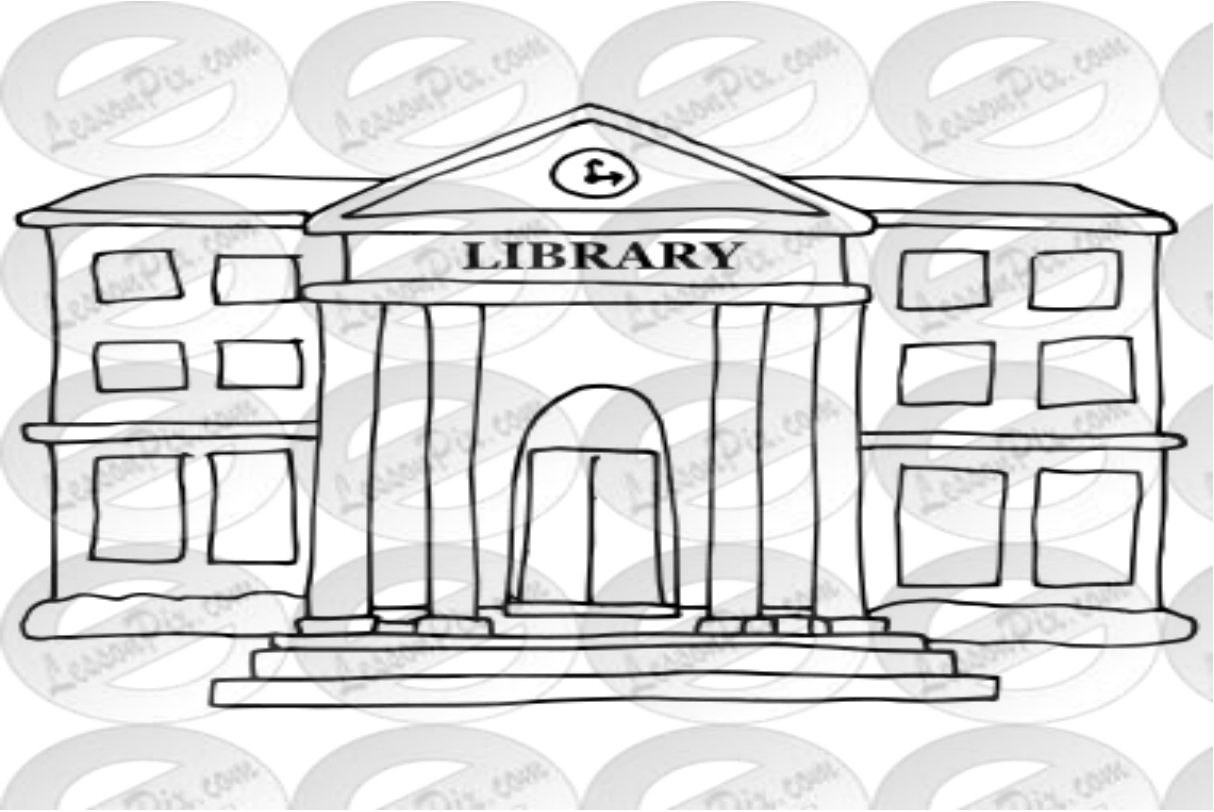
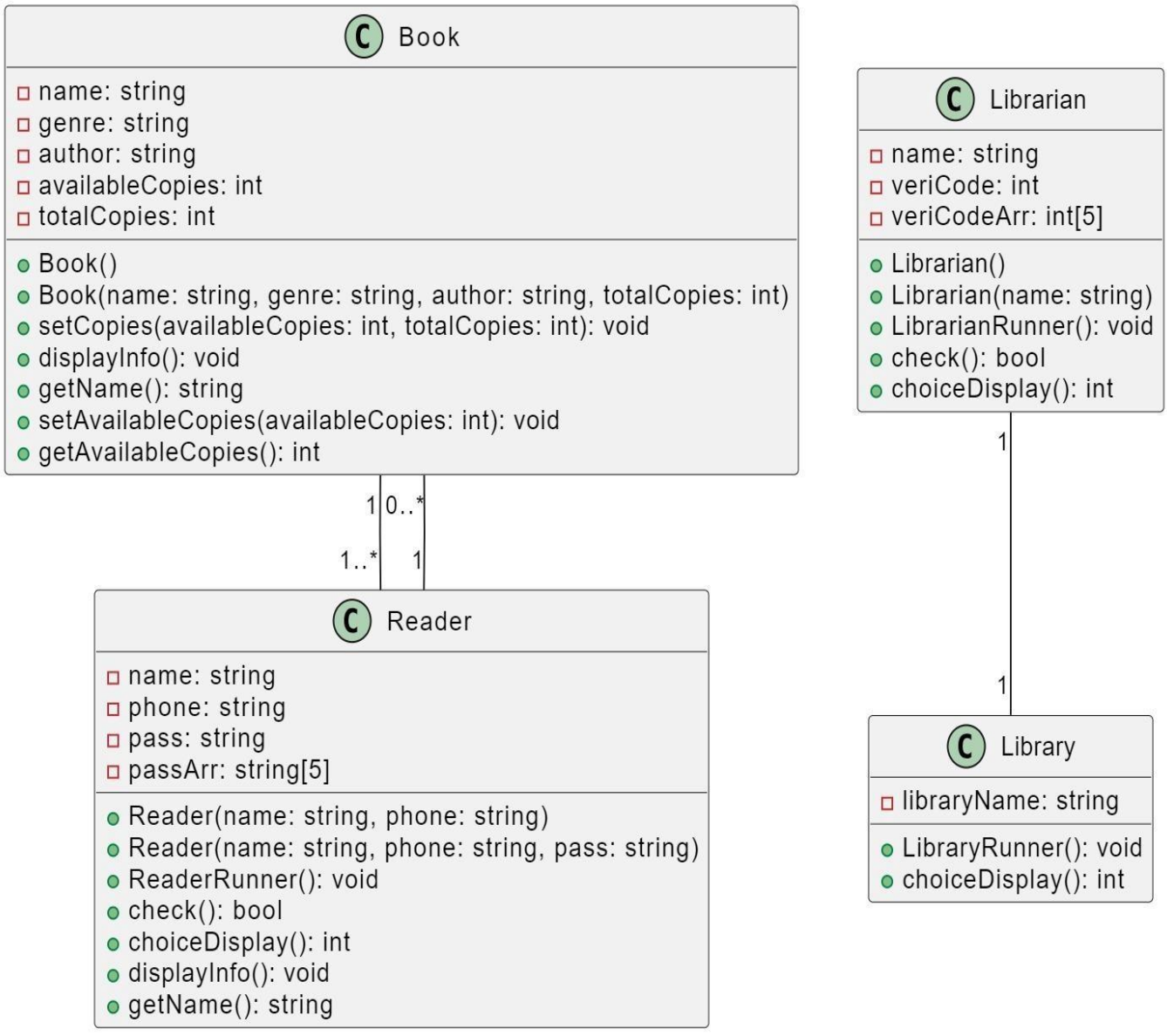
OOP PROJECT



