K.R. Mangalam University

School of Engineering & Technology

Fundamentals Of Java Programming Lab

(ENCA203) Assignment 4

City Library Management System

Submitted by:

Name: Khushi

Roll No: 2401201102

Course: BCA (AI & DS)

Section: B

Code:

```java
import java.io.*;
import java.util.*;

/**
 * City Library Digital Management System
 * Author: Khushi
 */

// --- Book Class ---
class Book implements Comparable<Book> {
    int bookId;
    String title, author, category;
    boolean isIssued;

    Book(int id, String t, String a, String c) {
        bookId = id;
        title = t;
        author = a;
        category = c;
        isIssued = false; // By default, a new book is not issued [cite: 4]
    }

    public void display() {
        // Output format follows the original code's style
        System.out.println(bookId + " " + title + " " + author + " " + category + " Issued: " + isIssued);
    }

    // Compares books based on title for sorting (case-insensitive)
    @Override
    public int compareTo(Book b) {
        return title.compareToIgnoreCase(b.title);
    }
```

```java
}

// --- Member Class ---
class Member {
    int memberId;
    String name, email;
    // Stores the IDs of books currently issued by this member
    List<Integer> issuedBooks = new ArrayList<>();

    Member(int id, String n, String e) {
        memberId = id;
        name = n;
        email = e;
    }

    public void display() {
        System.out.println(memberId + " " + name + " " + email + " Books: " + issuedBooks);
    }
}

// --- Main Class (Library Management Logic) ---
public class Main {
    // Stores books keyed by bookId
    static Map<Integer, Book> books = new HashMap<>();
    // Stores members keyed by memberId
    static Map<Integer, Member> members = new HashMap<>();
    static Scanner sc = new Scanner(System.in);
    // Initial IDs for new entries
    static int bookId = 101, memberId = 201;

```

```
63      public static void main(String[] args) {
64          System.out.println(x: "Welcome to City Library System");
65          while (true) {
66              System.out.println(x: "\n1.Add Book 2.Add Member 3.Issue 4. Return 5.Search 6
67              int ch = sc.nextInt();
68              sc.nextLine(); // Consume newline
69
70              switch (ch) {
71                  case 1 -> addBook();    //
72                  case 2 -> addMember();  //
73                  case 3 -> issueBook();  //
74                  case 4 -> returnBook(); //
75                  case 5 -> searchBook(); //
76                  case 6 -> sortBooks();  //
77                  case 7 -> {
78                      saveData();
79                      System.out.println(x: "Exiting program...");
80                      System.exit(status: 0); //
81                  }
82                  default -> System.out.println(x: "Invalid!"); //
83              }
84          }
85      }
86
87      static void addBook() {
88          System.out.print(s: "Title: ");
89          String t = sc.nextLine(); //
90          System.out.print(s: "Author: ");
91          String a = sc.nextLine(); //
92          System.out.print(s: "Category: ");
```

```
93          String c = sc.nextLine(); //
94
95          Book b = new Book(bookId++, t, a, c); // New ID is assigned and then incremented
96          books.put(b.bookId, b); //
97          saveData();
98          System.out.println("Book added with ID " + b.bookId); //
99      }
100
101     static void addMember() {
102         System.out.print(s: "Name: ");
103         String n = sc.nextLine(); //
104         System.out.print(s: "Email: ");
105         String e = sc.nextLine(); //
106
107         Member m = new Member(memberId++, n, e); //
108         members.put(m.memberId, m); //
109         saveData();
110         System.out.println("Member added with ID" + m.memberId); //
111     }
112
113     static void issueBook() {
114         System.out.print(s: "Book ID: ");
115         int bId = sc.nextInt(); //
116         System.out.print(s: "Member ID: ");
117         int mId = sc.nextInt(); //
118
119         Book b = books.get(bId);
120
121         // Check for book existence and if it's already issued
122         if (b == null || !members.containsKey(mId)) {
```

```
122         if (b == null || !members.containsKey(mId)) {
123             System.out.println(x: "Invalid ID");
124             return;
125         }
126 
127         if (b.isIssued) {
128             System.out.println(x: "Already issued");
129             return; //
130         }
131 
132         b.isIssued = true; //
133         members.get(mId).issuedBooks.add(bId); //
134 
135         saveData();
136         System.out.println(x: "Book issued successfully"); //
137     }
138 
139     static void returnBook() {
140         System.out.print(s: "Book ID: ");
141         int bId = sc.nextInt(); //
142         System.out.print(s: "Member ID: ");
143         int mId = sc.nextInt(); //
144 
145         // Check if both IDs are valid
146         if (books.containsKey(bId) && members.containsKey(mId)) {
147             books.get(bId).isIssued = false; //
148             // Remove the book ID from the member's list
149             members.get(mId).issuedBooks.remove((Integer) bId);
150             saveData();
```

```
            System.out.println(x: "Returned successfully"); //
        } else {
            System.out.println(x: "Invalid IDs"); //
        }
    }

    static void searchBook() {
        System.out.print(s: "Enter keyword: ");
        String k = sc.nextLine().toLowerCase(); // Keyword to lowercase

        // Iterate through all books
        for (Book b : books.values()) { //
            // Search in title OR author (case-insensitive)
            if (b.title.toLowerCase().contains(k) || b.author.toLowerCase().contains(k)) {
                b.display(); //
            }
        }
    }

    static void sortBooks() {
        // Create a list from the map's values
        List<Book> list = new ArrayList<>(books.values());
        Collections.sort(list); // Sorts based on the Book class's compareTo (by title)

        System.out.println(x: "\nSorted Books:"); //
        for (Book b : list) {
            b.display(); //
        }
    }
```



```
    }

    // --- Data Persistence ---
    static void saveData() {
        // Save Books to books.txt
        try (PrintWriter bw = new PrintWriter(fileName: "books.txt")) { //
            for (Book b : books.values()) {
                // Format: ID Title Author Category Issued
                bw.println(b.bookId + " " + b.title + " " + b.author + " " + b.category + " " + b.isIssued);
            }
        } catch (Exception ignored) {
            //
        }

        // Save Members to members.txt
        try (PrintWriter bw = new PrintWriter(fileName: "members.txt")) { //
            for (Member m : members.values()) {
                // Format: ID Name Email [Issued Book List]
                bw.println(m.memberId + " " + m.name + " " + m.email + " " + m.issuedBooks);
            }
        } catch (Exception ignored) {
            //
        }
    }
}
```

Output:

```
Welcome to City Library System

1.Add Book 2.Add Member 3.Issue 4. Return 5.Search 6.Sort 7.Exit
1
Title: never ends with us
Author: Collen Hoover
Category: love
Book added with ID 101

1.Add Book 2.Add Member 3.Issue 4. Return 5.Search 6.Sort 7.Exit
2
Name: khushi
Email: khushi@gmail.com
Member added with ID201

1.Add Book 2.Add Member 3.Issue 4. Return 5.Search 6.Sort 7.Exit
```