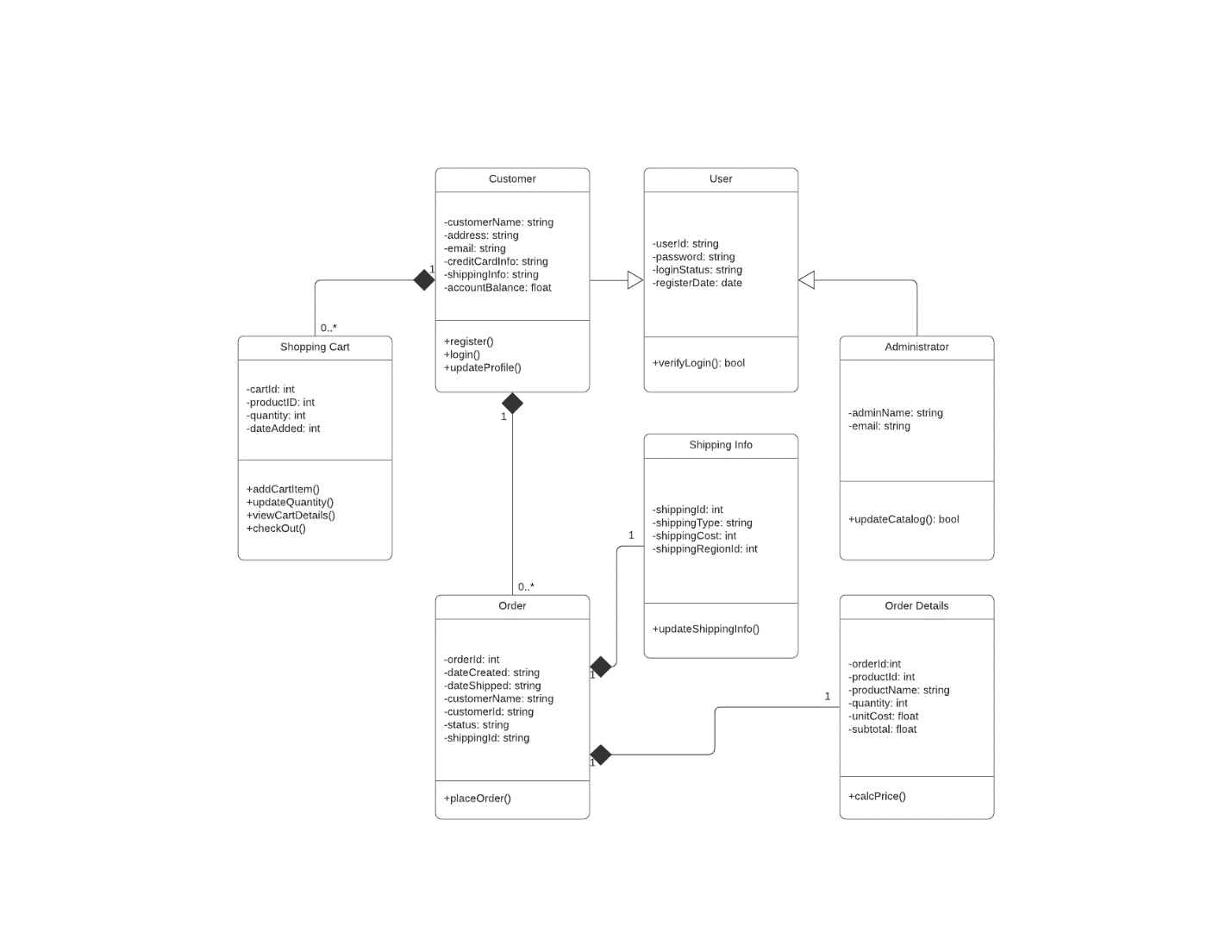
**4-2 Assignment: Evaluate an Object Model**



1. **Interpret the object model for the new online storefront by responding to the following prompts:** 
   * **What are the different functions of the online storefront? How are they represented in this type of model?**

