



# **TECNOLOGICO NACIONAL DE MEXICO**

## **INSTITUTO TECNOLÓGICO DE OAXACA**

### **Carrera:**

Ingeniería en Sistemas Computacionales

### **Materia:**

Diseño e Implementación de software con patrones

### **Documentación de patrón Interpreter**

### **Docente:**

Espinoza Pérez Jacob

### **Equipo:**

Ordaz Pacheco Ruudvan

Santos Manuel Julia Marlenny

Vera Acevedo Héctor Aramís

### **Grupo:**

7SB

### **Fecha de entrega:**

11/05/2025

# Documentación del Patrón Interpreter en el Sistema de Ventas

## Introducción

Este documento explica la implementación del **patrón Interpreter** para gestionar descuentos en un sistema de ventas Java con MySQL. El objetivo es permitir:

- Aplicar **descuentos automáticos** (por porcentaje, cantidad o cliente VIP).
- Mantener un **código limpio y escalable**.
- Facilitar la **adición de nuevas reglas** sin modificar la lógica principal.

## Interfaz ExpresionDescuento

```
public interface ExpresionDescuento {  
    double calcularDescuento(double precio, int cantidad);  
    String getDescripcion();  
}
```

## Estrategias de Descuento

### Descuento por Porcentaje (DescuentoPorcentaje)

```
public class DescuentoPorcentaje implements ExpresionDescuento {  
    private double porcentaje;  
  
    public DescuentoPorcentaje(double porcentaje) {  
        this.porcentaje = porcentaje;  
    }  
  
    @Override  
    public double calcularDescuento(double precio, int cantidad) {  
        return (precio * cantidad) * (porcentaje / 100);  
    }  
  
    @Override  
    public String getDescripcion() {  
        return porcentaje + "% de descuento";  
    }  
}
```

### Descuento por Cantidad (DescuentoPorCantidad)

```
public class DescuentoPorCantidad implements ExpresionDescuento {  
    private int cantidadMinima;  
    private double descuentoFijo;  
  
    public DescuentoPorCantidad(int cantidadMinima, double descuentoFijo) {  
        this.cantidadMinima = cantidadMinima;  
        this.descuentoFijo = descuentoFijo;  
    }  
  
    @Override  
    public double calcularDescuento(double precio, int cantidad) {  
        return cantidad >= cantidadMinima ? descuentoFijo : 0;  
    }  
  
    @Override  
    public String getDescripcion() {  
        return "Descuento de $" + descuentoFijo + " por comprar " + cantidadMinima + " o más unidades";  
    }  
}
```

## Descuento VIP (DescuentoClienteVIP)

```
public class DescuentoClienteVIP implements ExpresionDescuento {
    private boolean esVIP;

    public DescuentoClienteVIP(boolean esVIP) {
        this.esVIP = esVIP;
    }

    @Override
    public double calcularDescuento(double precio, int cantidad) {
        return esVIP ? (precio * cantidad) * 0.10 : 0; // 10% de descuento para VIPs
    }

    @Override
    public String getDescripcion() {
        return esVIP ? "10% de descuento VIP" : "Sin descuento VIP";
    }
}
```

## Clase ContextoDescuento

```
public class ContextoDescuento {
    private List<ExpresionDescuento> descuentos = new ArrayList<>();

    public void agregarDescuento(ExpresionDescuento descuento) {
        descuentos.add(descuento);
    }

    public double aplicarDescuentos(double precio, int cantidad) {
        double descuentoTotal = 0;
        for (ExpresionDescuento descuento : descuentos) {
            descuentoTotal += descuento.calcularDescuento(precio, cantidad);
        }
        return descuentoTotal;
    }

    public String getDescripcionDescuentos() {
        StringBuilder sb = new StringBuilder();
        for (ExpresionDescuento descuento : descuentos) {
            sb.append(descuento.getDescripcion()).append("\n");
        }
        return sb.toString();
    }
}
```

## Flujo de Trabajo

### 1. Añadir Producto con Descuentos

2. El usuario selecciona un producto y cantidad.
3. El sistema calcula:
  - **Subtotal:** precioUnitario \* cantidad.
  - **Descuentos:** Usando ContextoDescuento.
  - **IVA:** Según el porcentaje del producto.

2. Se actualiza la tabla de venta.

```
private void jButton_añadir_productoActionPerformed(ActionEvent evt) {  
    configurarDescuentos();  
    double descuento = contextoDescuento.aplicarDescuentos(precioUnitario, cantidad);  
    double total = (precioUnitario * cantidad) - descuento + iva;  
}
```


### ◆ 2. Registrar Venta

1. Se registra la cabecera en tb\_cabecera\_venta.
2. Por cada producto:
  - Se aplican descuentos.
  - Se guarda el detalle en tb\_detalle\_venta.
  - Se actualiza el stock.

```
private void jButton_RegistrarVentaActionPerformed(ActionEvent evt) {  
    CabeceraVenta cabecera = new CabeceraVenta(idCliente, totalPagar, fecha);  
    ctrlVenta.guardar(cabecera);  
    for (DetalleVenta detalle : listaProductos) {  
        ctrlVenta.guardarDetalle(detalle);  
        actualizarStock(detalle);  
    }  
}
```

## Resultados

Message

 ✓ Venta registrada y factura generada

OK



NOMBRE: Miscelanea Calicanto  
TELEFONO: 9547894569  
DIRECCION: Av Montoya  
RAZON SOCIAL: Apoyando a la  
economia oaxaqueña

Nota: 022  
Fecha: 2025/05/11

Datos del cliente:

<b>Cedula/RUC:</b>	<b>Nombre:</b>	<b>Telefono:</b>	<b>Direccion:</b>
9581732455	Teresa Hernandez	9581732455	Privada Texcoco

<b>Cantidad:</b>	<b>Descripcion:</b>	<b>Precio Unitario:</b>	<b>Precio Total:</b>
10	Cerveza	25.0	227.5
3	CocaCola 600ml	22.0	31.25

Total a pagar: 258.75

¡Gracias por su compra!

## Conclusión

La implementación del patrón **Interpreter** para gestionar descuentos en el sistema de ventas representa una solución robusta y escalable que aborda las necesidades actuales del negocio mientras sienta las bases para futuras expansiones. A continuación, se presenta un análisis detallado de los beneficios, lecciones aprendidas y oportunidades de mejora:

- Flexibilidad para añadir reglas.
- Código mantenible.
- Descuentos configurables sin cambiar la lógica principal.