COURSEWORK 1 PRESENTATION

CST-2550

INTRODUCTION

Student number – M00905923

Student Name – Shiv Deshmukh

BREIF DESCRIPTION –

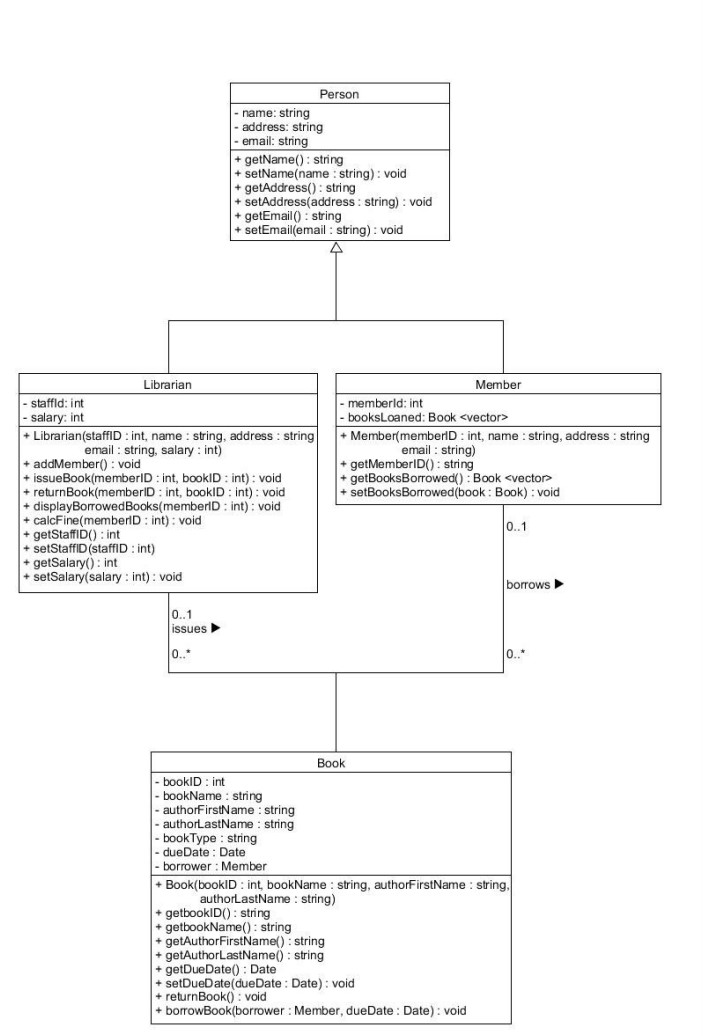
It is a simple library management system implemented in C++. It includes classes for BOOK PERSON (base class for Member), Member, and Librarian. The main function provides a basic user interface for interacting with the library system.

It allows users to

* Register new members.
* Issue and return books.
* Display borrowed books.
* Calculate fines for overdue book.,
* Exit the system

BREIF OVERVIEW-

* Implements a simple command-line interface for the library management system.
* Displays a menu for actions like registering members, issuing/returning books, displaying borrowed books, calculating fines, and exiting the system.
* Utilizes the Librarian class to perform these actions.



* DESCRIPTION OF THE UML DIAGRAMS –
* Classes:
* Book: Represents a book in the library. It has attributes such as book ID, name, author, due date, borrower information, etc.
* Person: A base class representing a person with attributes like name, address, and email.
* Member: A derived class from Person, representing a library member. It has a unique member ID and a list of books loaned.
* Librarian: Another derived class from Person, representing a librarian. It manages library operations such as adding members, issuing and returning books, calculating fines, etc.
* LibraryDate Structure:
* Represents a date in the library, with day, month, and year.
* Functions:
* getLibraryDateFromUser(): Takes user input for a date in DD MM YYYY format and returns a LibraryDate structure.
* addDaysToLibraryDate(): Adds a specified number of days to a given LibraryDate.
* dateDifference(): Calculates the difference in days between two LibraryDates.
* Book::borrowBook(): Allows a member to borrow a book by setting the borrower and due date.
* Book::returnBook(): Marks a borrowed book as returned.
* Librarian::addMember(): Adds a new member to the system after checking for existence based on ID, Name, Address, and Email.
* Librarian::displayLibraryBooks(): Displays information about library books from a CSV file.
* Librarian::issueBook(): Issues a book to a member by updating the due date.
* Librarian::returnBook(): Records the return of a borrowed book.
* Librarian::displayBorrowedBooks(): Displays books borrowed by a specific member.
* Librarian::calculateFine(): Calculates fines for overdue books for a specific member.

