10/30/2025

LAB TASK # 06

SOFTWARE CONSTRUCTION AND DEVELOPMENT -LAB

**TAYYABA CHEEMA**

**Code without Refactoring**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

// Global Variables (bad practice!)

vector<string> bookName;

vector<string> bookAuthor;

vector<int> bookYear;

vector<bool> issued;

vector<string> issuedTo;

vector<string> studentList;

vector<string> studentRoll;

vector<int> fineList;

void addBook() {

string n, a;

int y;

cout << "Enter Book Name: ";

getline(cin >> ws, n);

cout << "Enter Author: ";

getline(cin >> ws, a);

cout << "Enter Year: ";

cin >> y;

bookName.push\_back(n);

bookAuthor.push\_back(a);

bookYear.push\_back(y);

issued.push\_back(false);

issuedTo.push\_back("None");

cout << "Book Added Successfully!\n";

}

void showAllBooks() {

cout << "\n--- Library Books ---\n";

for (int i = 0; i < bookName.size(); i++) {

cout << "Book ID: " << i + 1 << endl;

cout << "Title: " << bookName[i] << endl;

cout << "Author: " << bookAuthor[i] << endl;

cout << "Year: " << bookYear[i] << endl;

cout << "Issued: " << (issued[i] ? "Yes" : "No") << endl;

cout << "Issued To: " << issuedTo[i] << endl;

cout << "----------------------\n";

}

}

void registerStudent() {

string n, r;

cout << "Enter Student Name: ";

getline(cin >> ws, n);

cout << "Enter Roll No: ";

getline(cin >> ws, r);

studentList.push\_back(n);

studentRoll.push\_back(r);

fineList.push\_back(0);

cout << "Student Registered Successfully!\n";

}

void showStudents() {

cout << "\n--- Registered Students ---\n";

for (int i = 0; i < studentList.size(); i++) {

cout << "ID: " << i + 1 << endl;

cout << "Name: " << studentList[i] << endl;

cout << "Roll: " << studentRoll[i] << endl;

cout << "Fine: Rs." << fineList[i] << endl;

cout << "----------------------\n";

}

}

void issueBook() {

string roll;

int id;

cout << "Enter Student Roll: ";

getline(cin >> ws, roll);

bool foundStudent = false;

int studentIndex = -1;

for (int i = 0; i < studentRoll.size(); i++) {

if (studentRoll[i] == roll) {

foundStudent = true;

studentIndex = i;

break;

}

}

if (!foundStudent) {

cout << "Student not found!\n";

return;

}

cout << "Enter Book ID to Issue: ";

cin >> id;

id -= 1;

if (id < 0 || id >= bookName.size()) {

cout << "Invalid Book ID!\n";

return;

}

if (issued[id]) {

cout << "Book already issued to: " << issuedTo[id] << endl;

return;

}

issued[id] = true;

issuedTo[id] = studentList[studentIndex];

cout << "Book issued successfully to " << studentList[studentIndex] << endl;

}

void returnBook() {

int id;

string roll;

cout << "Enter Book ID to Return: ";

cin >> id;

id -= 1;

cout << "Enter Student Roll: ";

getline(cin >> ws, roll);

if (id < 0 || id >= bookName.size()) {

cout << "Invalid Book ID!\n";

return;

}

if (!issued[id]) {

cout << "This book was not issued.\n";

return;

}

bool foundStudent = false;

int studentIndex = -1;

for (int i = 0; i < studentRoll.size(); i++) {

if (studentRoll[i] == roll) {

foundStudent = true;

studentIndex = i;

break;

}

}

if (!foundStudent) {

cout << "Student not found!\n";

return;

}

int lateDays;

cout << "Enter Late Days (if any): ";

cin >> lateDays;

if (lateDays > 0) {

int fine = lateDays \* 10;

fineList[studentIndex] += fine;

cout << "Fine of Rs." << fine << " added to student account.\n";

}

issued[id] = false;

issuedTo[id] = "None";

cout << "Book returned successfully.\n";

}

void searchBook() {

string keyword;

cout << "Enter keyword to search in book names: ";

getline(cin >> ws, keyword);

bool found = false;

for (int i = 0; i < bookName.size(); i++) {

if (bookName[i].find(keyword) != string::npos) {

cout << "Book Found: " << bookName[i] << " by " << bookAuthor[i] << endl;

found = true;

}

}

if (!found) cout << "No matching books found.\n";

}

void deleteBook() {

int id;

cout << "Enter Book ID to delete: ";

cin >> id;

id -= 1;

if (id < 0 || id >= bookName.size()) {

cout << "Invalid Book ID!\n";

return;

}

bookName.erase(bookName.begin() + id);

bookAuthor.erase(bookAuthor.begin() + id);

bookYear.erase(bookYear.begin() + id);

issued.erase(issued.begin() + id);

issuedTo.erase(issuedTo.begin() + id);

cout << "Book deleted successfully!\n";

}

void deleteStudent() {

int id;

cout << "Enter Student ID to delete: ";

cin >> id;

id -= 1;

if (id < 0 || id >= studentList.size()) {

cout << "Invalid Student ID!\n";

return;

