Swashk Sharma
2401730032

Assignment-4

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

Class book implements comparable <Book> {

private int bookId;
private string title;
private string author;
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 get bookId() { return bookId; }
public string get title() { return title; }
public string get author() { return author; }
public string get category() { return category; }

public boolean issued() { return issued }
public void mark As issued() { issued = true; }
public void mark As returned() { issued = false; }

public void display Book details() {
        system.out.printf("Book ID" + bookId);
        system.out.printf("Title:" + title);
```

```
system.out.println("author"+author);
system.out.println("category:"+category);
system.out.println("issued:"+issued?
        "yes":"no"));
}
public int compareTo(Book other){
  return
this.title.compareToIgnoreCase(other.title);
}
public static Book fromCSVString(String csvLine){
String[] p = csv.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 email){
this.memberId = memberId
 this.name = name;
this.email = email;
}
 public int getMemberId(){ return memberId;}
 public void addIssuedBook(int bookId){
  issuedBooks.add(bookId);
}
 public void returnIssuedBook(int bookId){
 issuedBook.remove(Integer.valueOf(bookId));
}

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

String[] p = s.split(",");
 Member m = new
Member(Integers.parseInt(p[0]), p[1], p[2]);
   if(p.length > 3){ p[3].length() > 2}
   String books = p[3].substring(1, p[3].
                          length()-1)
     if(!book.Empty()){
```

```
for (String x : Books.split(":")){
    m.addIssuedBook(Integer.parseInt(x));
}

}

}

    return;
}

}
public class main{
private Map<Integer, Book> Book = new HashMap<>();
    private Map<Integer, Member> member =
                        new LinkedHashMap<>();
private Set<String> categories = new HashSet<>();

private final String bookFile = "Books.txt";
private final String memberFile = "member.txt";

Scanner s = new Scanner(System.in);
    public main(){
    loadBook();
    loadMember();
}
    void loadBook();
```

```
try(BufferedReader br = new BufferedReader
        (new FileReader(f-file)))
{

String line;
while ((line = br.readline()) != null)
 book b = Book.fromFileString(line);
 books.put(b.getBookId(),b);
 categories.add(b.getCategory());

}
} catch (Exception) {}
}
void loadMember()
 try(BufferedReader br = new BufferedReader()
        new FileReader(MEMBER.file))}

String line;
while((line = br.readline()) != null)}
book b = book.fromString(file);
books.put(b.getBookId(),b)}
categories.add(b.getCategory());
}

} catch(exception){}
} void saveBooks()
```

```
try (BufferedWriter bw = new BufferedWriter
(new BufferedWriter(new FileWriter(Book-file));
for (Book b: books.values())
    bw.write(b.toFileString());
    bw.newLine();
}
} catch (Exception e)}
}
void saveMembers()}
try (BufferedWriter bw = new BufferedWriter
for member m; members.value()}
    bw.newLine();
}
} catch (exception)}}
}
public void addBook()}
System.out.println("Enter Book ID");
int ID: sc.nextInt(); sc.nextInt();
bw.nwLine
String title = sc.nextLine();
String author = sc.nextLine();
Book b: new book id, title, author, category, false;
books.put(id b);
categories.add(category);
save.Books();
system.out.println(new book);
```

```
public void addMember() {
System.out.println("Enter Member ID");
int id = sc.nextInt(); sc.nextLine();
System.out.println("Enter Email:");
String mail = sc.nextLine();
System.out.println("Book added");
}
public void issueBook() {
System.out.println("Enter Book ID");
int BookId = sc.nextInt();
System.out.println("Enter Member ID:");
int MemberId = sc.nextInt();
Book b = books.get(BookId);
Member m = members.get(memberId);
if (b == null || m == null) {
System.out.println("Invalid book");

return;
}
if b.isIssued() {
System.out.println("Book already issued");
return;
}
b.markAsIssued();
m.addIssuedBook(BookId)
saveMembers()
System.out.println("Book issued");
```

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

public void sortBook() {
List<Book> list = new
ArrayList<>(Book.values());
System.out.println("1.sort by Title");
System.out.println("2.sort by Author:");
int choice = sc.nextInt();
if (choice == 1) {
    for (Book b : list) {
        b.displayBookDetails();
        System.out.println("-----");
    }
}
}
```

```
public void menu() {
int choice;
do {
System.out.println("Library Digital
    Management system:");

System.out.println("1. Add Book");
System.out.println("2 Add Members");
System.out.println("3 Issue Book");
System.out.println("4 search book");
System.out.println("5 Return Books");
System.out.println("6 sort Book");
System.out.println("7. Exit");

switch(choice)

Case 1 : addBook();break;
Case 2 : addMember();break;
Case 3 : displayBook();break;
Case 4 : return book();break;
Case 5 : search book();break;
Case 6 : sort Books();break;
Case 7 :
   save book();
   save member();
   System.out.println("Goodby!");
   break;
   default;
   System.out.println("Invalid choice");
}
```

```
    } while (choice != 7);
}

Public Static void main (String [] args) {

Main In = new main ();
 In.menu();

}
}
```