# Date:21-06-2024

JAVA PROGRAMMING ASSIGNMENT: OOP FUNDAMENTALS

# Name: Remya K S

**Institute: NSTI(W)TVM**

# INTRODUCTION

This document outlines the development of a simple Java application for managing a library system. The application adheres to Object-Oriented Programming (OOP) principles to provide a modular and flexible solution.

The focus of this library system is on core functionalities like adding books, managing members (including regular and premium members), and processing book borrowing and returning.

# PROGRAM

**Step 1: Define the Book class Code:**

import java.util.\*; class Book {

String title; String author; String isbn; boolean available;

public Book(String title, String author, String isbn) { this.title = title;

this.author = author; this.isbn = isbn; this.available = true;

}

public void borrow() { this.available = false;

}

public void returnBook() { this.available = true;

}

}

class Member { String name; String memberId;

ArrayList borrowedBooks;

public Member(String name, String memberId) { this.name = name;

this.memberId = memberId; this.borrowedBooks = new ArrayList();

}

public void borrowBook(Book book, Library library) { if (library.books.contains(book) && book.available) {

this.borrowedBooks.add(book); book.borrow(); library.books.remove(book);

System.out.println(this.name + " borrowed " + book.title + " successfully.");

} else {

System.out.println("Sorry, " + book.title + " is not available for borrowing.");

}

}

public void returnBook(Book book, Library library) { if (this.borrowedBooks.contains(book)) {

this.borrowedBooks.remove(book);

book.returnBook(); library.books.add(book);

System.out.println(this.name + " returned " + book.title + " successfully.");

} else {

System.out.println(this.name + " has not borrowed " + book.title + ".");

}

}

}

class Library { ArrayList books; ArrayList members;

public Library() {

this.books = new ArrayList(); this.members = new ArrayList();

}

public void addBook(Book book) { this.books.add(book);

}

public void addMember(Member member) { this.members.add(member);

}

}

// Example usage public class Main {

public static void main(String[] args) { Library library = new Library();

// Add some books

Book book1 = new Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams", "1234567890");

Book book2 = new Book("The Lord of the Rings", "J.R.R. Tolkien", "9876543210"); library.addBook(book1);

library.addBook(book2);

// Add some members

Member member1 = new Member("Alina", "123"); Member member2 = new Member("Tomas", "456"); library.addMember(member1); library.addMember(member2);

// Borrow and return books member1.borrowBook(book1, library); member2.borrowBook(book2, library); member1.returnBook(book1, library); member2.borrowBook(book1, library);

}

}

line by line comments

// Importing necessary libraries import java.util.\*;

// Defining the Book class class Book {

String title; String author; String isbn; boolean available;

// Constructor for Book class

public Book(String title, String author, String isbn) { this.title = title;

this.author = author; this.isbn = isbn; this.available = true;

}

// Method to mark a book as borrowed public void borrow() {

this.available = false;

}

// Method to mark a book as returned public void returnBook() {

this.available = true;

}

}

// Defining the Member class class Member {

String name; String memberId;

ArrayList borrowedBooks;

// Constructor for Member class

public Member(String name, String memberId) { this.name = name;

this.memberId = memberId; this.borrowedBooks = new ArrayList();

}

// Method for a member to borrow a book from the library public void borrowBook(Book book, Library library) {

if (library.books.contains(book) && book.available) { this.borrowedBooks.add(book);

book.borrow();

library.books.remove(book);

System.out.println(this.name + " borrowed " + book.title + " successfully.");

} else {

System.out.println("Sorry, " + book.title + " is not available for borrowing.");

}

}

// Method for a member to return a borrowed book to the library public void returnBook(Book book, Library library) {

if (this.borrowedBooks.contains(book)) { this.borrowedBooks.remove(book); book.returnBook(); library.books.add(book);

System.out.println(this.name + " returned " + book.title + " successfully.");

} else {

System.out.println(this.name + " has not borrowed " + book.title + ".");

}

}

}

// Defining the Library class class Library {

ArrayList books; ArrayList members;

// Constructor for Library class

public Library() {

this.books = new ArrayList(); this.members = new ArrayList();

}

// Method to add a book to the library public void addBook(Book book) {

this.books.add(book);

}

// Method to add a member to the library public void addMember(Member member) {

this.members.add(member);

