CS 203 HW #7 — Online Book Store

Due Wednesday, March 18

In this assignment, you will implement a few classes needed for an online bookstore software system.

Important: Some of the instructions in this assignment are deliberately vague. In these cases, you are expected to consider the scenario and make appropriate choices.

Specification

To complete this part of the assignment, you must complete three classes: `Customer`, `Book`, and `Main`. You must implement these three classes exactly as described in this document. *All the code you write should have complete and appropriate comments.*

Customer Class

An instance of this class contains all the information about a single customer. It should have the following attributes stored in its instance variables:

- customer name
- customer account ID
- customer address
- customer credit card number
- customer password
- an array of `Book` objects that this customer buys. To be clear, this is not an array of Strings with just the titles of the books.
- a count of how many books this customer has bought. (This will tell you how many cells in the Book array are currently being used.)

The `Customer` class should support the following actions (methods):

- A `toString()` method that returns a summary of the customer's information, but does not include the credit card number or password.
- A getter method to get the customer's address
- A getter method to get the customer's ID.
- A getter method to get the customer's name.
- A getter method to get the credit card number.
- A getter method to get the password.
- A setter method to set/change the password.
- A setter method to set/change the address
- A setter method to set/change the credit card number.
- A constructor for your `Customer` class. The constructor should set the customer's name and address and ID. The constructor should also create the array of `Book` objects as an empty array and set the number of purchases to zero. You can assume that no customer ever buys more than five hundred books, so it's safe to use an array of length 500. You should not need to add any parameters to the constructor to initialize the Book array or for the number of book purchases.

- A method to add a given `Book` to the array of books that the customer has bought. This method should also update the `numPurchases` instance variable appropriately. The customer can buy the same book more than once. The customer can also buy one or more copies of the same book at one time. Add one item to `Book` array for each copy.

Book Class

An instance of this class contains all information about a book that the store carries.

It should have the following attributes stored in its instance variables:

- Book ID number
- Book title
- Book author
- Book publisher
- Year of publication
- A count of how many copies of this book the store currently has in stock
- A count of how many copies of this book the store has sold so far
- Book price

The `Book` class should support the following actions (methods):

- A `toString()` method that returns the summary of the book's information.
- A getter for each instance variable
- A method that records the sale of this book to a given `Customer`. This method should also take the number of requested copies as a parameter. The inventory information should be updated, *and* the `Customer`'s book array should also be updated.
- If there are no copies of this book in stock, do not change the values of the instance variables, and print an appropriate error message.
- If the number of copies requested is greater than the number available, do not change the values of the instance variables, and print an appropriate error message.
- If the number of books to buy is negative, do not change the values of the instance variables, and print an appropriate error message.
- A constructor that sets the ID number, amount in stock, title, author, publication year, publisher, and price according to arguments required by the constructor. The number sold should be set to an appropriate initial value.
- A method that is called when the store (re)stocks the book. The method should increase the number of available copies of the book by the amount given. Print an appropriate error message to the console if the given number of books is less than one.

Main Class

This class will be used to test the code written for the `Book` and `Customer` classes. It should contain only a single, static `main` method that creates multiple instances of both `Book` and `Customer` classes with made-up values of your choice. The main method should also call various methods on these objects to provide what you believe is a complete test of <u>all</u> the functionality therein. If you wish, you may add a few more "get" methods for Book and Customer classes to support this.

UML Diagram

Create a UML diagram that accurately reflects your completed source code. Don't forget about access (public/private) and the relationships between the classes. This diagram must be an electronic image file or PDF file. The Engineering build contains two different software tools capable of creating UML diagrams: Dia and Visio. Both of these tools can export the diagram you draw as an image. You can also use another program of your choice (Word, PowerPoint, etc), or you can hand draw the diagram and photograph or scan the result to acquire a file you can turn in. Your diagram will be graded on both accuracy and overall neatness.

Additional Enrichment

You are welcome to add additional functionality to your program as you desire. Some suggestions:

- Create a new class called Accounting that tracks the sales of the bookstore. Accounting should track the store current bank balance. The `Customer` and `Book` classes should use `Accounting` in some of their methods.
- Create a GUI using Java's AWT and Swing libraries that allows the user to use your Book and Customer classes.

Code Quality

A good computer program not only performs correctly, it is also easy to read and understand:

- A comment at the top of the program includes the name of the program, a brief statement of its purpose, your name, and the date that you finished the program.
- The comment header should also include a list of known bugs or deficiencies, or a statement that the program has no known deficiencies.
- Variables have names that indicate the meaning of the values they hold and use conventional case.
- Code is indented consistently to show the program's structure.
- The body of if and else clauses are enclosed in braces and indented consistently, even if they consist of only a single line of code.
- Opening braces are placed consistently, either at the end of a statement or on a line by itself directly after it.
- Within the code, major tasks are separated by a blank line and prefaced by one or more single-line comments identifying the task.
- Methods are separated by blank lines and prefaced by a multi-line comment describing what they do and what their parameters mean.
- Very long statements (such as long print statements or complex boolean expressions) are broken across lines and indented to show their structure.
- Now that you are familiar with loops and methods, your code should not contain redundant or repeated sections.
- Your program does not contain extraneous or commented out code. It does not contain out-of-date or irrelevant comments.
- When a constant value is used repeatedly in your program, declare a `final` variable with a descriptive name to contain that value. Use it instead of a literal.

Logistics and Hints

- No starter code for this assignment. Create your own classes.
- **Hint:** Use the `Main` class to your advantage by writing it bit-by-bit as you write the other two classes. For example, each time you write a new method in `Book` or `Cus-tomer`, add some code to your `Main` that calls that method. Don't wait until the other two classes are finished.

Turning in this Assignment

You are responsible for turning in your homework assignments properly.

- Be sure your name is in the comment header at the top of your `.java` files.
- Remember to include comments to remind yourself and your reader the functionality of the different parts of your program.
- If you did any of the Additional Enrichment (above), carefully document this functionality in the comment header at the top of your `.java` files.
- Zip up your <u>entire</u> BlueJ project folder and submit the zip file via Moodle.