**Implement the following projects which focus on different aspects of Java programming, including:**

Object-Oriented Programming principles

Exception handling

File operations

Data structures

Input validation

User interface design

**Library Book Management**

Implement a simple library system to track books, their availability, and borrowing status using object oriented programming in Java.

Key points:

- Use ArrayList for book collection

- Track due dates

- Implement search functionality

import java.io.\*;

import java.time.\*;

import java.time.format.\*;

import java.util.\*;

// Custom exceptions

class BookNotFoundException extends Exception {

public BookNotFoundException(String message) {

super(message);

}

}

class BookAlreadyExists**Library Book Management**Exception extends Exception {

public BookAlreadyExistsException(String message) {

super(message);

}

}

// Book class representing a library book

class Book implements Serializable {

private final String isbn;

private final String title;

private final String author;

private boolean isAvailable;

private LocalDate dueDate;

private String borrower;

public Book(String isbn, String title, String author) {

this.isbn = isbn;**Library Book Management**

this.title = title;

this.author = author;

this.isAvailable = true;

this.dueDate = null;

this.borrower = null;

}

// Getters and setters

public String getIsbn() { return isbn; }

public String getTitle() { return title; }

public String getAuthor() { return author; }

public boolean isAvailable() { return isAvailable; }

public LocalDate getDueDate() { return dueDate; }

public String getBorrower() { return borrower; }

public void borrowBook(String borrower, int loanPeriodDays) {

this.isAvailable = false;

this.borrower = borrower;

this.dueDate = LocalDate.now().plusDays(loanPeriodDays);

}

public void returnBook() {

this.isAvailable = true;

this.borrower = null;

this.dueDate = null;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append(String.format("ISBN: %s\nTitle: %s\nAuthor: %s\nStatus: %s",

isbn, title, author, isAvailable ? "Available" : "Borrowed"));

if (!isAvailable) {

sb.append(String.format("\nBorrowed by: %s\nDue Date: %s",

borrower, dueDate.format(DateTimeFormatter.ISO\_LOCAL\_DATE)));

}

return sb.toString();

}

}

// Library class to manage the book collection

class Library implements Serializable {

private final List<Book> books;

private final String dataFile = "library\_data.ser";

public Library() {

this.books = new ArrayList<>();

loadData();

}

// Add a new book

public void addBook(String isbn, String title, String author)

throws BookAlreadyExistsException {

if (findBookByIsbn(isbn).isPresent()) {

throw new BookAlreadyExistsException("Book with ISBN " + isbn + " already exists");

}

books.add(new Book(isbn, title, author));

saveData();

}

// Remove a book

public void removeBook(String isbn) throws BookNotFoundException {

Book book = findBookByIsbn(isbn)

.orElseThrow(() -> new BookNotFoundException("Book not found"));

books.remove(book);

saveData();

}

// Borrow a book

public void borrowBook(String isbn, String borrower, int loanPeriodDays)

throws BookNotFoundException {

Book book = findBookByIsbn(isbn)

.orElseThrow(() -> new BookNotFoundException("Book not found"));

if (!book.isAvailable()) {

throw new IllegalStateException("Book is already borrowed");

}

book.borrowBook(borrower, loanPeriodDays);

saveData();

}

// Return a book

public void returnBook(String isbn) throws BookNotFoundException {

Book book = findBookByIsbn(isbn)

.orElseThrow(() -> new BookNotFoundException("Book not found"));

book.returnBook();

saveData();

}

// Search functions

public Optional<Book> findBookByIsbn(String isbn) {

return books.stream()

.filter(book -> book.getIsbn().equals(isbn))

.findFirst();

}

public List<Book> searchBooks(String query) {

String lowercaseQuery = query.toLowerCase();

return books.stream()

.filter(book ->

book.getTitle().toLowerCase().contains(lowercaseQuery) ||

book.getAuthor().toLowerCase().contains(lowercaseQuery) ||

book.getIsbn().toLowerCase().contains(lowercaseQuery))

.toList();

}

// Get overdue books

public List<Book> getOverdueBooks() {

LocalDate today = LocalDate.now();

return books.stream()

.filter(book -> !book.isAvailable() &&

book.getDueDate().isBefore(today))

.toList();

}

// Get all books

public List<Book> getAllBooks() {

return new ArrayList<>(books);

}

// Save data to file

private void saveData() {

try (ObjectOutputStream oos = new ObjectOutputStream(

new FileOutputStream(dataFile))) {

oos.writeObject(books);

} catch (IOException e) {

System.err.println("Error saving library data: " + e.getMessage());

