A D Patel Institute of Technology
(A Constituent College of CVM University)

A.D. PATEL INSTITUTE OF TECHNOLOGY

(A Constituent College of CVM University)

New V. V. Nagar

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATASCIENCE

Mini Project Seminar

Submitted By

Name of Student: Tweency Marvaniya (12202120601051)

Programming with Java (202044502)

Semester 5 A.Y. 2024-

1. Introduction

- **Project Title**: Library Management System

- **Objective**:
  The objective of this project is to create a Library Management System that manages a collection of books. The system allows users (students and faculty) to borrow and return books, while the librarian can add books to the library's inventory. The project aims to automate the process of book management, reducing manual errors and improving the efficiency of the library operations.
- **Technology Stack**:
  The project is developed using Java, leveraging object-oriented programming principles. The following technologies are used:
  - Java Standard Edition (Java SE) for core programming.
  - Java's Collection framework for managing the library inventory.
  - File handling using java.io package for data persistence.
  - Java's concurrent utilities for thread-safe operations.
  -

2. Project Overview

- **System Architecture**:
  The system is designed with a modular architecture that follows object-oriented principles. The main components are:
  1. **Library**: Manages the inventory of books and user activities (borrow, return).
  2. **User**: Abstract class that defines the base properties of a library user.
  3. **Student** and **Faculty**: Specific user types inheriting from User.
  4. **Book**: Represents the book entity.
  5. **InventoryManager**: Manages the library's book inventory and handles inventory updates.
- **Features**:
  - User management: Students and faculty can borrow/return books.
  - Inventory management: Admins can add new books and check stock.
  - File handling for inventory persistence.
  - Exception handling for managing errors like book unavailability.
- **Modules**:
  1. **Library Module**: Manages all library operations, including adding books and managing the inventory.
  2. **User Module**: Handles different types of users (students and faculty) with different borrowing limits.
  3. **Book Module**: Stores book details like title, author, and ISBN.

4. **File Handling Module**: Manages saving and loading inventory data to/from a text file.

3. Implementation Details

- **Inheritance and Polymorphism**:
  The project uses inheritance to define a base class User from which Student and Faculty inherit. Polymorphism is demonstrated by the fact that both Student and Faculty have different borrow limits, and this behavior is determined dynamically at runtime through method overriding.
- **Inner Classes**:
  The InventoryManager class is implemented as an inner class within the Library class. This allows tight encapsulation, ensuring that only the Library has access to the inventory management functions.
- **Exception Handling**:
  The system includes robust exception handling. For example, if a user tries to borrow a book that is not available, an exception is thrown with a meaningful message. This prevents system crashes and guides users with proper feedback.
- **Threading**:
  The inventory operations (adding or removing books) are thread-safe, thanks to the use of ConcurrentHashMap in the InventoryManager class. This ensures that multiple users can borrow or return books simultaneously without data corruption.

- **Collection API**:
  The ConcurrentHashMap<Book, Integer> is used to manage the library's book inventory. It provides thread-safe operations and efficient access to the inventory, allowing easy addition and removal of books and checking their availability.

- **File Handling**:
  The system uses Java's file handling capabilities to persist data. It saves the current inventory to a text file (library_inventory.txt) and loads the inventory back into memory when the system starts. This ensures that data is not lost when the application is closed.

4. Project Demonstration

- **Live Demo**:
  During the live demo, the following actions will be performed:
  1. Adding new books to the library.
  2. A student borrowing a book.
  3. A faculty member borrowing multiple books.

4. Returning a book and checking the updated inventory.
5. Saving the inventory to a text file and reloading it.

- **Key Functionalities**:
    - Borrowing and returning books.
    - Inventory management, with real-time availability checks.
    - Persisting inventory data to a file for later use.
- **User Interface**:

  This is a command-line based system where users interact with the application by entering commands and book details. The user interface is simple and intuitive, guiding the user step-by-step through borrowing or returning books.

5. Challenges and Solutions

- **Development Challenges**:
    - Managing concurrent access to the inventory, especially when multiple users might try to borrow the same book at the same time.
    - Handling exceptions for various scenarios like trying to borrow a book that's out of stock.
    - File handling was tricky when parsing inventory data, especially ensuring data integrity while reading and writing files.

- **Solutions Implemented**:
    - The use of ConcurrentHashMap ensured thread-safe operations for managing the inventory.
    - Exception handling was implemented to provide meaningful feedback to users without crashing the system.
    - Robust file parsing mechanisms were added to handle different edge cases (like missing or incorrect data in the file).

