Functions

User Class

verifyLogin(): This function is used to verify the login status of a user. It returns a boolean value to indicate whether the login is successful or not

Customer Class (Inherited from User)

register(): This function is used for customer registration.

login(): This function is used for customer login.

updateProfile(): This function allows customers to update their profile information

Administrator Class (Inherited from User)

UupdateCatalog(): This function is used by administrators to update the product catalog. It returns a boolean value to indicate the success of the update.

Shopping Cart Class

The ShoppingCart class is composed of a customer and represents the shopping cart functionality.

It contains attributes such as cartId, productId, quantity, and dateAdded to manage items in thecart.

Order Class

The Order class is composed of a customer and represents the order management functionality.

It contains attributes such as orderId, dateCreated, dateShipped, customerName, customerId, status, and shippingId to manage orders.

placeOrder(): This function is used to place an order.

ShippingInfo Class

The ShippingInfo class is composed of an order and represents shipping information for orders. It contains attributes like shippingId, shippingType, shippingCost, and shippingRegionId.

updateShippingInfo(): This function is used to update shipping information.

OrderDetails Class Functions:

The OrderDetails class is composed of an order and represents the details of items within an order.

It contains attributes such as orderId, productId, productName, quantity, unitCost, and subtotal.

calcPrice(): This function calculates the subtotal for a product in an order.

User management (registration, login, profile updates).

Administrator functions for catalog management.

Shopping cart management (adding/removing items).

Order placement and tracking.

Shipping information management.

Detailed order item information.

The provided object model captures some basic functionality for Hamp Crafts' online storefront, such as user management, shopping cart, order management, and shipping information. However, it may not fully meet all of their desired features and requirements. The model is relatively simplified and does not account for more advanced functionalities that are typically present in real-world e-commerce platforms. It lacks features like product variations, inventory management, payment processing, security measures, reporting and analytics, customer support tools, and integration with third-party services. Additionally, it does not address aspects of scalability, user roles beyond customers and administrators, and the complexities associated with product management and categorization. To fully encompass Hamp Crafts' desired functionality, the model would need to be expanded and refined to include these missing elements, ensuring that it aligns with the specific needs and complexities of their online business.

A solid diamond shape represents a form of aggregation known as "shared aggregation." Shared aggregation implies a whole-part relationship between classes, where one class (the whole or container) contains or is composed of multiple instances of another class (the part).The solid diamond is the appropriate choice for shared aggregation because it signifies that the part class can be reused across different instances of the whole class. This is commonly used when one class contains a collection of another class, and those instances of the other class can be shared among multiple instances of the containing class. It is a way to represent a more specific and meaningful relationship between classes compared to a simple association, where instances of the associated class are not necessarily shared.

Different models

Process models, including flowcharts and BPMN diagrams, excel at portraying the sequential flow of activities, tasks, and interactions within a system. They are particularly effective at elucidating step-by-step processes, revealing how they progress from one stage to another, and highlighting decision points and conditional flows. This clarity aids stakeholders in comprehending how various scenarios influence the overall process and in assigning roles and responsibilities efficiently. Process models also play a vital role in process optimization by identifying bottlenecks and enhancing workflow efficiency. However, their strength lies predominantly in representing process flow, and they may be less suited for capturing dynamic system behaviors, intricate data flows, and complex structural relationships among system components. In addition, Functional programming's declarative and compositional nature aligns well with the flow-based representation of process models. Process models can illustrate how data flows through a series of pure functions, providing a visual guide to the transformation of inputs into outputs.

Object Models, based on object-oriented principles, provide a comprehensive view of the structural aspects of a system. These models encompass classes, objects, attributes, and methods, as well as the relationships connecting them. Object models are distinguished by their ability to offer structural clarity, revealing how a system is organized, including class hierarchies and component interdependencies. They embrace abstraction, simplifying complex systems by breaking them down into reusable and well-defined components. The concept of encapsulation establishes clear boundaries for data and behaviors within classes, enhancing system design. Additionally, object models support code reuse and hierarchy management through inheritance and polymorphism. Nevertheless, object models may struggle to capture dynamic system behavior, data flow, user interface intricacies, and interactions with external systems comprehensively.