PROGRAMACIÓN ORIENTADA A OBJETOS

Ayudantía 11

Semestre I – 2024

ICCI - ITI

Docentes: **Eric Ross Cortés**
Moisés Moraga
Alejandro Paolini

Ayudantes: **María Victoria Quiroga**
Vicente Briceño
Javiera Berríos

Actividad:

Eres un desarrollador para un Ecommerce (tienda de productos online) este cuenta con productos, clientes y pedidos. Los productos pueden ser de dos tipos: **computadores** o **ropa** (indicados por COMPUTER o CLOTHING respectivamente), estos productos se encuentran guardados en un archivo de texto llamado "*productos.txt*" con la siguiente estructura:

`COMPUTER,nombre,marca,precio,almacenamiento`

`CLOTHING,nombre,marca,precio,talla`

Mediante el uso de patrones de diseño, el sistema requiere lo siguiente:

- ✓ Una configuración global que sea instanciada una única vez con los siguientes atributos:

```
private String rutaDB = "jdbc:mysql://localhost:3306/mydatabase";
private String hostServidor = "http://localhost:8080";
private String operacion = "dev";
private String temaUI = "dark";
```

- ✓ La utilización de **Factory** para la creación de productos (computadores o prendas) con **Flyweight** que retorne el mismo objeto en caso de que ya esté creado.
- ✓ Implementación de un **Visitor** para calcular y definir el nuevo precio de los productos según su descuento. El descuento para computadores es de 10% y el descuento para prendas es de 20%.
- ✓ Implementación de un **Observer** que notifique por consola a todos los clientes con este cuando el estado de un pedido cambie.
- ✓ Implementación de **Strategy** para definir los envíos de los pedidos como estándar o express.

Consideraciones:

- ✓ **Cliente** solo tiene de atributo un nombre, se crea en la aplicación e implementa **Observer**.
- ✓ **Pedido** tiene asociado un **Producto**, estado (Ejemplo: "en camino", "entregado", etc.), días restantes para la entrega (entero) y al modificarse el estado se notifica a los **Observers** asociados.
- ✓ **Strategy** se compone de una interfaz **PedidoProcessingStrategy** con la función `procesarPedido(Pedido pedido)`, un **PedidoProcessor** con la estrategia a utilizar, **PedidoEstandarProcessing** y **PedidoExpressProcessing** que definen los días de entrega del pedido (3 para express y 7 para estándar).

Se pide:

- Modelamiento
- Código de la aplicación

Ejemplo de Main:

```
ArrayList<Producto> productos = new ArrayList<Producto>();
leerArchivo(productos);

// Mostrar configuración
Configuracion config = Configuracion.getInstancia();
System.out.println(config.toString());

// Aplicar descuento
DescuentoVisitor descuentoVisitor = new DescuentoVisitor();
for (Producto producto : productos) {
	producto.aceptar(descuentoVisitor);
}

// Obtener productos
Producto producto1 = productos.get(0);
Producto producto2 = productos.get(5);

// Crear pedido
Pedido pedido1 = new Pedido(producto1);
Pedido pedido2 = new Pedido(producto2);
Cliente cliente1 = new Cliente("Alice");
Cliente cliente2 = new Cliente("Bob");

// Procesar pedido
PedidoProcessor pedidoProcessor = new PedidoProcessor();
pedidoProcessor.setStrategy(new PedidoEstandarProcessing());
pedidoProcessor.procesar(pedido1);

pedidoProcessor.setStrategy(new PedidoExpressProcessing());
pedidoProcessor.procesar(pedido2);

// Notificar a clientes del envío
pedido1.addObserver(cliente1);
pedido1.addObserver(cliente2);
pedido1.setEstado("En camino");
```