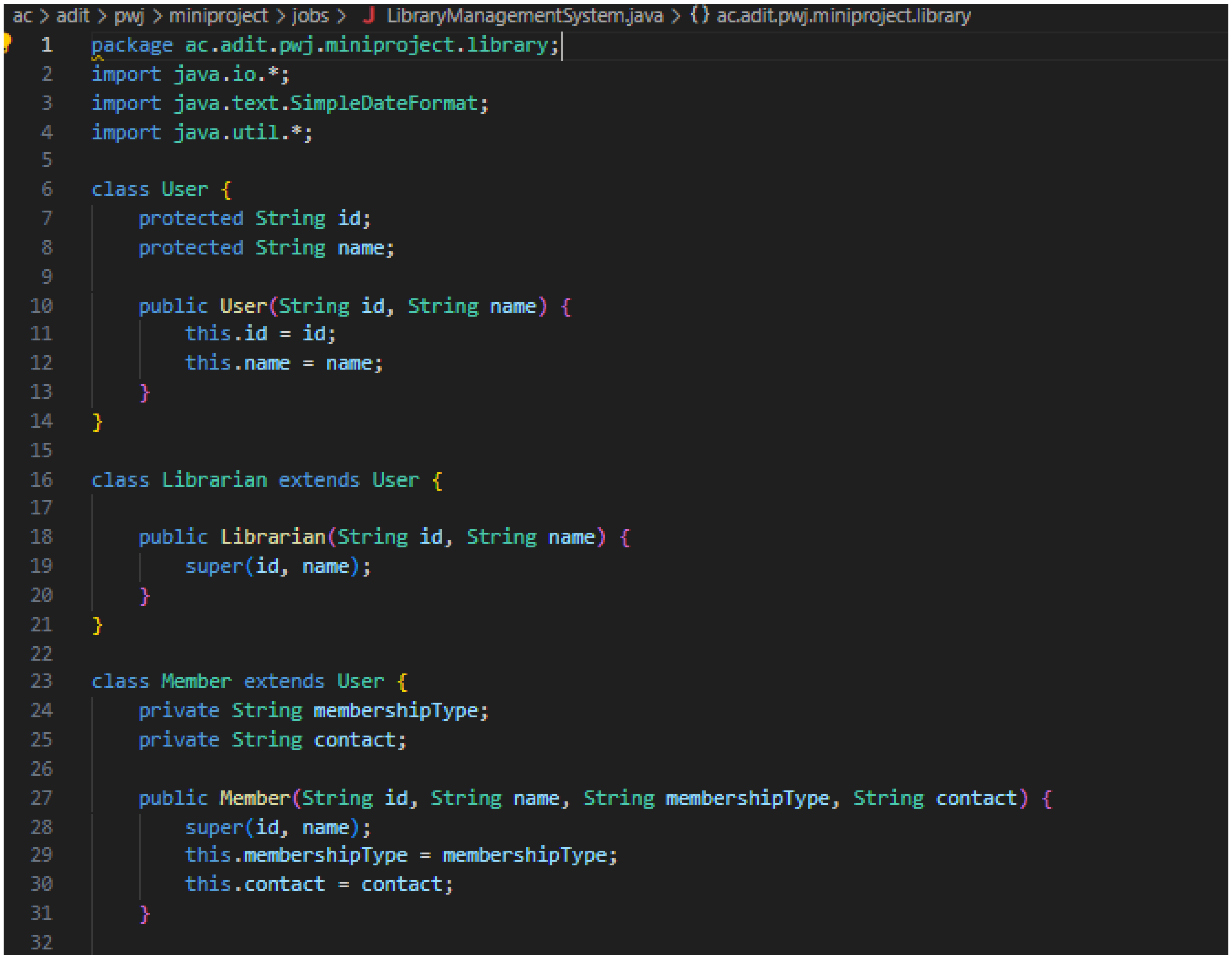
6. Conclusion and Future Work

- **Conclusion**:
  The Library Management System successfully automates the management of books, users, and the borrowing process. It provides a streamlined experience for users and reduces the manual work of managing a physical inventory. The project demonstrates effective use of Java's object-oriented programming features, including inheritance, polymorphism, and file handling.

