P03 Library Management System

Overview

This assignment focuses on building a simple library management system in Java, where you will create a Library class to manage a collection of Book objects. The system allows for adding, removing, updating, and searching for books. It also includes functionalities to test the library's operations and demonstrate its usage, giving you hands-on experience with object-oriented programming concepts such as encapsulation and array manipulation.

Grading Rubric

5 points	**Pre-assignment Quiz**: accessible through Canvas until 11:59PM on **09/29**.
+2.5 points	**Bonus Points**: students whose ***final*** submission to Gradescope has a timestamp earlier than **5:00 PM CDT** on **WED 10/02** will receive an additional 2.5 points toward this assignment, up to a maximum total of 50 points.
25 points	**Immediate Automated Tests**: accessible by submission to Gradescope. You will receive feedback from these tests *before* the submission deadline and may make changes to your code in order to pass these tests. Passing all immediate automated tests does **not** guarantee full credit for the assignment.
10 points	**Additional Automated Tests**: these will also run on submission to Gradescope, but you will not receive feedback from these tests until after the submission deadline.
10 points	**Manual Grading Feedback**: TAs or graders will manually review your code, focusing on algorithms, use of programming constructs, and style/readability.
50 points	**POINTS_POSSIBLE**

Learning Objectives

After completing this assignment, you should be able to:

- **Implement** a custom data type in Java by following a given design structure with private data fields and public methods
- **Demonstrate** the utility of defining a constructor and accessor/mutator methods by validating the implementation in a separate tester class
- **Identify** and **use** appropriate methods from Java's ArrayList class to store objects of your custom data type

Additional Assignment Requirements and Notes

Keep in mind:

- Pair programming is **NOT ALLOWED** for this assignment.
- The ONLY external libraries you may use in your program are:
  ```
  java.util.ArrayList
  java.util.Random
  ```
  Use of *any* other packages (outside of java.lang) ***in submitted files*** is NOT permitted.
- You are allowed to define any **local** variables you may need to implement the methods in this specification (inside methods). **You are NOT allowed to define any additional instance or static variables or constants beyond those specified in the write-up**.
- You are allowed to define additional **private** helper methods in any class.
- Only the Driver class and Tester class may contain a main method.
- All classes and methods must have their own Javadoc-style method header comments in accordance with the CS 300 Course Style Guide.
- Any source code provided in this specification may be included verbatim in your program without attribution.
- **Run your program locally before you submit to Gradescope**. If it doesn't work on your computer, *it will not work on Gradescope*.

Need More Help?

Check out the resources here: https://canvas.wisc.edu/courses/427315/pages/resources

CS 300 Assignment Requirements

You are responsible for following the requirements listed on both of these pages on all CS 300 assignments, whether you've read them recently or not. Take a moment to review them if it's been a while:

- Academic Conduct Expectations and Advice, which addresses such questions as:
  - How much can you talk to your classmates?
  - How much can you look up on the internet?
  - What do I do about hardware problems?
  - and more!

- Course Style Guide, which addresses such questions as:
  - What should my source code look like?
  - How much should I comment?
  - and more!

Getting Started

1. Create a new project in Eclipse, called something like **P03 Library Management System**
   1. Ensure this project uses Java 17. Select "JavaSE-17" under "Use an execution environment JRE" in the New Java Project dialog box.
   2. Do **NOT** create a project-specific package; **use the default package**.

2. Create two (2) Java source files within that project's src folder:
   1. **Book.java** (does **NOT** include a main method)
   2. **Library.java** (does **NOT** include a main method)

3. Download three (3) Java source files from the assignment page on Canvas:
   1. **LibraryDriver.java** (includes a main method)
   2. **LibraryTester.java** (includes a main method)

You do not need to modify anything or submit the **LibraryDriver.java**. It is only for your own benefit.

If you add the downloaded files to the project right away, you'll probably see quite a few errors, because they assume there are methods in your source files that don't exist yet, but don't worry! You'll fix that shortly.

Implementation Requirements Overview

The Javadocs for the classes you will write in their entirety are found here:

- Book
- Library

The Book and Library define a simple Library Management System. The Library contains a class that manages a collection of books, allowing for functionality such as adding, removing, finding, and updating books. It also includes methods to retrieve the total number of books and print all books in the library. The Book file defines a Book class, which represents a book with a title and an author, and includes accessor and mutator methods to retrieve and modify these attributes. Together, these classes provide a basic framework for managing a library's book collection.

In this program, you are required to complete both of these instantiable classes, as well as **1 tester file** (for better code organization). We've provided a couple of example tests to get you started in each of the tester files, but be sure to complete them so that your code passes all tests by the time you make your final submission to Gradescope.

The Book Class *and its tester methods*

In this step we are going to create our first class called ***Book***. Please refer to the Book JavaDoc for the correct field and method names of this class.

1. Define private fields according to the Book JavaDoc:
    - The **Book** class should have five (5) ***private*** fields:
        - `String title`
        - `String author`
        - `int yearOfPublication`
        - `String publisher`
        - `int numberOfPages`
2. Each of these fields should be ***private***, meaning they can only be accessed or modified through methods, and ***non-static***, meaning that each Book object has its own value for each field.
3. Create a constructor:
    - The constructor should take 5 parameters (one for each field) and initialize the corresponding fields. For validating our arguments, we need to apply some validity checks. If the `yearOfPublication` is negative or more than the current year, an **IllegalArgumentException** should be thrown with an appropriate message. If the

number `numberOfPages`, an **IllegalArgumentException** should be also thrown with the appropriate message.