SOFTWARE TESTING –

User testing was my approach specifically –

* ***Unit testing***
* ***Component testing***
* ***System testing***

I used the Catch2 framework to perform my software tests as it is

* Header only
* No external dependencies
* Default main function

USING MAKEFILE –

Makefile was used in this software to –

* To have automated builds
* To have consistent builds
* To have easy maintenance
* To have clean organisation

There were two directories made

* ‘src’ (contains cpp file)
* ‘obj’(contains .exe file)

**CXX** specifies the C++ compiler (**g++** in this case).

**CXXFLAGS** specifies compiler flags, such as the C++ standard version (**-std=c++11**) and enabling compiler warnings (**-Wall**)

**SRC\_DIR** is the directory where your C++ source files are located.

**SOURCES** is a list of all C++ source files in the **src** directory.

**OBJ\_DIR** is the directory where object files will be stored.

**OBJECTS** is a list of all object files corresponding to source fi

The **clean** target removes the object files and the executable.

SUMMARY

It is a simple implementation of a Library Management System (LMS). Here's a summary

**Classes:**

**Book:** Represents a book in the library with attributes such as book ID, name, author, due date, borrower information, etc.

**Person:** A base class representing a person with attributes like name, address, and email.

**Member:** A derived class from Person, representing a library member. It has a unique member ID and a list of books loaned.

**Librarian:** Another derived class from Person, representing a librarian. It manages library operations such as adding members, issuing and returning books, calculating fines, etc.

**LibraryDate Structure:**

Represents a date in the library, with day, month, and year.

**Functions:**

Input functions (**getLibraryDateFromUser**, **addDaysToLibraryDate**, **dateDifference**) for handling dates.

Book-related methods (**borrowBook**, **returnBook**) to manage book borrowing and returning.

Librarian methods (**addMember**, **displayLibraryBooks**, **issueBook**, **returnBook**, **displayBorrowedBooks**, **calculateFine**) to perform various library management tasks.

**Main Function:**

Implements a simple command-line interface for the library management system.

Displays a menu for actions such as registering members, issuing/returning books, displaying borrowed books, calculating fines, and exiting the system.

Uses the Librarian class to perform these actions.

**User Interaction:**

Users (librarians) interact with the system through the console by selecting options from the menu, entering relevant information, and executing tasks.

**File Handling:**

Reads and displays information about library books from a CSV file named "library\_books.csv" using the **displayLibraryBooks** method.

**Basic Flow:**

The main loop continuously prompts the user for actions until the user chooses to exit.

Actions include registering new members, issuing/returning books, displaying borrowed books, calculating fines, and exiting the system.

**Purpose:**

The code provides a simplified demonstration of a library management system, allowing librarians to manage members, books, and transactions through a command-line interface.

CRITICAL REFLECTIONS

* Personally I feel the code should have more robust for my liking,
* I should have implemented more software tests which I failed to do.
* I should have first understood the UML diagram and the basic infrastructure of the software and then proceeded to start implementing it.

FUTURE APPROACH

* I will first understand the UML diagram and work on understanding the structure of the software before writing code
* Learn how the implement makefie and version control into my projects to make a robust outcome
* I will carry more software tests using catch2 for the robustness of my code and project

THANK YOU