Adam Khales

Id: 2493077

Final Project:

Library-Management-System

420-SF2-RE DATA STRUCTURES AND OBJECT-ORIENTED PROGRAMMING

Table of contents

Project description (deliverable 1):

Library Management System (LLMS) Project Documentation

Scenario :

The Limited Library Management System (LLMS) is a small-scale application where librarians can manage books in a catalog, issue and return books, and view the catalog and issued books. The system allows librarians to verify student details and check book availability before issuing. Once a book is issued, the book's details are updated in the system. The students can search for books by title, author, or publication year. Librarians can manage the books, while students can borrow and return them.

Design Paradigm:

The system will demonstrate the following functionalities:

- **Book Management:** Librarians will add books to the catalog, track their availability, and issue/return them.
- **Book Search:** Students will search for books by title, author, or publication year.
- **Issuing Books:** Books will be issued to students, with tracking of availability and issued quantity.
- **Returning Books:** Books can be returned by students, and the system will update the catalog accordingly.
- **Catalog Display:** Students and librarians can view the full catalog, with books sorted by serial number (SN).
- **Read/write** from a database using text IO

Librarian:

-search books
- add books

-track availability of books (boolean isAvailable)

-issue books

Students:

-search books

- track availability of books (boolean isAvailable)

- return books

Expected Output:

The system will allow:

- **Librarians** to add, issue, return, and manage books in the catalog.
- **Students** to search for books, borrow and return them.
- A dynamic catalog that tracks book availability and issued books.
- Proper error handling if a book is not available or if a student tries to return a book not issued to them.

Example outputs:

- "Enter book title to add: The Great Gatsby"
- "Enter student ID to issue book: 12345"
- "Book issued successfully."
- "Enter title, author, or publication year to search:"

Hierarchies:

The system will contain the following class hierarchies:

- **Book Class**: represents books in the catalog (title, author, serial number, etc)
    - Audio Book extends Book, adds duration
    - Paper Book extends Book, adds page count
- **User Class** (Parent class): Represents users of the system.
    - **Student Class**: Inherits from User, with additional functionality for borrowing and returning books.

    - **Librarian Class**: Inherits from User, with functionality for managing the catalog, issuing books, and viewing issued books.

Interface:

- **Borrowable Interface**: This interface will define the operations related to books, such as borrow, return, and isAvailable. This is needed to ensure that any class implementing this interface (such as the `Librarian` class) can perform these operations in a standardized way.

Runtime Polymorphism:

- A displayInfo() method will be inherited by Audio book and Paper book from Book and will have to be overridden in AudioBook and PaperBook.

Text I/O:

- **LibraryManagementSystem Class**: This class will use text-based I/O to interact with a database. It will use the database to see if there is already a library made with already created books to avoid having to create everything from scratch every time.

Comparable and Comparator:

- The Book class will implement the `Comparable` interface. This will allow books to be compared and sorted based on their serial number (SN).
- A **Comparator** will be implemented in the child classes if in need of another sorting method specific to them.

Class Diagram:

Here's a description of the classes and their relationships:

Class: Book (abstract)

Implements: `Comparable<Book>`
Implements: `Borrowable`

- **Fields:**
    - `title: String`
    - `author: String`

        - `serialNumber: int`
        - `Copies: int`
- **Methods:**
    - `+displayInfo():`
    - `+compareTo(Book):`

Subclass: AudioBook extends Book

- **Fields:**
    - `duration: int`
- **Methods:**
    - `+displayInfo():`(overridden)

Subclass: PaperBook extends Book

- **Fields:**
    - `pageCount: int`
- **Methods:**
    - `+displayInfo():`(overridden)

Interface: Borrowable

- **Methods:**
    - `+borrowBook():`
    - `+returnBook():`
    - `+isAvailable():`

Class: User (abstract)

- **Fields:**
    - `name: String`
    - `userId: int`

Subclass: Student extends User, implements `Borrowable`

- **Methods:**
  - `+searchBook():`
  - `+borrowBook():`
  - `+returnBook():`

Subclass: Librarian extends User, implements `Borrowable`

- **Methods:**
  - `+addBook():`
  - `+issueBook():`
  - `+returnBook():`
  - `+searchBook():`

Class: LibraryManagementSystem

- **Responsibilities:** Handles I/O (text-based)
- **Methods:**
  - `+read():`
  - `+write():`

Deliverable 2:

