Assignment-4

Name - Smruthi Jha, Roll no. - 2401730081

Code →

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

class Book implements Comparable<book>{
        int bookId;
        String title;
        String author;
        String category;
        boolean isIssued;

    Book(int BookId, String title, String author, String category){
        this.bookId = bookId;
        this.title = title;
        this.author = author;
        this.isIssued = false;
    }
    void displayBookDetails(){
        System.out.println("ID: "+bookId+", Title: "+title+", Author: "
                +author+", Category: "+category+", Issued
                +isIssued);
    }
    void markAsIssued() {isIssued = true;}
    void markAsReturned() {isIssued = false;}
    @Override
    public int compareTo(Book b){
        return this.title.compareToIgnoreCase(b.Title);
}}
```

```
class AutherSort implements Comparator<Book> {
    public int compare(Book a, Book b) {
        return a.author.compareToIgnoreCase(b.author);
    }
}
class Member {
    int memberId;
    String name;
    String email;
    List<Integer> issuedBooks = new ArrayList<>();

    Member(int memberId, String name, String email) {
        this.memberId = memberId;
        this.name = name;
        this.email = email;
    }
    void displayMemberDetails() {
        System.out.println("ID: " + memberId + ", Name: " + name +
            ", Email: " + email + ", Books: " + issuedBooks);
    }
    void addIssuedBook(int id) { issuedBooks.add(id); }
    void returnIssuedBook(int id) { issuedBooks.remove(Integer.valueOf(id)); }
}
public class LibrarySystem {
    Map<Integer, Book> books = new HashMap<>();
    Map<Integer, Member> members = new HashMap<>();

    String BOOK_FILE = "books.txt";
    String MEMBERS_FILE = "members.txt";
```

```
public static void main(String[] args) {
    LibrarySystem lib=new LibrarySystem();
    lib.loadData();
    lib.menu();
    lib.saveData();
}
void menu(){
    Scanner sc = new Scanner(System.in);

    while(true){
    System.out.println("\n===City Library Digital System===");
    System.out.println("1.Add Book");
    System.out.println("2.Add member");
    System.out.println("3.Issue Book");
    System.out.println("4.Return Book");
    System.out.println("5.Search Book");
    System.out.println("6.Sort Books");
    System.out.println("7.Exit");
    System.out.println("Enter choice:");
    int ch =sc.nextInt();
    sc.nextLine();

    switch(ch){
        case 1 ->addBook(sc);
        case 2 ->addMember(sc);
        case 3 ->issueBook(sc);
        case 4 ->returnBook(sc);
        case 5 ->searchBook(sc);
        case 6 ->sortBooks();
        case 7 -> {
            System.out.println("Saving & Exiting");
            saveData();
```

```
            return;
        }
        default -> System.out.println("Invalid Choice");
    }
}
}
void addBook(Scanner sc){
    System.out.print("Enter ID: ");
    int id = sc.nextInt(); sc.nextLine();
    System.out.println("Enter Title");
    String t = sc.nextLine();
    System.out.println("Enter Author:");
    String a = sc.nextLine();
    System.out.println("Enter category:");
    String c = sc.nextLine();

    books.put(id, new Book(id, t, a, c));
    System.out.println("Book added!");
    saveData();
}
void addMember(Scanner sc){
    System.out.println("Enter Member ID:");
    int id = sc.nextInt(); sc.nextLine();
    System.out.println("Enter name:");
    String n = sc.nextLine();
    System.out.println("Enter email:");
    String e = sc.nextLine();

    members.put(id, new Member(id, n, e));
    System.out.println("Member added!");
    saveData();
}
```

```
void issueBook(Scanner sc) {
    System.out.println("Enter Book Id: ");
    int bid = sc.nextInt();
    System.out.print("Enter Member Id: ");
    int mid = sc.nextInt();

    if(!books.containsKey(bid)){
        System.out.println("Book not found!");
        return;
    }
    if(!members.containsKey(mid)){
        System.out.println("Member not found!");
        return;
    }
    Book b = books.get(bid);
    if(b.isIssued){
        System.out.println("Book Already Issued!");
        return;
    }
    b.markAsIssued();
    members.get(mid).addIssuedBook(bid);

    System.out.println("Book issued!");
    saveData();
}
void returnBook(Scanner sc){
    System.out.print("Enter Book ID: ");
    int bid=sc.nextInt();
    System.out.print("Enter member ID: ");
    int mid=sc.nextInt();
}
```

```
if (!books.containsKey(bid) || !members.containsKey(mid)) {
        System.out.println("Invalid IDs!");
        return;
}
books.get(bid).markAsReturned();
members.get(mid).returnIssuedBook(bid);

System.out.println("Book returned!");
saveData();
}
void searchBook(Scanner sc) {
    System.out.println("Search by Title: ");
    String key = sc.nextLine().toLowerCase();

    for (Book b : books.values()) {
        if (b.title.toLowerCase().contains(key)) {
            b.displayBookDetails();
        }
    }
}
void sortBooks() {
    List<Book> list = new ArrayList<>(books.values());

    System.out.println("\n Sort By: 1) Title 2) Author ");
    Scanner sc = new Scanner(System.in);
    int ch = sc.nextInt();

    if (ch == 1) Collections.sort(list);
    else Collections.sort(list, new AuthorSort());

    System.out.println("\n Sorted Books: ");
    list.forEach(Book::displayBookDetails);
}
```

```
void saveData() {
    try{
      BufferedWriter bw = new BufferedWriter(new FileWriter(
                                                BOOK_FILE));
      for (Book b : books.values()) {
          bw.write(b.bookId + "," + b.title + "," + b.author + "," +
                   b.category + "," + bw.newLine();
      }
      bw.close();

      bw = new BufferedWriter(new FileWriter(MEMBER_FILE));
      for (Member m : members.values()) {
          bw.write(m.memberId + "," + m.name + "," + m.email +
                   "," + m.issuedBooks);
          bw.newLine();
      }
      bw.close();

    }catch(Exception e){
        System.out.println("Error Saving!");
    }
}

void loadData() {
    try{
      BufferedReader br = new BufferedReader(new FileReader(BOOK_FILE));
      String line;
      while((line = br.readLine()) != null){
          String[] a = line.split(",");
          Book b = new Book(Integer.parseInt(a[0]), a[1], a[2], a[3]);
          b.isIssued = Boolean.parseBoolean(a[4]);
          books.put(b.bookId, b);
      }
```

```
        br.close();
    }catch (Exception ignored) {}

    try{
        BufferedReader br = new BufferedReader(new FileReader(MEMBER_FILE));
        String line;
        while ((line = br.readLine()) != null) {
            String[] a = line.split(",");
            Member m = new Member(Integer.parseInt(a[0]), a[1], a[2]);
            members.put(m.memberId, m);
        }
        br.close();
    }catch (Exception ignored) {}
}
```