CST2550 Coursework 1

Ahmed Eissa, Adam Philpot, Olugbenga Oluwagbemi

November 24, 2023

1. Brief Task Description

A small library needs a system to track the details of their available books and members. They provided you with the data source CSV file. The file contains details of all books that are available for members to borrow and return within three days. It is made available to you for implementation and testing, but the system should allow any file to be used - a different file will be used for your CW marking (do NOT hardcode the filename). You have been provided with the UML diagram in Figure 1 to design the software and must implement the classes as shown – any deviation will be marked down. You must use the Git version control system with regular commits pushed to a repository on either Bitbucket or GitHub.

You should NOT use any third-party libraries or code as a part of your solution and all code must be written by you, e.g., not automatically generated by an A.I. or IDE. Do not use any non-standard code or libraries or system calls in your program.

2. Submission

The submission will be done via submitting a link to your Bitbucket or GitHub repository. The online repository MUST consist of all-required source code; a single MP4 video file of a screencast presentation and a single PDF file of the presentation slides. The deadline is 5pm on Friday, 5th January 2024.

The repository should include:

• C++ source code files of your program.

• C++ source code of "Catch2" tests which test your code.

• a makefile to compile your program.

In the presentation you should show your slides while you describe each section. Full details of what should be included in the presentation are given below. The video should be a maximum of **6 minutes** long. If the video is longer, only the first 6 minutes will be marked. The video must also include a short demonstration of your software where you demonstrate understanding of the implementation (note: just running the program will not demonstrate any knowledge of the implementation).

Notes:

- Only the required files MUST be included in your repository, no other files (any other files will not be used when marking your work).
- If code does not compile and run, it will severely limit your marks for the code.
- Do not use any non-standard or OS specific libraries or system calls, which will prevent your program from working correctly when being marked.
- Do not make any further commits to the repository once you have made the submission – the date of commits will be monitored and anything committed after the deadline will be ignored.
- Requests to 'pre-assess' summative assessment will be (politely) refused.