- **Future Enhancements**:

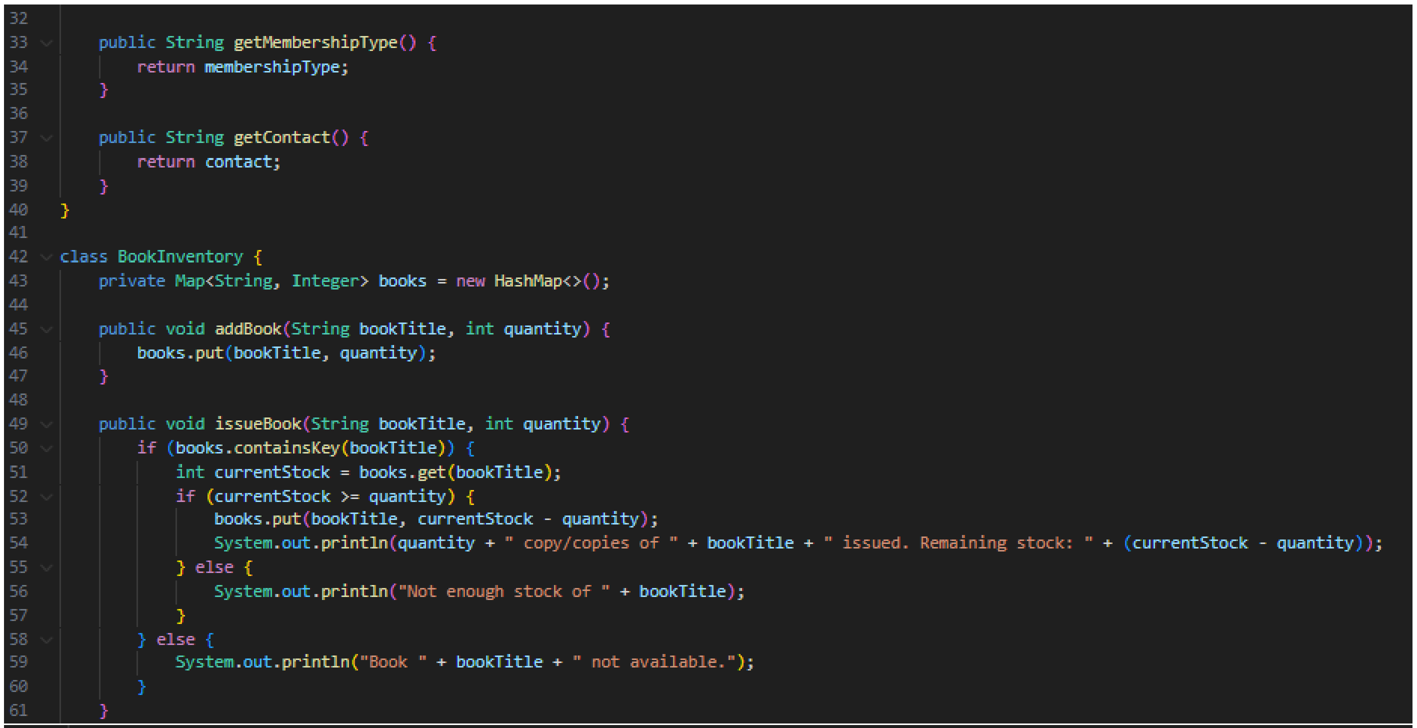
- Adding a graphical user interface (GUI) using JavaFX or Swing to improve user interaction.
- Extending the system to support digital media (e.g., e-books, audiobooks).
- Implementing user authentication to differentiate between students, faculty, and admins.
- Adding more advanced search and filtering options for the inventory.

DEMO code:

