Adam Sissoko

November 13, 2021

Hamp Crafts Object Model Evaluation

**Interpret the object model** for the new online storefront by responding to the following prompts:

1. What are the different functions of the online storefront? How are they represented in this type of model?
   * In a UML diagram, the functions of each class are represented at the bottom of each class block. For example, The “Customer” class holds three functions titled “register”, “login”, and “updateProfile”. These functions handle the processes that are specific to the customer class, such as registering online as a customer of the store. These functions also have access to the attributes of an instance of the customer class. This holds true for all functions in each of their respective classes. Functions with a ‘+’ are public, and can be accessed outside the class using an instance of the class. In this case, only a customer object can access the “login” function; a “shopping cart” object does not have the ability to access the “customer” functions.
2. What are the different classes of “users” represented by this object model? What are the associations between these classes?
   1. Admin and Customer
   2. Each has a “userid”, a “password”, a “loginstatus”, and a “registrationdate”
   3. Customer “user” has more properties to accommodate for shopping logic (address for shipping, credit card info for purchases, etc). The Admin user class has a function called “UpdateCatalog” for updating the catalog of items on the website, which the Customer class cannot do.
3. How would the objects “use” their respective variables and functions?
   1. As mentioned above, each instance of a class (object) is able to access its functions within that class. By accessing the functions of their class, the object is able to perform certain duties within the environment of the website. The variables of each object are accessed through these functions as well, but only by using an instance of the object itself. Access to any private variables from outside the class is forbidden, and only functions within that class may access or change them.
4. Does this object model capture all of Hamp Crafts’ desired functionality? Why or why not?
   1. Missing functions:
      1. Lack of function to send billing to outside party
      2. Lack of notification interface
      3. No order history variable or function
   2. Based on the information provided regarding what Hamp Craft’s would like to see, I would say that this object model captures ALMOST all of the functionality that the company desires. One item that is not explicitly included (but may exist in the overall model) is a function to send billing information to an outside party. Functions related to billing and orders include calculating the price, updating the shipping, and placing the order. However, no function seems to be specific to sending billing information on to a third party biller. Again, this functionality may be embedded within one of the listed functions, but there is no way to know based solely on this diagram. Additionally, there does not seem to be a function dedicated to sending a notification to customers once an order has been placed / received. Finally, there is no order history variable for storing customer orders, nor a function to access such a variable should the customer wish to see past orders.
5. The above diagram uses a solid diamond shape to represent a form of aggregation. What type of aggregation does this represent?
   1. This shows a composition, or “non-shared” dependency between the classes
6. What does it imply about the relationship between the classes?
   1. This type of relationship implies that the parent class has exclusive ownership over the child class(es). In essence, if a member of the parent class is deleted, then the corresponding member of the child class is also deleted (see next answer for a more detailed explanation).
7. Why is a solid diamond the appropriate choice here?
   1. Looking at the diagram, one can see that certain relationships are only possible between certain classes. For instance, a shopping cart (child class) can only belong to a customer (parent class). A shopping cart cannot belong to an order, or any other class for that matter. If a particular customer account is deleted, then that customer’s shopping cart is also deleted. However, a customer can delete their shopping cart object without any adverse repercussions to their own account. This helps to limit system complexity as a whole, and ensures that only certain classes belong to certain parent classes. Logically speaking, a customer can have a shopping cart, and can also have an order. Their order can have shipping info, and can have order details. These child class objects, however, compose their parents only, and just like real-world children, cannot be transferred to a new parent with new inheritances.

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:

1. 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?
   1. The process model does a very good job of giving the general idea of what the data flow procedures are in a process. This type of model is good for seeing the “big picture”, and getting an idea for what details may be necessary when developing the solution to the proposed problem. In the case of Hamp Craft’s, the process model simplifies what type of data will flow between stores and processes, as well as what processes will need to be developed. The finer details, such as how each process will complete its task or manipulate the input data, is left out, allowing an observer to focus on the whole process itself. This is important because it helps a developer create a “road map” for development in the SDLC. Understanding which processes interact with which data stores and sources will help the developer design the processes themselves later on during the object model creation stage.