

Actividad PostContenido 2 - Unidad 1

Esteban Mauricio Calderón Bayona

Universidad de Santander

Patrones de Diseño

Jhonatan Castillo

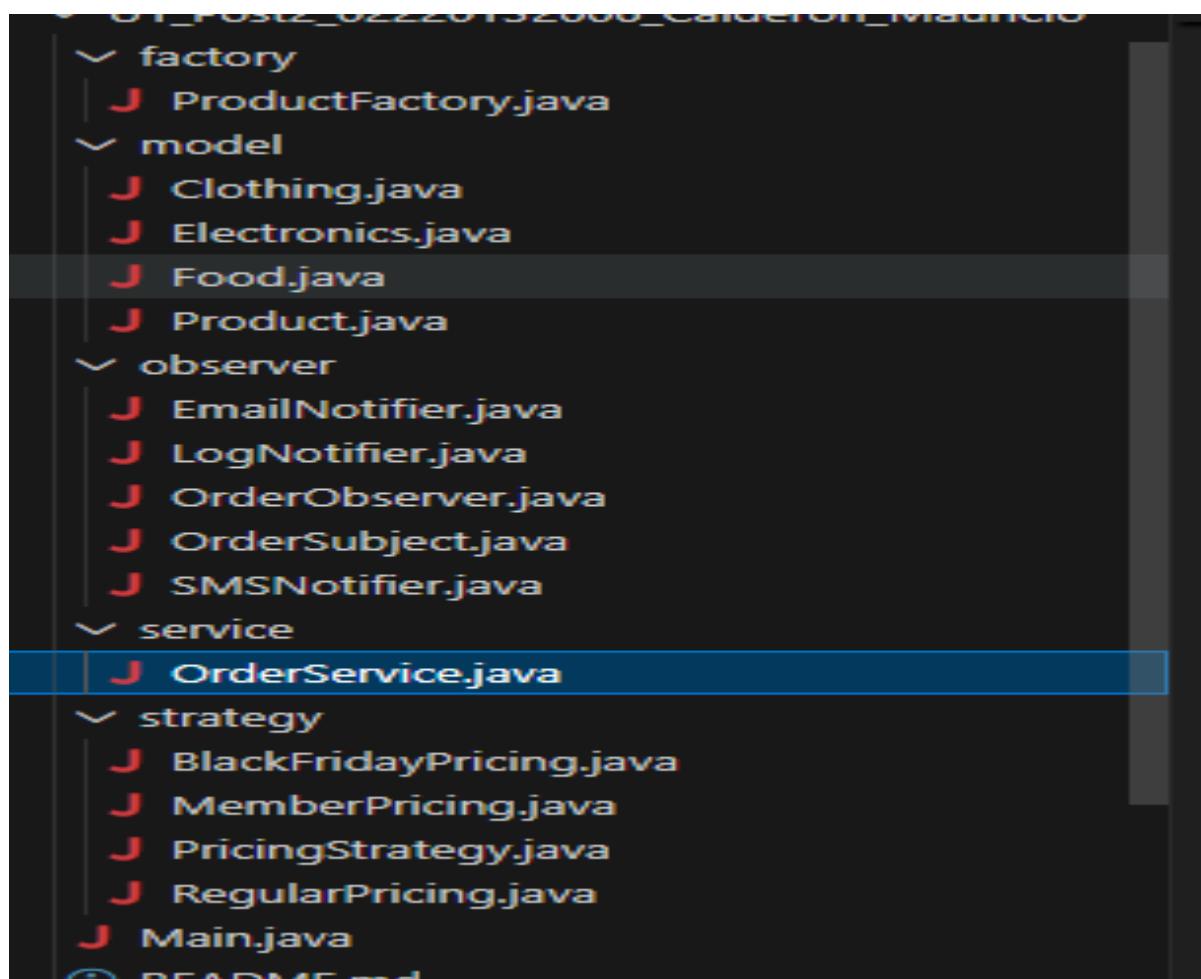
21/02/2026

CODIGO

https://github.com/maocalderon/U1_Post2_02220132006_Calderon_Mauricio.git

CAPTURES

Estructura del Proyecto:



Aca podemos evidenciar los paquetes y clases que cree para ejecutar el código

Implementacion de Patrones

```
U1_Post2_02220132006_Calderon_Mauricio > factory > J ProductFactory.java > ProductFactory
1 package factory;
2 import model.*; // Esto importa todas las clases del paquete model
3
4 public class ProductFactory {
5     public static Product createProduct(String type, String name, double price, int warranty) {
6         return switch (type.toUpperCase()) {
7             case "ELECTRONICS" -> new Electronics(name, price, warranty);
8             case "CLOTHING" -> new Clothing(name, price, size: "M");
9             case "FOOD" -> new Food(name, price, days: 30);
10            default -> throw new IllegalArgumentException("Tipo desconocido");
11        };
12    }
13 }
```

```
01_Post2_02220132000_Calderon_Mauricio / service > OrderService.java > OrderService
  8  public class OrderService implements OrderSubject { // [cite: 140]
12      public void processOrder(Product product, PricingStrategy strategy) {
13          double finalPrice = strategy.calculateFinalPrice(product.getBasePrice());
14          System.out.println("Procesando: " + product.getDescription());
15          System.out.println("Estrategia aplicada: " + strategy.getDescription());
16          System.out.println("Total con envío: $" + (finalPrice + product.getShippingCost()));
17      }
18
19      public void changeStatus(String orderId, String newStatus) {
20          String oldStatus = this.status;
21          this.status = newStatus;
22          notifyObservers(orderId, oldStatus, newStatus); // [cite: 53, 63]
23      }
24
25      @Override
26      public void subscribe(OrderObserver observer) { observers.add(observer); }
27
28      @Override
29      public void unsubscribe(OrderObserver observer) { observers.remove(observer); }
30
31      @Override
32      public void notifyObservers(String orderId, String oldS, String newS) {
33          for (OrderObserver o : observers) {
34              o.update(orderId, oldS, newS); // [cite: 64]
35          }
36      }

```

Ejecución del código

```
16  
17     System.out.println(x: "--- INICIO DE PRUEBA E-COMMERCE UDES ---  
18  
19         // 3. Creacion de productos usando Factory (Punto a de la guia)  
20         Product laptop = ProductFactory.createProduct(type: "ELECTRONIC")  
21         Product camisa = ProductFactory.createProduct(type: "CLOTHING")  
22         Product snack = ProductFactory.createProduct(type: "FOOD", name:  
23  
24             // 4. Calculo de precios con diferentes estrategias (Punto b de la guia)  
25             ...  
26  
PROBLEMAS 9 SALIDA TERMINAL ... Debug: Main + v
```

Procesando: Camiseta Polo [Ropa] Talla: M
Estrategia aplicada: Descuento de Miembro (10%)
Total con envío: \$81500.0

--- Procesando Pedido 3 ---
Procesando: Barra de Cereal [Alimento] Vence en: 30 días
Estrategia aplicada: Precio regular (Sin descuento)
Total con envío: \$5500.0

--- Actualización de Estados (Notificaciones) ---
[EMAIL] ? Enviando correo para el pedido ORD-001: Pasó de CREATED a PROCESSING
[SMS] ? Pedido ORD-001: Nuevo estado -> PROCESSING
[LOG] ? Pedido ORD-001: Cambio de CREATED a PROCESSING
[EMAIL] ? Enviando correo para el pedido ORD-001: Pasó de PROCESSING a SHIPPED
[SMS] ? Pedido ORD-001: Nuevo estado -> SHIPPED
[LOG] ? Pedido ORD-001: Cambio de PROCESSING a SHIPPED
[EMAIL] ? Enviando correo para el pedido ORD-001: Pasó de SHIPPED a DELIVERED
[SMS] ? Pedido ORD-001: Nuevo estado -> DELIVERED
[LOG] ? Pedido ORD-001: Cambio de SHIPPED a DELIVERED

--- PRUEBA FINALIZADA CON ÉXITO ---
PS C:\Users\USUARIO\Desktop>