21/11/25

Assignment - 4

City Library Digital Management System

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

public class CityLibrary {
  static class Book implements Comparable<Book> {
  int id;
  String title, author, category;
  boolean isIssued;
  Book(int id, String t, String a, String c, boolean i) {
    this.id = id ; title = t; author = a; category = c; isIssued
                                                    = i; }

  void markAsIssued() { isIssued = true; }
  void markAsReturned() { isIssued = false; }
  String toFile() { return id + "|" + title + "|" + author
                  + "|" + category + "|" + isIssued; }
  static Book fromFile(String line) {
    if (line == null || line.trim().isEmpty())
    return null;
    String[] p = line.split("\\|", -1);
    if (p.length < 5) return null;
    try { return new Book(Integer.parseInt(
    ⊗ p[0]), p[1], p[2], p[3], Boolean.parse
                                Boolean(p[4]));
  }
```

```
catch (Exception e) { return null; }
}

public int compareTo(Book o) { return title.compareToIgnoreCase(o.title); }
}
void display() {
    System.out.println("ID:"+id+"|"+title+"|"+author
        +"|"+category+"|Issued:"+isIssued);
}
}

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

    Member(int id, String n, String e) { this.id=id; name=n; email=e; }
    String toFile() {
        StringBuilder sb = new StringBuilder();
        for(int i=0;i<issued.size();i++) {
            if(i>0) sb.append(","); sb.append(issued.get(i));
        }
        return id+"|"+name+"|"+email+"|"+sb.toString();
```

```
}
static Member fromFile(String line) {
    if (line == null || line.trim().isEmpty())
        return null;
    try {
        Member m = new Member(Integer.parseInt(p[0]),
                              p[1], p[2]);
        if (p.length > 3 && !p[3].trim().isEmpty()) {
            for (String s : p[3].split(",")) if (!s.trim()
                .isEmpty()) m.issued.add(Integer.
                    parseInt(s.trim()));
        }
        return m;
    } catch (Exception e) { return null; }
}
private final Map<Integer, Book> books = new
                                    TreeMap<>();
                    Member
private final Map<Integer> members = new TreeMap<>();

private final Set<String> categories = new TreeSet<>();
private final String BOOKS_FILE = "books.txt", MEMBERS_
                                    FILE = "members.txt";
private int nextBookId = 101, nextMemberId = 201;
private final Scanner sc = new Scanner(System.in);

public static void main(String[] args) {
    new CityLibrary().run();
}
```

```
void run () {
    loadAll ();
    while (true) {
        showMenu ();
        String ch = sc.nextLine().trim ();
        switch (ch) {
            case "1": addBook (); break;
            case "2": addMember (); break;
            case "3": issueBook (); break;
            case "4": returnBook (); break;
            case "5": searchBooks (); break;
            case "6": sortBooks (); break;
            case "7": listBooks (); break;
            case "8": listMembers (); break;
            case "9": saveAll ();
                System.out.println ("saved Exiting."); return;
            default: System.out.println ("Invalid choice. Try
                                        again.");
        }
    }
}

void showMenu () {
    System.out.println ("\n Welcome to City Library Digital
                        Management 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 Books");
System.out.println("6. Sort Books");
System.out.println("7. List All Books");
System.out.println("8. List All Members");
System.out.println("9. Exit");
System.out.println("Enter your choice:");
}

void addBook() {
    System.out.print("Enter Title:");
    String t = sc.nextLine().trim();
    System.out.print("Enter author:");
    String a = sc.nextLine().trim();
    System.out.print("Enter Category:");
    String c = sc.nextLine().trim();
    int id = nextBookId++;
Book b = new Book(id, t, a, c, false);
books.put(id, b); categories.add(c);
saveAll();
System.out.println("Book added with ID:"+id);
}

void addMember() {
    System.out.print("Enter name:");
    String n = sc.nextLine().trim();
    System.out.print("Enter email:");
```

```
String e = sc.nextLine().trim();
int id = nextMemberId++;
Member m = new Member(id, n, e);
members.put(id, m);
saveAll();
System.out.println("Member added with ID: " + id);
}

void issueBook() {
try {
System.out.print("Enter book ID: ");
int bid = Integer.parseInt(sc.nextLine().trim());
Books b = books.get(bid);
if (b == null) { System.out.println("Book not
found");
return; }
if (b.isIssued) {
System.out.println("Book already issued.");
return; }
System.out.println("Enter Member ID: ");
int mid = Integer.parseInt(sc.nextLine().trim());
Member m = members.get(mid);
if (m == null) { System.out.println("Member
not found");
return; }
b.markAsIssued();
if (!m.issued.contains(bid))
m.issued.add(bid);
```

