Working with Object-Oriented Principles — Order Entry Domain

Name: Tykyrah Strickland

Course: Large Scale Programming (Fall 2025)

Date: October 05, 2025

1) Classes

I focused on three main classes that seem most important for an Order Entry system:

Customer

— This represents the person or business buying from us. Customers have a name, address, phone number, and an ID to tell them apart. If the customer is a company, it also keeps track of a contact person and any discount they get. If it's just a person, we use a license number to identify them.

Order

— Each order has details about what was purchased. It keeps the order number, the date, the shipping method (air or ground), the order's current status, and who placed it. An order can have up to 10 items.

Product

— This describes each item we sell, like the product's name, description, price, warranty info, supplier, catalog link, and weight category. It also knows which categories it belongs to (like a main category with subcategories) and which warehouses have it in stock.

2) Attributes

Here's what I think each class would need to store:

Customer
- customerId
- name
- address
- phone
- type (Individual or Company)

- licenseNumber (for Individuals)
- companyContact (for Companies)
- companyDiscountPercent (for Companies)

Order
- orderNumber
- orderDate
- shippingMode (AIR or GROUND)
- status (pending, paid, shipped, etc.)
- customerId (connects to Customer)
- lineItems (list of products, max 10)

Product
- productId
- name
- description
- listPrice
- warrantyPeriodMonths
- supplierId
- catalogUrl
- weightClass (Light, Medium, Heavy)
- categories
- inventoryByWarehouse (stock counts)

3) Behaviors / Methods

Some actions each class might do:

Customer

- applyDiscount(amount) → lowers the price if it's a company with a discount.
- validateContactInfo() → checks that all the right info is filled in (license number for people, contact person for companies).

Order
- addLineItem(product, price, qty, category) → adds a product to the order, but won't go over 10 items.
- getSubtotal() and getTotal() → add up the price of everything, apply discounts, and include shipping.
- estimateShippingCost() → figures out cost based on shipping method and weight.
- setStatus(newStatus) → updates the order's stage safely (pending → paid → shipped, etc.).

Product
- isAvailable(qty) → checks stock across all warehouses.
- reserve(qty, warehouse) → sets aside items for an order.
- weightBracket() → helps with shipping cost.

4) Inheritance and Relationships

It makes sense to have Customer as a general idea and then make two subtypes:

- IndividualCustomer — has a license number.
- CompanyCustomer — has a contact person and a discount percent.

Other relationships:
- Orders are made by one customer and can hold up to 10 items.
- Each item in an order is linked to a product.
- Products can belong to categories, and categories can be single or broken down into smaller parts.
- Products know which warehouses have them available.