E-commerce Product Management System

Step 1: Object-Oriented Analysis (OOA) Model

1. Identify Objects (Nouns):

- Product
- Electronics
- Clothing
- Grocery
- ShoppingCart
- Order
- GenericCatalog
- IDiscount

2. Identify Attributes for Each Object:

- Product: id, name, price, sku
- Electronics: warranty_months
- Clothing: size, on_clearance
- Grocery: expiry_date
- ShoppingCart: items (map of product_id -> (Product, quantity))
- Order: order_id, items, status, created_at
- GenericCatalog: items (vector of Product)
- IDiscount: (no attributes, only behavior)

3. Identify Methods for Each Object:

- Product: getId(), getName(), getBasePrice(), getSku(), finalPrice(), toString()
- Electronics: applyDiscount(), finalPrice(), getType()
- Clothing: applyDiscount(), finalPrice(), toString(), getType()
- Grocery: toString(), getType()
- ShoppingCart: addProduct(), removeProduct(), operator+=, operator+, total(), toString(), clear()
- Order: pay(), ship(), cancel(), total(), statusString(), toString()
- GenericCatalog: add(), getItems(), size()
- IDiscount: applyDiscount() (pure virtual)

4. Identify Inheritance Relationships:

- Electronics inherits Product and implements IDiscount
- Clothing inherits Product and implements IDiscount
- Grocery inherits Product
- Order uses ShoppingCart (composition)
- GenericCatalog is a template class for collections of Products

Explanation of Class Design

The system is designed around a base Product class, with Electronics, Clothing, and Grocery as derived classes. Electronics and Clothing implement the IDiscount interface to provide polymorphic discount handling. ShoppingCart manages multiple products using operator overloading (+, +=) to add items conveniently. Order represents a snapshot of a cart with a unique order ID, status transitions (Created, Paid, Shipped, Cancelled), and total calculation. GenericCatalog is implemented as a template class to store collections of products in a type-safe way.

Code Walkthrough

Key parts of the implementation: - Pure virtual function: IDiscount::applyDiscount ensures consistent discount behavior across discountable products. - Operator overloading: ShoppingCart uses operator+= and operator+ for intuitive product addition. - Templates: GenericCatalog

demonstrates generic programming by storing any product type. - Polymorphism: Product is extended by specialized product classes, and their behavior is customized (e.g., discounts, toString).

Test Results

The demo (main) shows: - Adding various products to a catalog and displaying them. - Creating a shopping cart, adding products using both += and + operators. - Generating an order snapshot from the cart, then paying the order. - Totals are correctly calculated with discounts applied (e.g., Electronics 10% off, Clothing clearance 30% off). - Cart is cleared at the end, demonstrating state changes.

LLM Usage

A Large Language Model (LLM) like ChatGPT was used to assist in brainstorming template class design and operator overloading approaches. For example, prompts such as 'Suggest a template class for inventory in C++' guided the GenericCatalog implementation. The code itself was adapted and finalized by the developer.