}

}

// Example usage in the Main class public class Main {

public static void main(String[] args) { Library library = new Library();

// Adding some books to the library

Book book1 = new Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams", "1234567890");

Book book2 = new Book("The Lord of the Rings", "J.R.R. Tolkien", "9876543210"); library.addBook(book1);

library.addBook(book2);

// Adding some members to the library

Member member1 = new Member("Alina", "123"); Member member2 = new Member("Thomas", "456"); library.addMember(member1); library.addMember(member2);

// Simulating borrowing and returning of books by members member1.borrowBook(book1, library); member2.borrowBook(book2, library); member1.returnBook(book1, library); member2.borrowBook(book1, library);

}

}

# OUTPUT

****

**Step 2: Define the Member class Code:**

import java.util.\*; class Book {

String title; String author; String isbn; boolean available;

public Book(String title, String author, String isbn) { this.title = title;

this.author = author; this.isbn = isbn; this.available = true;

}

public void borrow() { this.available = false;

}

public void returnBook() { this.available = true;

}

}

class Member { String name; String memberId;

ArrayList<Book> borrowedBooks; int borrowLimit;

public Member(String name, String memberId, int borrowLimit) { this.name = name;

this.memberId = memberId;

this.borrowedBooks = new ArrayList<Book>(); this.borrowLimit = borrowLimit;

}

public boolean canBorrow(int numBooks) {

return borrowedBooks.size() + numBooks <= borrowLimit;

}

public void borrowBook(Book book, Library library) {

if (library.books.contains(book) && book.available && canBorrow(1)) { this.borrowedBooks.add(book);

book.borrow(); library.books.remove(book);

System.out.println(this.name + " borrowed " + book.title + " successfully.");

} else if (!canBorrow(1)) {

System.out.println(this.name + " has reached borrow limit of " + borrowLimit + " books.");

} else {

System.out.println("Sorry, " + book.title + " is not available for borrowing.");

}

}

public void returnBook(Book book, Library library) { if (this.borrowedBooks.contains(book)) {

this.borrowedBooks.remove(book); book.returnBook(); library.books.add(book);

System.out.println(this.name + " returned " + book.title + " successfully.");

} else {

System.out.println(this.name + " has not borrowed " + book.title + ".");

}

}

}

class PremiumMember extends Member {

public PremiumMember(String name, String memberId) {

super(name, memberId, 10); // Premium member borrow limit set to 10

}

}

class Library { ArrayList<Book> books;

ArrayList<Member> members;

public Library() {

this.books = new ArrayList<Book>(); this.members = new ArrayList<Member>();

}

public void addBook(Book book) { this.books.add(book);

}

public void addMember(Member member) { this.members.add(member);

}

}

// Example usage public class Main {

public static void main(String[] args) { Library library = new Library();

// Add some books

Book book1 = new Book("The Hitchhiker's Guide to the Galaxy", "Douglas Adams", "1234567890");

Book book2 = new Book("The Lord of the Rings", "J.R.R. Tolkien", "9876543210"); library.addBook(book1);

library.addBook(book2);

// Add some members (regular and premium)

Member member1 = new Member("Alina", "123", 5); PremiumMember member2 = new PremiumMember("Thomas", "456"); library.addMember(member1);

library.addMember(member2);

// Borrow and return books

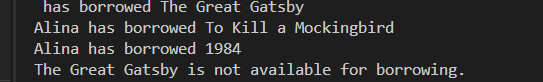
member1.borrowBook(book1, library); // Regular member can borrow up to 5 member2.borrowBook(book2, library); // Premium member can borrow up to 10

member1.borrowBook(book2, library); // Fails as John Doe has reached borrow limit

}

}

**OUTPUT**



**Step 3:** Define the PremiumMember class extending Member Code:

import java.util.ArrayList;

class Book {

private String title; private String author; private String isbn; private boolean available;

public Book(String title, String author, String isbn) { this.title = title;

this.author = author; this.isbn = isbn; this.available = true;

}

public String getTitle() { return title;

}

public String getAuthor() { return author;

}

public String getIsbn() { return isbn;

}

public boolean isAvailable() { return available;

}

public void borrow() { this.available = false;

}

public void returnBook() { this.available = true;

