Nneka Hamilton

CS 255

4-2 Assignment: Evaluate an Object Model

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

- Shopping Cart -addCartItem() +updateQuantity() +viewCartDetails() +checkOut()
- Customer +register() +login() +updateProfile()
- User: verifyLogin(): bool o
- Administrator: updateCatalog(): bool
- Order: placeOrder()
- Shipping Info: updateShippingInfo()
- Order Details: calcPrice()

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

Under the umbrella of the User superclass, there are two unique user classes: Customer and Administrator. Both of these classes are subclasses of the User class. The User class provides them with the ability to inherit common properties such as userId, password, loginStatus, registerDate, and the verifyLogin() method. This structure enables them to contain unique functions and data while also inheriting common attributes.

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

The variables that objects in a system use to store information are called variables, and the functions that objects use to modify their own variables or the variables of other objects are called functions. A good example of this is the shopping cart object, which has a function called addCartItem. This function most likely updates its own variables, such as the productID, quantity, and dateAdded information. Similar to the last example, the Order object has a function called placeOrder() that most likely generates an instance of the Shipping Info object.

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

This model does an excellent job of encapsulating all the functionality that is wanted; however, the one thing that I believe is lacking is any form of payment confirmation. Despite the fact that the customer has a creditCardInfo variable, I am unable to locate the needed confirmation variable or function for when the payment is successful. It is true that the object Order contains a status variable but such status would refer to the status of the orders shipment rather than the status of the financial transaction.

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

A more robust kind of aggregation is called composition, and it is the type illustrated here. Through the establishment of this connection, it is made clear that the first class's presence is necessary for the second class. It is possible for an Order to exist apart from its Order Details counterpart; however, it is impossible for an Order Details object to exist without being connected to an Order object. This reliance is effectively demonstrated by the solid diamond in this diagram, which depicts the things as being interrelated in a domino fashion. The information regarding the shipping and the details of the order are dependent on the presence of an Order, which, in addition to a shopping cart, is dependent on the presence of a Customer.

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?

Composition is a more robust kind of aggregation, and it is denoted by the term aggregation in this instance. By virtue of this connection, it may be deduced that the existence of the first class is necessary for the operation of the second class. Although it is possible for an Order to exist without reference to Order Details, it is impossible for an Order Details object to exist without being connected to an Order object. Because the elements are interconnected in a domino fashion, the solid diamond in this diagram can depict this dependence. Not only does a customer's presence depend on a shopping cart, but the shipping information and order details depend on an order's existence.

Finally, think through the two different models you’ve explored for Hamp Crafts’ systems: a process model and an object model. Then compare these models by responding to the following prompts:

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?

Process models offer a comprehensive picture that enables clients and stakeholders to understand the system's functionality without being mired in technical specifics. They emphasize the progression of data and processes, which can be particularly advantageous for conveying information to non-technical audiences. This abstraction can mask the complexities of the foundational elements, like class structures and interrelations within the code. To address that gap, it may be beneficial to augment process models with supplementary documentation, such as UML diagrams or technical specifications, which can provide a more comprehensive perspective of the system architecture. In this manner, both technical and non-technical stakeholders can comprehend their individual issues.

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?

It effectively encapsulates the classes and their operations, facilitating developers' understanding of the system's structure. Although it proficiently delineates classes and qualities, it may inadequately represent the interactions and relationships among those classes.

The status variable in the Order object underscores a crucial element of object-oriented design: the necessity for explicit documentation. Ambiguities regarding whether status pertains to shipment, transaction, or a broader order status can result in misconceptions throughout development. Thorough documentation, encompassing class diagrams, interaction diagrams, or sequence diagrams, can elucidate these aspects and furnish a more comprehensive understanding of the system's behavior.