For the second deliverable, the following parts will be implemented:

- **Class Structures**: Implementation of the Book, `User`, `Student`, `Librarian`, `Audio book`, `Paper book` classes.
- **Interface**: borrowable
- **Text-Based Interaction**: read method of text I/O
- **JUnit Tests**:
  - Methods who will be tested
    - `+addBook():`
    - `+issueBook():`

- ♣ `+returnBook():`
- ♣ `borrowBook():`
- ♣ `SearchBook():`

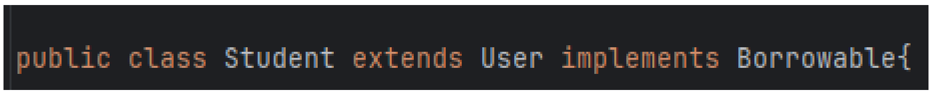
Project Features and Screenshots:

Requirements:

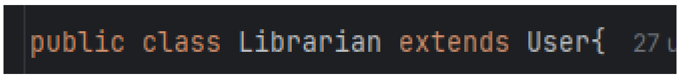
<u>Two hierarchies:</u>

Hierarchy 1:

User, Student, Librarian



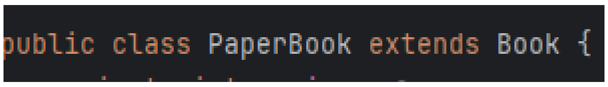
```
public class Student extends User implements Borrowable{
```



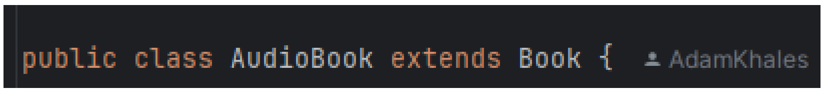
```
public class Librarian extends User{  27 u
```

Hierarchy 2 :

Book, AudioBook, PaperBook



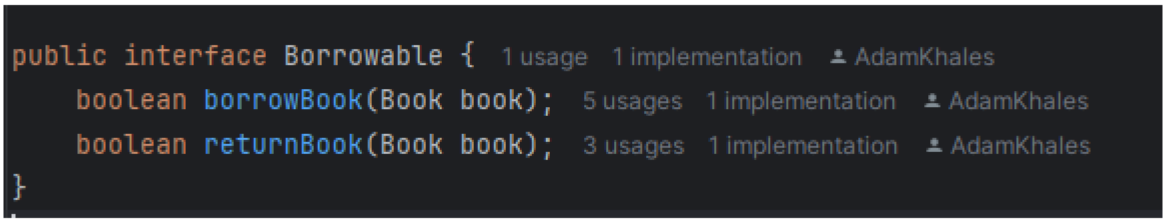
```
public class PaperBook extends Book {
```



```
public class AudioBook extends Book {  AdamKhales
```

<u>Interface:</u>

Borrowable interface with at least 1 abstract method:

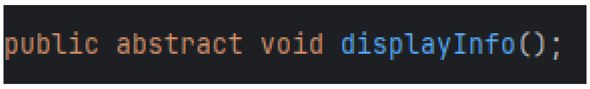


```
public interface Borrowable {  1 usage  1 implementation  AdamKhales
    boolean borrowBook(Book book);  5 usages  1 implementation  AdamKhales
    boolean returnBook(Book book);  3 usages  1 implementation  AdamKhales
}
```

Runtime-Polymorphism:

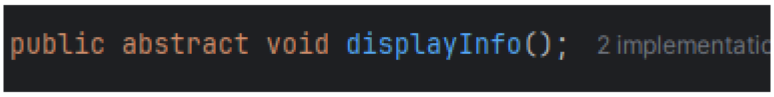
displayInfo method implemented in subclasses but are different:

In User class :



```
public abstract void displayInfo();
```

In Book class :



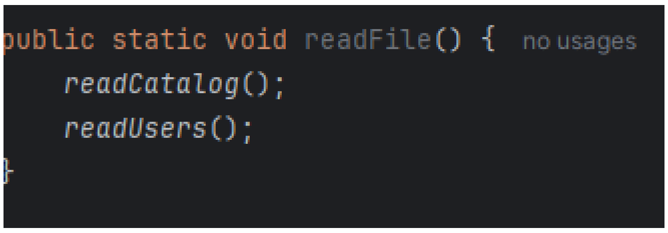
```
public abstract void displayInfo();  2 implementatio
```

