatting of input/output.
- Correct exception handling (your program should handle exceptions properly.)

Design and Code Style

- Use of appropriate types, program control structures and classes from the core API.
- Structure of appropriate classes, functions and design patterns.
- Effective usage of class, method, and variable names.
- The code layout is readable and consistent

Documentation and presentation

- Appropriate use of comments.
- Clear and appropriate screenshots.

	Criteria	**Distinction (70-100%)**	**Merit (60-69%)**	**Pass (50-59%)**	**Fail (0-49%)**
Questions 1+2	Functionality	Fully working application that demonstrates an excellent understanding of the assignment problem using a relevant	All required functionality is met, and the application is working properly with no errors. Good formatting of	Program works correctly or with only minor errors. Suitable formatting of input/output, catch exception	Faulty application with wrong implementation and wrong output. Bad formatting of input/output, no exception

		Java approach. Excellent formatting of input/output, catch exception handling, the application deals well with invalid user input and doesn't crash for any data.	input/output, catch exception handling, but the application may have errors given invalid input data.	handling, but the application doesn't deal with invalid input data and could crash for some data.	handling, the application doesn't deal with invalid user input and crashes for most data.
	Code Style	Most effective class, method, and variable names (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity.) Excellent attention to code layout. The layout of your code should be readable and consistent. This means things like the placement	Pays good attention to class, method, and variable names (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity.) Good attention to code layout. The layout of your code should be readable and consistent.	Pays some attention to the most effective class, method, and variable names (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity.) Some attention to code layout. The layout of your code should be readable and consistent. This means things like	Pays little/no attention to the effective class, method, and variable names (names chosen for classes, methods, and variables should convey the purpose and meaning of the named entity.) Little or no attention to code layout. The layout of your code should be readable and consistent. This means things like

		of curly braces, code indentation, wrapping of long lines, the layout of parameter lists, etc.	This means things like the placement of curly braces, code indentatio n, wrapping of long lines, the layout of parameter lists, etc.	the placement of curly braces, code indentation, wrapping of long lines, the layout of parameter lists, etc.	the placement of curly braces, code indentation, wrapping of long lines, the layout of parameter lists, etc.
	Documentati on and Presentation	Appropriate use of comments. Clear and appropriate write-up.	Some use of comments. Reasonabl e write-up.	No comments. Unclear write-up.	No comments. No write-up.
Question 3	Code	Code is fully correct and written in good style	Code is fully correct	Code has minor errors	Code has major errors
	Write-up	Is correct, clearly written and shows understandi ng	n/a	Is correct	Is incorrect

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on the 10th of June 2023 via Learning Central.

Feedback from this assignment will be useful for other programming and data structure and algorithms related modules.

Submission Instructions

Your coursework should be submitted through Learning Central.
Your submission should include:

	Description		Type	Name
	ONE ZIP archive file (and no more than one) containing all the source code files for your answer to questions	**Compulsory**	One ZIP (.zip) archive	[student number] _ Code.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].pdf

Any code submitted will be run on a university machine 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 of 20% of the attained mark.

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

Support for assessment

Questions about the assessment can be asked during lectures and labs, and on the departmental StackOverflow (StackOverflow questions must be tagged cmt-219). A specific session will be scheduled for coursework help; check the module map/timetable for details. Support for the programming elements of the assessment will be available in the lab classes.