ITSC 1213 – Project 1 – Bookstore Management System

The main idea behind this activity is to create a system that allows a bookstore owner to keep track of their inventory and members.

Preliminaries

The owner of a bookstore has asked for your help designing a system that allows them to keep track of the store's inventory and members. The owner provides you with the following details about what the store offers:

- The store sells three types of products: books, CDs, and DVDs.
- The store offers two types of memberships to customers: regular memberships and premium memberships. The regular membership is free, while the premium members pay a fee every month.
- The store keeps track of payment method for their premium members and whether the monthly fee is due or has been paid.
- The system should keep track of the members and how much money each has spent at the store regardless if they are regular or premium members.
- The system also keeps track of the inventory of each product.

Your design of this system must include 3 distinct functionality. A function, in this context, are abstract ideas like "register a new premium member", or "complete a purchase". While in many programming contexts a specific function refers to what we refer to as methods in Java, here function and method are NOT synonymous. Functions here refer to the holistic application/program/system that we are creating for managing a bookstore. To implement your functions it may take one or more methods and will likely involve 1 or more of your classes.

One of these functions MUST be to complete a purchase. In order for that to happen the user should be able to select one or more items, and those items should be deducted from the inventory after the purchase is completed. The other two functions are entirely your choice.

As you are thinking about these features, think of any application or a website you use to purchase something online. What functionality, you as the user, are able to complete using this application? Answering this question should give you plenty of options to choose from.

You are welcome to program or map out additional functions. These do NOT necessarily need to be implemented, but can be included using empty methods (stub methods) to make your system look more complete.

Important Considerations

- We want you to take an Object-Oriented approach in this project. That means you will be creating a good number of classes that will all interact with each other, rather than having everything contained within one class and one main method. The FastFoodKitchen lab is a good example of this kind of design. In the same way that the FastFoodKitchen contained a list of BurgerOrders, the Bookstore may contain a list of members, or a list of books that can be purchased.
- It is very important to consider what attributes a book would need, that a member would need, etc. This is especially important when you consider functionalities like purchasing.. which of these attributes do you need to access during a purchase, and if you need that information where/how can you find it?

Part A – UML Diagram and Feature Description (5 points)

Examine the description of the system above. Using one of the UML Diagramming tools (See Course Resources), design the different classes of your system. Think of the various things (objects) that you need to represent in the system and what properties they need to have and actions to perform. For now we don't need to worry about relationships between the classes. We will add that in a later project.

Your diagram must show:

- All of the classes, fields, and methods. Remember that you will need methods that
  accomplish appropriate functionality, not just getters and setters.
- You must include visibility (i.e., public or private), data types, return types, and any
  other pertinent information.
- Show the data types/structures (e.g., Strings, arrays or ArrayLists, Book,...) that your program will use to hold the important data.

In addition, you should also submit a document that describes the 3 functions your system will implement. This description should include any arguments, returns, or fields needed in order to make this functionality work. Save the description in a file (file format can be text, pdf or doc) with the name *BookstoreDesign*.

Part B – Implementation and Test Harness (20 points)

- Create your project in Netbeans, using the naming format [your last name][your first name]Project1. For instance, mine would be "LastFirstProject1"
    - Create all of the classes you mapped out in the UML diagram from Part A
    - Add all the fields and methods described in your UML, including getters/setters/constructors.
    - Your project and its contents should match your UML exactly. If you name a method in the UML, I will expect to see it in the Netbeans project.

Note that you can complete this step in its entirety before adding any functionality. You can use empty stub methods to start.

- Now you need to add enough code and logic to your methods to implement the “complete a purchase” functionality and create a test harness so this function can be executed.

In order to test your functionality, you should create a Test Harness class. This will be a class whose function is to create an object that represents the book store, and use Scanner objects to take input and direct the user through the system. For clues as to how this can be accomplished, look back at labs like the BurgerOrder project. Text menus should be presented to the user that will allow them to pick from options such as "Make a purchase" or "Create a new member". For options that are not implemented you can simply print a message and redisplay the menu options. This should be the only class that includes a main method.

Example of a book store system output/interaction

Coding Style – (5 points)

This grade is awarded for proper coding styles. This includes:

- Appropriate prompts and informative output
- Appropriate method, variable and object names
- Proper indentation
- Good commenting (explains what code is doing)
- Well-organized, elegant solution

Submission Requirements:

1. Submit the UML and functionality description from Part A.
2. Submit your NetBeans project export (**lastNameFirstNameProject1.zip**) that includes your program to fulfill the requirements.
3. Submit a PDF document with the name **project1Info.pdf** that includes the following assignment information:
    1. Explanation of any special or additional features you added, if any.
    2. If your solution to this problem is incomplete, you may explain the status as of your submission and describe any issues that kept you from completing the assignment..
    3. Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?
    4. Did you learn anything that will help you perform better on future assignments?