Text I/O:

Text I/O in LibraryManagementSystem class:

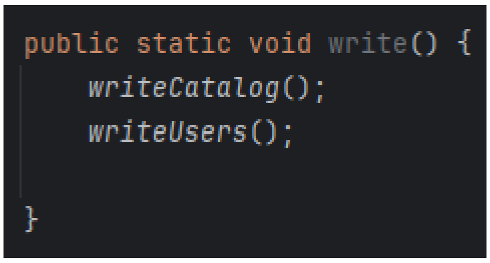
The methods write in two files: catalog.csv and users.csv

Read method using helpers :



```
public static void readFile() {  no usages
    readCatalog();
    readUsers();
}
```

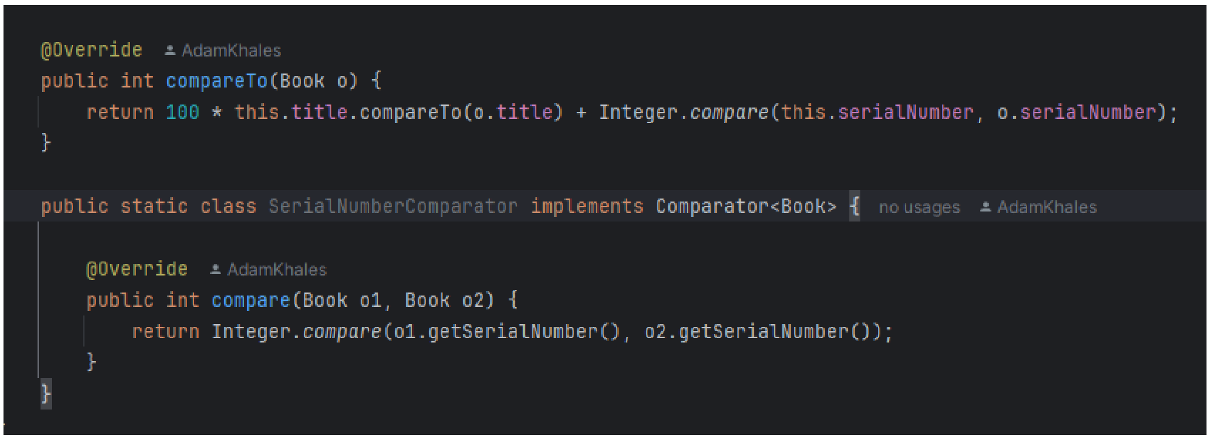
Write method using helpers:



```
public static void write() {
    writeCatalog();
    writeUsers();

}
```

Implementation of comparable and comparator interfaces:

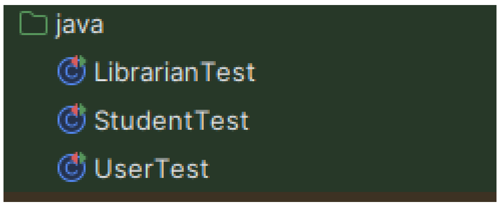


```
@Override  AdamKhales
public int compareTo(Book o) {
    return 100 * this.title.compareTo(o.title) + Integer.compare(this.serialNumber, o.serialNumber);
}

public static class SerialNumberComparator implements Comparator<Book> {  no usages  AdamKhales

    @Override  AdamKhales
    public int compare(Book o1, Book o2) {
        return Integer.compare(o1.getSerialNumber(), o2.getSerialNumber());
    }
}
```

<u>Unit testing for methods:</u>



```
java
    LibrarianTest
    StudentTest
    UserTest
```

Each tested method has 3 or more testing cases.

Challenges:

The biggest challenge i faced in this project was to design it for deliverable 1. I did not have any ideas and I had troubles trying to come up with different methods to implement in my project. Another challenge I faced in this project was that I did not know how to use github repositories the way I am supposed to. However, I finally figured it out and completed my project. Other than that, I did not face any challenges or issues when it came to implementing my methods and classes, everything went smoothly.

Learning Outcomes:

During this project, I got to learn a bit more about github and how to use it. I learned to initialize repositories with git init, track changes using git status, how to use git add and git commit and how to push my work to my github repository using git push. This will be useful for future schoolwork as well as for personal projects I may start.

Other than learning more about how to use github, this project allowed me to practice most of the material learned this semester and allowed me to make sure that I understood everything as well as being able to implement the topics in projects.