EXPLANATION OF THE CODE

UML DIAGRAM



HEADER INCLUSIONS AND NAMESPACE

#pragma once: Ensures that the file is only included once throughout the compilation process, avoiding duplicate inclusions.
#include <iostream>: Contains the standard input/output stream library.
#include<string>: Include the string library.
#include<windows.h> : Contains Windows-specific API functions.
#include<vector>: Uses the vector library for dynamic arrays.
#include<iterator>: Provides iterators for STL containers.
#inlcude<iomanip>: for setting space such as setw()
Using namespace std;: Avoids prefixing standard functions and objects with std:: s

CLASS BOOK

class Book: Describes a class that contains book operations and details.
Private: Variables held by members are inaccessible from outside the class.
(string name, author, genre; Stores book information.) Copy counts are held
in int availableCopies, totalCopies;.
public: Constructors and member functions that are available from outside the class.
default constructor: is book(). **parameterized constructor**: Book(string name, string genre, string author, int totalCopies):
Initializes a book object using a parameterized constructor.
Available and total copies are set using the void setCopies(int availableCopies, int totalCopies) function. **void displayInfo():** Shows details about the book. **string getName():** Gives back the name of the book.
The amount of available copies is set using the void setAvailableCopies(int availableCopies) function. **int getAvailableCopies():** Gets the number of available copies.

GLOBAL BOOK LIST

vector<Book> bookList;
Declares a global vector to store a list of Book objects.

CLASS READER

class Reader: Defines a Reader class for managing reader operations.
public: string name, phone, pass: :Stores
reader details. string passArr[5]; Stores
predefined passwords.
Reader(string name, string phone): Constructor to initialize a reader without a password.
Reader(string name, string phone, string pass): Constructor to initialize a reader with a password. **void ReaderRunner():** Main function for reader operations. **getline(cin, pass);:** Gets password input.

if (!check()): Verifies password. **START::**
Label for restarting operations.
int chosenOption = choiceDisplay();: Displays choices and gets input.
switch (chosenOption): Processes the chosen option. case 1::
Searches for a book. case 2:: Borrows a book. case 3:: Returns a
book.
default: :Handles invalid choices.
bool check(): Checks if the entered password matches any in passArr.
int choiceDisplay(): Displays the choices and gets the input. **void displayInfo():** Displays reader information. **string getName():**
Returns the reader's name.

GLOBAL READER LIST

vector<Reader> readerList;

Declares a global vector to store a list of Reader objects.

