COURSE PROJECT: E-Commerce Application in Java

PREAMBLE

Project Description

This course project will allow you to get practice developing a complete Java application. You will work to develop a prototype for a virtual store interface in Java only - it will not be web-based. In the store, you will be able to sell any product of your choice, but it must be specialized. That means, you need to choose a product and focus on selling varieties of that product, for example, varieties of cars, computers, plants, or any other product you would like.

The application will provide an interface for a user to shop for products from you store, which, as we already mention is specialized in something you selected. The user should be able to: (1) browse the products, (2) add products to a virtual cart, and eventually, (3) checkout. At first, the interaction will be textual, i.e., though the console. But later, a GUI will be added to complete the experience.

The system will be implemented in Java and will have the following classes: (1) *StoreManager*, (2) *StoreView*, (3) *Inventory*, (4) *ShoppingCart*, (5) *Product*, and (6) several other classes of your choosing (these will be the classes modelling the products you sell). A brief overview of these classes is presented below. More detailed requirements will be outlined in the relevant milestones.

StoreManager: This class will be the controller of the system. *StoreManager* will (1) manage an *Inventory,* (2) maintain *ShoppingCarts* for users and (3) respond to queries/prompts/inputs received from the user via the *StoreView* class.

StoreView: This class will be the starting point of your application. Ideally, *StoreManager* and *StoreView* would be able to exist independently. However, communication in this project will be done entirely between objects (no web, sockets, multi-threading, etc.). Therefore, the *StoreView* will be responsible for initializing the entire system. *StoreView* will allow the user to interact with the store via an interface. This interface will be textual at first (through the console), and later refactored as a GUI.

Inventory: This will be the class responsible for keeping track and maintaining information relating to the products available in the store. The *Inventory* will be managed by the *StoreManager*.

ShoppingCart: A *ShoppingCart* will be maintained for any user using the store interface (likely just one; but that could be subject to change!). A user will be able to view the contents of their cart and modify them appropriately. Interactions between the user and their cart will be moderated via the *StoreManager*. That means, the *StoreView* class cannot actually see any *ShoppingCart*. The *StoreManager* will create and manage *ShoppingCart* objects for users as requested by the *StoreView* class.

Product: This class models the base behavior for all products. You will be defining your own classes that will extend this base Product class.

General Submission Requirements

For each milestone, you need to submit:

1. The specified milestone deliverables.
2. From milestone 2 onwards, complete Javadoc documentation comments for all of your classes.
3. From milestone 2 onwards, an updated UML class diagram of your software.
4. A report. The report must contain:
    4.1. A Change Log explaining design choices and specifying any changes you made in the design from the previous milestone. That means, in Milestone 2, you need to explain and justify the changes made from the design in Milestone 1. In Milestone 3, you need to explain and justify the changes made from the design in Milestone 2, and so on.
    4.2. Answers to all the questions in the milestone. Questions will be based on the material covered in the lectures. They will require you to think about the decisions you made for each deliverable. There will be different types of questions, mostly open-ended and interview-type as these will likely be the type of questions you will encounter in the future and when looking for a job as a software developer.

General submission rules:

1. Each file you submit must contain your name and student ID. Otherwise, a 50% penalty will be applied.
2. The name of the files must be exactly as stated in the deliverable. Otherwise, a 50% penalty will be applied.
3. If there are compilation errors in your java code, a 50% penalty will be applied.