E-commerce System with Product Hierarchy

---

1. PROJECT OVERVIEW

Project Title

E-Commerce System with Product Hierarchy using Advanced OOP Concepts

Project Goal

To design and implement a console-based E-Commerce system using advanced Object-Oriented Programming concepts including:

- Inheritance
- Polymorphism
- Abstraction
- Interfaces
- Static vs Instance members
- Package organization
- Order management system

Learning Objectives

- Understand class hierarchy design
- Apply polymorphism in real-world systems
- Implement abstract classes and interfaces
- Manage multi-package Java projects
- Develop modular, scalable architecture

---

2. SETUP INSTRUCTIONS

Requirements

- Java JDK 17 or above
- VS Code / IntelliJ / Eclipse
- Java Extension Pack (if using VS Code)

Installation Steps

Step 1: Install Java

Download and install JDK from Oracle website.

Verify installation:

java -version

Step 2: Project Folder Structure

Create this structure:

E-commerce System

├── products/

├── cart/

├── customers/

├── orders/

├── payments/

└── ECommerceSystem.java

Step 3: Compile

From root folder:

javac */*.java ECommerceSystem.java

Step 4: Run

java ECommerceSystem

---

3. CODE STRUCTURE

Package Hierarchy

products

| ├── Product.java (Abstract)

| ├── ElectronicsProduct.java

| ├── ClothingProduct.java

| └── BookProduct.java

|

cart

| ├── CartItem.java

| └── ShoppingCart.java

|

customers

| └── Customer.java

|

payments

| ├── PaymentMethod.java (Interface)

| ├── CreditCardPayment.java

| └── UpiPayment.java

|

orders

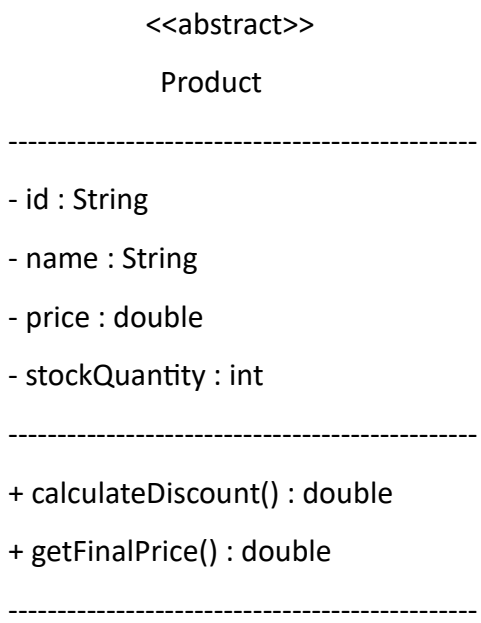
| ├── Order.java

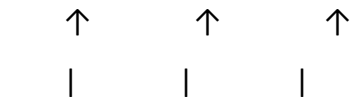
| └── OrderManager.java

└── ECommerceSystem.java (Main Program)

---

4. UML CLASS DIAGRAM





Customer

- id : String
- name : String
- email : String

ShoppingCart

- items : List<CartItem>

+ addProduct()
+ removeProduct()
+ updateQuantity()
+ getTotalAmount()

Order

- orderId : String
- customer : Customer
- finalAmount : double

OrderManager

- orders : List<Order>

<<interface>>

PaymentMethod

+ pay(amount)

CreditCardPayment implements PaymentMethod

UpiPayment implements PaymentMethod

---

5. INHERITANCE HIERARCHY

Product (Abstract)

```
    |
    ├── ElectronicsProduct
    ├── ClothingProduct
    └── BookProduct
```

This demonstrates **IS-A relationship**:

- ElectronicsProduct IS-A Product
- ClothingProduct IS-A Product
- BookProduct IS-A Product

---

6. POLYMORPHISM EXAMPLES

Example 1: Product Polymorphism

```
List<Product> products = new ArrayList<>();

products.add(new ElectronicsProduct(...));
products.add(new ClothingProduct(...));
```

products.add(new BookProduct(...));

All treated as Product.

Example 2: Interface Polymorphism

PaymentMethod payment;

payment = new CreditCardPayment();

payment = new UpiPayment();

Both share same method:

payment.pay(amount);

This is runtime polymorphism.

---

7. SYSTEM ARCHITECTURE

Architecture Type

Layered Architecture

Presentation Layer

↓

Business Logic Layer

↓

Data Layer (In-memory)

Flow

User → ShoppingCart → Order → OrderManager

---

8. DATA STRUCTURES USED

Structure	Usage
ArrayList	Store products
ArrayList	Store cart items
ArrayList	Store order history

Why ArrayList?

