In [3]:
```python
class Product:
    total_products = 0  # Class variable

    def __init__(self, name, price, stock):
        self.name = name
        self.price = price
        self.stock = stock
        Product.total_products += 1

    def update_stock(self, quantity):
        self.stock += quantity

    # Simulated method overloading
    def display_info(self, detailed=False):
        if detailed:
            print(f"Product: {self.name}, Price: ₹{self.price}, Stock: {self.stock}
        else:
            print(f"{self.name} - ₹{self.price}")

    @staticmethod
    def product_info():
        return "Products are items listed for sale with a name, price, and stock co

    @classmethod
    def get_total_products(cls):
        return cls.total_products


class Customer:
    customer_count = 0  # Class variable

    def __init__(self, name, email):
        self.name = name
        self.email = email
        self.order_history = []
        Customer.customer_count += 1

    def place_order(self, order):
        self.order_history.append(order)

    @staticmethod
    def customer_info():
        return "Customers can place orders and maintain an order history."

    @classmethod
    def get_customer_count(cls):
        return cls.customer_count


class Order:
    order_count = 0  # Class variable

    def __init__(self, order_id, customer):
        self.order_id = order_id
```

```
        self.customer = customer
        self.products = {}
        Order.order_count += 1

    def add_product(self, product, quantity):
        if product.stock >= quantity:
            self.products[product] = quantity
            product.update_stock(-quantity)
        else:
            print(f"Insufficient stock for {product.name}!")

    @staticmethod
    def order_info():
        return "Orders contain a unique ID, customer, and product list with quantit

    @classmethod
    def get_order_count(cls):
        return cls.order_count
# Creating products
p1 = Product("Laptop", 55000, 10)
p2 = Product("Mouse", 500, 100)

# Display info with simulated overloading
p1.display_info()            # Basic
p1.display_info(detailed=True)  # Detailed

# Creating customer
c1 = Customer("Alice", "alice@example.com")

# Creating order
o1 = Order("ORD001", c1)
o1.add_product(p1, 2)
o1.add_product(p2, 5)

# Place order for customer
c1.place_order(o1)

# Check static method usage
print(Product.product_info())
print(Customer.customer_info())
print(Order.order_info())

# Check class method usage
print("Total Products:", Product.get_total_products())
print("Total Customers:", Customer.get_customer_count())
print("Total Orders:", Order.get_order_count())

# Optional: View customer's order history
print(f"\nCustomer: {c1.name}")
for order in c1.order_history:
    print(f"Order ID: {order.order_id}")
    for prod, qty in order.products.items():
        print(f"- {prod.name} x{qty}")
```

```
Laptop - ₹55000
Product: Laptop, Price: ₹55000, Stock: 10
Products are items listed for sale with a name, price, and stock count.
Customers can place orders and maintain an order history.
Orders contain a unique ID, customer, and product list with quantities.
Total Products: 2
Total Customers: 1
Total Orders: 1

Customer: Alice
Order ID: ORD001
- Laptop x2
- Mouse x5
```

In [ ]: