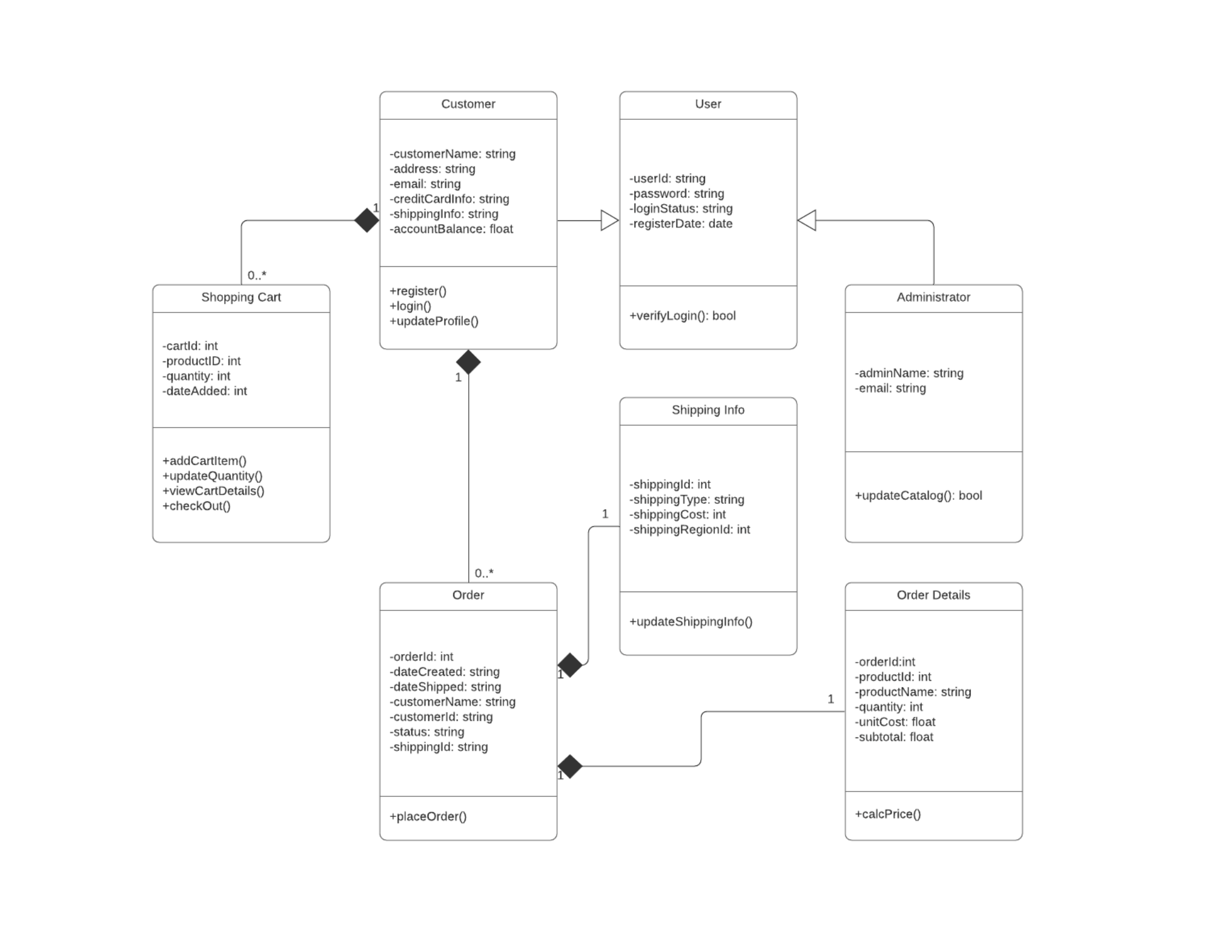
**Prompt**

Hamp Crafts is a family-owned craft store that has been in business for decades. Its current operations are limited to its physical (brick-and-mortar) storefront. Recently, the owners have determined that one of the best ways to gain additional revenue is through online sales. You have reassured Hamp Crafts’ owners that adding in an online storefront will work with their current processes. In this assignment, you will review an object model, specifically a UML class diagram for the new online store. You will interpret the diagram and reflect on how well it meets Hamp Crafts’ needs.

**1.** **First, review Hamp Crafts**’ **needs for their online storefront.**

**UML:**

* Hamp Crafts would like: customers to be able to create an account with their shipping, billing, and contact information
* For customer orders, Hamp Crafts would like: to accept credit and debit cards for transactions. Hamp Crafts plans on: using an established credit card vendor service (e.g., Square, Shopify) to receive customer payments.
* Once a transaction is complete: the customer should receive a notification based on the information in their personal profile regarding order status and confirmation.
* On the administrative side of the online storefront: Hamp Crafts should receive an alert of the transaction.
* Customers: should be able to check the status of their order any time online from their personal account profile under order history.
* The business owners: also need an administrative back end for customer support and updates to customer information and the website.

**2. 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?**

**Class Interpretation:**

The different functions of the storefront are clearly represented in the UML diagram through 7 main classes, each handling specific responsibilities.

The functionality is modeled using methods (functions prefixed with +) in each class.

* **Customer :**

**Functions from this class include:**

Register()

login()

updateProfile()

This class is for the user facing accounts and there functionality hence the functions.

**UML Observation:** after this it has a arrow going from admin to customer and user.

In the diagram, arrows show:

• Customer → User which is inheritance

• Administrator → User, which again is inheritance.

