Brooke Slampak

CS 255

**Module Four Assignment**

* What are the different functions of the online storefront? How are they represented in this type of model?

The online storefront system has several features, each represented through different classes and functions in the diagram. Customers can register for an account, log in, and update their profile through the Customer class, which gets its login features from the User class. Once logged in, they can view and manage their shopping cart by adding items, changing quantities, and checking out using functions like addCartItem() and checkOut(). When ready, customers place orders using the Order class, which keeps track of things like order status and shipping. The Shipping Info class handles how the order gets delivered, including the shipping method and cost.

There are also tools to manage order details and calculate prices using the Order Details class. On the admin side, the Administrator class, which also inherits from User, can update the product catalog using updateCatalog(). The diagram also shows how everything connects. The model covers all the basics you’d expect from an online store, from browsing and buying to managing orders and shipping.

* What are the different classes of “users” represented by this object model? What are the associations between these classes?

In this object model, there are a couple of different types of users. There's the User class, which includes basic things like user ID, password, login status, and when the user registered. Both customers and admins are built on top of this base class, they just add their own specific details and actions. The verifyLogin() method in User handles login checks for everyone.

From there, the Customer class adds things like name, address, email, credit card info, and shipping info. It also includes functions like register(), login(), and updateProfile() to let users manage their own accounts. There’s also the Administrator class, which also comes from User, but it’s focused on store management. It has its own attributes like admin name and email and a function called updateCatalog() for managing products. While both types of users share some core features, they’re set up to do totally different things in the system.

* How would the objects “use” their respective variables and functions?

In this model, each object would "use" its variables and functions to carry out specific tasks based on its role. For example, a Customer object would hold details like their name, email, and shipping address, and use functions like register() to sign up, login() to get into their account, or updateProfile() to make changes to their info. They’d also use a Shopping Cart object to add items, change quantities, or check out using methods like addCartItem() and checkOut().

An Administrator object works a bit differently. It still has basic login info from the User class, but its main job is to manage the store. So, it would use the updateCatalog() function to add or edit products. Both Customer and Administrator objects use the verifyLogin() function, which is inherited from User, to check if their login info is correct. Each object uses its own variables to keep track of its state and calls its functions when it needs to perform actions within the system.

* Does this object model capture all of Hamp Crafts’ desired functionality? Why or why not?

The object model covers the main features Hamp Crafts would need for an online store, such as signing up, logging in, managing a shopping cart, placing orders, adding shipping info, and letting admins update the product catalog. That said, it might not cover every feature Hamp Crafts may want. There’s nothing in the model for browsing or searching products, leaving reviews, tracking inventory, or processing payments beyond just storing credit card information. It also doesn’t show aspects like order history, customer messages, promo codes, or wishlists. While it’s a sold model for a baseline online store, a few more features might need to be added to fully match what the business is looking for.

* The above diagram uses a solid diamond shape to represent a form of aggregation. What type of aggregation does this represent? What does it imply about the relationship between the classes? Why is a solid diamond the appropriate choice here?

The solid diamond shape in the diagram represents **composition**. Composition simply means that one thing completely "owns" another. If the main object is removed, the thing it owns is removed, too. Like how if a house is destroyed, its rooms are also destroyed. For example, the Customer class has a composition relationship with the Shopping Cart, meaning a shopping cart only exists if there’s a customer using it. Same with Order and Order Details, if the order is deleted, the details tied to that order are also gone. It shows a strong connection where one part can't exist without the other.

A solid diamond is appropriate in the diagram because it shows the relationship isn’t just a loose connection, it’s a bond where the components are reliant on the existence of each other. Things like the shopping cart or order details don’t make sense on their own, they only exist as part of a customer or an order. Using composition like this helps keep the system organized and makes it easier to manage when components of the system are created or deleted.

* How well do you think a process model describes the system? What information does it make easier to understand? What aspects of the system are more difficult to understand or are not represented?

A **process model** does a good job of showing **how a system works step by step** because it lays out the flow of interactions between parts of the system. It’s helpful for visualizing the **logic and flow of operations**, which is useful for planning and communicating how the system should behave. However, a process model doesn’t show everything. It doesn’t show the **structure of the system**, like how data is stored and organized. You also don’t get much insight into **object lifecycles**. So, while it’s great for understanding behavior and workflows, it falls short when it comes to representing the **static design and internal architecture** of the system.

* How well do you think an object model describes the system? What information does it make easier to understand? What aspects of the system are more difficult to understand or are not represented?

An **object model** does a good job of describing the **structure** of the system. It helps you understand what classes exist, what their responsibilities are, and how they relate to each other through inheritance, association, aggregation, or composition. This makes it much easier to understand the **overall design**, such as how objects interact or depend on one another. It’s especially useful during development because it clearly defines the building blocks of the system.

However, object models don’t show the **flow of activities**, user interactions, or the sequence in which operations occur. For example, you can see that a Customer has a Shopping Cart, but you won’t see the steps involved in adding an item to the cart or placing an order. While the object model is great for understanding the static structure, it doesn’t represent the **dynamic behavior or process flow**, which you’d need a **process or sequence model** to fully understand.