**CS5003 –Data Structures and Specialist Programming Individual Coursework 1**

**2024-25**

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, Friday 13th of December 2024, 3pm via WebLearn**

**Mandatory Software Demo (Panopto Recording) Deadline: 3pm, 13th of December 2024 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 workshop 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.**

# Software Requirements and Functionalities

You are required to develop a software in Java using relevant and data structures you have learned during the module. The software 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 four transactions** for each account as specified below. To simplify the matter the software does not require saving data on disk, however if you wish you can implement this as an additional feature (see the marking scheme table criteria no.12).

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:
   * Current balance will be also updated in response to applied transactions, and the balance shouldn’t go below zero.
   * For each account only the information of the last four transactions is maintained by the system. **You should test this by feeding your system with input of more than four transactions and confirm that only last four transactions can be stored and maintained (show this test in your demo and report).**
5. Given an account number, display on the computer screen details of the account's last four 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.

## Implementation Requirements

1. Choose the most appropriate data structures and algorithms to implement this software.

1. One of your data structure should be used from the Java Collections Framework.

1. Another one data structure (other than the one you choose in (b)) should be your own implementation (i.e. the data structure and its operations/functions should be implemented from scratch by you). For example, if you use LinkedList as a data structure, you should provide your own implementation of a linked list class from scratch in order to provide the required functions of your application.

1. You should also provide your own implementation of **one** standard searching/sorting algorithm (i.e. linear search doesn’t fulfil this requirement). For example, you could write a Java method of your own to perform any of the insertion sort algorithm or binary search algorithm, each can be used in the implementation of Function no.5 of the application.

# Deliverables

This coursework requires three deliverables to be submitted via WebLearn before the submission deadline above. **Please note you will automatically fail the coursework if any of (1 or 3) has not been submitted.**

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 requirements of the coursework. The artefact should be in **zipped** folder with the following file naming format:

**'YourStudentID\_StudentLastName\_PartNumber.zip'.**

1. The report in MS Word compatible or PDF format. Follow the naming format:

**'YourStudentID\_StudentLastName\_Report.pdf'**

1. The recorded software demo in **mp4 format, using Panopto application**.

## (1) 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.

## (2) A reflective report (1000 words), Please be concise (clear statements in less number of words), use bullets and numbering for text descriptions not paragraphs

1. Detailed instructions to run the program: concise steps in bullet or flowchart (not in paragraph). Example: download the zip file, open the project in Netbeans, run the following java file, ….etc**. Note that these instructions are not a User Manual of your application.**
2. 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.).**
3. Detailed description of the classes’ purpose, properties and methods. **Which data structures**

**and which algorithms you have used, in which part of your programs, and why.**

1. Screen dumps (live, 2 per page) including test plan, test data and test results
2. A reflection of your experience of the development task, what issues you experienced, your solution to overcome it and any lessons learned.

**Make sure all text and diagrams are visible within the pages margines before you submit your report file.**

**(3) A recorded software demo (10 minutes) recorded using Panopto application in mp4 format, which concisely demo any implemented functionalities (1-6) , implementation requirements (a-d and features (1-5) of your software with your voice over and your face included for authentication.** You should clarify in the demo:

* How you designed Java classes to hold the data of a bank account details (i.e. properties and functionalities). Show your classes implementation and **demonstrate your understanding to the implemented code.**
* Which classes, functions and data structures you have implemented from scratch and which you have used from the java collections or any other sources (i.e. from textbooks, online sources, etc.);
* Which data structure(s); searching/sorting algorithms you chose and **Why?**
* Live test the implemented functionalities as they appear in the marking scheme. **Demonstrate your understanding on how you implemented them.**
* Demonstrate any additional features you have added to your software.

.

# 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. **Note that it is crucial to demonstrate your understanding of how you did/implemented each criteria in the marking scheme in your coursework. Otherwise, you will lose marks.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Item | Weighting(%) | |
| SOFTWARE DEMO (Design and Implementation) | |  | 70% |
| 1 | *Design quality* of your Java classes 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 |
| 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, and update the relative account balance accordingly (6.1). Note that for each account, only the last four transactions are maintained by the system (6.2), and neither an account balance nor a transaction amount can be negative numbers, regardless whether a transaction is a deposit or withdrawal (6.3). | (6.1)  (6.2)  (6.3) | 4  4  2 |
| 7 | Given an account number, display on the computer screen details of the account's last 4 transactions (7.1). These transactions should be ***sorted*** by transaction amounts (7.2). | (7.1) (7.2) | 3  4 |
| 8 | Provide an appropriate system user interface that allows testing of the above functions. |  | 5 |
| 9 | One of the data structures is implemented from scratch (9.1)  The code to test the implemented data structure (9.2) | (9.1)  (9.2) | 5  5 |
| 10 | Implementation of a standard searching/sorting algorithm, and  testing it |  | 6 |
| 11 | Additional marks will be awarded for new features |  | 5 |
| REPORT | |  | 18% |
| a | Detailed instructions to run the program in bullets or flowchart (not in paragraph) |  | 2 |
| b | The architecture of your software in terms of software classes in UML class design, clearly indicating which classes are of your own work and which classes are from other sources (e.g. from textbooks, online sources). |  | 2 |
| c | Detailed description of each of the classes’ purpose, properties and methods. |  | 2 |
| d | A reflection of your experience. |  | 2.5 |
| e | Which data structures and which algorithms you have used, in which part of your program, and why. |  | 5 |
| f | Screen dumps (live, 2 per page) including test data/plan and test results |  | 2.5 |
| g | Report clarity and format: use concise statements, replace text with diagrams when you can, bullets and numbering for text descriptions not paragraphs. | 2 | |
| PROGRAMMING QUALITY AND STYLE | | 12% | |
| 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 | |
|  | Total Mark | 100% | |

Get Started.. Don't leave it to the last minute. I am sure you can do a great job. "A little progress every day can adds up to big results" With my best wishes in your project!

Dr. Mona Abdelgayed