Library Program Documentation

Introduction:

Student Number: M00865822

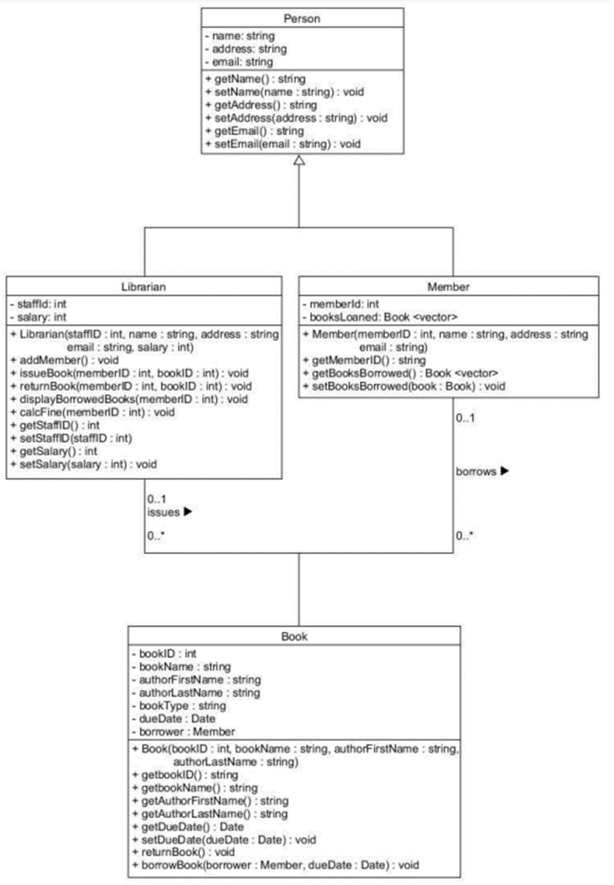
Brief description:
My project is a Library system designed for a Librarian. The librarian can add members to the library and issue books based on the available books in the CSV file. The Librarian can return, display books borrowed by a member and calculate a fine for said books. I will review the Design, Implementation, Testing, and Conclusion.

Design:

UML Diagrams: The UML diagram shows the structure of the library system. It has four classes: Person, Librarian, Member, and Book.

The Person class is the base class with private attributes 'name', 'address', and 'email'. It also has public setter and getter methods for each attribute. These methods have access and can modify private values.

The Librarian class has private attributes 'staffId' and 'salary' and it also inherits from the 'Person' class.



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
issues ▶
0..*
0..1
borrows ▶
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

The class has public methods that allow the librarian to add a member, issue a book, return a book, display borrowed books, and calculate fines. It also has getter and setter methods for its attributes. The “0..1 issues 0..” shows that there is a relationship between ‘Librarian’ and ‘Book’ so that the librarian can issue zero or more books.

The Member class also inherits from the ‘Person’ class and has its private attribute ‘memberId’. It also has a private association with the ‘Book’ class. It is represented by the vector of ‘Book’ objects, named ‘booksLoaned’, which indicates that a member can have zero or more books loaned to them. The public methods include getter and setter methods for ‘memberId’ and borrowed books. And a method to add a borrowed book to the member’s vector of books.

The Book class is independent and has private attributes for the book details and a pointer to ‘borrower’, which is of the type ‘Member’. Public methods include a constructor, getters, setters, and methods to manipulate the due date and handle borrowing.

Overall, the UML diagram shows how the classes are related to each other and their methods.

Implementation:

