



# **TECNOLOGICO NACIONAL DE MEXICO**

## **INSTITUTO TECNOLÓGICO DE OAXACA**

### **Carrera:**

Ingeniería en Sistemas Computacionales

### **Materia:**

Diseño e Implementación de software con patrones

### **Documentación de patrón Flyweight**

### **Docente:**

Espinoza Pérez Jacob

### **Equipo:**

Ordaz Pacheco Ruudvan

Santos Manuel Julia Marlenny

Vera Acevedo Héctor Aramís

### **Grupo:**

7SB

### **Fecha de entrega:**

07/04/2025

# **Documentación del Patrón Flyweight en el Sistema de Ventas**

## **Introducción**

El patrón Flyweight es un patrón de diseño estructural que permite minimizar el uso de memoria compartiendo la mayor cantidad de datos posible entre objetos similares. En el sistema de gestión de ventas, este patrón se implementó para optimizar el manejo de productos en las ventas, reduciendo significativamente el consumo de memoria cuando se procesan múltiples ventas con los mismos productos.

## Interfaz ProductFlyweight

La clase ProductFlyweight encapsula los datos intrínsecos (compartidos) de los productos:

### Responsabilidades:

- Almacenar datos invariables de productos (nombre, precio, IVA, etc.)
- Proporcionar acceso seguro a los datos compartidos
- Garantizar inmutabilidad para seguridad en entornos concurrentes

```
public class ProductFlyweight {  
    private final int idProducto;  
    private final String nombre;  
    private final double precio;  
    private final int porcentajeIva;  
    private final String descripcion;  
    private final String categoria;  
  
    public ProductFlyweight(int idProducto, String nombre, double precio,  
        int porcentajeIva, String descripcion, String categoria) {  
        this.idProducto = idProducto;  
        this.nombre = nombre;  
        this.precio = precio;  
        this.porcentajeIva = porcentajeIva;  
        this.descripcion = descripcion;  
        this.categoria = categoria;  
    }  
  
    // Getters  
    public int getIdProducto() { return idProducto; }  
    public String getNombre() { return nombre; }  
    public double getPrecio() { return precio; }  
    public int getPorcentajeIva() { return porcentajeIva; }  
    public String getDescripcion() { return descripcion; }  
    public String getCategoria() { return categoria; }  
}
```

## Clase ProductFlyweightFactory (Fábrica Flyweight)

Gestiona la creación y reutilización de instancias Flyweight:

### Flujo de trabajo:

1. Recibe solicitud de producto con parámetros completos
2. Verifica si ya existe un Flyweight para ese ID de producto
3. Reutiliza la instancia existente o crea una nueva
4. Devuelve la referencia al Flyweight

```
import java.util.HashMap;
import java.util.Map;

public class ProductFlyweightFactory {
    private static final Map<Integer, ProductFlyweight> flyweights = new HashMap<>();

    public static ProductFlyweight getFlyweight(int idProducto, String nombre, double precio,
                                                int porcentajeIva, String descripcion, String categoria) {
        ProductFlyweight flyweight = flyweights.get(idProducto);

        if (flyweight == null) {
            flyweight = new ProductFlyweight(idProducto, nombre, precio, porcentajeIva, descripcion, categoria);
            flyweights.put(idProducto, flyweight);
        }

        return flyweight;
    }

    public static int getFlyweightCount() {
        return flyweights.size();
    }
}
```

## Modificaciones en DetalleVenta

La clase DetalleVenta fue adaptada para:

- Contener referencia al ProductFlyweight
- Almacenar solo datos extrínsecos (cantidad, descuentos, etc.)
- Mantener compatibilidad con el sistema existente

```
public class DetalleVenta {  
    private int idDetalleVenta;  
    private int idCabeceraVenta;  
    private ProductFlyweight productFlyweight;  
    private int cantidad;  
    private double subTotal;  
    private double descuento;  
    private double iva;  
    private double totalPagar;  
    private int estado;  
  
    // Constructor con Flyweight  
    public DetalleVenta(int idDetalleVenta, int idCabeceraVenta,  
                        ProductFlyweight productFlyweight, int cantidad,  
                        double subTotal, double descuento, double iva,  
                        double totalPagar, int estado) {  
        this.idDetalleVenta = idDetalleVenta;  
        this.idCabeceraVenta = idCabeceraVenta;  
        this.productFlyweight = productFlyweight;  
        this.cantidad = cantidad;  
        this.subTotal = subTotal;  
        this.descuento = descuento;  
        this.iva = iva;  
        this.totalPagar = totalPagar;  
        this.estado = estado;  
    }  
  
    // Getters  
    public int getIdDetalleVenta() { return idDetalleVenta; }  
    public int getIdCabeceraVenta() { return idCabeceraVenta; }  
    public int getCantidad() { return cantidad; }  
    public double getSubTotal() { return subTotal; }  
    public double getDescuento() { return descuento; }  
    public double getIva() { return iva; }  
    public double getTotalPagar() { return totalPagar; }  
    public int getEstado() { return estado; }  
    public ProductFlyweight getProductFlyweight() { return productFlyweight; }
```

```

// Métodos de compatibilidad
public int getIdProducto() { return productFlyweight.getIdProducto(); }
public String getNombre() { return productFlyweight.getNombre(); }
public double getPrecioUnitario() { return productFlyweight.getPrecio(); }

// Setters
public void setIdDetalleVenta(int idDetalleVenta) { this.idDetalleVenta = idDetalleVenta; }
public void setIdCabeceraVenta(int idCabeceraVenta) { this.idCabeceraVenta = idCabeceraVenta; }
public void setCantidad(int cantidad) { this.cantidad = cantidad; }
public void setSubTotal(double subTotal) { this.subTotal = subTotal; }
public void setDescuento(double descuento) { this.descuento = descuento; }
public void setIva(double iva) { this.iva = iva; }
public void setTotalPagar(double totalPagar) { this.totalPagar = totalPagar; }
public void setEstado(int estado) { this.estado = estado; }
}

```

## Adaptación de InterFacturacion

La interfaz gráfica fue modificada para:

1. Obtener datos completos del producto al añadirlo
2. Crear Flyweights a través de la fábrica
3. Construir DetalleVenta con referencia al Flyweight

```

String combo = this.jComboBox_producto.getSelectedItem().toString();
if (combo.equalsIgnoreCase("Seleccione producto:")) {
    JOptionPane.showMessageDialog(null, "Seleccione un producto");
} else {
    if (!txt_cantidad.getText().isEmpty()) {
        boolean validacion = validar(txt_cantidad.getText());
        if (validacion == true) {
            if (Integer.parseInt(txt_cantidad.getText()) > 0) {
                cantidad = Integer.parseInt(txt_cantidad.getText());
                this.DatosDelProducto();

                if (cantidad <= cantidadProductoBBDD) {
                    subtotal = precioUnitario * cantidad;
                    totalPagar = subtotal + iva + descuento;

                    // Obtener datos adicionales para el Flyweight
                    String descripcion = "";
                    String categoria = "";
                    try {
                        Connection cn = Conexion.getConnection();
                        String sql = "SELECT p.descripcion, c.descripcion as categoria " +
                                    "FROM tb_producto p JOIN tb_categoria c ON p.idCategoria = c.idCategoria " +
                                    "WHERE p.idProducto = ?";
                        PreparedStatement pst = cn.prepareStatement(sql);
                        pst.setInt(1, idProducto);
                        ResultSet rs = pst.executeQuery();
                        if (rs.next()) {
                            descripcion = rs.getString("descripcion");
                            categoria = rs.getString("categoria");
                        }
                        Conexion.cerrarConexion();
                    } catch (SQLException e) {
                        System.out.println("Error al obtener datos del producto: " + e);
                    }

                    // Crear Flyweight
                    ProductFlyweight flyweight = ProductFlyweightFactory.getFlyweight(
                        idProducto, nombre, precioUnitario, porcentajeIva, descripcion, categoria);

                    // Crear detalle de venta con Flyweight
                    producto = new DetalleVenta(
                        auxIdDetalle, 1, flyweight, cantidad,
                        subtotal, descuento, iva, totalPagar, 1
                    );

                    listaProductos.add(producto);
                    JOptionPane.showMessageDialog(null, "Producto Agregado");
                    auxIdDetalle++;
                    txt_cantidad.setText("");
                    this.CargarComboProductos();
                    this.CalcularTotalPagar();
                    txt_efectivo.setEnabled(true);
                    jButton_calcular_cambio.setEnabled(true);
                } else {
                    JOptionPane.showMessageDialog(null, "La cantidad supera el Stock");
                }
            }
        }
    }
}

this.listaTablaProductos();

```

## Adaptación de Ctrl\_RegistrarVenta

El controlador mantuvo su funcionalidad, pero ahora:

