E-COMMERCE PRODUCT MANAGEMENT SYSTEM DOUCMENTATION

Object-Oriented Analysis (OOA)

The E-commerce Product Management System is built on clearly defined object-oriented components. First, the main objects in the system include: Product (regular products), Electronics (electronic products inheriting from Product), Discountable (an interface that defines the ability to apply discounts), InventoryList (a list managing the inventory for each type of product), ShoppingCart (the customer's shopping cart), and Order (customer orders).

Each object contains attributes that describe its specific characteristics. The Product class has a name, identifier, base price, discount percentage, discounted price, and stock quantity. The Electronics class, inheriting from Product, adds power, warranty period, extra fees, and a total price (including the base price plus the extra fees). InventoryList maintains the list of products in stock, while ShoppingCart manages the selected products and their quantities. Order gathers information about both regular and electronic products that have been ordered.

Regarding methods, the system provides various operations to manage products and shopping carts. Discountable defines the applyDiscount(rate) method to apply discounts. Product and Electronics implement this method and also provide other functions such as displaying information (displayInfo), retrieving the name, price, discount rate, and stock quantity, updating stock levels, or comparing two products using the == and > operators. The InventoryList class supports adding, removing, and searching for products in stock. ShoppingCart allows adding or removing items, calculating the total order value, displaying the cart, and searching for products. Finally, Order is responsible for creating orders and displaying order information.

In terms of inheritance, Product implements the Discountable interface, while Electronics inherits all attributes and behaviors from Product and extends them with its own information. The classes InventoryList, ShoppingCart, and Order do not participate in inheritance but primarily play the role of data managers, demonstrating a composition relationship with the product classes.

Thanks to this structure, the system has a clear and extensible architecture, fully reflecting the principles of object-oriented analysis, ensuring effective support for future software development and maintenance.

Explanation About Class Design

In this system, the class design is built to take full advantage of the features of object-oriented programming. The Discountable class is defined as an interface, helping to separate the discount behavior from the specific implementation. Thanks to this, both Product and Electronics can flexibly implement the applyDiscount(rate) method in their own way without affecting the overall structure. Next, inheritance is applied when Electronics extends the Product class, reusing all the basic information of a product while adding specific elements such as power, warranty, and extra fees.

In addition, the system leverages operator overloading to provide a natural syntax when working with data. Comparison operators such as == and > are redefined to compare two products based on name or price, while the += and -= operators in ShoppingCart allow adding or removing products in an intuitive way. The use of templates for the InventoryList and ShoppingCart classes allows them to work with different data types (for example, Product or Electronics) without rewriting the source code for each type. This organization not only reduces duplication but also enhances scalability and code reusability.

Thanks to the combination of interfaces, inheritance, operator overloading, and template classes, the entire system architecture becomes clear and easy to maintain, while also supporting future functional expansion.

Code Walkthrough

The program about an e-commerce product management system describes a simple shopping system that includes classes for managing products, electronic items, inventory, shopping carts, and orders. The Discountable class acts as an interface for objects that can apply discounts. The Product class represents regular products with attributes such as name, code, base price, discount rate, stock quantity, and methods to display information or calculate the price after a discount. The Electronics class inherits from Product, adding specific details of electronic devices such as power, warranty period, and additional fees, while overriding functions to calculate the final price and display detailed information. The template class InventoryList manages the list of items in stock, supporting adding, removing, or searching for products by name. The ShoppingCart class, also a template, stores pairs of products and purchased quantities, allowing the calculation of the total cost, displaying the cart, adding or removing items, and searching for products in the cart. The Order class records order information from the shopping cart, updates stock quantities, and prints detailed invoices.

In the main function, the program initializes a list of regular and electronic products, then lets the user choose a role: customer or manager. If the customer role is selected, the user can search for products, view information, add them to the cart, continue shopping, or switch to functions such as placing an order, adding or removing items, and comparing prices. If the manager role is selected, the user can add or remove products from the inventory, enter full details, and the system will update and display the newly entered data.

