Smart Library Management System

Lucas Cameron Dash

420-SF2-RE DATA STRUCTURES AND OBJECT-ORIENTED PROGRAMMING section 00002

Outline

- Project Description
- Program features
- Challenges and obstacles

Project Description

Project Title

Smart Library Management System

Scenario

This application simulates a library system for managing books, users, and transactions. Librarians can add/remove books, register members, and view borrowing history. Students/members can borrow books or magazines at a reduced price and non-memebrs will be able to rent books at regular prices.

Design paradigms

- **User Role Differentiation:**
  - Students can browse, borrow, and return books.
  - Librarians can add, remove, and manage book inventory, as well as register new users.
  - Books while be held in a Map that shows the amount of each book available.
- **Search & Sort:**
  - Users can search for books by title, author, or publication year using method overloading.
  - Books can be sorted by title or publication year using Comparable and Comparator.
- **Data Persistence:**

    - All books and user data are stored in text files and are loaded when the program starts and saved when it ends.
    - The text files holds the rented books so when they need to be returned, they can be taken from there.
- **Class Hierarchies:**
    - User is a base class with subclasses Student Librarian and non-Member.
    - Item is an abstract class with Book and Magazine as concrete subclasses.
- **Interfaces:**
    - Borrowable interface defines a method borrowItem(User user) that is implemented by borrowable items like books and magazines.

Expected Output

- Users can log in as student or librarian

- View/search/sort books

- Borrow/return books (students/non-memebers)

- Add/remove books, register users (librarians)

- Data is saved to and loaded from `.txt` files

Hierarchies

1. User Hierarchy:

  - `User` (abstract)

    - `Student`

    - `Librarian`

    - `non-Member`

2. Item Hierarchy:

  - `Item` (abstract)

    - `Book`

    - `Magazine`

Interface

- Interface: `Borrowable`

- Method: `boolean borrowItem(User user)`

- Purpose: Common logic for borrowable items like books or magazines.

Runtime Polymorphism

- `User.displayDashboard()` overridden in `Student` and `Librarian`

- `borrowItem()` method overridden in `Book`, `Magazine`

TextIO

- Purpose: Load/save book and user data to text files

-when book is being rented to a user it will write to that users file and when it will be returned it will be taken from the user's files

Comparable and Comparator

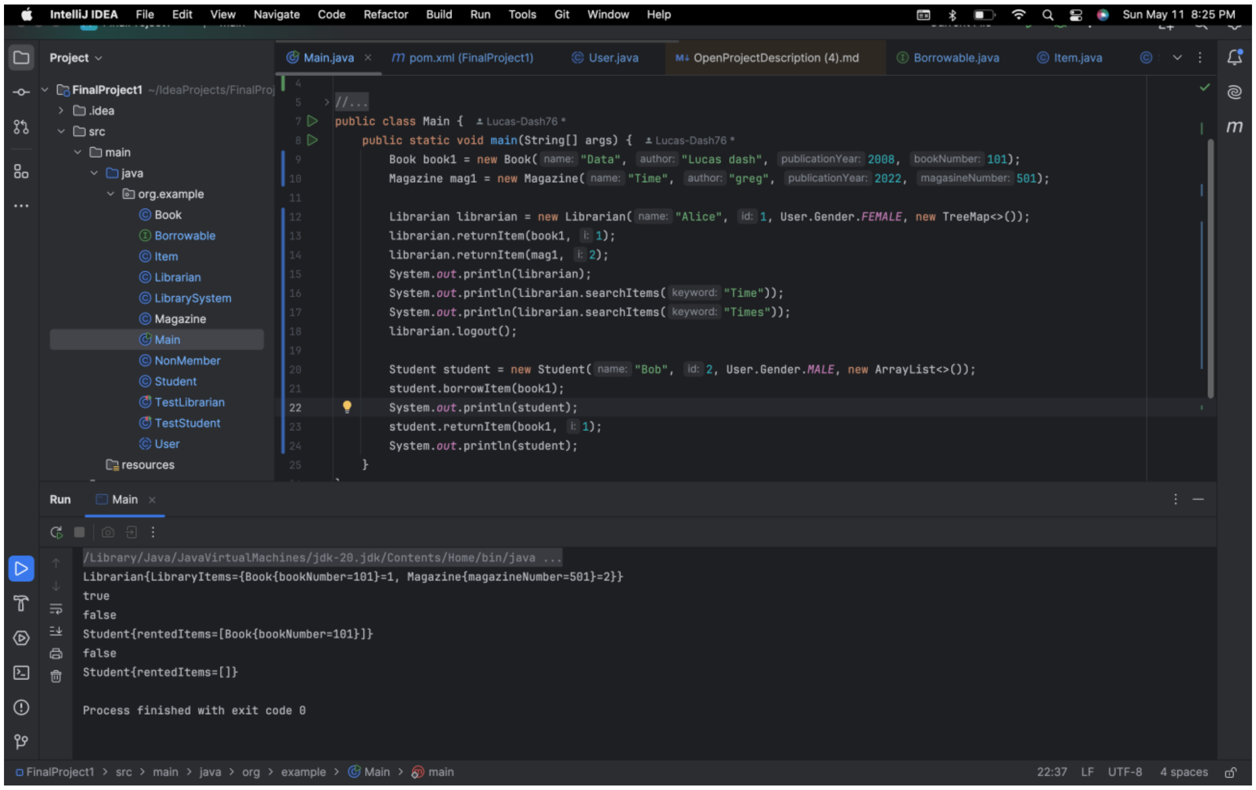
- `item implements Comparable<item>` for sorting by title

