# E-commerce Product Management System - Báo cáo (UML đã sửa)

## 1. OOA Analysis (tóm tắt)

Objects: Product, Electronics, ShoppingCart, Order, InventoryList, Discountable, CartItem.

## 2. Thiết kế lớp (class design)

Product: lớp cơ sở; Electronics: lớp kế thừa; InventoryList<T>: lớp template; ShoppingCart quản lý giỏ hàng và nạp chồng operator+=; Order lưu đơn hàng; Discountable là interface trừu tượng.

## 3. Code walkthrough (đoạn quan trọng)

Các phần chính: applyDiscount (interface), operator+= (ShoppingCart), InventoryList template. Xem phần phụ lục để xem mã nguồn đầy đủ.

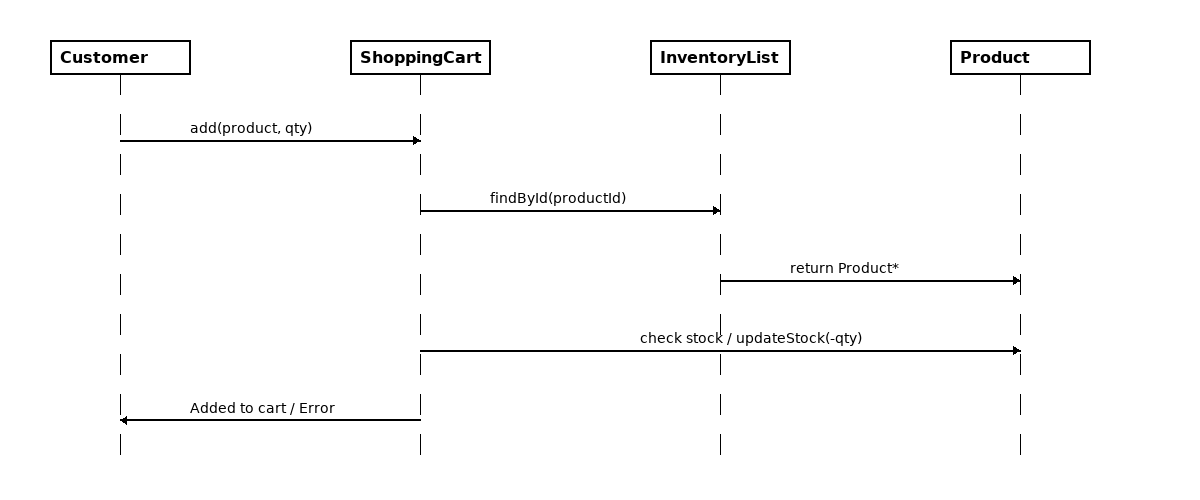
## 4. Kết quả kiểm thử (program output)

=== Inventory ===  
ID: 1 | Name: Book | Price: $10 | Stock: 5  
ID: 2 | Name: Laptop | Price: $800 | Stock: 2  
 Brand: Dell | Warranty: 24 months  
ID: 3 | Name: Phone | Price: $500 | Stock: 0  
 Brand: Samsung | Warranty: 12 months  
Added to cart: 10  
Added to cart: 800  
Out of stock!  
  
--- Cart Contents ---  
ID: 1 | Name: Book | Price: $10 | Stock: 4  
ID: 2 | Name: Laptop | Price: $800 | Stock: 1  
 Brand: Dell | Warranty: 24 months  
Total: $810  
  
After discount:  
ID: 2 | Name: Laptop | Price: $720 | Stock: 1  
 Brand: Dell | Warranty: 24 months  
  
Order confirmed. Total payment: $810

## 5. UML Diagrams (fixed)

Class Diagram:

Sequence Diagram (thêm sản phẩm vào giỏ):



## 6. LLM Usage

Tôi đã dùng ChatGPT để brainstorm tên lớp, cấu trúc template và ý tưởng nạp chồng toán tử.

## Appendix: Full C++ source code

#include <iostream>

#include <vector>

#include <string>

#include <iomanip>

using namespace std;

class Discountable {

public:

virtual double applyDiscount(double rate) = 0;

virtual ~Discountable() {}

};

class Product : public Discountable {

protected:

int id;

string name;

double price;

int stock;

public:

Product(int \_id, string \_name, double \_price, int \_stock)

: id(\_id), name(\_name), price(\_price), stock(\_stock) {}

virtual void display() {

cout << "ID: " << id << " | Name: " << name

<< " | Price: $" << price << " | Stock: " << stock << endl;

}

bool inStock() const { return stock > 0; }

void reduceStock() { if (stock > 0) stock--; }

int getId() const { return id; }

double getPrice() const { return price; }

double applyDiscount(double rate) override {

price = price - price \* rate;

return price;

}

bool operator==(const Product& other) const {

return this->id == other.id;

}

};

class Electronics : public Product {

string brand;

int warrantyMonths;

public:

Electronics(int \_id, string \_name, double \_price, int \_stock,

string \_brand, int \_warranty)

: Product(\_id, \_name, \_price, \_stock),

brand(\_brand), warrantyMonths(\_warranty) {}

void display() override {

Product::display();

cout << " Brand: " << brand

<< " | Warranty: " << warrantyMonths << " months" << endl;

}

};

template <typename T>

class InventoryList {

vector<T> items;

public:

void addItem(const T& item) { items.push\_back(item); }

void showAll() {

for (auto& item : items) item->display();

}

T getItem(int index) { return items[index]; }

int size() const { return items.size(); }

};

class ShoppingCart {

vector<Product\*> cartItems;

double total = 0;

public:

ShoppingCart& operator+=(Product\* p) {

if (p->inStock()) {

cartItems.push\_back(p);

total += p->getPrice();

p->reduceStock();

cout << "Added to cart: " << p->getPrice() << endl;

} else {

cout << "Out of stock!\n";

}

return \*this;

}

void showCart() {

cout << "\\n--- Cart Contents ---\\n";

for (auto& p : cartItems) p->display();

cout << "Total: $" << total << endl;

}

double getTotal() const { return total; }

};

class Order {

vector<Product\*> orderedItems;

double total;

public:

Order(ShoppingCart& cart, double totalAmount)

: total(totalAmount) {}

void confirmOrder() {

cout << "\\nOrder confirmed. Total payment: $" << total << endl;

}

};

int main() {

// Inventory

InventoryList<Product\*> inventory;

Product\* p1 = new Product(1, "Book", 10.0, 5);

Product\* p2 = new Electronics(2, "Laptop", 800.0, 2, "Dell", 24);

Product\* p3 = new Electronics(3, "Phone", 500.0, 0, "Samsung", 12);

inventory.addItem(p1);

inventory.addItem(p2);

inventory.addItem(p3);

cout << "=== Inventory ===\\n";

inventory.showAll();

// ShoppingCart demo

ShoppingCart cart;

cart += p1;

cart += p2;

cart += p3; // out of stock test

cart.showCart();

// Apply discount

p2->applyDiscount(0.1); // 10% off laptop

cout << "\\nAfter discount:\\n";

p2->display();

// Place order

Order order(cart, cart.getTotal());

order.confirmOrder();

// Cleanup

delete p1; delete p2; delete p3;

return 0;

}