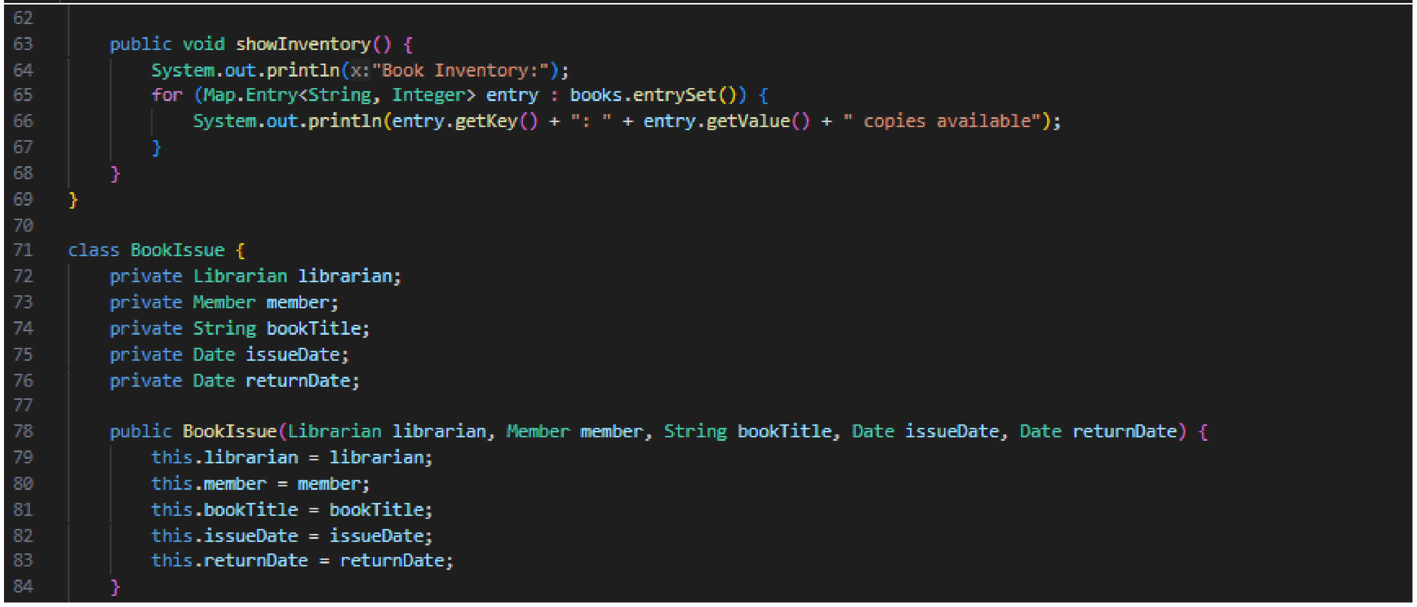


ac > adit > pwj > miniproject > jobs > LibraryManagementSystem.java > {} ac.adit.pwj.miniproject.library

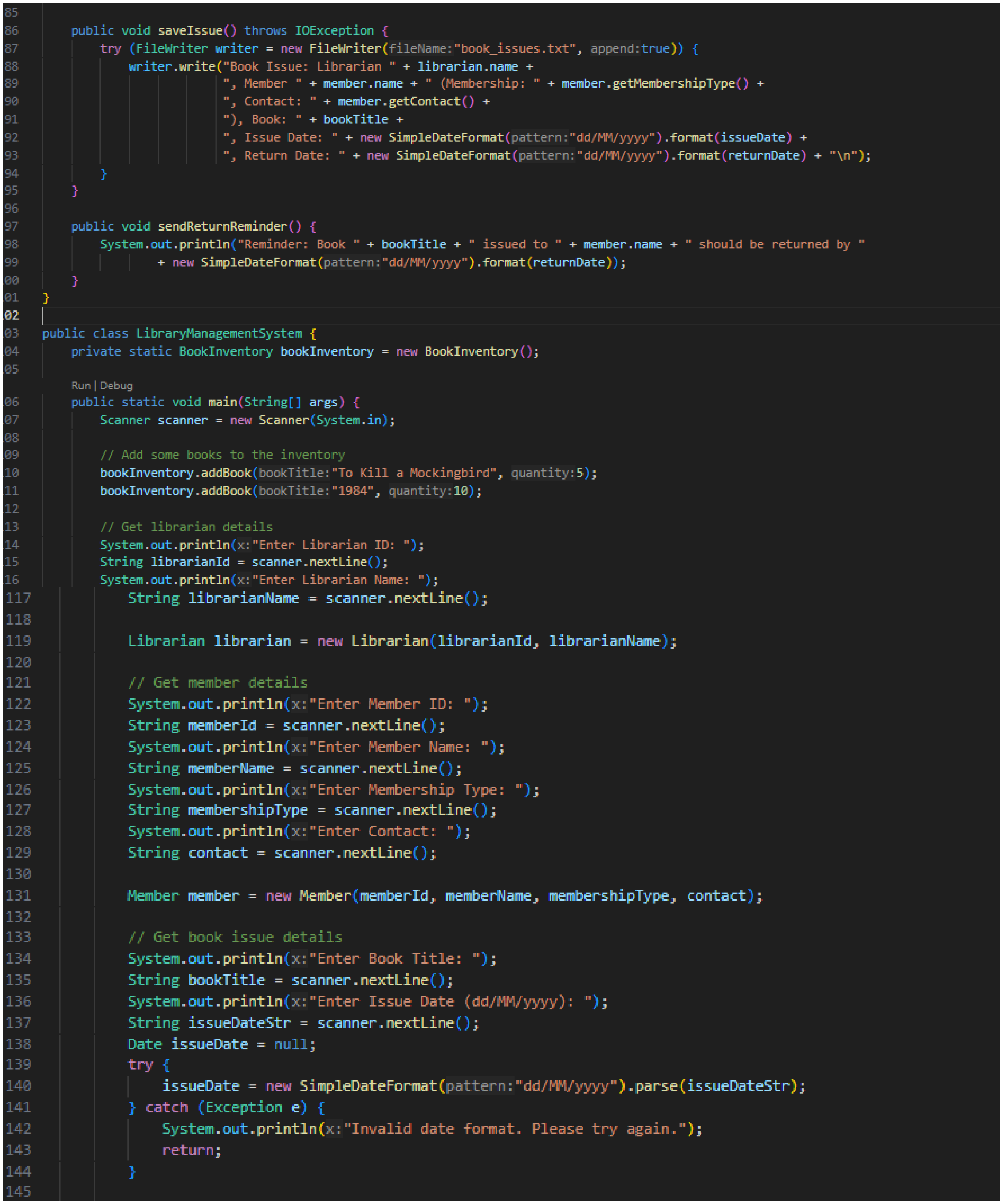
```java
package ac.adit.pwj.miniproject.library;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.*;

class User {
    protected String id;
    protected String name;

    public User(String id, String name) {
        this.id = id;
        this.name = name;
    }
}

class Librarian extends User {

    public Librarian(String id, String name) {
        super(id, name);
    }
}

class Member extends User {
    private String membershipType;
    private String contact;

    public Member(String id, String name, String membershipType, String contact) {
        super(id, name);
        this.membershipType = membershipType;
        this.contact = contact;
    }

