CS5003 –Data Structures and Specialist Programming

Individual Coursework 1

2022-23

This individual coursework requires developing and documenting Java application(s) using an object-oriented approach.

The coursework carries 30% of the module mark.

Submission Deadlines: 3pm, Tuesday 6 December 2022 via WebLearn

Mandatory Software Demo Recording Deadline: 3pm, Tuesday 6 December 2022 via WebLearn

NB– Anyone not meeting the deadline must submit their work with a completed *mitigating circumstances form*. It will only be marked if the mitigating circumstances are accepted.

Please note the rules on plagiarism

The application should be implemented individually. This is not a group/team effort. Any material which is a direct copy from someone else (student or other source) or a close paraphrase/code must be indicated where it is quoted i.e. it must be made clear what material is a quotation or close paraphrase e.g. by showing the text in italics or in quotation marks. It is not sufficient to show the source in a list of references or bibliography. If you are unclear, please discuss your examples with your seminar tutor or the module leader. Plagiarism is a serious offence and conviction for plagiarism may lead to suspension from the University, even for a first offence. Please see the section on Academic Misconduct in the Student Handbook.

Part 1: Use of Data Structures & Java Collections Framework

You are required to develop a software system in Java using relevant classes from the Java Collections framework. The system is to manage bank accounts. It should be able to manage account general details for an *unlimited* number of accounts, and details of the *last six* transactions for each account as specified below. To simplify the matter the system does not require saving data on disk, however if you wish you can implement this functionality into your system.

The data of a bank account must include the following details:

Account general details:

1. Account number
2. Account holder name
3. Account holder address
4. Opening Date
5. Current balance.

Transaction details:

1. Transaction type (i.e. deposit, withdrawal)
2. Transaction amount
3. Transaction date.

Your system is required to provide the following functionality.

1. Create a new account and add it to the system.
2. Display on the computer screen a list of the existing accounts with the account general details.
3. Delete a closed account from the system, given the account number.
4. Update the system with details of any new transaction of existing accounts. Note that for each account only the information of the last six transactions is maintained by the system.
5. Given an account number, display on the computer screen details of the account's last six transactions being *sorted* by transaction amounts. Note that all transaction amounts are positive numbers regardless whether a transaction is a deposit or withdrawal.
6. Provide an appropriate system user interface that allows testing of the above methods.

Part 2: Implementation of a Data Structure and an Algorithm

You are to provide your own implementation of a standard data structure and a standard searching/sorting algorithm to satisfy the requirement of this part of the coursework.

You should have used some classes and its methods from the Java Collections framework in your implementation of the system as requited by Part 1 of this assignment. In this part, you should do the following: -

(a) To construct your own implementation of the same data structure of *at least one* of the Java Collections classes which you used in your solution from Part 1. For example, if you used the LinkedList class you should provide your own implementation of a linked list class from scratch in order to provide the same functions of your application.

(b) To write your own Java code from scratch to implement a sorting or searching algorithm. For example, you could write a Java method of your own to perform the *insertion sort* algorithm, which can be used in the implementation of Function no.5 of the application from Part 1.

(c) To provide another version of your application which provides the same functions as specified in Part 1, but makes use of the class(s) and method(s) from your implementation of the chosen data structure and algorithm from (a) and (b) above.

Deliverables

This coursework requires three deliverables to be submitted via WebLearn before the submission deadline above.

(1) The software artefact with a complete set of Java classes source code (i.e. *.java files, or complete NetBeans Java application projects) to meet the requirement of Part 1 and Part 2 of the coursework.**

(2) The report in MS Word compatible or PDF format.

(3) The recorded software demo in mp3 or mp4 format.

Your software implementation should demonstrate/provide the following features

1. Use of appropriate data types (built-in and programmer-defined) to handle the application data.
2. Define and use your own class or classes.
3. Use of appropriate data structures for the required programming scenario.
4. Use suitable algorithms e.g. sorting and searching.
5. Provide either console-based or GUI-based user interface for your application.

A reflective report (1000 words), which concisely documents:

a. Detailed instructions to run the program. Note that these instructions are not a User Manual of your application.

b. The architecture of your software in terms of software classes using a UML class diagram. You are required to clearly indicate which classes are of your own work and which classes are from other sources (e.g. from textbooks, online sources etc.).

c. Detailed description of the classes' purpose, properties and methods.

d. Which data structures and which algorithms you have used, in which part of your programs, and why.

e. Screen dumps (live, 2 per page) including test plan, test data and test results

f. A reflection of your experience of the development task, what issues you experienced, your solution to overcome it and any lessons learned.

A recorded software demo (10 minutes), which concisely demo any implemented functionalities(1-6, a-c)) and features (1-5) of your software with your voice over

Marking Scheme for CS5003 Individual Coursework 1

This coursework counts for 30% of the module mark. Please see the table below for the marking criteria and its weighting.

	Item	**Weighting %**
SOFTWARE DEMO (Design and Implementation)		**70%**
	Part 1	
1	<u>*Design quality*</u> of your Java class/es to hold details of bank accounts and transactions, i.e. the class *public properties and methods*	5
2	Use of suitable data structures in your Java classes.	7.5
3	Create a new account and add it to the system.	5
4	Display on the computer screen a list of the existing accounts with the account general details.	5
5	Delete a closed account from the system, given the account number.	5
6	Update the system with details of any new transaction of existing accounts. Note that for each account only the information of the last six transactions is maintained by the system.	7.5
7	Given an account number, display on the computer screen details of the account's last 6 transactions being <u>*sorted*</u> by transaction amounts. Note that all transaction amounts are positive numbers regardless whether a transaction is a deposit or withdrawal.	7.5
8	Provide an appropriate system user interface that allows testing of the above functions.	7.5
	Part 2	
9	Implementation of a standard data structure	5
10	The code to test the implemented data structure	5
11	Implementation of a standard algorithm	5
12	The code to test the implemented algorithm	5
REPORT		**17.5%**
a	Detailed instructions to run the program	2.5
b	The architecture of your software in terms of software classes, clearly indicating which classes are of your own work and which classes are from other sources (e.g. from textbooks, online sources).	2.5
c	Concise description of each of the classes' purpose, properties and methods.	2.5

d	Which data structures and which algorithms you have used, in which part of your program, and *why*.	5
e	Screen dumps (live, 2 per page) including test data/plan and test results	2.5
f	A reflection of your experience.	2.5
PROGRAMMING QUALITY AND STYLE		**12.5%**
1	Clarity of code which shows the underlying algorithm	2.5
2	Sensible naming of programmer-defined variables, classes, properties and methods	2.5
3	Useful comments in code	2.5
4	Data validation and exception handling	2.5
5	User interface design and usability of the system	2.5
	Total Mark	**100%**