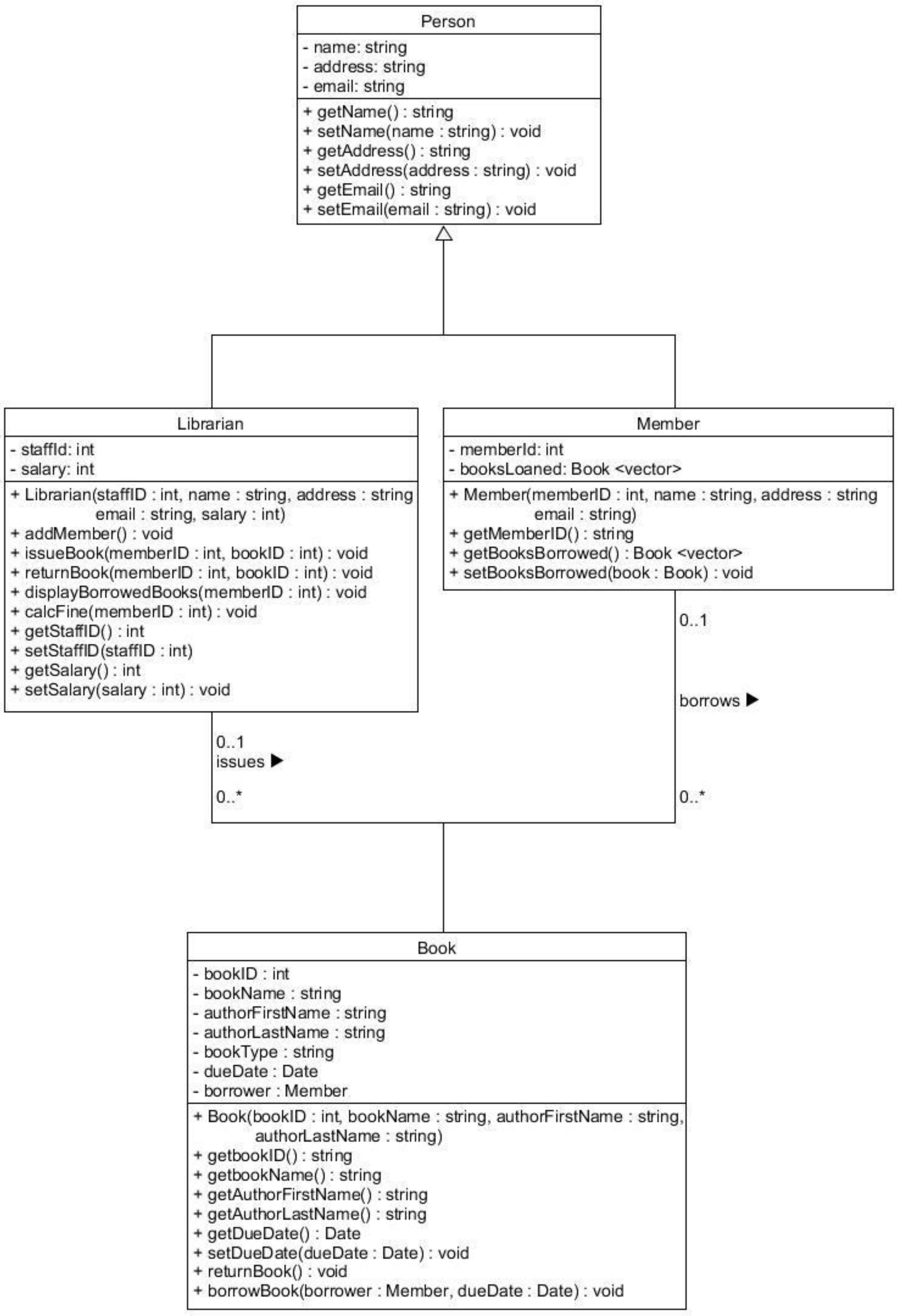
Person
- name: string
- address: string
- email: string
+ getName() : string
+ setName(name : string) : void
+ getAddress() : string
+ setAddress(address : string) : void
+ getEmail() : string
+ setEmail(email : string) : void
Librarian
- staffId: int
- salary: int
+ Librarian(staffID : int, name : string, address : string email : string, salary : int)
+ addMember() : void
+ issueBook(memberID : int, bookID : int) : void
+ returnBook(memberID : int, bookID : int) : void
+ displayBorrowedBooks(memberID : int) : void
+ calcFine(memberID : int) : void
+ getStaffID() : int
+ setStaffID(staffID : int)
+ getSalary() : int
+ setSalary(salary : int) : void
Member
- memberId: int
- booksLoaned: Book <vector>
+ Member(memberID : int, name : string, address : string email : string)
+ getMemberID() : string
+ getBooksBorrowed() : Book <vector>
+ setBooksBorrowed(book : Book) : void
0..1
borrows ▶
0..*
0..1
issues ▶
0..*
Book
- bookID : int
- bookName : string
- authorFirstName : string
- authorLastName : string
- bookType : string
- dueDate : Date
- borrower : Member
+ Book(bookID : int, bookName : string, authorFirstName : string, authorLastName : string)
+ getbookID() : string
+ getbookName() : string
+ getAuthorFirstName() : string
+ getAuthorLastName() : string
+ getDueDate() : Date
+ setDueDate(dueDate : Date) : void
+ returnBook() : void
+ borrowBook(borrower : Member, dueDate : Date) : void

Figure 1: The UML class diagram for the library system.

3. Scenario

A small library has these types of books that are available to borrow for members only:

1. Science fiction
2. Satire
3. Drama
4. Action and Adventure
5. Romance
6. Mystery
7. Horror
8. Health
9. Guide
10. Diaries
11. Comics
12. Journals
13. Biographies
14. Fantasy
15. History
16. Science
17. Art

All books have an id, name, author, type, and page count.

The program will only be used by the librarian at the library (not by customers/members) and will not take payment details.

The system should include the following functionality:

- Add a member – the librarian should be able to create a new member and display the new member's details directly following the creation of the member.
- Issue a book to a member – the librarian should be able to issue a book to an individual member with a valid due date from the date of issue (3 days).
- Return a book – the librarian should be able to return a book from an individual member.
- Display all books borrowed by any individual member – the librarian should be able to display all books borrowed by an individual member.
- Calculate a fine for any individual member for overdue book(s) – upon the return of a book, if the book's due date has expired, a fine should be calculated based on a rate of £1 per day overdue.

Your program must have input validation to ensure it performs correctly and helpful error messages informing the user of any incorrect input. It should be intuitive, i.e., the user shouldn't need to read the source code or attend training to know how to use the system.