CLASS LIBRARIAN

class Librarian: Defines a Librarian class for managing librarian operations. **private:**

string name;: Stores librarian name. **int**
veriCode;: Stores entered verification code.

int veriCodeArr[5];: Stores predefined verification codes. **public:**

Librarian(): Default constructor.

Librarian(string name): Parameterized constructor to initialize a librarian.

void LibrarianRunner(): Main function for librarian operations.
cin >> veriCode;: Gets verification code input. **if (!check()):**
Verifies code.

START:: Label for restarting operations.

int chosenOption = choiceDisplay();: Displays choices and gets input.

switch (chosenOption): Processes the chosen option.
case 1:: Adds a book. case 2:: Removes a book. case
3:: Searches for a book.

case 4:: Searches for a reader.

case 5:: Adds a reader.
case 6:: Removes a reader.
case 0:: Exits the program.

default:: Handles invalid choices. **bool check():** Checks if the entered code matches any in veriCodeArr.

int choiceDisplay(): Displays the choices and gets the input.

CLASS LIBRARY

class Library: Defines a Library class to manage the library system.
public: **string libraryName;:** Stores the name of the library.

void LibraryRunner(): Main function for running the library system. **getline(cin, libraryName);:** Gets the library name input.

START:: Label for restarting operations.
cout << "----Welcome to " << libraryName << " Library----" << endl;: Displays welcome message.

int chosenOption = choiceDisplay();: Displays choices and gets input.
switch (chosenOption): Processes the chosen option. case 0:: Exits the program. case 1:: Enters librarian mode. case 2:: Enters reader mode.

default:: Handles invalid choices. **int choiceDisplay():**
Displays the choices and gets the input.

OVERVIEW OF FUNCTIONALITY INITIALIZATION:

There is a global list of readers (readerList) and books (bookList).

Read Aloud:

Organises book information, including title, author, genre, and quantity.
Functions to set copies, show information, and retrieve book titles and available copies are provided.

Reading Group:

maintains the password, phone number, and name of the reader.
offers features for finding, borrowing, and returning books, among other reader actions.incorporates comparing passwords to pre-established passwords.

Class Librarian:

oversees the verification and details of librarians. offers features for managing readers as well as adding, deleting, and finding books for librarian operations. includes comparing the verification code to predetermined codes.

Class in the Library:

oversees the entire library system. gives the reader or librarian a way to start interacting with the system.
How It Operates
The patron walks into the library.

How It Operates

After logging in, the user selects whether to log in as a reader or a librarian in the library system. After confirming their code, they can handle books and readers if they log in as a librarian. Once they have the right password, they can look for, borrow, and return books if they log in as readers. The structure of the system is modular, with distinct responsibilities encapsulated in each class and function.

CPP FILE

#include <iostream>: Includes the input-output stream library needed for basic input and output operations.

#include "../bismaOOPProject/Header.h": Includes the header file where the classes (Book, Reader, Librarian, Library) are defined. This allows the main function to use these classes.

int main(): Defines the main function, which is the starting point of the program execution. Library library1;: Creates an instance of the Library class named library1. This instance will be used to manage the library system. library1.LibraryRunner();: Calls the LibraryRunner() function of the library1 object. This function handles the main logic for interacting with the library system, including choosing between librarian and reader modes and performing respective operations. return 0;: Ends the main function and returns 0, indicating that the program executed successfully.

The main function sets up the library system by creating a Library object and invoking its LibraryRunner() method. Here's how it fits with the rest of the code:

Library Initialization:

The LibraryRunner() method is called, prompting the user to enter the library's name and then presenting options to log in as either a librarian or a reader.

LibraryRunner Logic:

The user is greeted with a welcome message and prompted to choose between entering as a librarian or a reader.

Depending on the choice, the LibraryRunner() method directs the user to the respective functionalities:

Librarian Mode:

Prompts for the librarian's name and verification code.

If the verification code is correct, it presents a menu for adding, removing, and searching books, and managing readers.

Reader Mode:

Prompts for the reader's name and phone number.
Prompts for the reader's password. If correct, it presents a menu for searching, borrowing, and returning books.

Librarian and Reader Functionalities:

Librarian:

Can add new books, remove existing books, search for books, and manage reader information (add or remove readers).

Reader:

Can search for books, borrow books (decrement available copies), and return books (increment available copies).

DESIGN APPROACH

Using namespace std:

Font color :

The default text colour of the terminal output—gray text on a black background—is set by this function, fontColor. It accomplishes this by first getting a handle to the standard output of the console, which it then uses to set the console text properties. The Windows API is used by this feature, which is exclusive to the Windows operating system, to communicate with the console.

RESULT

The main function initializes the library system by creating an instance of the Library class and invoking its LibraryRunner() method. This method handles user interactions, including switching between librarian and reader modes and performing the respective operations. The integration of the main function with the rest of the code ensures that the program starts and runs the library system as intended, allowing users to manage books and readers effectively.