CWP1: Project Library Management

Himanshu Verma
(M00885750)

Project Overview:

objective: 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.

Key Components:

Enum and Struct for Book Types:

I have built a struct BookTypeMapping to map strings to these enum values, and an enum class BookType for different book genres. This method of handling book categories is effective.

Date Class:

Date-related operations, such as assigning due dates and adjusting dates that surpass allowable boundaries, are managed by a custom Date class. The scheduling of book due dates for borrowed materials is greatly aided by this class.

Person, Member, and Librarian Classes:

To demonstrate inheritance, the **Person class** acts as the foundation for the Member and Librarian classes.

Derived from Person, the **Member class** manages features exclusive to library users, such as checking out and returning books.

The **Librarian class**, which is similarly developed from Person, has more extensive duties such as managing books and members and lending and returning books.

Book Class:

Represents the specifics and workings of a library book, such as the procedures for borrowing and returning books.

File Reading for Book Initialization:

In order to demonstrate file handling in C++, the librarian class reads from a CSV file to initialise the book collection.

Interactive Menu-driven Interface:

The primary feature facilitates user involvement with the system by offering an interactive menu-driven interface for library administration operations.

UML Diagram:

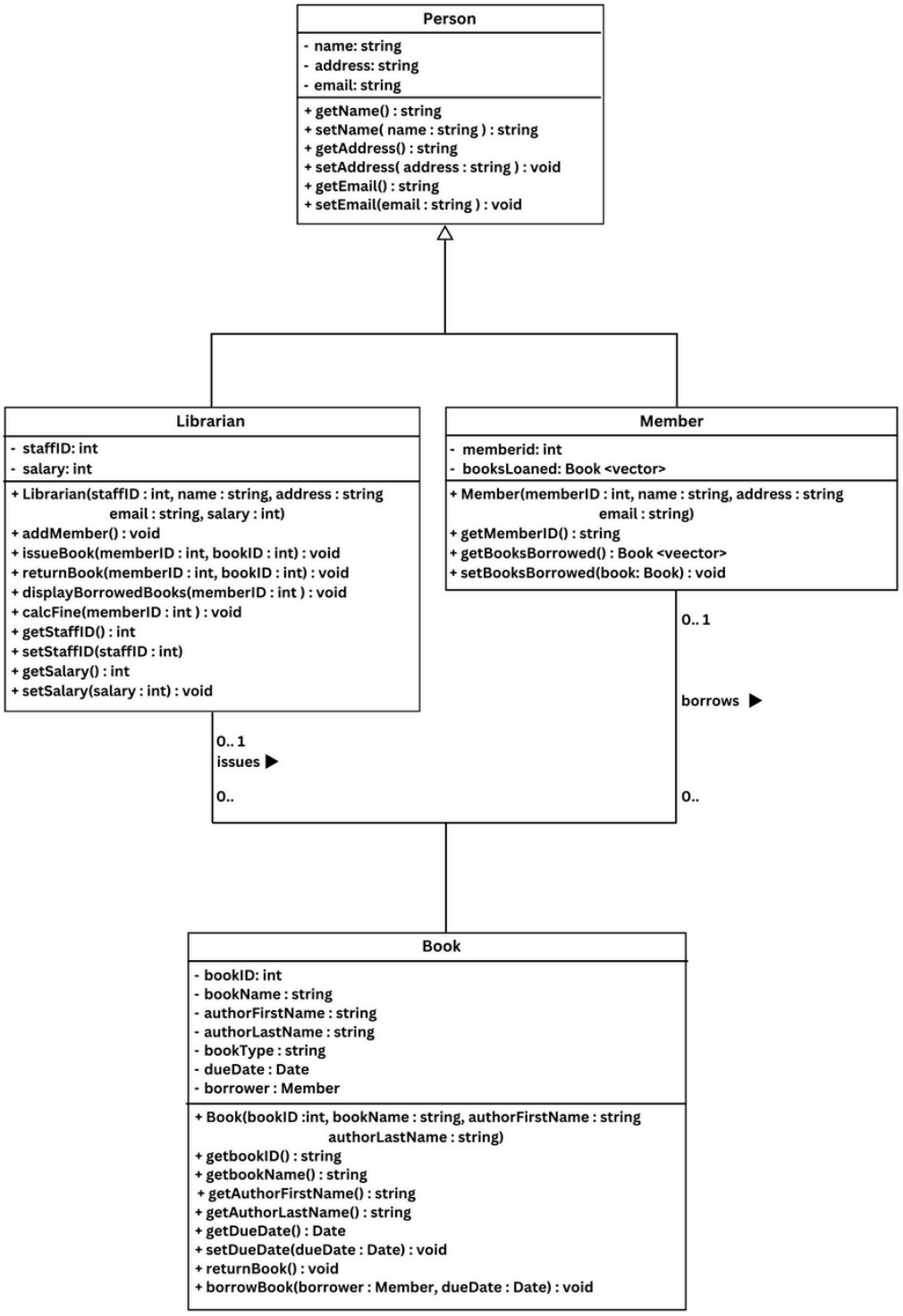

Figure 1: The UML class diagram for the library system.