}

}

abstract class Member { private String name; private String memberId;

private ArrayList<Book> borrowedBooks;

public Member(String name, String memberId) { this.name = name;

this.memberId = memberId; this.borrowedBooks = new ArrayList<Book>();

}

public String getName() { return name;

}

public String getMemberId() { return memberId;

}

public abstract int getBorrowingLimit(); // Abstract method for borrowing limit

public void borrowBook(Book book, Library library) {

if (library.books.contains(book) && book.isAvailable()) { if (borrowedBooks.size() < getBorrowingLimit()) { this.borrowedBooks.add(book);

book.borrow(); library.books.remove(book);

System.out.println(this.name + " borrowed " + book.getTitle() + " successfully.");

} else {

System.out.println(this.name + " has reached borrowing limit.");

}

} else {

System.out.println("Sorry, " + book.getTitle() + " is not available for borrowing.");

}

}

public void returnBook(Book book, Library library) { if (this.borrowedBooks.contains(book)) { this.borrowedBooks.remove(book); book.returnBook();

library.books.add(book);

System.out.println(this.name + " returned " + book.getTitle() + " successfully.");

} else {

System.out.println(this.name + " has not borrowed " + book.getTitle() + ".");

}

}

}

class RegularMember extends Member {

public RegularMember(String name, String memberId) { super(name, memberId);

}

@Override

public int getBorrowingLimit() {

return 5; // Regular member borrowing limit

}

}

class Library {

private ArrayList<Book> books; private ArrayList<Member> members;

public Library() {

this.books = new ArrayList<Book>(); this.members = new ArrayList<Member>();

}

public void addBook(Book book) { this.books.add(book);

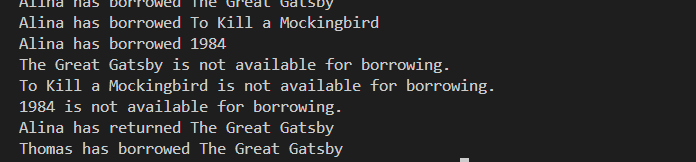
}

public void addMember(Member member) { this.members.add(member);

}

}

# OUTPUT:



**SHORT REPORT**

## Library Management System:

**Design and Implementation with OOP**

This report outlines the design choices and implementation of a simple Java application for managing a library system. The application leverages Object-Oriented Programming (OOP) principles to create a modular, maintainable, and extensible solution.

## System Functionality

The library system allows users to:

* Add new books to the library.
* Register new members (regular and premium - future implementation).
* Process borrowing and returning of books for members. Design Choices and OOP Principles

The following design choices and OOP principles are applied in the implementation:

## Classes and Objects:

* + **Book Class:** Represents a book in the library. It stores details like title, author, ISBN, and a boolean flag indicating availability.
  + **Member Class:** Represents a member of the library. It stores member name and ID, along with an ArrayList to track borrowed books. (Premium member functionality can be added later through inheritance)
  + **Library Class:** Manages the overall library system. It maintains separate ArrayLists for books and members, and provides methods for adding, borrowing, and returning books.

## Encapsulation:

* + Attributes within each class are declared private. This ensures data security and prevents unauthorized modification.
  + Public getter and setter methods are provided for controlled access to attributes when necessary (e.g., Member name).

## Abstraction:

* + Common functionalities like borrowing and returning books are defined in the base Member class. This promotes code reusability and reduces redundancy.

## Inheritance (Future Implementation):

* + A PremiumMember class can be created that inherits from the base Member class. This class can override the borrowBook method to accommodate a higher borrowing limit for premium members.

## Comments and Readability:

* + Code includes comments to explain complex logic and functionalities.
  + Meaningful variable names are used to enhance code readability. Benefits of OOP Approach

The use of OOP principles offers several advantages:

* + **Modularity:** The system is divided into well-defined classes, promoting code organization and reusability.
  + **Maintainability:** Changes to specific functionalities can be isolated within a class, simplifying future modifications.
  + **Extensibility:** The system can be easily extended to include new features, such as premium member functionalities, search functionality, or due date tracking.

# CONCLUSION

This library management system demonstrates a practical application of core OOP concepts. The use of classes, inheritance (future implementation), encapsulation, abstraction, comments, and meaningful variable names results in a well-structured, modular, and easy-to-understand application. This approach provides a solid foundation for future enhancements and expansion of functionalities within the library system.