Page No.
Date : / /

Assignment - 4

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

public class cityLibrary {
    static class Book implements Comparable
    <Book> {
        int bookId;
        String title, author, category;
        boolean isIssued;

        Book (int id, String t, String a, String
        c, boolean issued) {
            bookId = id; title = t; author
            a; category = c; isIssued =
            issued;
        }
        @Override
        public int compareTo (Book b) {
            return this.title.compare-
            ToIgnoreCase (b.title);
        }
        void display () {
            System.out.println (bookId + " | "
            + title + " | " + author + " | "
            + category + " | " + (isIssued
            ? "Issued" : "Available"));
        }
```

```
String toLine() {
    return bookId + "," +
    title + "," + author + "," +
    category + "," + isIssued;
}
static Book FromLine (String s)
{
    String [ ] p = s.split(",");
    return new Book(Integer.
    parseInt(p[0]), p[1], p[2],
    p[3], Boolean.parseBoolean
    (p[4]));
}
}

static class Member {
    int memberId;
    String name, email;
    List<Integer> issuedBooks = new
    ArrayList<>();

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

```
void display(){
    System.out.println(memberId +
    " | " + name + " | " + email +
    " | Books: " + issuedBooks);
}
String toLine(){
    return memberId + ";" + name +
    ";" + email + ";" + issuedBooks.
    toString();
}
static Member fromLine(String s){
    String[] p = s.split(";");
    Member m = new Member
    (Integer.parseInt(p[0]), p[1]
    p[2]);
    return m;
}
}

static class LibraryManager {
    Map<Integer, Book> books = new
    HashMap<>();
    Map<Integer, Member> members =
    new HashMap<>();

    File bookFile = new File("books.txt");
    File memberFile = new File("member.txt");
```

```
int nextBookId = 100;
int nextMemberId = 200;

LibraryManager() { load(); }

void load() {
    try (BufferedReader br = new
    BufferedReader(new FileReader
    (bookFile))) {
        String line;
        while ((line = br.readLine())
        != null) {
            Book b = Book.fromLine
            (line);
            books.put(b.bookId,
            b);
            nextBookId = Math.max(
            nextBookId, b.bookId
            + 1);
        }
    } catch (Exception ignored) {}

    try(BufferedReader br = new
    BufferedReader(new FileReader
    (memberFile))) {
        String line;
        while((line = br.readLine
        ()) != null) {
```

```
            Member m = member.fromLine
            (line);
            members.put(m.memberId, m);
            nextMemberId = Math.max(
            nextMemberId, m.memberId + 1);
        }
    } catch (Exception ignored) {}
}
void save() {
    try (PrintWriter pw = new PrintWriter
    (bookFile)) {
        for (Book b : books.values())
        pw.println(b.toLine());
    } catch (Exception ignored);

    try (PrintWriter pw = new PrintWriter
    (memberFile)) {
        for (Member m : members.values())
        pw.println(m.toLine());
    } catch (Exception ignored) {}
}
void addMember (String n, String e) {
    Member m = new Member (nextMember-
    Id++, n, e);
    members.put(m.memberId, m);
    save();
    System.out.println("Member added: ID "
    m.memberId);
}
```

```
void addBook(String t, String a,
String c) {
    Book b = new Book(nextBookId++,
    t, a, c, false);
    books.put(b.bookId, b);
    save();
    System.out.println("Book
    added: ID " + b.bookId);
}
void issueBook(int bId, int mId)
{
    Book b = books.get(bId);
    member m = members.get(mId);

    if (b == null || m == null) {
        System.out.println("Invalid
        ID");
        return;
    }
    if (b.isIssued) {
        System.out.println("
        Already issued");
        return;
    }
    b.isIssued = true;
    m.issuedBooks.add(bId);
    save();
    System.out.println("Book issued.");
}
```

```
void returnBook (int bId, int mId){
    Book b = books.get(bId);
    member m = members.get(mId);

    if (b == null || m == null) { System.
     out.println("Invalid ID"); return;
    }
    b.isIssued = false;
    m.issuedBooks.remove(Integer.valueOf
    (bId));
    save();
    System.out.println("Book returned
    ");
}
void search(String key) {
    key = key.toLowerCase();
    for (Book b : book.values()) {
        if(b.title.toLowerCase().contains(
        key) || b.author.toLowerCase().
        contains(key) || b.category.toLower
        Case().contains(key)) {
            b.display();
        }
    }
}
```

```
void sortByTitle(){
    List<Book> list = new ArrayList<>(books.values());
    Collections.sort(list);
    list.forEach(Book::display);
  }
}

public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);
    LibraryManager lm = new LibraryManager();

    while(true){
        System.out.println("\n1.Add Book 2.AddMember 3.Issue 4. Return 5.Search 6.Sort 7. Exit");
        System.out.print("Choice: ");

        int ch = Integer.parseInt(sc.nextLine());

        switch(ch){
            case 1:
                System.out.println("Title: "); String t = sc.nextLine();
```

```
System.out.print("Author: ");
String a = sc.nextLine();
System.out.print("Category: ");
String c = sc.nextLine();
lm.addBook(t, a, c);
break;
case 2:
System.out.print("Name: ");
String n = sc.nextLine();
System.out.print("Email: ");
String e = sc.nextLine();
lm.addMember(n, e);
break;

case 3:
System.out.print("Book ID: ");
int b1 = Integer.parseInt(sc.nextLine());
System.out.print("Member ID: ");
int m1 = Integer.parseInt(sc.nextLine());
lm.issueBook(b1, m1);
break;

case 4:
System.out.print("Book ID: ");
int b2 = Integer.parseInt(sc.nextLine());
```

```
            System.out.print("Book
            ID: "); int b2 = Integer.
            parseInt(sc.nextLine());
            lm.returnBook(b2, m2);
            break;
        case 5:
            System.out.print("
            Search keyword: ");
            lm.search(sc.nextLine());
            break;

        case 6:
            lm.sortByTitle();
            break;

        case 7:
            System.out.println("Exiting");
            return;

    }
}
```