- Trabaja con DetalleVenta que contiene Flyweights
- Almacena en BD solo la referencia al producto (ID)
- Recupera datos compartidos cuando es necesario

```
public class Ctrl_RegistrarVenta {

    //ultimo id de la cabecera
    public static int idCabeceraRegistrada;
    java.math.BigDecimal iDColVar;
    /**
     * *****
     * metodo para guardar la cabecera de venta
     * *****
     */
    public boolean guardar(CabeceraVenta objeto) {
        boolean respuesta = false;
        Connection cn = Conexion.getConnection();
        try {
            PreparedStatement consulta = cn.prepareStatement("insert into tb_cabecera_venta values(?, ?, ?, ?, ?)",
                Statement.RETURN_GENERATED_KEYS);
            consulta.setInt(1, 0); //id
            consulta.setInt(2, objeto.getIdCliente());
            consulta.setDouble(3, objeto.getValorPagar());
            consulta.setString(4, objeto.getFechaVenta());
            consulta.setInt(5, objeto.getEstado());

            if (consulta.executeUpdate() > 0) {
                respuesta = true;
            }

            ResultSet rs = consulta.getGeneratedKeys();
            while(rs.next()){
                iDColVar = rs.getBigDecimal(1);
                idCabeceraRegistrada = iDColVar.intValue();
            }

            Conexion.cerrarConexion();
        } catch (SQLException e) {
            System.out.println("Error al guardar cabecera de venta: " + e);
        }
        return respuesta;
    }
}
```



```

public boolean guardarDetalle(DetalleVenta objeto) {
    boolean respuesta = false;
    Connection cn = Conexion.getConnection();
    try {
        PreparedStatement consulta = cn.prepareStatement(
            "insert into tb_detalle_venta values(?,?,?,?,?,?,?,?,?,?)");

        consulta.setInt(1, 0); // id
        consulta.setInt(2, idCabeceraRegistrada);
        consulta.setInt(3, objeto.getIdProducto()); // Del Flyweight
        consulta.setInt(4, objeto.getCantidad());
        consulta.setDouble(5, objeto.getPrecioUnitario()); // Del Flyweight
        consulta.setDouble(6, objeto.getSubTotal());
        consulta.setDouble(7, objeto.getDescuento());
        consulta.setDouble(8, objeto.getIva());
        consulta.setDouble(9, objeto.getTotalPagar());
        consulta.setInt(10, objeto.getEstado());

        if (consulta.executeUpdate() > 0) {
            respuesta = true;
        }
        Conexion.cerrarConexion();
    } catch (SQLException e) {
        System.out.println("Error al guardar detalle de venta: " + e);
    }
    return respuesta;
}

```

#### Ventajas del Patrón Flyweight en este Código

1. **Optimización de memoria:** Reduce drásticamente el consumo cuando hay muchos productos repetidos en ventas
2. **Consistencia de datos:** Todos los detalles de venta comparten la misma información de producto
3. **Rendimiento mejorado:** Minimiza acceso a BD para datos de productos
4. **Escalabilidad:** Permite manejar grandes volúmenes de ventas eficientemente
5. **Mantenibilidad:** Centraliza la lógica de productos en un solo lugar

## Resultados

Facturacion

**Facturación**

Ciente:

Seleccione producto:

Cantidad:

N	Nombre	Cantidad	P. Unitario	SubTotal	Descuento	Iva	Total Pagar	Accion
1	Cerveza	20	25.0	500.0	0.0	0.0	500.0	Eliminar
2	Sabritas Cl...	2	18.0	36.0	0.0	0.0	36.0	Eliminar
3	BigCola	10	23.0	230.0	0.0	0.0	230.0	Eliminar



Registrar Venta

Subtotal:

766.0

Descuento:

0.0

Iva:

0.0

Total a pagar:


766.0

Efectivo:

Cambio:

234.0

Message



¡Venta Registrada!



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

Nota: 018  
Fecha: 2025/04/05

Datos del cliente:

<b>Cedula/RUC:</b>	<b>Nombre:</b>	<b>Telefono:</b>	<b>Direccion:</b>
9514565656	abdiel Tomas Diaz	9514565656	Av Tecnologico

<b>Cantidad:</b>	<b>Descripcion:</b>	<b>Precio Unitario:</b>	<b>Precio Total:</b>
20	Cerveza	25.0	500.0
2	Sabritas Clasicas	18.0	36.0
10	BigCola	23.0	230.0

Total a pagar: 766.0

¡Gracias por su compra!

## Conclusión

La implementación del patrón Flyweight en el sistema de gestión de ventas ha demostrado ser una solución efectiva para:

1. **Optimizar** el uso de recursos del sistema
2. **Centralizar** la gestión de datos de productos
3. **Mejorar** el rendimiento en operaciones con múltiples ventas
4. **Facilitar** el mantenimiento y extensión del módulo de productos

Este enfoque permite que el sistema maneje eficientemente grandes volúmenes de transacciones manteniendo un diseño limpio y escalable, particularmente útil en entornos con catálogos extensos de productos que se repiten frecuentemente en diferentes ventas.