Test Results

- Test case 1:

+ Input:

1

Desk Lamp

Yes

Yes

Book A

Yes

2

Yes

Phone X

Yes

1

No

2

Backpack X

Yes

1

No

1

+ Output:

Choose your role:

1. Customer

2. Manager

Choose: 1

==========

Enter product you want to buy: Desk Lamp

Name: Desk Lamp

Price: 80

Price (applying 8% discount): 73.6

(Out of stock)

Amount: 0

Would you want to add this product to your cart? (Yes/No)

Your answer: Yes

This product is out of stock

Would you want to look for other products? (Yes/No)

Your answer: Yes

Enter product you want to buy: Book A

Name: Book A

Price: 50

Price (applying 10% discount): 45

Amount: 20

Would you want to add this product to your cart? (Yes/No)

Your answer: Yes

Enter number of products you want to buy: 2

Add to the cart successful!

Would you want to look for other products? (Yes/No)

Your answer: Yes

Enter product you want to buy: Phone X

Name: Phone X

Price: 850

Price (applying 15% discount): 722.5

Power: 20

Warranty Time: 12

Amount: 5

Would you want to add this product to your cart? (Yes/No)

Your answer: Yes

Enter number of products you want to buy: 1

Add to the cart successful!

Would you want to look for other products? (Yes/No)

Your answer: No

==========

List of all products in the cart:

Name: Book A

Price: 50

Price (applying 10% discount): 45

Quantity: 2

Subtotal: 90

Name: Phone X

Price: 850

Price (applying 15% discount): 722.5

Quantity: 1

Total: 812.5

Choose function:

1. Order product

2. Add product

3. Remove product

4. Compare 2 products

Choose: 2

Enter product you want to buy: Backpack X

Name: Backpack X

Price: 60

Amount: 15

Would you want to add this product to your cart? (Yes/No)

Your answer: Yes

Enter number of products you want to buy: 1

Add to the cart successful!

Would you want to look for other products? (Yes/No)

Your answer: No

List of all products in the cart:

Name: Book A

Price: 50

Price (applying 10% discount): 45

Quantity: 2

Subtotal: 90

Name: Phone X

Price: 850

Price (applying 15% discount): 722.5

Quantity: 1

Name: Backpack X

Price: 60

Quantity: 1

Total: 872.5

Choose function:

1. Order product

2. Add product

3. Remove product

4. Compare 2 products

Choose: 1

==========

ORDER DETAILS:

Name: Book A

Price: 50

Price (applying 10% discount): 45

Quantity: 2

Subtotal: 90

Name: Phone X

Price: 850

Price (applying 15% discount): 722.5

Quantity: 1

Name: Backpack X

Price: 60

Quantity: 1

Total: 872.5

+ Explanation about test case:

In this test case, the user chooses the role of a customer. First, they search for the product Desk Lamp, but it is out of stock. Although the user wants to add it to the cart, the program displays the message “This product is out of stock” and prompts them to continue searching for other products. Next, the user searches for Book A, which has a price of 50 and a 10% discount, with 20 units available in stock. The user successfully adds 2 units to the cart. Then, they search for Phone X, an electronic product priced at 800 with an additional fee of 50 and a 15% discount, with 5 units remaining in stock. The user adds 1 unit to the cart. When asked whether to continue searching for other products, they choose “No.” At this point, the cart contains 2 Book A and 1 Phone X, with a total price of 812.5. The user

then selects the option to add more products and adds 1 Backpack X, priced at 60 with 15 units in stock and no discount. The cart now contains 2 Book A, 1 Phone X, and 1 Backpack X, with a total of 872.5. Finally, the user selects the option to place the order, the program creates the order, updates the stock quantities, and displays the order details. The total payment remains 872.5. This test case demonstrates handling out-of-stock products, adding multiple types of products to the cart, calculating the total price, and successfully placing an order.

Explain about UML Diagrams

- Class Diagram: