Leandro Frazão
Student ID: 2020093
Email: LeandroFrazao@hotmail.com

Git Hub: https://github.com/LeandroFrazao/Java_Project_2020/
Docker: https://hub.docker.com/r/leandrofrazao/java-library-app
After pulling the image, enter the command to create volume and container:
docker run -it --name java-library-app --mount source=java-library-app-data,target=/app leandrofrazao/java-library-app:1.4

Library System

Explanation of the structure of my code

I decided to create different classes for different uses. The objects Reader, Book, Borrows and Returns got an individual class. For the interface of the software, it was created a class called UserScreen, where it shows all different options for the user to control the application. To sort, search and other functions that manage data, it was created DataControl class. And the last class, ReadWriteFile is used to read files, load the array lists (arrays of books, readers, borrows and returns) and write data to files.

The main class starts, calls the UserScreen that has a constructer that creates an object DataControl. DataControl has also a constructor that creates an object ReadWriteFile, which will read books, readers, borrows and return files. When all data is loaded to the arrays, it returns to UserScreen Class, and brings the interface to the user screen.

The class UserScreen has a function to redirect the user choices for specific interface. It also checks and validate the input, returning errors if necessary. And depending on the user choices, it calls functions from DataControl class which allows:

- Search a Book by title and/or author, and show Details and Cover of a book if user wants to.
- List all Books in ascendant order by title or author;
- Search a Reader by Name and/or Surname;
- Search a Reader by ID;
- List all Readers in Ascendant order by Name or ID;
- To Borrow Book(s);
- List of Borrowed books by user ID;
- List of all Borrowed books;
- To Return Book(s);
- List of Returned books by user ID;
- List of all Returned books.

Justification for my logic design decisions

I tried to create a reliable application, that would handle all possible errors that the user might have. Indeed, I tried to make it efficient, using merge sort and binary search when possible. Also, I created a function to check and store information when the arrays is sorted, so it would avoid sorting the arrays again, for example, my books data has over 9 thousand items, and readers data has almost a thousand readers.

I decided to create a file to register borrows and another file to register returns. So, when user borrows a book, it will be recorded the user id, the book id, and the date of borrowing. However, when the user returns a book, it will automatically update the borrow file, removing the information of that specific book. That said, the Borrows file might have no register if the users returned their borrowed books, while the Returns file keeps track registering all returns, which contains information of the user, the books and date of borrowing and date of returning.

I created a function to check the number of each borrowed book in the beginning of the application, compare with the total of copies of the book in the “quantity” variable, so that the value of the variable in the book object called "copiesLeft" can be defined, and so, it is not necessary to rewrite the any file. Then every time the user borrows or returns a book, the variable “copiesLeft” changes, and it allows to check the books available to be borrowed. I also defined that the user can’t borrow two same titles of a book, and it is checked if they have not returned the borrowed book yet.

Explanation of the structure of the text files used to ensure data persistency

As I mentioned before, I am using 4 files in the csv format (Books, Readers, Borrows and Returns).

The Books file contains 9201 titles, that includes author, publication year, ISBN, image URL and quantity.

The Readers file contains 946 readers, that has information of first name, last name, email, gender, phone, address, town, and country.

The Borrows file contains data of borrowings that vary each time a borrows or returns occurs. This file records the user ID, book ID and date of borrowing.

The Returns file contains data of returning, and it records the user ID, book ID, date of borrowing and date of returning. The size of this file tends to grow, and it is possible to check all borrowing that had occurred since the beginning.