Large Assignment #2
Due: Friday, October 4, 2024 by 11:59 PM

Reminder About Academic Integrity: If you violate the academic integrity policy, the recommended sanction will be that you fail the course. Violations of the academic integrity policy include but are not limited to: using ChatGPT or AI software to help you write or debug code, looking at someone else's solution to the assignments, sharing your solution to the assignment, and posting any part of the assignment online.

Part 0. Note about submission.
This assignment will be submitted on `lectura` using `turnin`. So the first thing you should do, and you should do this *right away*, is verify that you have a current username and password to log in to lectura. If you do not, contact lab@cs.arizona.edu for help. Also, if you are unfamiliar with lectura or need a refresher, review this document.

You are expected to work in pairs for this project *unless you get special permission from me (Lotz) to work alone.* (You would have to make a good argument for it.)

Only one of you should submit the files on lectura, but you need to make sure that every file includes both names and usernames (as a block comment in the java files).

Part 1. Project Overview
In this project, you must design and implement a system for keeping track of a personal library. The program that will actually be run (the main) is the user interface for this library. The rest of the classes are for you to design and implement.
The user interface should be implemented in a class called `MyLibrary.java` and should contain a main method that runs a text-based user interface where the user can enter commands to get or add information to the library. Below is a description of the commands and functionality that must be provided.

Command	**Description – what should happen when this command is entered**
search	● allow the user to choose a method for searching: by title, author, or rating (should be a number from 1 to 5) ● ask for the appropriate information ● retrieve the Book's information or a list of Book's that fit the information (e.g. searching by author or rating could yield multiple results)
addBook	● ask the user for appropriate information about the book that should be added ● add the book to the collection
setToRead	● ask the user for the book they want to update

	• set that book to read • Note: It is impossible to *unread* a book.
rate	• ask the user what book they want to rate • ask for the rating • set the rating for that book • Note: ratings should be updatable
getBooks	• retrieve and display a list of books according to the following options (which should be given to the user to choose from) ○ all books sorted by title ○ all books sorted by author ○ all books that have been read (use a default sorting method – your choice) ○ all books that have not been read (use a default sorting method – your choice)
suggestRead	retrieve a random unread book from the library
addBooks	• ask the user for a file name • read the book list from the file and add them to the library • a sample file is provided so you know the appropriate format

Part 2. Coding Requirements
The description above gives the kinds of interactions the user should be able to have with the library. That does not mean that all the actual work is done in that class. That class should primarily be used for the user interactions and the rest of the work should be abstracted to other appropriate classes.

The coding requirements for the project are:
- provide a working user interface that meets the requirements above and
- appropriate, well-documented classes to make the whole system work
- good encapsulation and design practices

This is quite open-ended, but you are expected to follow good design principles and coding practices that have been covered in class through Thursday, September 19, 2024.

You are allowed to use any appropriate classes and interfaces from java libraries, but you need to make sure you are using them well.

You are expected to use appropriate design patterns and avoid antipatterns discussed in class.

You are expected to design and implement well-organized classes and apply the material we've covered in class.

There should be no main methods except for the one in MyLibrary.java.

Part 3. Documentation

There are several requirements for the documentation:

- each class should have a block comment at the top explaining how encapsulation is maintained
- the code should be commented with helpful comments to help someone familiar with java understand what it is doing; major design decisions should be included in comments as appropriate
- you should use appropriate javadoc tags that have been discussed in class
- you should provide two UML diagrams:
    - one for your Book class (Hint: You should have a Book class) with all of its fields and methods
    - one class diagram that shows all classes and interactions – the individual classes in this diagram don't necessarily need to show all the fields and methods – just the ones that matter for the interactions (see the examples in the slides for this)
- you should provide a well-organized text file called `doc.txt` file with a brief explanation of your design choices including
    - the classes
    - any interfaces involved
    - any library classes involved
    - any data structures you used
    - any design patterns you used
- All files need to include both your names and usernames at the top (in a block comment if it's a java file.)

Part 4. Grading

- If your code does not compile and run on lectura (without errors), you will receive an automatic 0.
- Assuming it does compile and run on lectura, here is the grading breakdown:

Criteria	**Points**
MyLibrary.java: ● runs and uses the correct commands, and works as specified. ● does not do work that should be done elsewhere – it should primarily be the UI ● provides all the required functionality	20
All the classes are well-designed, do not overlap, and illustrate good encapsulation. The overall design follows appropriate design patterns. Appropriate input validation.	10

The implementation uses appropriate library classes, interfaces, and data structures, and uses them well.	10
The documentation meets all the requirements and is clear and helpful.	10

Part 5. Submission.
As a reminder, if you are unfamiliar with lectura or need a refresher, take a look at the document linked above, which explains how to log in to lectura and transfer files to lectura.

Important Note: Transferring files to lectura is not enough. When you transfer files, they go to a local drive that *we do not have access to*. Once you have transferred your files, you should run the code to make sure it all works. Then you need to run the following command:

```
turnin csc335la2 <all files for submission>
```

Upon successful submission, you should see this message:

```
Turning in:
      MyLibrary.java – ok
      … (similar message for any other files you submit)
All done.
```

Important Note: Only properly submitted assignments will be graded. No assignments will be accepted by email or any other way except as described in this handout. If you are worried about your submission, feel free to ask us and we can check that it got into the right folder. Make very sure that you use the correct command. If you do not, we will not get your submission and even if we end up accepting it eventually, it will count as a resubmission (with the deduction and everything.)