Assignment - 4

```
import java.io.*
import java.util.*
class Book implements Serializable, Comparable
<Book> {
    private Integer bookid;
    private String title;
    private String author;
    private String category;
    private boolean isIssued;
    public Book(int bookid, String title,
      String author, String category) {
        this.bookid = bookid;
        this.title = title;
        this.author = author;
        this.category = category;
        this.isIssued = false;
    }
    public Integer getBookId() {
      return bookid;
    }
    public String getTitle() {
        return title;}
    public String getAuthor() {
        return author;}
    public String getCategory() {
        return category;}
    public boolean isIssued() {
        return isIssued;
    }
    public void markAsIssued() {
        this.isIssued = true;}
    public void AsReturned() {
```

```
this.isIssued = false; }
public void displayBookDetails() {
    System.out.println("Book Details:");
    System.out.println("ID:" + bookid);
    System.out.println("Title:" + title);
    System.out.println("Author:" + author);
    System.out.println("Category:" + category);
    System.out.println("Issued:" + (isIssued? "Yes" : "No"));
}
public int compareTo(Book b){
    return this.title.compareToIgnoreCase(b.title);
}}
class Member implements Serializable {
    private int mid;
    private String name;
    private String mail;
    private List<Integer> issuedBooks;
    public Member(int mid, String name, String mail){
        this.mid = mid;
        this.name = name;
        this.mail = mail;
        this.IssuedBooks = new ArrayList<>();
    }
    public int getMemberID(){
        return mid;
    } public List<Integer> getIssuedBooks(){
        return issuedBooks; }
    public void returnIssuedBook(int bookid){
        issuedBooks.remove(Integer.valueOf(bookid));
    }
    public void void displayMemberDetails(){
        System.out.println("Member Details");
```

```
        System.out.println("ID: "+mid);
        System.out.println("Name: "+name);
        System.out.println("Email"+mail);
        System.out.println("IssuedBooks"+issuedBooks);
    }}
class LibraryManager {
        private Map<Integer,Book>books=new HashMap();
        private Map<Integer,Member>members=new HashMap();
        private final String BOOK_FILE="books.dat";
        private final String MEMBER_FILE="members.dat";
        public LibraryManager(){
            loadData();
        }
        public void loadData(){
        try(ObjectInputStream oisBook=new
            ObjectInputStream(new FileInputStream
                (BOOK_FILE));
    ObjectInputStream oisMember=new ObjectInputStream(
      new FileInputStream(MEMBER_FILE))){
        books = (Map<Integer,Book>)oisBook.readObject();
        members=(Map<Integer,Member>)oisMember.readObject();
      }catch(Exception ignored){}
    }
      public void saveData(){
      try(ObjectOutputStream oosBook=new ObjectOutputStream(
        new FileOutputStream(BOOK_FILE));
        ObjectOutputStream oosMember=new ObjectOutputStream(
        new FileOutputStream(MEMBER_FILE))){
          oosBook.writeObject(books);
          oosMember.writeObject(members);
        }catch(IOException e){
          System.out.println("Error saving data");
Bittoo  }}
```

```
public void addBook(Book book) {
    books.put(book.getBookId(), book);
    saveData();
}
public void addMember(Member member) {
    members.put(member.getMemberId(), member);
    saveData();
}
public void issueBook(int bookId, int mid) {
    Book b = books.get(bookId);
    Member m = members.get(memberID);
    if (b==null || m==null) {
        System.out.println("Invalid ID!");
        return;
    }
    if (b.isIssued()) {
        System.out.println("Book already issued");
        return;
    } b.markAsReturned();
    m.returnIssuedBook(bookId);
    saveData();
    System.out.println("Book returned");
} public void searchBooks(String keyword) {
    books.values().stream().filter
(b -> b.getTitle().toLowerCase().contains
(keyword.toLowerCase())).forEach
(Book::displayBookDetails);
}
public void sortBooks() {
    List<Book> list = new ArrayList<>(books.values());
    Collections.sort(list);
    list.forEach(Book::displayBookDetails);
}}
```

Bittoo

```
public class Main {
    public static void main(String[] args){
    LibraryManager obj = new LibraryManager();
    Scanner sc = new Scanner(System.in);
    int choice;
    System.out.println("Welcome to CityLibrary!");
    do {
      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.print("Enter choice:");
       choice= sc.nextInt();
       switch(choice){
       case1:
         sc.nextLine();
         System.out.println("Enter book title");
         string t = sc.nextLine();
          System.out.println("Enter author");
          String a = sc.nextLine();
          System.out.println("Enter category:");
          String c = sc.nextLine
          int id = math.random();
          manager.addBook(new Book(id,t,a,c));
          break;
    case 2:
          sc.nextLine();
          String m = sc.nextLine();
          System.out.println("Enter email:");
```

```
String e = sc.nextLine();
int mid = Math.random();
manager.addMember(new Member(mid,m,e));
System.out.println("Member added:" +mid);
break;
case 3:
    System.out.print("Enter Book ID");
    int bidIssue = sc.nextInt();
    int midIssue = sc.nextInt();
    manager.issueBook(bidIssue,midIssue);
    break;
case 4:
    System.out.print("Enter Book ID");
    int bidreturn = sc.nextInt();
    int midreturn = sc.nextInt();
    manager.returnBook(bidreturn,midreturn);
    break;
case 5:
    sc.nextLine();
    System.out.print("Enter keyword:");
    String key = sc.nextInt();
    manager.searchBooks(key);
    break;
case 6:
    manager.sortBooks;
    break;

case 7:
    System.out.println("Exited");
    manager.saveData();
    break;
```

```
default:
    System.out.println("Invalid choice.");
  }
} while (choice != 7);
   sc.close()
}}
```