Homework - Software Design

In this assignment, you are asked to implement a program that allows a user to search through a collection of books and find books written by a specific author or the number of books written in a specified year. You will implement this program using the three-tier architecture discussed in recent lessons.

In completing this assignment, you will:

- Apply what you have learned about software architecture, software design, and internal quality
- Consider the tradeoffs of software design, especially the input and output of methods, as it relates to software quality

Getting Started

Download the following files:

- **PresentationTier.java:** the class that represents the “presentation tier,” which handles interaction with the user
- **LogicTier.java:** the class that represents the “logic tier,” which includes processing and performing computations on data
- **DataTier.java:** the class that represents the “data tier,” which handles reading data from some source and making it available to the rest of the program
- **Book.java:** represents a single book in our program
- **Main.java:** responsible for instantiating the objects and setting up their relations

Also download **books.txt**, which contains a list of popular books in tab-separated format. Each row represents one book and has the following format:

[*title*][*tab*][*author*][*tab*][*year*]

You may also use your own list of books for testing your application as you implement it.

Activity

This program is perhaps not difficult to implement but the challenge here is applying good software design principles (modularity, functional independence, and abstraction) in order to create code that has high internal quality: changeability, stability, understandability, and testability.

In particular, you need to apply the three-tier architecture, which we will simplify by saying that each tier only has one class. Your goal is to put the right functionality into the right tiers/classes, decide on method inputs and outputs, and then implement the code.

In practice, there may be some flexibility as to whether certain functionality goes into a single method or multiple methods, what the names of the methods would be, etc. And you generally would have multiple classes per tier, and not just a single class.

In this case the application you implement must have the following methods:

- **findBookTitlesByAuthor:** for a given name, search through all of the books and return the titles of those books whose author name includes the input name.
- **findNumberOfBooksInYear:** for a given year, search through all of the books and return the number of books published in that year
- **getAllBooks:** read the data file containing information about the books, create Book objects for each, and then return the Book objects.
- **showBookTitlesByAuthor:** using the command-line (i.e., reading from System.in), ask the user to enter part or all of an author's name, then display (using System.out) the titles of those books whose author name includes the input name.
- **showNumberOfBooksInYear:** using the command-line (i.e., reading from System.in), ask the user to enter a year, then display (using System.out) the number of books published in that year

Both *findBookTitlesByAuthor* and *showBookTitlesByAuthor* should look for partial matches and should ignore case, i.e. be case-insensitive. The titles of the books that are returned/displayed should be sorted based on the year they were published in non-descending order; if two or more books have the same publication year, those books should be sorted alphabetically.

For each of the five methods, you need to decide:

- In which class (*PresentationTier*, *LogicTier*, or *DataTier*) should it be implemented?
- What should its input parameter(s) be?
- What should its return type be?

In addition to considering modularity and functional independence in applying the three-tier architecture, be sure to also consider abstraction when designing each method: the caller of a method should be able to use it knowing only what it does, and not the details of how it works.

Once you have completed your design, implement each of the five methods according to the specifications above.

In all cases, you can handle error conditions in any way you choose, e.g. if the user enters a year that is non-numeric in *showBookTitlesByAuthor*, if the name specified in *findBookTitlesByAuthor* is null, etc. Just be sure your code works correctly for "normal" inputs.

Likewise, it is okay for *getAllBooks* to assume that the input file exists and is well-formatted.

You can use any prompt you choose for asking for input in *showBookTitlesByAuthor* and *showNumberOfBooksInYear*, and can likewise choose any output formatting you like.

Finally, implement *PresentationTier.start* so that it asks the user which feature they would like to use and invokes the appropriate methods in the appropriate classes. Once the output has been displayed, the program should terminate. As above, you can handle error cases in any way that you deem appropriate.

Please be sure that your code adheres to the following guidelines and restrictions:

- Each method listed above must go into exactly one of the three classes (*PresentationTier*, *LogicTier*, or *DataTier*).
- Do not add any other classes, though you may add additional fields and methods as necessary.
- Do not change the *Book* class or *Main* class.
- All methods must be public.

Helpful Hints

As you have probably already noticed, the examples discussed in the lessons are very similar to this program. Use those as guidelines and consider where the functionality went in those examples, and what inputs/outputs each method had.

The *getAllBooks* method needs to read from the input file. If you do not have prior experience writing Java code to read a text file, feel free to look online for help. There is good documentation at:

https://docs.oracle.com/javase/tutorial/essential/io/file.html

and good tutorials elsewhere online