https://github.com/Lucas-Dash76/FinalProject

Project Features

- The project has the requirements needed for the project.
- One of the requirements is the text files which can be seen in the screenshot below
- This project has features of adding and returning books seen in this screenshot.

- This screenshot also shows the different sub classes of user and has experimentation with the searchItems class.



```java
//...
public class Main {  Lucas-Dash76 *
    public static void main(String[] args) {  Lucas-Dash76 *
        Book book1 = new Book( name: "Data",  author: "Lucas dash",  publicationYear: 2008,  bookNumber: 101);
        Magazine mag1 = new Magazine( name: "Time",  author: "greg",  publicationYear: 2022,  magasineNumber: 501);

        Librarian librarian = new Librarian( name: "Alice",  id: 1, User.Gender.FEMALE, new TreeMap<>());
        librarian.returnItem(book1,  i: 1);
        librarian.returnItem(mag1,  i: 2);
        System.out.println(librarian);
        System.out.println(librarian.searchItems( keyword: "Time"));
        System.out.println(librarian.searchItems( keyword: "Times"));
        librarian.logout();

        Student student = new Student( name: "Bob",  id: 2, User.Gender.MALE, new ArrayList<>());
        student.borrowItem(book1);
        System.out.println(student);
        student.returnItem(book1,  i: 1);
        System.out.println(student);
    }
```

```
/Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java ...
Librarian{LibraryItems={Book{bookNumber=101}=1, Magazine{magazineNumber=501}=2}}
true
false
Student{rentedItems=[Book{bookNumber=101}]}
false
Student{rentedItems=[]}

Process finished with exit code 0
```

Challenges and Obstacles

During the implementation of my project i encountered multiple challenges that i had to overcome. Below is a list of these challenges and how I overcame them.

- The biggest obstacle I faced was when I was writing my LibrarySystem class. I wanted to ensure that the implementation was up to my standard and because I included a map as my library system it made it more difficult. To correct myself I always came back to the notes and our lessons and slowly but surely worked the class to my standards.

- Another obstacle i face when implementing this project is my searchItems class. I wanted to write a quick and simple way for this class but at first i found it difficult to know which stream method i should use and what order it should be in eventually I figured it out by looking what methods i could use after stream and finding out what they do.
- A smaller obstacle I faced is when I was Trying to create my hierarchy and I didn't know how to incorporate an interface. At first I wanted to include it in my item class but then I realized it would be more useful in my user class and subclasses since they had similar methods.