```



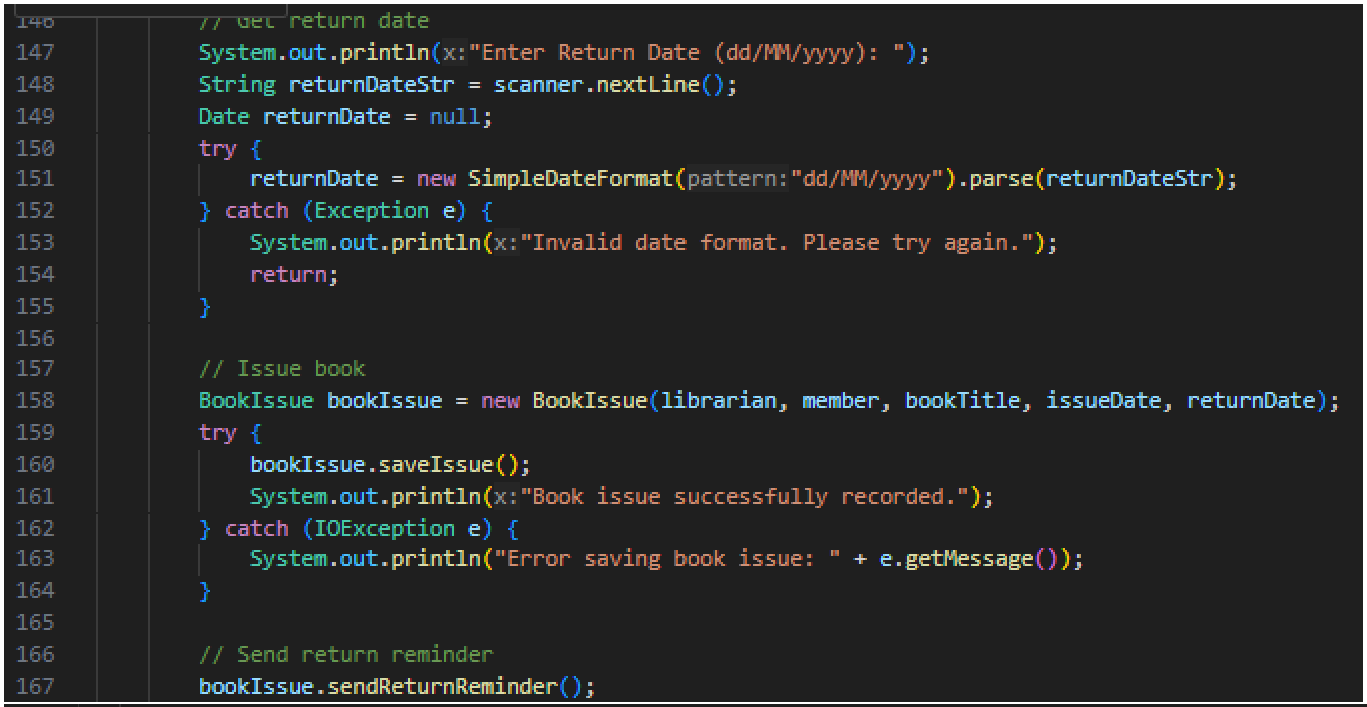
```java

    public String getMembershipType() {
        return membershipType;
    }

    public String getContact() {
        return contact;
    }
}

class BookInventory {
    private Map<String, Integer> books = new HashMap<>();

    public void addBook(String bookTitle, int quantity) {
        books.put(bookTitle, quantity);
    }

    public void issueBook(String bookTitle, int quantity) {
        if (books.containsKey(bookTitle)) {
            int currentStock = books.get(bookTitle);
            if (currentStock >= quantity) {
                books.put(bookTitle, currentStock - quantity);
                System.out.println(quantity + " copy/copies of " + bookTitle + " issued. Remaining stock: " + (currentStock - quantity));
            } else {
                System.out.println("Not enough stock of " + bookTitle);
            }
        } else {
            System.out.println("Book " + bookTitle + " not available.");
        }
    }
```