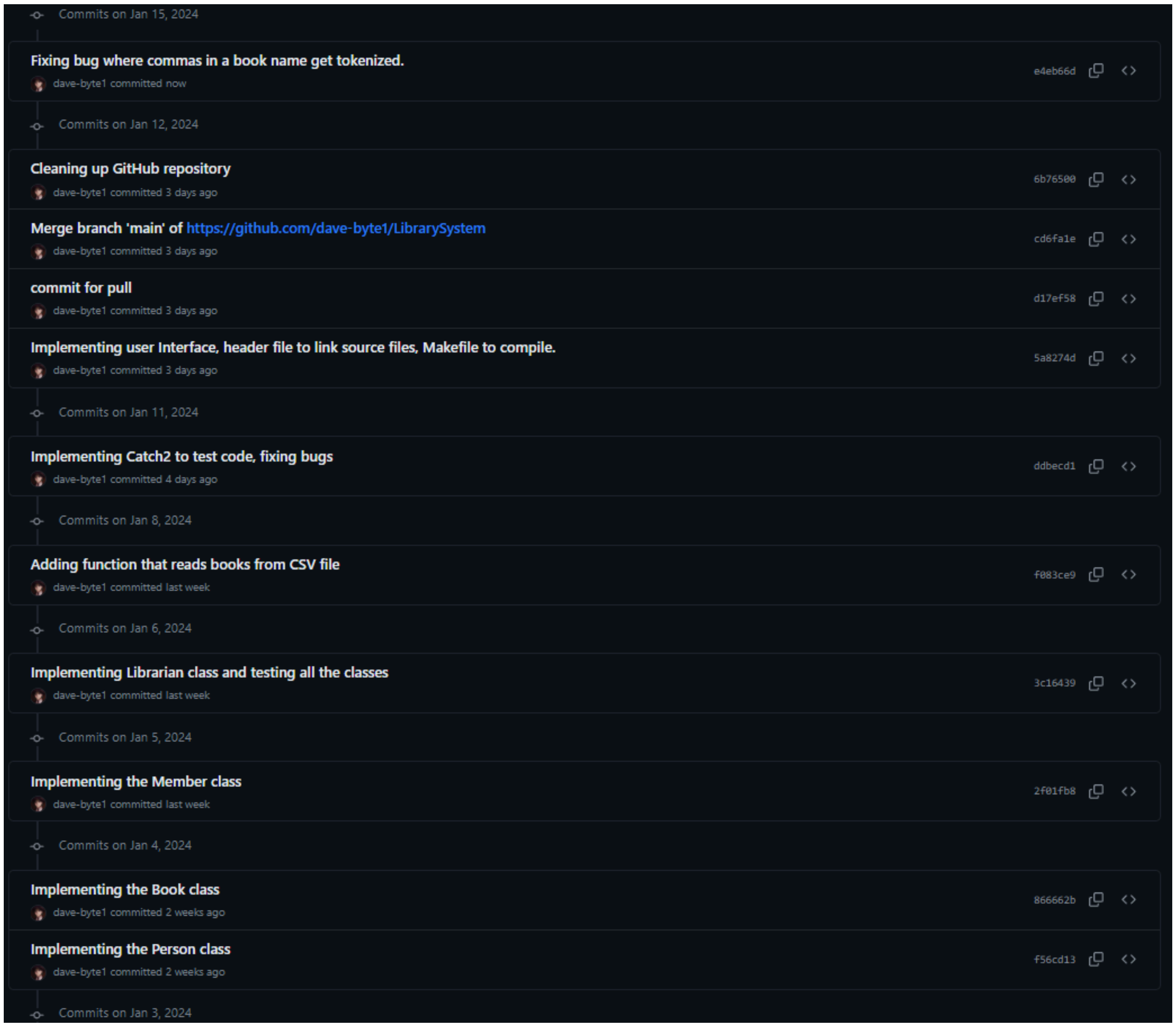
Approach:
I approached the implementation of the program by using the UML diagrams as an indication of where to start. For example, I implemented the classes one by one from top to bottom.
Constructors for the classes were created to initialize objects with parameters. Methods were implemented to perform tasks like setting and getting private data fields and issuing, returning, displaying, and calculating the fine of the Books.
Error handling was added to inform the Librarian that there was an error. For example, if a book is taken or if an invalid book ID was inserted.

Read books from CSV function was created to read the books from the CSV file, tokenizing the commas and setting the private data fields in the Book class e.g. ‘bookName’.
Then I went to test my classes to ensure they worked using Catch2.
And finally implementing the interface and creating a Makefile.