}

studentList.erase(studentList.begin() + id);

studentRoll.erase(studentRoll.begin() + id);

fineList.erase(fineList.begin() + id);

cout << "Student deleted successfully!\n";

}

void mainMenu() {

while (true) {

cout << "\n=== Library Management System ===\n";

cout << "1. Add Book\n";

cout << "2. Show All Books\n";

cout << "3. Register Student\n";

cout << "4. Show Students\n";

cout << "5. Issue Book\n";

cout << "6. Return Book\n";

cout << "7. Search Book\n";

cout << "8. Delete Book\n";

cout << "9. Delete Student\n";

cout << "10. Exit\n";

cout << "Enter your choice: ";

int ch;

cin >> ch;

switch (ch) {

case 1: addBook(); break;

case 2: showAllBooks(); break;

case 3: registerStudent(); break;

case 4: showStudents(); break;

case 5: issueBook(); break;

case 6: returnBook(); break;

case 7: searchBook(); break;

case 8: deleteBook(); break;

case 9: deleteStudent(); break;

case 10: cout << "Goodbye!"; return;

default: cout << "Invalid choice!\n"; break;

}

}

}

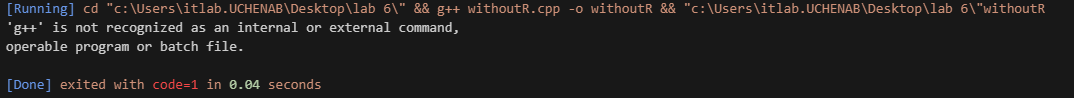
int main() {

mainMenu();

return 0;

}

Output:



**Code after refactoring:**

#include <iostream>

#include <string>

#include <vector>

// Constants

const int FINE\_PER\_DAY = 10;

// Book class

class Book {

public:

Book(std::string name, std::string author, int year)

: name(name), author(author), year(year), issued(false), issuedTo("None") {}

std::string getName() const { return name; }

std::string getAuthor() const { return author; }

int getYear() const { return year; }

bool isIssued() const { return issued; }

std::string getIssuedTo() const { return issuedTo; }

void issueBook(const std::string& studentName) {

issued = true;

issuedTo = studentName;

}

void returnBook() {

issued = false;

issuedTo = "None";

}

private:

std::string name;

std::string author;

int year;

bool issued;

std::string issuedTo;

};

// Student class

class Student {

public:

Student(std::string name, std::string roll) : name(name), roll(roll), fine(0) {}

std::string getName() const { return name; }

std::string getRoll() const { return roll; }

int getFine() const { return fine; }

void addFine(int amount) {

fine += amount;

}

void clearFine() {

fine = 0;

}

private:

std::string name;

std::string roll;

int fine;

};

// LibraryManager class

class LibraryManager {

public:

void addBook(const Book& book) {

books.push\_back(book);

}

void registerStudent(const Student& student) {

students.push\_back(student);

}

void issueBook(const std::string& roll, int bookId) {

if (bookId < 1 || bookId > books.size()) {

throw std::invalid\_argument("Invalid book ID");

}

auto& book = books[bookId - 1];

if (book.isIssued()) {

throw std::runtime\_error("Book already issued");

}

auto student = findStudentByRoll(roll);

if (!student) {

throw std::invalid\_argument("Student not found");

}

book.issueBook(student->getName());

}

void returnBook(int bookId, int lateDays) {

if (bookId < 1 || bookId > books.size()) {

throw std::invalid\_argument("Invalid book ID");

}

auto& book = books[bookId - 1];

if (!book.isIssued()) {

throw std::runtime\_error("Book not issued");

}

book.returnBook();

if (lateDays > 0) {

// Find student and add fine

for (auto& student : students) {

if (student.getName() == book.getIssuedTo()) {

student.addFine(lateDays \* FINE\_PER\_DAY);

break;

}

}

}

}

// Other methods...

private:

std::vector<Book> books;

std::vector<Student> students;

Student\* findStudentByRoll(const std::string& roll) {

for (auto& student : students) {

if (student.getRoll() == roll) {

return &student;

}

}

return nullptr;

}

};

// Input/Output functions

void addBook(LibraryManager& library) {

std::string name, author;

int year;

std::cout << "Enter book name: ";

std::getline(std::cin, name);

std::cout << "Enter author: ";

std::getline(std::cin, author);

std::cout << "Enter year: ";

std::cin >> year;

std::cin.ignore();

Book book(name, author, year);

library.addBook(book);

}

// Main function

int main() {

LibraryManager library;

while (true) {

std::cout << "\n=== Library Management System ===\n";

std::cout << "1. Add Book\n";

std::cout << "2. Register Student\n";

std::cout << "3. Issue Book\n";

std::cout << "4. Return Book\n";

std::cout << "5. Exit\n";

int choice;

std::cin >> choice;

std::cin.ignore();

switch (choice) {

case 1:

addBook(library);

break;

// Other cases...

case 5:

return 0;

default:

std::cout << "Invalid choice\n";

}

}

return 0;

}

Output:

