# E-COMMERCE DEMO MANAGEMENT SYSTEM - REPORT

**Name:** Huỳnh Vũ Minh Hiền
**Student's ID:** 24110091
**Subject:** Object-Oriented Programming

---

# I. Object-Oriented Analysis (OOA) Model

## 1. Objects and Attributes

**Objects:**

Product

Electronics (specialized Product)

InventoryList

ShoppingCart

Order

**Attributes:**

**Product:** id, name, price, stock

**Electronics (inherits Product):** brand

**InventoryList:** items[], count

**ShoppingCart:** items[], count, total

**Order:** orderId, customer, amount

---

## 2. Identify Methods

**Product:** getId(), getName(), getPrice(), getStock(), updateStock(), printInfo(), applyDiscount()

Operator overload: operator==

**Electronics:** updateStock(), printInfo() (overridden)

**InventoryList:** add(), printAll(), size(), getAt()

**ShoppingCart:** Operator overload += (add product to cart), applyDiscount(), printCart()

**Order:** printOrder()

---

### 3. Inheritance Relationships

**Product** is the base class.

**Electronics** inherits from **Product**.

**ShoppingCart**, **InventoryList**, and **Order** interact with **Product** objects.

---

# II. Overview of the E-commerce Demo System

**Product (Base Class):** Represents a general product in the system with attributes such as id, name, price, and stock. Provides basic methods to get product info and apply discounts.

**Electronics (Derived Class):** Inherits from Product and adds the attribute *brand*. Overrides methods to update stock (with a bonus when restocking) and print product information.

**InventoryList:** Maintains a list of available products. Provides functionality to add and display all products.

**ShoppingCart:** Allows customers to add products (using overloaded += operator). Tracks total price and applies discounts to the total.

**Order:** Represents a customer's purchase with order id, customer name, and total amount.

This design applies **encapsulation** (private attributes, getters/setters), **inheritance** (Electronics inherits Product), and **polymorphism** (method overriding in Electronics).

---

# III. Code Walkthrough

**Product class:**
Stores product information (id, name, price, stock). Includes discount calculation and operator overload for equality comparison.

**Electronics class:**
Derived from Product, adds brand attribute, and overrides methods for stock update and info printing.

**InventoryList class:**
Simple list structure to store multiple products and display them.

**ShoppingCart class:**
Contains a list of products, supports operator += for adding products, calculates total, and applies discount rates.

**Order class:**
Stores order details and prints them.

**Main function:**

Creates product objects (Notebook, Pencil, Headphone).

Adds products to the inventory and displays them.

Compares two products using the overloaded == operator.

Demonstrates shopping cart operations (adding items, printing details, applying discounts).

Creates an order and prints order details.

# IV. System Operations

**Inventory Operations:**

Add products into the inventory.

Display product information.

**Product Operations:**

Update stock.

Apply discounts.

Compare products using operator ==.

**ShoppingCart Operations:**

Add products to cart using +=.

Display cart contents.

Apply discount to total amount.

**Order Operations:**

Create an order with order id, customer, and amount.

Print order details.

# V. Testing the System & Results

**Test Case 1: Create and Display Products**

```
Product p1(1, "Notebook", 5.5, 10);
Product p2(2, "Pencil", 1.2, 0);
Electronics e1(3, "Headphone", 25.0, 5, "Sony");
```

**=> Output:** Product and Electronics details displayed.

**Test Case 2: Add Products to Inventory**

```
inventory.add(p1);
inventory.add(p2);
inventory.add(e1);
inventory.printAll();
```

**=> Output:** Inventory shows all three products.

**Test Case 3: Compare Products**

```
if (p1 == p2) cout << "Same"; else cout << "Different";
```

**=> Output:** "Different".

**Test Case 4: Add Products to ShoppingCart**

```
cart += p1;
cart += p2;   // Out of stock
cart += e1;
```

**=> Output:** Notebook and Headphone added, Pencil rejected.

**Test Case 5: Apply Discount and Create Order**

```
double newTotal = cart.applyDiscount(0.1);Order o1(101, "Nguyen Van A",
newTotal);
o1.printOrder();
```

**=> Output:** Discounted total displayed

Order information printed

# VI. LLM Usage

I used **ChatGPT (LLM)** for:

Structuring the class design and suggesting attributes/methods.

Debugging syntax and logical errors.

Writing explanatory English comments in the code.

# VII. Conclusion

This project demonstrates key **Object-Oriented Programming** concepts:

**Encapsulation:** Private attributes with public getters/setters.

**Inheritance:** Electronics inherits Product.

**Polymorphism:** Overridden methods in Electronics.

**Operator Overloading:** For adding products to ShoppingCart and comparing products.