# Assignment - 4

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

class Book implements Comparable<Book>{
    private int bookId;
    private String title;
    private String author;
    private String category;
    private boolean issued;
    public Book(int bookId, String title, String author,
                String category, boolean issued){
        this.bookId = bookId;
        this.title = title;
        this.author = author;
        this.category = category;
        this.issued = issued;
    }
    public int getBookId(){return bookId;}
    public String getTitle(){return title;}
    public String getAuthor(){return author;}
    public String getCategory(){return category;}
    public boolean isIssued(){return issued;}

    public void markAsIssued(){issued = true;}
    public void markAsReturned(){issued = false;}
    public void displayBookDetails(){
        Sout("BookId" + bookId);
        Sout("Title" + title);
        Sout("Author" + author);
        Sout("Category" + category);
        Sout("Issued" + (issued? "Yes": "No"
    }
}
```

```
@Override
public int compareTo(Book other){
    return this.title.compareToIgnoreCase(other.title
}
public String toFileString(){
    return bookId+";"+title+";"+author+";"+
        category+";"+issued;
}
public static Book fromFileString(String s){
    String[] p = s.split(";");
    return new Book(
        Integer.parseInt(p[0]),
        p[1],
        p[2],
        p[3],
        Boolean.parseBoolean(p[4])
    );
}
}

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

    public Member(int memberId, String name, String mail
        this.memberId = memberId;
        this.name = name;
        this.email = email;
    }
    public int getMemberId(){ return memberId;}
```

```
public void addIssuedBook(int bookId){
    issuedBooks.remove(Integer.valueOf(bookId));
}

public String toFileString(){
    return memberId+";"+name+";"+email+
    issuedBooks.toString();
}

public static Member fromFileString(String
    String[] p = s.split(";", 4);
    Member m = new Member(Integer.parseInt(p[0]),p[1],p[2],
                          p[3]);

    if(p.length == 4 && p[3].length()>2){
        String books = p[3].substring(1, p[3].length()-1
        if(!books.isEmpty()){
            for(String x: books.split(",")){
                m.addIssuedBook(Integer.parseInt(x));
            }
        }
    }
    return m;
}
}

Class Main{

    private Map<Integer, Book> books = new HashMap<>(
    private Map<Integer, Member> member = new HashMap
    private Set<String> categories = new HashSet<>();

    private final String BOOK_FILE = "books.txt";
    private final String MEMBER_FILE = "member.txt";
```

```
Scanner sc = new Scanner (System.in);
public Main (){
    loadBooks();
    loadMembers();
}
Void loadBooks (){
try(BufferedReader br = new BufferedReader(new File
    Reader (BOOK_FILE))){
    String line;
    While((line = br.readLine()) != null){
        book b = Book.fromFileString (line);
        books.put(b.getBookId(), b);
        categories.add(b.getCategory());
    }
}catch (Exception e){}
}
Void loadMembers(){
try (BufferedReader br = new BufferedReader (new
                FileReader (MEMBER_FILE))){
    String line;
    While((line = br.readline()) != null ){
        Member m = Member.fromFileString (line);
        member.put(m.getMemberId(), m);
    }
}catch (Exception e ){}
}

Void saveBooks(){
try (BufferedWriter bw = new BufferedWriter (new FileWriter
                                (BOOK_FILE))){
    }
    for (Member m : member.values ()){
        bw.write(m.toFilestring ());
        bw.newLine ();
    }
}Catch (Exception e){}
```

```
public void addBook(){
    Sout("Enter book ID");
    int id = sc.nextInt();
    sc.nextline();
    Sout("Enter Title");
    String title = sc.nextline();
    Sout("Enter Author");
    String author = sc.nextline();
    Sout("Enter Category");
    String category = sc.nextline();
    Book b = new book(id, title, author, category, fa
    books.put(id, b);
    categories.add(category);
    savebook();
    Sout("book added");
}
public void issueBook(){
    Sout("Enter Book ID");
    int bookId = sc.nextInt();
    Sout("Enter MemberID");
    int memberId = sc.nextInt();
    Book b = books.get(bookId);
    Member m = members.get(memberId);
    if(b == null || m == null){
        Sout("Invalid book or member ID");
        return;
    }
    if(b.isIssued()){
        Sout("Book already issued");
        return;
    }
    b.markAsIssued();
    m.addIssuedBook(bookId);
```

```
            saveBooks();
            saveMembers();
            Sout("Book Successfully issued");
        }

        public void returnBook(){
            Sout("Enter bookID");
            int bookId = sc.nextInt();
            Sout("Enter MemberID");
            int memberId = sc.nextInt();
            Book b = books.get(bookId);
            Member m = members.get(memberId);
            if(b == null || m == null){
                Sout("Invalid book or member");
                return;
            }
            b.markAsReturned();
            m.returnIssuedBook(bookId);
            saveBooks();
            saveMember();
            Sout("Book returned");
        }

        public void searchBooks(){
            sc.nextLine();
            Sout("search by title/author");
            String key = sc.nextLine().toLowerCase();
            for(Book b : books.values()){
                if(b.getTitle().toLowerCase().contains(key)||
                    b.getAuthor().toLowerCase().contains(key)||
                    b.getCategory().toLowerCase().contains(key)){
                b.displayBookDetails();
                Sout("---");
                }
        }
    }
}
```

```
public void sortBooks(){
List<Books> list = new Arraylist <>(books.value
Sout("1. sort by title");
Sout("2. sort by Author");
int choice = sc.nextInt();
if(choice==1) Collections.sort(list);
else list.sort(comparators.comparing(Book::getA
for(Book b: list){
    b.displayBookDetails();
    Sout("----");
}
}
public void menu(){
  int choice;
  do{
      Sout("\n Elicity library Managmt Sys
      Sout("1. Add book");
      Sout("2. Add Member");
      Sout("3. Issue book");
      Sout("4. Return book");
      Sout("5. Search book");
      Sout("6. Sort book");
      Sout("7. Exit");
      Sout("Enter choice");
      choice=sc.nextInt();
      switch(choice){
        case 1: addBook(); break;
        case 2: addMember(); break;
        case 3: issueBook(); break;
        case 4: returnBook(); break;
        case 5: searchBook(); break;
        case 6: sortBook(); break;
        case 7:
```

```
            saveBooks();
            saveMembers();
            Sout("Good Bye");
            break;
        default:
            Sout("Invalid choice")
        }
    }while(choice != 7);
}

PSVm(String Args[]){
    Main lm = new Main();
    lm.mainU();
}
}
```