```
saveAll(); System.out.println("Book issued to member "+mid+".");
} catch (Exception ex) {
System.out.println("Invalid input."); }
}

void returnBook() {
    try {
System.out.print("Enter book ID: ");
    int bid = Integer.parseInt(sc.nextLine().trim());
Book b = books.get(bid);
if (b==null) { System.out.println("Book not found");
    return; }
if (!b.isIssued) { System.out.println("Book is not issued");
    return; }
System.out.print("Enter Member ID: ");
    int mid = Integer.parseInt(sc.nextLine().trim());
Member m = members.get(mid);
    if (m==null) {
System.out.println("Member not found");
    return; }
if (!m.issued.contains(bid)) {
System.out.println("This member does not have the book."); return;
    }
```

b.markAsReturned();
m.issued.remove(Integer.valueOf(bid));
saveAll();
System.out.println("Book returned");
} catch (Exception ex) {
System.out.println("Invalid Input");}
}

void searchBooks() {
System.out.println("Search by: 1. Title 2. Author
3. Category");

System.out.print("Choice: ");
String c = sc.nextLine().trim();
System.out.print("Enter search term: ");
String term = sc.nextLine().trim().toLowerCase();

boolean found = false;
for (Book b: books.values()) {
boolean match = false;
if ("1".equals(c) && b.title.toLowerCase().
contains(term)) match = true;

else if ("2".equals(c) && b.author.toLowerCase().
contains(term)) match = true;

else if ("3".equals(c) && b.category.toLowerCase() &&
contains(term)) match = true;

```
else if (!Arrays.asList("1","2","3").contains
                                    (c) &&
    (b.title.toLowerCase().contains(term)||b.author.
     toLowerCase().contains(term)||b.
     category.toLowerCase().contains(term)))

    match = true;
    if (match) { b.display();
        found = true;}
    }
    if(!found) System.out.println("No books found");
    }

void sortBooks() {
    System.out.println("Sort by: 1.Title 2.Author
                                3.Category");

System.out.print("Choice:");
String c = sc.nextLine().trim();
List<Book> list = new ArrayList<>(books.values());
if("2".equals(c)) Collections.sort(list, new
                    Comparator<Book>(){

public int compare(Book a,Book b) {
    return a.author.compareToIgnoreCase(b.author);
    }
    }
);
```

```
else if ("3".equals(c)) Collections.sort(list, new
                    Comparator<Book>() {
    public int compare(Book a, Book b) {
    int r = a.category.compareToIgnoreCase(b.category);
    else Collections.sort(list);
    for (Book b: list) b.display();
    }

void listBooks() { if (books.isEmpty()) {
System.out.println("No books.");
    return; }
for (Book b: books.values()) b.display();
System.out.println("Categories: " + categories); }

void listMembers() { if (members.isEmpty()) }
System.out.println("No members");
    return; }
for (Member m: members.values()) m.display();
    }
void loadAll() { loadBooks();
    loadMembers();
if (!books.isEmpty()) nextBookId = Collections.max
                    (books.keySet()) + 1;

    if (!members.isEmpty())
nextMemberId = Collections.max(members.keySet()) + 1;
    }
```

```
void loadBooks() {
File f = new File(BOOKS_FILE);
try {
  if(!f.exists()) f.createNewFile();
BufferedReader br = new BufferedReader(new File
                                  Reader(f));
String line;
while((line = br.readLine()) != null){
  Book b = Book.fromFile(line);
  if(b != null){ books.put(b.id, b);
  categories.add(b.category);}
  }
  br.close();
} catch(IOException e){
  System.out.println("Error loading books.");
  }
}
void loadMembers(){
  File f = new File(MEMBERS_FILE);
  try{
    if(!f.exists()) f.createNewFile();
  BufferedReader br = new BufferedReader(new
                                  FileReader(f));
  String line;
  while((line = br.readLine()) != null){
    Member m = Member.fromFile(line);
    if(m != null) members.put(m.id, m);
    }
    br.close();
```

```
        }
        catch (IOException e) {
            System.out.println("Error loading members");
        }
    }

    void saveAll() {
        try {
            BufferedWriter bw = new BufferedWriter(new
                    FileWriter(BOOKS_FILE, false));
            for (Book b : books.values()) {
                bw.write(b.toFile()); bw.newLine();}
            bw.close();
            bw = new BufferedWriter(new FileWriter(
                    MEMBERS_FILE, false));

            for (Book b : books.values()) {
                bw.write(b.toFile()); bw.newLine();}
            bw.close();
            bw = new BufferedWriter(new FileWriter(
                    MEMBERS_FILE, false));

            for (Member m : members.values()) {
                bw.write(m.toFile());
                bw.newLine();
            }
            bw.close();
        }
```

```
catch (IOException e) {
    System.out.println("Error saving data");
}
}
```