Library Catalogue System

Merrick MacInnis 6237800

Data Structures and Object Oriented Programming

Scenario:

- This project will be a library catalogue system. It will allow users to search up books based on a topic, the publication date, call number, title or author; displaying the media with the searched properties and none of nothing if no media falls under the search.
- Employees will be able to check out pieces of media to a user by associating their id number with the media being checked out.
- Employees will also be able to import and export a list of media information to and from the library catalogue system (in the form of csv files),
- as well as add and remove individual pieces of media.
- Users will also be able to sign up to become a new patron of the library by providing an email that hasn't been used by a previous patron of the library.

There will be two types of users, employees and patrons. Both will have names and identification numbers.

- Patrons will have fees, a limit on how many books they can borrow (let's say 5), and an email address.
- Employees will have a phone number, and a password which can be used to access employee exclusive parts of the program.

There will be multiple forms of media that can be checked out from the library all of which will have a borrow time (ex: 1 week, 1 month) and a fee charge (the amount a person would be charged for each day for a late return) as well as a status on whether the book is checked out or not and if it is being borrowed, the id number of the person borrowing it. All media will have a call number and topic as well.

- There will be books which have author(s), a publication date, a title, and a page count.
- There will be DVD's which will have a title, release date, director.

Both books and DVDs will implement the Comparator interface, being able to compare to the same subclasses of media, and if searched for in a way that the two can be compared then they will be compared in the same way. If both forms of media can't be compared by the searched for attribute then they will be compared by call numbers.
All media will implement the MediaInfo interface which will have the abstract method displayInfo() which will return a string of the media's information.

There will be a libraryDataBase class which will contain:

- A list of all the registered users
- A list of all books within the library
- A map with users as the keys and lists of what books they borrow as the values to those keys

TextIO will be used here to export all the media in the library and to import new media to the library.

Challenges:

When testing any methods that required comparing any objects that are a part of the media or user class families I found that the resultant call number/id number would differ from that of the expected number. The reason behind this is that these numbers are given a value using a static variable within the classes upon creation; meaning that all objects share this number between them. The solution to this problem within a single test case was to change the value of the call/id number of the expected class value to that of what was actually expected, however whenever a new test case was needed it would still share the same id counters making accounting for what the actual id of each object really annoying. After a bit of research I discovered that Junit 5 has the @AfterEach annotation which will make the method annotated with it run after every test method. I used it with a method that reset the counts of the call/id numbers that way I wouldn't have to keep track of those values between different tests. I also had difficulties testing the export and import methods using unit testing when it came to creating files, but seeing as we never looked at testing external files I went under the assumption that we were not meant to test them. (the instructions for this project said "Unit testing for user defined methods (if applicable)" which is the basis for that assumption.)

I also had to change a lot of the method outputs throughout the project because they were originally values that would lead to a Library class that would rely heavily on values outside of the class, which didn't seem very practical to me in the long term.

Learning Outcomes:

I learnt how to apply unit testing for objects with static variables and got to apply my knowledge of different data structures on my own giving me a better understanding of how they work compared to before this project.