E-Commerce System-Documentation

Name: Khong Dinh Tu

ID: 24110145

---

1. Object-Oriented Analysis (OOA)

Following the 4-step OOA model, the system has the following objects:

1.1 Objects

- Product
- ElectricProduct (inherits from Product)
- ShoppingCart
- Order
- Invoice
- Shopee (manages products and orders)
- Discountable (abstract class / interface)

1.2 Attributes for Each Object

- **Product:** id, name, category, weight, price, stock
- **ElectricProduct:** warrantyMonths (inherits attributes from Product)
- **ShoppingCart:** itemList (InventoryList<Product*>), total
- **Order:** orderID, items, total
- **Invoice:** items, shippingFee
- **Shopee:** inventory, orders, nextOrderId
- **Discountable:** *(no attributes, only abstract behavior)*

1.3 Methods

- **Product:** displayInfo(), applyDiscount(), updateStock(), reduceStock(), getId(), getName(), getPrice(), operator==, operator<
- **ElectricProduct:** displayInfo() (override), updateStock() (override), applyDiscount() (override)
- **ShoppingCart:** operator+=, operator-=, displayCart(), applyDiscount(), checkout(), getTotal()
- **Order:** display()
- **Invoice:** calcTotal(), displayInvoice()
- **Shopee:** addProduct(), showInventory(), findProduct(), createOrder(), showOrders()
- **Discountable:** applyDiscount() *(pure virtual)*

1.4 Inheritance Relationships

- Discountable is an **abstract base class**.
    - Product implements Discountable.
    - ElectricProduct inherits from Product (and thus also from Discountable).
    - ShoppingCart implements Discountable.
- Shopee is a manager class that uses Product and Order.

2. Class Design

2.1 Encapsulation

- All attributes are private or protected.
- Public getters and setters (or controlled methods like updateStock() and reduceStock()) are used for controlled access.

2.2 Inheritance

- Product inherits from Discountable.
- ElectricProduct inherits from Product.
- This hierarchy allows different types of products to reuse and extend common behavior.

2.3 Polymorphism

- The applyDiscount() function is pure virtual in Discountable and implemented differently in Product, ElectricProduct, and ShoppingCart.
- Polymorphism allows treating all these classes uniformly as Discountable*.

2.4 Operator Overloading

- Implemented in **Product**:
  - operator== → compare product weights.
  - operator< → compare product prices.
- Implemented in **ShoppingCart**:
  - operator+= → add a product to the cart.
  - operator-= → remove a product from the cart.

3. Code Walkthrough

Abstract Class

Discountable defines the contract for discounts. Both Product and ShoppingCart must implement applyDiscount().

Operator Overloading

- operator+= and operator-= make adding/removing products from the cart intuitive.
- operator== and operator< make product comparison intuitive (weight and price).

Order Flow Example

- User adds products to a **ShoppingCart**.
- At checkout, the cart generates an **Order**.
- An **Invoice** is created, adding a shipping fee.
- **Shopee** stores both products and orders for management.

4. Sample Output

When running the code, the ouput includes:

```
=== Test 1: Show Inventory ===
=== Inventory (3 items) ===
[ID]: 101
[Name]: C++ Primer
[Category]: Book
[Weight]: 500 g
[Price]: 150000 VND
[Stock]: 5
[ID]: 102
[Name]: Gaming Laptop
[Category]: Electronics
[Weight]: 2500 g
[Price]: 20000000 VND
[Stock]: 3
[Warranty Months]: 24 months
[ID]: 103
[Name]: Smartphone
[Category]: Electronics
[Weight]: 180 g
[Price]: 12000000 VND
[Stock]: 0
```

5. UML Diagrams

5.1 Class Diagram

- **Product** (abstract methods from Discountable)
- **ElectricProduct** inherits Product
- **ShoppingCart** implements Discountable
- **Order** and **Invoice** aggregate Products
- **Shopee** aggregates InventoryList<Product*> and Orders

Relationships:

- Inheritance: Product → ElectricProduct
- Interface realization: Discountable → Product, ShoppingCart
- Aggregation: Shopee → Orders, InventoryList
- Association: ShoppingCart → Product

5.2 Sequence Diagram (Checkout → Order)

Actors: User → ShoppingCart → Shopee → Order → Invoice

Steps:

1. User adds products to the cart with +=.
2. User calls checkout() on ShoppingCart.
3. ShoppingCart sends checkout result to Shopee.
4. Shopee creates an Order with items and total.
5. Shopee generates an Invoice with shipping fee.
6. Invoice displays order details back to the User.

6. Use of LLM (ChatGPT)

I used ChatGPT for:

- Brainstorming the design of the template class InventoryList.
- Suggesting intuitive operator overloads (`+=`, `-=`).
- Refining test case scenarios (out-of-stock products, restocking, multiple orders).
- Guidance on UML notation for class and sequence diagrams.

Example Prompt:
"Can you suggest test cases for an OOP-based C++ E-commerce cart and order system with operator overloading?"

Response:
ChatGPT suggested:

- *Attempting to add out-of-stock products to the cart.*
- *Adding the same product multiple times and displaying quantities correctly.*
- *Checking out twice and verifying correct stock updates.*
- *Restocking products and testing overridden methods in ElectricProduct.*

All final code and documentation were written and verified personally.