}

}

// Load data from file

@SuppressWarnings("unchecked")

private void loadData() {

File file = new File(dataFile);

if (!file.exists()) return;

try (ObjectInputStream ois = new ObjectInputStream(

new FileInputStream(file))) {

List<Book> loadedBooks = (List<Book>) ois.readObject();

books.addAll(loadedBooks);

} catch (IOException | ClassNotFoundException e) {

System.err.println("Error loading library data: " + e.getMessage());

}

}

}

// Main class with user interface

public class LibraryManagementSystem {

private static final Scanner scanner = new Scanner(System.in);

private static final Library library = new Library();

public static void main(String[] args) {

while (true) {

try {

displayMenu();

String choice = scanner.nextLine();

switch (choice) {

case "1" -> addBook();

case "2" -> removeBook();

case "3" -> borrowBook();

case "4" -> returnBook();

case "5" -> searchBooks();

case "6" -> viewAllBooks();

case "7" -> viewOverdueBooks();

case "8" -> {

System.out.println("Thank you for using the Library Management System!");

return;

}

default -> System.out.println("Invalid choice. Please try again.");

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

}

private static void displayMenu() {

System.out.println("\n=== Library Management System ===");

System.out.println("1. Add Book");

System.out.println("2. Remove Book");

System.out.println("3. Borrow Book");

System.out.println("4. Return Book");

System.out.println("5. Search Books");

System.out.println("6. View All Books");

System.out.println("7. View Overdue Books");

System.out.println("8. Exit");

System.out.print("Enter your choice: ");

}

private static void addBook() throws BookAlreadyExistsException {

System.out.print("Enter ISBN: ");

String isbn = scanner.nextLine();

System.out.print("Enter Title: ");

String title = scanner.nextLine();

System.out.print("Enter Author: ");

String author = scanner.nextLine();

library.addBook(isbn, title, author);

System.out.println("Book added successfully!");

}

private static void removeBook() throws BookNotFoundException {

System.out.print("Enter ISBN of book to remove: ");

String isbn = scanner.nextLine();

library.removeBook(isbn);

System.out.println("Book removed successfully!");

}

private static void borrowBook() throws BookNotFoundException {

System.out.print("Enter ISBN of book to borrow: ");

String isbn = scanner.nextLine();

System.out.print("Enter borrower name: ");

String borrower = scanner.nextLine();

System.out.print("Enter loan period (days): ");

int days = Integer.parseInt(scanner.nextLine());

library.borrowBook(isbn, borrower, days);

System.out.println("Book borrowed successfully!");

}

private static void returnBook() throws BookNotFoundException {

System.out.print("Enter ISBN of book to return: ");

String isbn = scanner.nextLine();

library.returnBook(isbn);

System.out.println("Book returned successfully!");

}

private static void searchBooks() {

System.out.print("Enter search term: ");

String query = scanner.nextLine();

List<Book> results = library.searchBooks(query);

if (results.isEmpty()) {

System.out.println("No books found matching your search.");

return;

}

System.out.println("\nSearch Results:");

results.forEach(book -> System.out.println("\n" + book));

}

private static void viewAllBooks() {

List<Book> allBooks = library.getAllBooks();

if (allBooks.isEmpty()) {

System.out.println("No books in the library.");

return;

}

System.out.println("\nAll Books:");

allBooks.forEach(book -> System.out.println("\n" + book));

}

private static void viewOverdueBooks() {

List<Book> overdueBooks = library.getOverdueBooks();

if (overdueBooks.isEmpty()) {

System.out.println("No overdue books.");

return;

}

System.out.println("\nOverdue Books:");

overdueBooks.forEach(book -> System.out.println("\n" + book));

}

}

o/p

=== Library Management System ===

1. Add Book

2. Remove Book

3. Borrow Book

4. Return Book

5. Search Books

6. View All Books

7. View Overdue Books

8. Exit

Enter your choice: 1

Enter ISBN: 12934

Enter Title: Discovery of India

Enter Author: Jwaharlal Nehru

Book added successfully!

=== Library Management System ===

1. Add Book

2. Remove Book

3. Borrow Book

4. Return Book

5. Search Books

6. View All Books

7. View Overdue Books

8. Exit

Enter your choice: 6

All Books:

ISBN: 12934

Title: Discovery of India

Author: Jwaharlal Nehru

Status: Available

=== Library Management System ===

1. Add Book

2. Remove Book

3. Borrow Book

4. Return Book

5. Search Books

6. View All Books

7. View Overdue Books

8. Exit

Enter your choice: 6