* **Customer :** The customer class features different functions including:
  + - **Register() –** Allowing the customer to register in the system and become a user.
    - **Login() –** This function allows the customer to login to the associated system.
    - **UpdateProfile() –** Allows the customer to update any relevant information to the system as needed.
    - **How:** These functions are represented as operations under the Customer Class Object Model. They provide the necessary functions for customers to enter personal information for system access.
* **Shopping Cart :** The Shopping Cart is an instance of the customer class with the ability to have (zero-to-many) instances of itself while also being of a composite relationship type to the Customer class.
  + - **AddCartItem() –** This function allows the customer to add any items in the online shop to the cart.
    - **UpdateQuantity() –** This function lets the customer increase or decrease the quantity of an item when viewing their shopping cart.
    - **ViewCartDetails() –** This function lets the customer view details about their current shopping cart, like items added, quantity, and price.
    - **CheckOut() –** This function lets the customer move to the check out phase of their order to complete their shopping.
    - **How:** All these functions are represented through the Shopping Cart Class and enable the customer to update their shopping cart details and items.
* **Order :** The order class includes details related to the customers’ information along with their current orders. Functions included:
  + - **PlaceOrder() –** This function allows the customer to complete their order by submitting it to the online ordering tool. Once the customer’s order is sent it creates an instance of two classes, Shipping Info and Order Details.
    - **How:** The Order class is an instantiation of the Customer class represented by a composite (Has-a) relationship , meaning it cannot exist with a Customer class. The PlaceOrder() function allows the customer to move to the next phase of their online shopping by sending their order to the online shop, Hamp Crafts.
* **Shipping Info :** The Shipping Info class contains details about the customer’s current shipment, specifically; the shipping id, shipping type, shipping cost, and shipping region ID. The Shipping Info class is an instantiation of the Order class related by a composite (has-a) relationship. Functions included:
  + - **UpdateShippingInfo() –** The updateShippingInfo() function pulls from the Order class relevant shipping details and updates that information in the system for the customer and Hamp Crafts to view.
    - **How:** The updateShippingInfo() function is represented as an operation of the Shipping Info Class which is an instantiation of the Order Class.
* **Order Details :** The Order Details class is an instantiation of the Order Class with relevant order details such as; Order Id, Product Id, Product Name, Quantity, Unit Cost, and the Sub-Total. The Order Details class features a function:
  + - **CalcPrice() –** The CalcPrice() operation enables the total price to be calculated using the relevant information obtained from the order details from the Order class.
    - **How:** The CalcPrice() operation is a part of the Order Details class which is an instantiation of the Order Class represented by a composite (has-a) relationship, so the price can only be calculated when an Order class is present, otherwise Order Details can not exist.
* **User :** The User Class is the parent class of the Customer Class and Administrator Class, which both inherit the attributes and operations of the User Class. The relevant User class information contains a User ID, Password, Login Status, and Register Date. The functions of the User Class include:
  + - **VerifyLogin() –** This function allows the current user to verify their login credentials through the initial login page of the online shop.
    - **How:** The VerifyLogin() function is of type Boolean so if True then the login is verified for the user and therefore able to access the system. If False, then user is unable to access. Both the Customer Class and Administrator Class inherit from the Parent User Class.
* **Administrator :** The Administrator class enables privileged access to the system through verification of two attributes; an Admin Name, and an Email. The Administrator also inherits from the User Class so it inherits the aforementioned classes’ attributes. The Administrator Class features a single function:
  + - **UpdateCatalog() –** This function allows the user logged in as administrator to update the shopping catalog of Hamp Crafts’.
    - **How:** The function is of type Boolean, so if True then the catalog is updated, if False, then the catalog is not updated. It’s an operation of the Administrator Class which also inherits from the User Class.
  + **What are the different classes of “users” represented by this object model? What are the associations between these classes?**
* **User :** The User is the Parent Class of the object model for Hamp Crafts. The User includes information related to the users’ login credentials used to verify access. The User can be of type Customer or of type Administrator depending on username information.
* **Administrator :** The administrator inherits the attributes and functions of the User class, so a user ID, password, Login Status, and Register Date can be accessed by the Administrator while the Administrator includes its own private attributes such as Admin Name, and Email.
* **Customer :** The Customer Class also inherits the attributes of the User Class but includes its own attributes relevant to a customer. The Customer class is also able to instantiate other Classes, such as, Shopping Cart and Order, which features information related to a customer’s order for the Online shop to use.
  + **How would the objects “use” their respective variables and functions?**

The associated objects would use their respective variables (attributes) and functions (operations) by performing multiple actions included in the online shop. For instance, the Customer object takes different variables such as a customer’s name, address, email, credit card info, shipping info, and account balance and uses these attributes to perform different functions such as register, login, and update profile. All these attributes and operations combined help the online storefront to process the relevant information to instantiate multiple objects of composite relationships. Both the Shopping Cart and Order Objects derive their information from the Customer’s Class with the ability to instantiate multiple instances of themselves but can only exist as a part of the Customer class and note separate too.

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

While the object model for Hamp Crafts captures most of the functionality including the ability to process credit cards and receive customer payment information. It also portrays the customer’s ability to receive a notification in their profile that contains status updates of their order. The Object model is useful when relaying information regarding what the object model is doing and how the different objects are interrelated to each other but doesn’t convey the necessary information of exactly how these systems are being implemented and what the ordering tools, customer information, and online storefront look like. I think the object model could benefit from relaying additional information regarding how the customer is accessing their order information and status updates. Additional functions could be implemented to convey this information for increased customer satisfaction.

* + **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 represents a relationship between two classes that are related by a composite relationship. This implies a (has-a) relationship so that one instantiation cannot exist separately from the original entity. For example, the Shopping Cart and Order objects cannot exist separately from the Customer Class because without a Customer they simply would not exist. This is different from a typical aggregate association which is of a lesser strength relationship in terms of instantiation and dependency. The solid diamond shape composite relationship is the appropriate choice here because it accurately portrays the dependency of the relationship amongst the related classes.

1. **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?**

A Process model is beneficial for describing the flow of logic and sequence of actions in a system. Process Models excel at modeling the lifecycle of a system from each stage like the Planning, Analysis, Design, Implementation, Maintenance cycles and the use of Data Flow Diagrams. It also makes it easier to understand how the data flows from one point to another in a system. Some aspects of the system that are more difficult to understand include how these processes interact with each other. For instance, how does an order process information regarding a customer’s shopping cart, and how that order provides information for the shipping information to process. The Process Model falls short when needing to portray complex relationships amongst multiple Classes as opposed to the logical flow of information within a single Class.

* + **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?**

The Object Model picks up where the Process Model leaves off, by providing a comprehensive model of the structure behind the system. The Object Model captures how Classes are related to each other by representing the relationships between each one. This helps to make the reader understand how Classes inherit and instantiate objects from one another and what type of relationships are bound by certain conditions. The Object model’s weaknesses, however, lie in its inability to properly capture the data flow and sequence of actions in a Class diagram.

Valacich, Joseph, S. and Joey F. George. Modern Systems Analysis and Design. Available from: MBS Direct, (9th Edition). Pearson Education (US), 2019.