4. Implement the accessor methods:
    - For each field, write a public method that returns the value of that field.
    - For example, `getTitle()` should return the value of `title`.
5. This is a good point to start implementing your tester methods:
    - In the LibraryTester.java, start with the constructor and accessor method testers:
    - Create a Book object using the constructor and test if the values you provided are returned correctly by the accessor methods.
    - We have provided for help the first tester method `testGetTitle()`for `getTitle()` to help you get started.
6. Implement the tester methods for the mutators:
    - After creating a Book object, use the mutator methods to modify some fields (for example, change the Author name and check if it was actually changed).
    - We have provided for help the tester method `testSetTitle()`for `setTitle()` to help you get started.
7. Implement the mutator methods:
    - For each field, write a public method that allows the field to be modified.
    - For example, `setTitle(String newTitle)` allows you to change the value of `title`.
    - Again, we need to implement some validity checks before passing the arguments to the fields. We need to make sure that there is no negative year or a future year (more than 2024) passed as an argument for `yearOfPublication`, otherwise an **IllegalArgumentException** should be thrown. Negative values should also be checked for `numberOfPages` and an **IllegalArgumentException** should be thrown.

Tips:

- Pay attention to naming conventions: method names and fields are in lowerCamelCase, and the class name is in UpperCamelCase.
- Keep the tests simple; you only need to check if the accessors and mutators are functioning correctly.

The Library Class *and its tester methods*

In this step we are going to create our second class called ***Library***. Please refer to the Library JavaDoc for the correct field and method names of this class.

1. Create an ArrayList to store the books:
    - In Library.java, create a private field called `books` which is an ArrayList of Book objects. This will be used to store all books in the library.
2. Implement `getTotalBooks()` and `getAllBooks()`. To help you get started we have provided the tester method `testGetTotalBooks()`.

- This method should return the total number of books in the library using books.size().

3. Implement `testAddMultipleBooks()` in the LibraryTester.java. To help you get started we have provided the tester method `testAddBook()`:
   - In LibraryTester.java, create a Library object and add a few Book objects to it **in order of the year they were published**, from the older to the newer.
   - Use `getTotalBooks()` to verify that the books were added correctly.
4. Implement `addBook(Book book)` in the Library.java:
   - This method should add a new book to the books list ordered by the year the book was published, from the older to the newer.
5. Implement `testRemoveOneOfManyBooks()` in the LibraryTester.java. We have provided `testRemoveBookByTitle()` for help:
   - Add a few books to the library, remove one by title, and check if the total number of books decreased and the arraylist is as expected after removing the book.
6. Implement `removeBookByTitle(String title)` in the Library.java:
   - This method should remove the first book that matches the given title.
7. Implement `testFindBooksByAuthor()` and `testFindBooksByMultipleAuthors()` in the LibraryTester.java :
   - Add multiple books by different authors and use this method to verify that it correctly returns the list of books by the specified author or authors.
8. Implement `findBooksByAuthor(String author)` in the Library.java:
   - This method should return an ArrayLlist of books written by the given author.
9. Implement `testUpdateBookTitle()` and `testUpdateMultipleBookTitles()` in the LibraryTester.java:
   - Add a book, update its title and return true if that was done successfully or false otherwise. Follow the same logic for multiple books.
10. Implement `updateBookTitle(String oldTitle, String newTitle)` in the Library.java:
    - This method should find a book by its current title and update it to the new title.
11. Implement `testUpdateBookAuthor()` and `testUpdateMultipleBookAuthors()` in the LibraryTester.java:
    - Add a book, update its author and return true if that was done successfully or false otherwise. Follow the same logic for multiple books.
12. Implement `updateBookAuthor(String title, String newAuthor)` in the Library.java:
    - This method should find a book by its current title and update the author.
13. Implement `printAllBooks()`:
    - This method should print the title and author of every book in the library.

Tips:

- Use ArrayList methods such as `add()`, `remove()`, and `size()` to manage the collection of books.

- For searching and updating, iterate over the list and use `equalsIgnoreCase()` to compare titles and authors.
- Use the `compareBooks()` method to compare the expected and actual list of books in your tests.

Assignment Submission

Hooray, you've finished this CS 300 programming assignment!

Once you're satisfied with your work, both in terms of adherence to this specification and the academic conduct and style guide requirements, submit your source code through Gradescope.

For full credit, please submit ONLY the following files (source code, *not* .class files):

- **Book**.java
- **Library**.java
- **LibraryTester**.java

Do ***NOT*** submit LibraryDriver.java for grading; this class is for your benefit only and may result in you losing points if it is submitted.

Additionally, if you used generative AI at any point during your development, ***you must include screenshots*** showing your FULL interaction with the tool(s).

Your score for this assignment will be based on the submission marked "**active**" prior to the deadline. You may select which submission to mark active at any time, but by default, this will be your most recent submission.

Students whose final submission (which must pass ALL immediate tests) is made before 5pm on the Wednesday before the due date will receive an additional 5% bonus toward this assignment. Submissions made after this time are NOT eligible for this bonus, but you may continue to make submissions until 10:00PM Central Time on the due date with no penalty.

Copyright notice