TECNOLOGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE OAXACA

Carrera:

Ingeniería en Sistemas Computacionales

Materia:

Diseño e Implementación de software con patrones

Documentación

Docente:

Espinoza Pérez Jacob

Equipo:

Ordaz Pacheco Ruudvan

Santos Manuel Julia Marleny

Vera Acevedo Héctor Aramís

Grupo:

7SB

Fecha de entrega:

18/03/2025

Documentación del Patrón Memento para Sistema de Productos

Descripción General

Esta implementación utiliza el patrón de diseño Memento para gestionar los cambios de estado de diferentes tipos de productos (estándar, con descuento y promocionales), permitiendo el seguimiento del historial de cambios y la capacidad de revertir modificaciones. El patrón Memento permite capturar y externalizar el estado interno de un objeto sin violar la encapsulación, posibilitando restaurar el objeto a estados previos.

Estructura del Sistema

Diagrama de Clases

El sistema está compuesto por las siguientes clases principales:

- **ProductoMemento**: Objeto inmutable que almacena el estado de un producto
- **HistorialProductos (Caretaker)**: Gestor del historial de estados de productos
- **Producto (Originator)**: Clase abstracta base para todos los productos
- **Clases concretas de productos**: ProductoEstandar, ProductoConDescuento, ProductoPromocion
- **Ctrl_Producto**: Controlador para operaciones de productos con soporte para historial

Relaciones entre Clases

- **Producto** es el Originator que crea instancias de Memento para guardar su estado
- **ProductoMemento** almacena el estado inmutable de un producto en un momento específico
- **HistorialProductos** actúa como Caretaker, gestionando la colección de mementos por producto
- **Ctrl_Producto** coordina las operaciones de producto y la gestión del historial

Componentes Principales

ProductoMemento

Almacena el estado inmutable de un producto en un momento específico.

```java
public class ProductoMemento {
    private final int idProducto;
    private final String nombre;
    private final int cantidad;
    private final double precio;
    private final String descripcion;
    private final int porcentajeIva;
    private final int idCategoria;
    private final int estado;
    private final String tipo;
    private final Object atributoEspecial; // Para almacenar atributos específicos de cada tipo

    public ProductoMemento(Producto producto) {
        this.idProducto = producto.getIdProducto();
        this.nombre = producto.getNombre();
        this.cantidad = producto.getCantidad();
        this.precio = producto.getPrecio();
        this.descripcion = producto.getDescripcion();
        this.porcentajeIva = producto.getPorcentajeIva();
        this.idCategoria = producto.getIdCategoria();
        this.estado = producto.getEstado();

        // Guardar información específica según el tipo de producto
        if (producto instanceof ProductoEstandar) {
            this.tipo = "estandar";
            this.atributoEspecial = null;
        } else if (producto instanceof ProductoConDescuento) {
            this.tipo = "descuento";
            this.atributoEspecial = ((ProductoConDescuento) producto).getPorcentajeDescuento();
        } else if (producto instanceof ProductoPromocion) {
            this.tipo = "promocion";
            this.atributoEspecial = ((ProductoPromocion) producto).getDescripcionPromocion();
        } else {
            this.tipo = "desconocido";
            this.atributoEspecial = null;
        }
    }
```

```
    }

    // Solo getters - Memento es inmutable
    public int getIdProducto() {
        return idProducto;
    }

    public String getNombre() {
        return nombre;
    }

    public int getCantidad() {
        return cantidad;
    }

    public double getPrecio() {
        return precio;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public int getPorcentajeIva() {
        return porcentajeIva;
    }

    public int getIdCategoria() {
        return idCategoria;
    }

    public int getEstado() {
        return estado;
    }

    public String getTipo() {
        return tipo;
    }

    public Object getAtributoEspecial() {
        return atributoEspecial;
    }
}
```

Atributos:

- Todos los atributos básicos de Producto (idProducto, nombre, cantidad, precio, etc.)
- `tipo`: Identifica el tipo de producto (estándar, descuento, promoción)
- `atributoEspecial`: Almacena cualquier atributo específico del tipo de producto

Características:

- Es inmutable (solo tiene getters, sin setters)
- Captura el estado completo de un producto, incluyendo atributos específicos del tipo