4. Detailed Description

It is recommended that you complete the tasks in the following order as the later sub-tasks will require the earlier ones.

4.1. Plan Software

Begin by planning which classes you need and how the software will work using the provided UML diagram.

You should include:

- use case diagram(s)
- we provided you with the class diagram(s)
- activity diagram(s)

4.2. Create a Git Repository

Create a Git repository for your coursework on Bitbucket or GitHub. As you create new files, you should add them to the Git repository. Commit regularly with clear, useful commit messages. You will need to submit the URL for your Git repository, as file submissions are NOT accepted.

4.3. Implement Classes

Once you are happy with your design, you should implement the classes. You should have a base class with any shared member variables and member functions, and derived classes for each specific type of object with specific member variables and functions.

You should write test cases and Catch2 test code to ensure these classes are working correctly.

4.4. Write a Makefile

Create a Makefile to compile the classes. As you implement other source code files add them to the Makefile. Your Makefile should be able to compile you project using the command 'make' and should be able to clean your project (remove any compiled files) using make clean.

4.5. Implement the Program

Implement the system using objects of the various classes mentioned above. Write additional tests for any functions to ensure they work correctly and implement a menu for the user to interact with the program. The user input should be validated.

4.6. Plan Presentation

Plan a presentation and create presentation slides detailing the design, implementation, testing and evaluation of your work. This should be about your work, not generic comments about programming in general.

The presentation should include:

- Introduction:
  - Your student number.
  - A brief description of your project (do not read the coursework task).
  - A brief overview of the presentation.
- Design:
  - Clearly show each of your UML diagrams.
  - You should briefly describe what the diagrams are showing (do not just read the diagrams).
- Implementation, including:
  - Your approach, i.e., how you translated the design into working software.
  - Explain what the Makefile was used for.
  - Explain how and why version control was used.
  - Include a screenshot of the Bitbucket/GitHub repository which clearly shows all commits and commit messages.
- Testing approach:
  - A statement of the approach used.
  - How you applied this approach.
  - Details of test cases, i.e., what was being tested (don't show the code).
- Software demonstration:
  - Show your understanding of the implementation, not just that you can run the program.
- Conclusion:
  - A brief summary of the work.
  - Limitations of your work (don't list features which aren't required).
  - How you would approach a similar project in the future to avoid the limitations (don't list features which aren't required by the specification).

5. Academic Misconduct

This is individual work, and you should complete it yourself. You should not work as a group, and each submit the same work (even with minor changes) as your own. Any material or ideas found online, in textbooks etc. should be properly referenced.

You should familiarise yourself with the university's academic integrity and misconduct policy:

https://www.mdx.ac.uk/about-us/policies/university-regulations

6. Extenuating Circumstances

There may be difficult circumstances in your life that affect your ability to meet an assessment deadline or affect your performance in an assessment. These are known as extenuating circumstances or 'ECs'. Extenuating circumstances are exceptional, seriously adverse, and outside of your control.

Please see link for further information and guidelines:

https://unihub.mdx.ac.uk/your-study/assessment-and-regulations/extenuating-circumstances

7. Marking

The presentation video and code will be marked according to the attached marking scheme. Marking will be from the online repository for your work.

As the presentation also includes your demonstrated understanding of your program, if there is no presentation video you will not be awarded any marks for this coursework task.

8. Feedback

Provisional marks and written feedback will be available on Moodle within 21 working days of your submission. If you would like clarification or more detailed feedback on your coursework, contact your module tutor.

9. Marking Scheme

9.1. Presentation

Item	Marks
Correctly submitted files (correct naming, format, contents, etc.) and repository	5
Introduction (description of the project, not the coursework task)	6
Software design (description and UML diagrams)	9
Software testing (description of testing approach used and evidence of testing)	6
Implementation (description of approach and how the makefile and version control were used).	9
Conclusion, including: - summary of work done - limitations and critical reflection - how would change approach on similar task in future	5
Layout and clarity of presentation and slides	5
Video timing (6 minutes long)	5

9.2. Code

Item	Marks
Code quality (follows code guidelines and user input validation)	10
Use of Git (regular commits, clear commit messages)	5
Makefile	5
Implementation matches design	10
Implementation meets requirements	10
Quality of test code (and matches testing approach described in presentation)	10