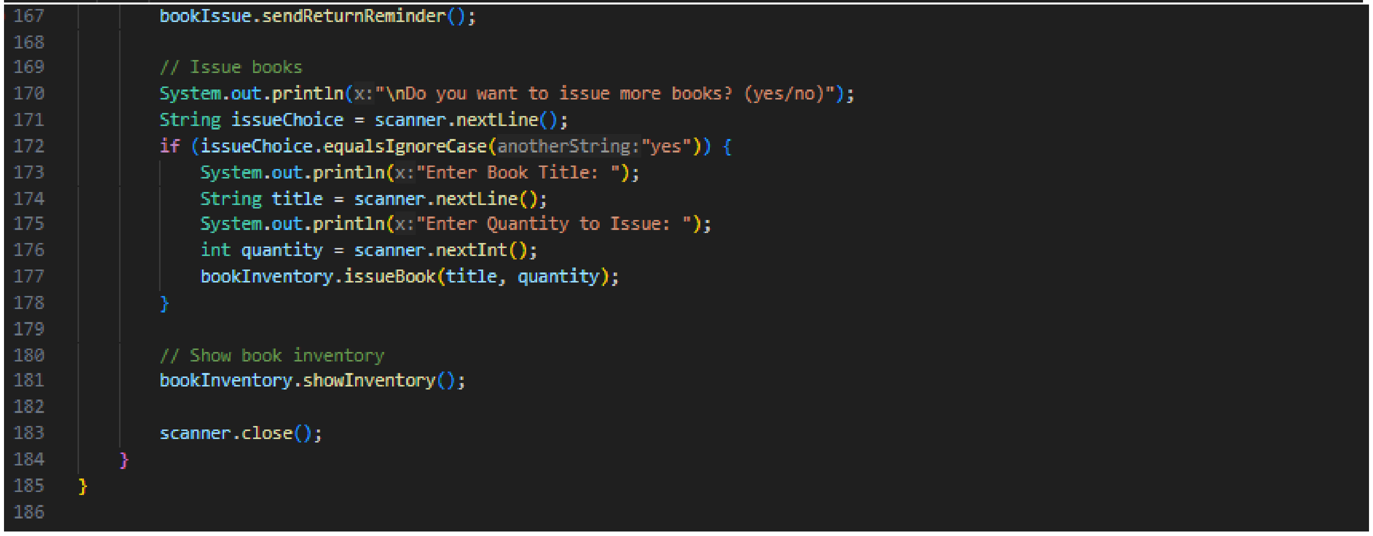
```java

    public void showInventory() {
        System.out.println(x:"Book Inventory:");
        for (Map.Entry<String, Integer> entry : books.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue() + " copies available");
        }
    }
}

class BookIssue {
    private Librarian librarian;
    private Member member;
    private String bookTitle;
    private Date issueDate;
    private Date returnDate;

    public BookIssue(Librarian librarian, Member member, String bookTitle, Date issueDate, Date returnDate) {
        this.librarian = librarian;
        this.member = member;
        this.bookTitle = bookTitle;
        this.issueDate = issueDate;
        this.returnDate = returnDate;
    }
```



```java

    public void saveIssue() throws IOException {
        try (FileWriter writer = new FileWriter(fileName:"book_issues.txt", append:true)) {
            writer.write("Book Issue: Librarian " + librarian.name +
                         ", Member " + member.name + " (Membership: " + member.getMembershipType() +
                         ", Contact: " + member.getContact() +
                         "), Book: " + bookTitle +
                         ", Issue Date: " + new SimpleDateFormat(pattern:"dd/MM/yyyy").format(issueDate) +
                         ", Return Date: " + new SimpleDateFormat(pattern:"dd/MM/yyyy").format(returnDate) + "\n");
        }
    }

    public void sendReturnReminder() {
        System.out.println("Reminder: Book " + bookTitle + " issued to " + member.name + " should be returned by "
                + new SimpleDateFormat(pattern:"dd/MM/yyyy").format(returnDate));
    }
}

public class LibraryManagementSystem {
    private static BookInventory bookInventory = new BookInventory();

    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Add some books to the inventory
        bookInventory.addBook(bookTitle:"To Kill a Mockingbird", quantity:5);
        bookInventory.addBook(bookTitle:"1984", quantity:10);

        // Get librarian details
        System.out.println(x:"Enter Librarian ID: ");
        String librarianId = scanner.nextLine();
        System.out.println(x:"Enter Librarian Name: ");
        String librarianName = scanner.nextLine();

        Librarian librarian = new Librarian(librarianId, librarianName);

        // Get member details
        System.out.println(x:"Enter Member ID: ");
        String memberId = scanner.nextLine();
        System.out.println(x:"Enter Member Name: ");
        String memberName = scanner.nextLine();
        System.out.println(x:"Enter Membership Type: ");
        String membershipType = scanner.nextLine();
        System.out.println(x:"Enter Contact: ");
        String contact = scanner.nextLine();

        Member member = new Member(memberId, memberName, membershipType, contact);

        // Get book issue details
        System.out.println(x:"Enter Book Title: ");
        String bookTitle = scanner.nextLine();
        System.out.println(x:"Enter Issue Date (dd/MM/yyyy): ");
        String issueDateStr = scanner.nextLine();
        Date issueDate = null;
        try {
            issueDate = new SimpleDateFormat(pattern:"dd/MM/yyyy").parse(issueDateStr);
        } catch (Exception e) {
            System.out.println(x:"Invalid date format. Please try again.");
            return;
        }