HistorialProductos (Caretaker)

Responsable de almacenar y gestionar el historial de estados de los productos.

```
public class HistorialProductos {
    private final Map<Integer, Stack<ProductoMemento>> historialPorProducto = new HashMap<>();

    // Guarda el estado actual de un producto
    public void guardarEstado(Producto producto) {
        int idProducto = producto.getIdProducto();

        // Verificar si existe historial para este producto
        if (!historialPorProducto.containsKey(idProducto)) {
            historialPorProducto.put(idProducto, new Stack<>());
        }

        // Añadir un nuevo memento al historial
        ProductoMemento memento = new ProductoMemento(producto);
        historialPorProducto.get(idProducto).push(memento);

        System.out.println("Estado guardado para el producto ID: " + idProducto);
    }

    // Restaura el estado previo de un producto
    public boolean restaurarEstado(Producto producto) {
        int idProducto = producto.getIdProducto();

        // Verificar si existe historial para este producto
        if (!historialPorProducto.containsKey(idProducto) || historialPorProducto.get(idProducto).isEmpty()) {
            System.out.println("No hay historial para restaurar para el producto ID: " + idProducto);
            return false;
        }

        // Quitar el estado actual (el más reciente)
        historialPorProducto.get(idProducto).pop();

        // Verificar si quedan estados para restaurar
        if (historialPorProducto.get(idProducto).isEmpty()) {
            System.out.println("No hay estados previos para restaurar para el producto ID: " + idProducto);
            return false;
        }

        // Obtener el estado previo
        ProductoMemento memento = historialPorProducto.get(idProducto).peek();
        //restaurarDesdeMemento(producto, memento);

        System.out.println("Estado restaurado para el producto ID: " + idProducto);
        return true;
    }
```

```
    // Restaura a un estado específico por índice (0 es el más antiguo)
    public boolean restaurarEstadoPorIndice(Producto producto, int indice) {
        int idProducto = producto.getIdProducto();

        // Verificar si existe historial para este producto
        if (!historialPorProducto.containsKey(idProducto) || historialPorProducto.get(idProducto).isEmpty()) {
            System.out.println("No hay historial para restaurar para el producto ID: " + idProducto);
            return false;
        }
        return false;

}
}
```

Atributos:

- `historialPorProducto`: Mapa que asocia cada ID de producto con su pila de estados (mementos)

Métodos:

- `guardarEstado()`: Guarda el estado actual de un producto
- `restaurarEstado()`: Restaura un producto a su estado previo
- `restaurarEstadoPorIndice()`: Restaura un producto a un estado específico en el historial
- `obtenerHistorial()`: Retorna la lista de estados guardados de un producto

Producto (Originator)

Representa la interfaz común para todos los tipos de productos.

```java
public abstract class Producto {
    protected int idProducto;
    protected String nombre;
    protected int cantidad;
    protected double precio;
    protected String descripcion;
    protected int porcentajeIva;
    protected int idCategoria;
    protected int estado;

    public Producto() {
    }

    public Producto(int idProducto, String nombre, int cantidad, double precio,
                    String descripcion, int porcentajeIva, int idCategoria, int estado) {
        this.idProducto = idProducto;
        this.nombre = nombre;
        this.cantidad = cantidad;
        this.precio = precio;
        this.descripcion = descripcion;
        this.porcentajeIva = porcentajeIva;
        this.idCategoria = idCategoria;
        this.estado = estado;
    }

    // Método abstracto que será implementado por las clases hijas
    public abstract double calcularPrecioFinal();

    // Getters y setters
    public int getIdProducto() {
        return idProducto;
    }

    public void setIdProducto(int idProducto) {
        this.idProducto = idProducto;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
```

```java
public int getCantidad() {
    return cantidad;
}

public void setCantidad(int cantidad) {
    this.cantidad = cantidad;
}

public double getPrecio() {
    return precio;
}

public void setPrecio(double precio) {
    this.precio = precio;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

public int getPorcentajeIva() {
    return porcentajeIva;
}

public void setPorcentajeIva(int porcentajeIva) {
    this.porcentajeIva = porcentajeIva;
}

public int getIdCategoria() {
    return idCategoria;
}

public void setIdCategoria(int idCategoria) {
    this.idCategoria = idCategoria;
}

public int getEstado() {
    return estado;
}

public void setEstado(int estado) {
    this.estado = estado;
}
```