**This implies to me that both Customer and Administrator are treated as subclasses of User.**

**So i’m assuming:**

**User:**

├─ +role: Enum["customer", "admin"]

├─ +verifyLogin()

**So the logic would look like(but in python my favorite language:**

if user.role == “customer”:

# Allow storefront to be used

enable\_cart\_access()

enable\_order\_tracking()

allow\_profile\_update()

elif user.role == "admin":

# Allow backend/admin behavior and privileges

enable\_catalog\_editing()

enable\_user\_support\_tools()

**Security & Pattern Relevance**

**This structure:**

• Keeps authentication logic centralized in one place (The User)

• Prevents misuse of inheritance (Admin ≠ Customer, this makes it extra safe)

• Follows good architecture regarding the separation of concerns

**It mirrors what frameworks like Dropwizard in Java encourage:**

• Centralized identity

• Role-based routing

• Secure resource control

**This is also similar to the singleton pattern:**

Where I’m imagining: a shared SessionManager/ UserManager object that:

**•**  Verifies users

• Holds session tokens

• Routes based on role

That’s a valid addition to this design especially in a security-aware framework like Dropwizard!We are actually doing this exact same thing in another class and this reminded me of this!

* **ShoppingCart:**

This class handles the selection for products and the cart items/check out.

**You can tell this by its functions which are:**

+addCartItem() : to add an item

+updateQuantity() : to change how the quantity of each item

+viewCartDetails(): to view what’s in the users cart

+checkOut() : to check out whats in the cart and purchase the items.

* **Order:**

This handles the purchasing/placement and finalization of the order. Its functions are:

+placeOrder(): which places the order, and will need to be linked to payment and shipping info.

* **Order-Details:**

This managers the order contents and the cost you can tell this by the function:

+calcPrice()

* **ShippingInfo:**

This handles the delivery info and cost by its function:

+updateShippingInfo() which gives differences prices for various shipping methods.

* **User:**

This is the class base, that is able to be shared for user variables/properties.

+verifyLogin(): bool is used for login for both admin/users.

* **Administrator:**

This is the backend/admin/security

**It’s function:**

+updateCatalog() shows that an administrant can change the products that are available, price, listing etc.

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

There are Users, Customers, and Administrators

**User:**

This is the class base. It is used and shared by all of the types of users.

**Customer:**

Customer inherits from user class - > buyers

**Admin:**

Inherits from the user class -> backend

* **What are the associations between these classes?:**

**Associations**

**A Customer is associated with:**

0..\* ShoppingCart objects: this is representing one or more active/abandoned carts

0..\* Order objects: this is representing their order history

This means each customer can maintain multiple carts ( in-progress sessions) and be able to place multiple orders over time which is a good design feature.

**In relation to Admin:**

The Administrator class doesn’t need a direct link to ShoppingCart or Order because admins don’t place orders or use carts, they work on the backend and manage everything across the whole system.

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

**Each object has variables:**

Such as email, address, creditCardInfo. All of these hold the data that is needed.

**Functions like:** register(), login(), checkOut() are all the function actions**.**

-They will update that data or trigger something else to happen checking out,

logging in, etc..

**Example:**

A customer signs up register(). Is able to fill out their info. Is able to add stuff to the cart using addCartItem()), then can check out with checkOut(), which places the order using placeOrder()

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

It was mentioned above that camp crafts wants Hamp Crafts would like: customers to be able to create an account with their shipping, billing, and contact information:

* For customer orders, Hamp Crafts would like: to accept credit and debit cards for transactions. Hamp Crafts plans on: using an established credit card vendor service (e.g., Square, Shopify) to receive customer payments.
* Once a transaction is complete: the customer should receive a notification based on the information in their personal profile regarding order status and confirmation.
* On the administrative side of the online storefront: Hamp Crafts should receive an alert of the transaction.
* Customers: should be able to check the status of their order any time online from their personal account profile under order history.
* The business owners: also need an administrative back end for customer support and updates to customer information and the website.

**So I think they are still missing :**

**What they wanted that is in UML:**

-Customers can make accounts, save shipping, billing, and contact info

-Credit/debit support is shown with creditCardInfo

-Order history works, customers are linked -> to multiple orders -> status is stored

-Admin backend exists: so the Administrator can update the catalog

**But it**’**s still missing:**

-No system confirmation system to send confirmation to the customer after an order

-No administration alert when a customer places an order

-No product catalog/inventory model shown in this diagram that I saw I don't see any functions like receiveAlert() o for admin. I also see no class for a catalog, or inventory/product.

-So it’s close, but it has not been fully implemented in the UML at least.

* **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?**

- That solid diamond means composition. This is when one thing fully owns the other or “has-a” relationship.

-This means : Strong ownership: one object contains another

If the parent is deleted -> the child gets deleted too

**Example:**

• A Customer owns their ShoppingCart/ Order

• The cart and order only “exist” because the customer does

• If the Customer is removed from the equation, the cart and orders should be removed too, they don’t stand separate but as one.

- These are dependent objects in the sense that are dependent with the customer. That’s why using a solid diamond here makes sense. It is showing how they’re tightly bound and not are not meant to exist all on their own.

**3. 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 great at for how a system works over a period of time. This can show all the the steps users take: login -> add to cart ->checkout. This helps a programmer understand flow and sequence.

-But This doesn’t show the structure of the system . Which is what data is stored where or how things are related. You “see the actions”, not the full blueprint.

**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 shows what’s IN the system: classes, data/how things are connected.

It makes the structure easier to understand.

-But it doesn’t show HOW or IN WHAT order things happen, so it’s not good for “understanding the actual flow or steps of the system.”