- Dynamic resizing

- Efficient traversal
- Suitable for small-medium datasets

---

9. ALGORITHMS USED

1. Discount Calculation

Each subclass overrides:

calculateDiscount()

Formula:

discount = price × percentage

---

2. GST Calculation

finalAmount = subtotal × 1.18

---

3. Cart Update Logic

Loop through items:

```
for each item:
   if productId matches:
      update quantity
```

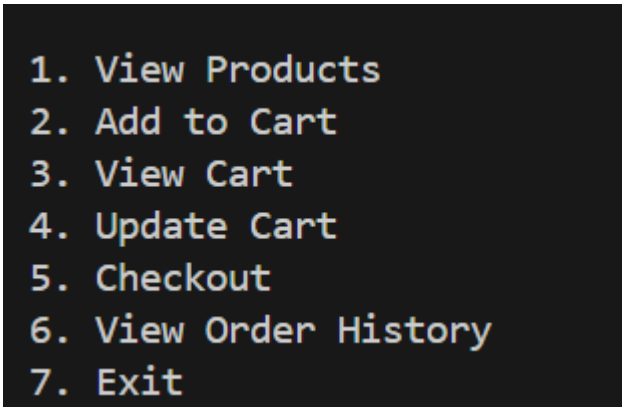
Time Complexity: O(n)

---

10. VISUAL DOCUMENTATION (Screenshots Required)

You should take screenshots of:

1. Main menu display



2. View Cart

```
Enter Your Choice: 1
ID: E1
Name: Smartphone
Price: ?50000.0
Discount: ?5000.0
Final Price: ?45000.0
Type: Electronics
Brand: Samsung
Warranty: 24 months
---------------------------
ID: C1
Name: T-Shirt
Price: ?1200.0
Discount: ?180.0
Final Price: ?1020.0
Type: Clothing
Size: M
Color: Blue
---------------------------
ID: B1
Name: Java Programming
Price: ?800.0
Discount: ?40.0
Final Price: ?760.0
Type: Book
Author: John Doe
ISBN: ISBN123
---------------------------
```

3. Add to Cart

```
Enter Your Choice: 2
Enter Product ID: C1
Enter Quantity: 3
Added to cart!

1. View Products
2. Add to Cart
3. View Cart
4. Update Cart
5. Checkout
6. View Order History
7. Exit
Enter Your Choice: 2
Enter Product ID: B1
Enter Quantity: 2
Added to cart!
```

4. Update Cart

```
1. View Products
2. Add to Cart
3. View Cart
4. Update Cart
5. Checkout
6. View Order History
7. Exit
Enter Your Choice: 4
1. Update Quantity
2. Remove Item
1
Enter Product ID: B1
Enter New Quantity: 1
Quantity updated.
```

5. Checkout

```
Enter Your Choice: 5
1. Credit Card
2. UPI
Paid ?110707.59999999999 using UPI.
Order ID: ORD1000
Customer ID: CUST1
Name: Rahul
Email: rahul@email.com
Final Amount (incl GST): ?110707.59999999999
Order Placed Successfully!
Cart cleared after checkout.
```

6. Order history display

```
Enter Your Choice: 6

===== ORDER HISTORY =====
Order ID: ORD1000
Customer: Rahul
Final Amount: ?110707.59999999999
----------------------------
```

11. TESTING EVIDENCE

Test Case 1: Add Product to Cart

Input:

2 → E1 → 1

Expected:

Added to cart!

Test Case 2: Checkout

Input:

5 → Credit Card

Expected:

Order Placed Successfully!

Test Case 3: Empty Cart Checkout

Input:

5

Expected:

Cart is empty!

Test Case 4: Order History

Expected:

===== ORDER HISTORY =====

Order ID: ORD1000

Customer: Rahul

Final Amount: ₹XXXX

📈 12. QUALITY STANDARDS CHECKLIST

Requirement	Status
Inheritance Used	✅
Polymorphism Implemented	✅

Requirement	Status
Abstract Class	✅
Interface	✅
Static Members	✅
Package Structure	✅
UML Diagram	✅
Testing Evidence	✅
Documentation	✅

13. CONCLUSION

This project successfully demonstrates advanced Object-Oriented Programming principles in a real-world E-Commerce scenario.

It shows:

- Modular design
- Reusability
- Scalability
- Clean architecture
- Industry-standard coding structure