Atributos:

- `idProducto`: Identificador único del producto

- `nombre`: Nombre del producto
- `cantidad`: Cantidad disponible en inventario
- `precio`: Precio base del producto
- `descripcion`: Descripción del producto
- `porcentajeIva`: Porcentaje de IVA aplicable
- `idCategoria`: Identificador de la categoría
- `estado`: Estado del producto (activo/inactivo)

Métodos:

- `calcularPrecioFinal()`: Método abstracto que calcula el precio final del producto
- Getters y setters para todos los atributos

Productos Concretos

ProductoEstandar

Implementación para productos estándar sin descuentos o promociones.

```
public class ProductoEstandar extends Producto {

    public ProductoEstandar() {
        super();
    }

    public ProductoEstandar(int idProducto, String nombre, int cantidad, double precio,
                            String descripcion, int porcentajeIva, int idCategoria, int estado) {
        super(idProducto, nombre, cantidad, precio, descripcion, porcentajeIva, idCategoria, estado);
    }

    @Override
    public double calcularPrecioFinal() {
        return precio + (precio * porcentajeIva / 100.0);
    }
}
```

Métodos:

- `calcularPrecioFinal()`: Calcula el precio con IVA incluido

ProductoConDescuento

Implementación para productos con descuento porcentual.

```java
public class ProductoPromocion extends Producto {
    private String descripcionPromocion;

    public ProductoPromocion() {
        super();
    }

    public ProductoPromocion(int idProducto, String nombre, int cantidad, double precio,
                             String descripcion, int porcentajeIva, int idCategoria, int estado,
                             String descripcionPromocion) {
        super(idProducto, nombre, cantidad, precio, descripcion, porcentajeIva, idCategoria, estado);
        this.descripcionPromocion = descripcionPromocion;
    }

    @Override
    public double calcularPrecioFinal() {
        // Implementación específica para productos en promoción
        // Por ejemplo, aplicar un descuento fijo o alguna lógica especial
        return precio + (precio * porcentajeIva / 100.0);
    }

    public String getDescripcionPromocion() {
        return descripcionPromocion;
    }

    public void setDescripcionPromocion(String descripcionPromocion) {
        this.descripcionPromocion = descripcionPromocion;
    }
}
```

Atributos Adicionales:

- `porcentajeDescuento`: Porcentaje de descuento aplicable

Métodos:

- `calcularPrecioFinal()`: Calcula el precio con IVA y aplica el descuento
- Getters y setters para porcentajeDescuento

ProductoPromocion

Implementación para productos en promoción especial.

Atributos Adicionales:

- `descripcionPromocion`: Descripción textual de la promoción

Métodos:

- `calcularPrecioFinal()`: Calcula el precio según la promoción
- Getters y setters para descripcionPromocion

Propósito

El patrón Memento permite:

1. **Capturar el estado interno** de un objeto sin violar la encapsulación
2. **Almacenar estados** para permitir su restauración posterior
3. **Implementar funcionalidad de deshacer/rehacer** sin exponer los detalles de implementación
4. **Mantener el historial de cambios** con posibilidad de análisis o auditoría

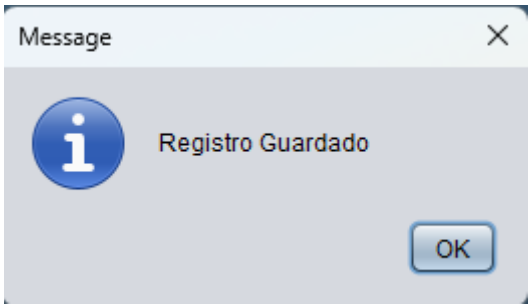
Implementación

En este sistema:

- **ProductoMemento** captura y almacena el estado completo del producto
- **HistorialProductos** gestiona la colección de mementos organizados por producto
- La implementación utiliza una estructura de pila para facilitar las operaciones de "deshacer"
- También se incluye la capacidad de restaurar a cualquier estado histórico específico

Resultado

ProductoNormal



Con promoción