**NAME: Muhammad Haris**

**REG: BCS221027**

CODE

#include <iostream>

#include <vector>

#include <string>

#include <ctime>

using namespace std;

class Item

{

protected:

int itemID;

string title;

bool availability;

public:

Item(int id, string t) : itemID(id), title(t), availability(true)

{}

virtual void getItemDetails() const = 0;

virtual bool checkAvailability() const

{

return availability;

}

virtual void checkOut() = 0;

virtual void checkIn() = 0;

virtual ~Item() {}

};

class Book : public Item

{

private:

string author;

string ISBN;

public:

Book(int id, string t, string a, string isbn) : Item(id, t), author(a), ISBN(isbn)

{}

void getItemDetails() const override

{

cout << "Book ID: " << itemID << "\nTitle: " << title << "\nAuthor: " << author << "\nISBN: " << ISBN << "\nAvailability: " << (availability ? "Yes" : "No") << endl;

}

void checkOut() override

{

if (availability)

{

availability = false;

cout << "Book checked out successfully.\n";

}

else

{

cout << "Book is currently unavailable.\n";

}

}

void checkIn() override

{

availability = true;

cout << "Book checked in successfully.\n";

}

};

class Journal : public Item

{

private:

string publisher;

int issueNumber;

public:

Journal(int id, string t, string p, int issue) : Item(id, t), publisher(p), issueNumber(issue)

{}

void getItemDetails() const override

{

cout << "Journal ID: " << itemID << "\nTitle: " << title << "\nPublisher: " << publisher << "\nIssue Number: " << issueNumber << "\nAvailability: " << (availability ? "Yes" : "No") << endl;

}

void checkOut() override

{

if (availability)

{

availability = false;

cout << "Journal checked out successfully.\n";

}

else

{

cout << "Journal is currently unavailable.\n";

}

}

void checkIn() override

{

availability = true;

cout << "Journal checked in successfully.\n";

}

};

class Member

{

private:

int memberID;

string name;

string contactInfo;

int loanLimit;

vector<Item\*> loanHistory;

public:

Member(int id, string n, string contact, int limit = 5) : memberID(id), name(n), contactInfo(contact), loanLimit(limit)

{}

void borrowItem(Item& item)

{

if (loanHistory.size() < loanLimit && item.checkAvailability())

{

item.checkOut();

loanHistory.push\_back(&item);

}

else

{

cout << "Cannot borrow item. Either the loan limit has been reached or the item is unavailable.\n";

}

}

void returnItem(Item& item)

{

auto it = find(loanHistory.begin(), loanHistory.end(), &item);

if (it != loanHistory.end())

{

item.checkIn();

loanHistory.erase(it);

}

else

{

cout << "Item not found in loan history.\n";

}

}

void getLoanHistory() const

{

cout << "Loan History for " << name << ":\n";

for (const auto& item : loanHistory)

{

item->getItemDetails();

}

}

};

class Loan

{

private:

int loanID;

Item\* item;

Member\* member;

time\_t dueDate;

time\_t returnDate;

public:

Loan(int id, Item\* i, Member\* m, time\_t due) : loanID(id), item(i), member(m), dueDate(due), returnDate(0) {}

void returnLoan()

{

returnDate = time(0);

item->checkIn();

}

double calculateFine() const

{

if (returnDate > dueDate)

{

double finePerDay = 2.0;

return difftime(returnDate, dueDate) / (60 \* 60 \* 24) \* finePerDay;

}

return 0.0;

}

};

int main()

{

Book book1(1, "THE Great Expectation", "Charles Dickens", "123456789");

Journal journal1(2, "Pakistan Journal of Computer and Information Systems", "Pakistan Scientific and Technological Information Centre", 202);

Member member1(1, "Haris", "haris@gmail.com");

member1.borrowItem(book1);

member1.returnItem(book1);

member1.getLoanHistory();

time\_t now = time(0);

time\_t due = now + 7 \* 24 \* 60 \* 60;

Loan loan1(1, &book1, &member1, due);

loan1.returnLoan();

cout << "Fine for late return: $" << loan1.calculateFine() << endl;

return 0;

}

**OUTPUT**