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.

Array of mappings for converting strings to BookType enum values

```
// Array of mappings for converting strings to BookType enum values
BookTypeMapping bookTypeMappings[] = {
    {"guide", BookType::Guide},
    {"journals", BookType::Journals},
    {"diaries", BookType::Diaries},
    {"drama", BookType::Drama},
    {"science fiction", BookType::ScienceFiction},
    {"art", BookType::Art},
    {"romance", BookType::Romance},
    {"history", BookType::History},
    {"action and adventure", BookType::ActionAndAdventure},
    {"satire", BookType::Satire},
    {"mystery", BookType::Mystery},
    {"fantasy", BookType::Fantasy},
    {"horror", BookType::Horror},
    {"comics", BookType::Comics},
    {"science", BookType::Science},
    {"health", BookType::Health}};

// Function to convert a string to a BookType enum value
BookType stringToBookType(const std::string &str)
{
```

```
Class representing a basic Person with name, address, and email information
```

```
class Person
{
protected:
    // Name of the person
    string name;
    // Address of the person
    string address;
    // Email address of the person
    string email;

public:
    // Setter method to update the name of the person
    void setName(string &newName)
    {
        name = newName;
    }

    // Setter method to update the email address of the person
    void setEmail(string &newEmail)
    {
        email = newEmail;
    }
```

Person:

This is a base class that has email, address, and name attributes.

It provides ways to set and get these properties.

```
Class representing a book in the library & Constructor for Book class
```

```
// Class representing a book in the library
class Book
{
private:
    // Unique identifier for the book
    int bookID;
    // Title of the book
    string bookName;
    // First name of the book's author
    string authorFirstName;
    // Last name of the book's author
    string authorLastName;
    // Genre or type of the book
    BookType bookType;
    // Due date for the book (default set to December 12, 2025)
    Date dueDate = Date(12, 12, 2025);
    // Member who currently has the book on loan
    Member *borrower;
    // Number of pages in the book
    int pageCount;

public:
    // Constructor to initialize a Book object with specified attributes
    Book(int id, string name, int pages, string firstName, string lastName, BookT
    {
        bookID = id;
        bookName = name;
        authorFirstName = firstName;
        authorLastName = lastName;
        bookType = type;
        pageCount = pages;
    }
```

```
// Main function representing the Library Management System menu and interactions.
```

```
// Main function representing the Library Management System menu and interactions
int main()
{
    // Create a Librarian object with initial information
    Librarian librarian(1, "Librarian", "Librarian", "librarian@email.com", 50000

    // Variable to store the user's menu choice
    int choice;

    // Display the menu and handle user input until the user chooses to exit
    do
    {
        cout << "\nLibrary Management System Menu:\n";
        cout << "1. Add Member\n";
        cout << "2. Issue Book\n";
        cout << "3. Return Book\n";
        cout << "4. Display Borrowed Books\n";
        cout << "5. Calculate Fine\n";
        cout << "6. Display All Books\n";
        cout << "7. Exit\n";
        cout << "Enter your choice (1-7): ";

        // Get user input for menu choice
        cin >> choice;

        // Switch statement to handle different menu choices
        switch (choice)
        {
        case 1:
            // Call the Librarian's method to add a new member
            librarian.addMember();
```

```
// Class representing a librarian, inheriting from the Person class
```

```
class Librarian : public Person
{
private:
    // Unique identifier for the librarian
    int staffID;
    // Salary of the librarian
    int salary;
    // Vector to store library members
    vector<Member> members;
    // Vector to store library books
    vector<Book> books;

    // Private helper method to find a member in the library by their unique ID.
    // Returns a pointer to the found member or nullptr if not found.
    Member *findMemberById(int memberID)
    {
        for (auto &member : members)
        {
            if (member.getMemberID() == memberID)
            {
                return &member;
            }
        }
        return nullptr; // Member not found
    }

    // Private helper method to find a book in the library by its unique ID.
    // Returns a pointer to the found book or nullptr if not found.
    Book *findBookById(int bookID)
    {
        for (auto &book : books)
```

Librarian Class: This class inherits from Person and has additional attributes:
staffID: A unique identifier for a librarian.
salary: The salary of the librarian