Makefile:
The Makefile was a key component in building my program. The Makefile can compile multiple source files and turn them into object files which are invoked by the compiler.
After the compilation, the Makefile links the objects in an executable.
The reason why a Makefile is useful is that without it developers would manually have to constantly type lengthy compiler commands every time they try to build their software. This can be very time-consuming and the Makefile simplifies this by just needing to type ‘make’ in the terminal.

How and why version control was used:
I used version control whenever I felt like the program was in a safe state to commit. It helped me look back at previous versions of what I’ve done, revert to previous states if I need to, and if needed I could create branches while maintaining the integrity of my code.

GitHub is designed for collaboration which allows multiple developers to work on the same project from anywhere in the world! Features like pull requests and code reviews are essential when it comes to collaborative coding. With Git/GitHub you can create branches to develop features, fix bugs, or create experiments without affecting the main branch. Branches can also be merged into the main branch. If the project is open source, it allows contributions from anyone around the world by forking the repository, making the change, and submitting it back to the developer for consideration. It also allowed me to access my work from my PC to my laptop by simply cloning the repository.



Commits on Jan 15, 2024

Fixing bug where commas in a book name get tokenized.
dave-byte1 committed now — e4eb66d

Commits on Jan 12, 2024

Cleaning up GitHub repository
dave-byte1 committed 3 days ago — 6b76500

Merge branch 'main' of https://github.com/dave-byte1/LibrarySystem
dave-byte1 committed 3 days ago — cd6fa1e

commit for pull
dave-byte1 committed 3 days ago — d17ef58

Implementing user Interface, header file to link source files, Makefile to compile.
dave-byte1 committed 3 days ago — 5a8274d

Commits on Jan 11, 2024

Implementing Catch2 to test code, fixing bugs
dave-byte1 committed 4 days ago — ddbecd1

Commits on Jan 8, 2024

Adding function that reads books from CSV file
dave-byte1 committed last week — f083ce9

Commits on Jan 6, 2024

Implementing Librarian class and testing all the classes
dave-byte1 committed last week — 3c16439

Commits on Jan 5, 2024

Implementing the Member class
dave-byte1 committed last week — 2f01fb8

Commits on Jan 4, 2024

Implementing the Book class
dave-byte1 committed 2 weeks ago — 866662b

Implementing the Person class
dave-byte1 committed 2 weeks ago — f56cd13

Commits on Jan 3, 2024

Testing Approach:

After completing each class, I used ‘Unit Testing’ by testing each class one by one to ensure that individual methods and functions work correctly.
After all my classes and functions were complete, I created Catch2 test cases to further help me test issues in my program and I managed to find one in the Book class.
I used ‘Continuous Testing’ to ensure that whenever new features were added they worked correctly with already existing functionality.

Test cases:
The test cases in ‘library_tests.cpp’ cover a range of functionality across the ‘Book’, ‘Member’, and ‘Librarian’ classes to ensure they work correctly.
In the Book class tests, I created a constructor to validate that the constructor correctly initializes the ‘Book’ object with the correct specific details. Then I checked that each getter and setter functions return the correct values. I ensured that setting a new due date updates the ‘dueDate’ attribute of a ‘Book’. And testing that borrowing a book correctly sets its due date and returning a book returns the due date to 0.
In the Member class tests I verified that creating a ‘Member’ correctly initializes the object and the getter and setter return the expected values. It also verifies that the ‘setBooksBorrowed’ function adds a ‘Book’ to the member’s ‘booksLoaned’ vector and that ‘getBooksBorrowed’ returns the correct list of borrowed books.
Finally, the Librarian class tests ensured that adding a member function works, the librarian can issue and return books without exceptions. And that the ‘calculateFine’ function calculates fines on overdue books.

Conclusion:

In summary, I utilized a range of features to develop the library management system, including the implementation of classes and interfaces, reading book data from a CSV file, employing version control with Git, conducting comprehensive testing, and using a Makefile for the build of the system and automation, all to make sure that the Library system can be used by a librarian without prior knowledge of the code.

Some limitations can include the lack of error handling in my user interface, Even though I have error handling in my classes I didn’t have enough time to add proper error handling for the user interface which could be one limitation to my system.
To avoid this limitation in the future I could probably start developing the program earlier to ensure I have enough time to provide error handling for my user interface.