```

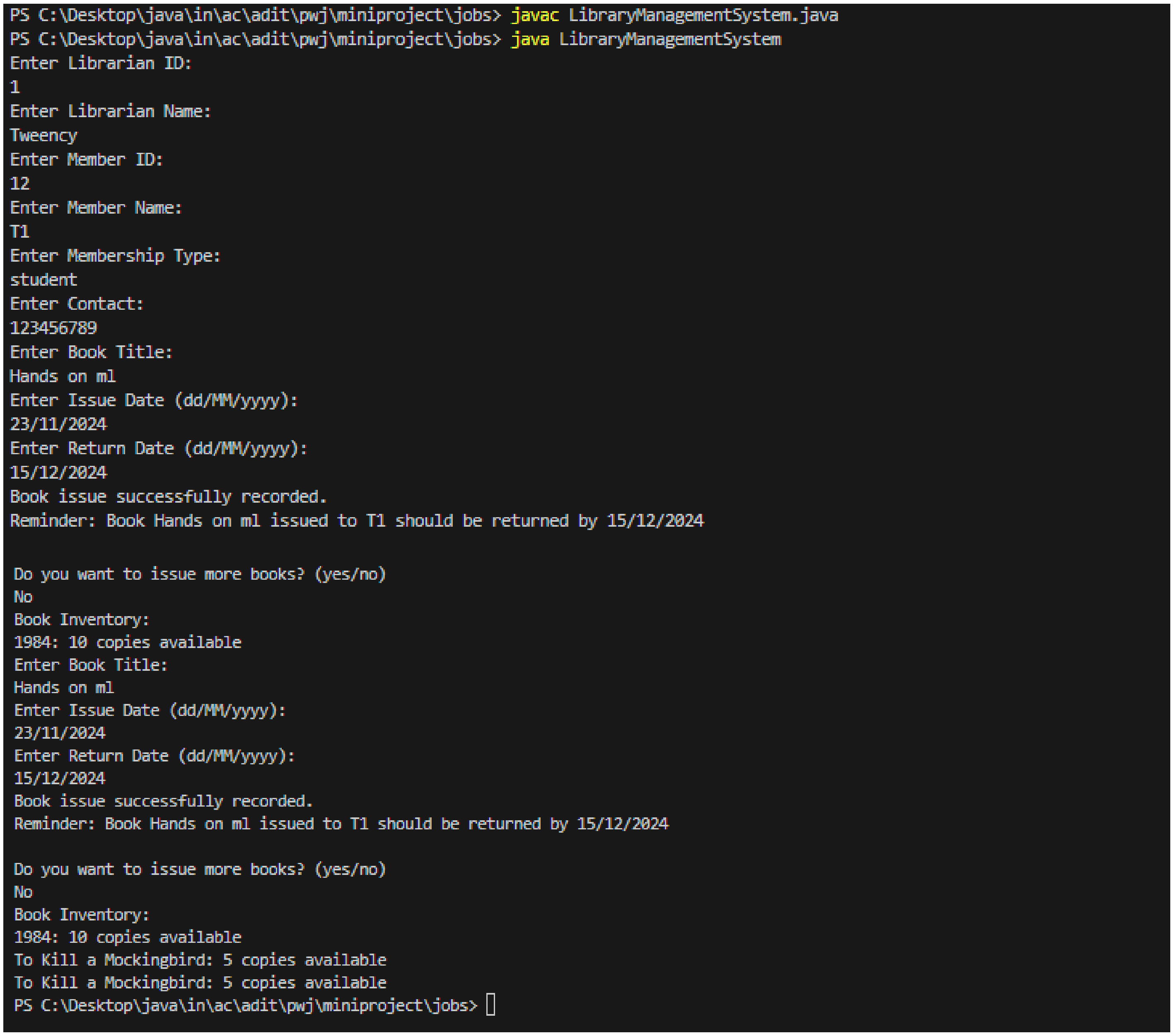


```java
        // Get return date
        System.out.println(x:"Enter Return Date (dd/MM/yyyy): ");
        String returnDateStr = scanner.nextLine();
        Date returnDate = null;
        try {
            returnDate = new SimpleDateFormat(pattern:"dd/MM/yyyy").parse(returnDateStr);
        } catch (Exception e) {
            System.out.println(x:"Invalid date format. Please try again.");
            return;
        }

        // Issue book
        BookIssue bookIssue = new BookIssue(librarian, member, bookTitle, issueDate, returnDate);
        try {
            bookIssue.saveIssue();
            System.out.println(x:"Book issue successfully recorded.");
        } catch (IOException e) {
            System.out.println("Error saving book issue: " + e.getMessage());
        }

        // Send return reminder
        bookIssue.sendReturnReminder();
```



```java
        bookIssue.sendReturnReminder();

        // Issue books
        System.out.println(x:"\nDo you want to issue more books? (yes/no)");
        String issueChoice = scanner.nextLine();
        if (issueChoice.equalsIgnoreCase(anotherString:"yes")) {
            System.out.println(x:"Enter Book Title: ");
            String title = scanner.nextLine();
            System.out.println(x:"Enter Quantity to Issue: ");
            int quantity = scanner.nextInt();
            bookInventory.issueBook(title, quantity);
        }

        // Show book inventory
        bookInventory.showInventory();

        scanner.close();
    }
}
```

Output :



```
PS C:\Desktop\java\in\ac\adit\pwj\miniproject\jobs> javac LibraryManagementSystem.java
PS C:\Desktop\java\in\ac\adit\pwj\miniproject\jobs> java LibraryManagementSystem
Enter Librarian ID:
1
Enter Librarian Name:
Tweency
Enter Member ID:
12
Enter Member Name:
T1
Enter Membership Type:
student
Enter Contact:
123456789
Enter Book Title:
Hands on ml
Enter Issue Date (dd/MM/yyyy):
23/11/2024
Enter Return Date (dd/MM/yyyy):
15/12/2024
Book issue successfully recorded.
Reminder: Book Hands on ml issued to T1 should be returned by 15/12/2024

 Do you want to issue more books? (yes/no)
 No
 Book Inventory:
 1984: 10 copies available
 Enter Book Title:
 Hands on ml
 Enter Issue Date (dd/MM/yyyy):
 23/11/2024
 Enter Return Date (dd/MM/yyyy):
 15/12/2024
 Book issue successfully recorded.
 Reminder: Book Hands on ml issued to T1 should be returned by 15/12/2024

 Do you want to issue more books? (yes/no)
 No
 Book Inventory:
 1984: 10 copies available
 To Kill a Mockingbird: 5 copies available
 To Kill a Mockingbird: 5 copies available
 PS C:\Desktop\java\in\ac\adit\pwj\miniproject\jobs> 
```