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 Marlenny

Vera Acevedo Héctor Aramís

Grupo:

7SB

Fecha de entrega:

18/03/2025

Documentación del Patrón Builder para Sistema de Productos

Descripción General

Esta implementación utiliza el patrón de diseño Builder para crear diferentes tipos de productos (estándar, con descuento y promocionales) de manera flexible y estructurada. El patrón Builder separa la construcción de un objeto complejo de su representación, permitiendo crear diferentes representaciones usando el mismo proceso de construcción.

Estructura del Sistema

Diagrama de Clases

El sistema está compuesto por las siguientes clases principales:

- **ProductoBuilder** (interfaz): Define las operaciones para construir un producto paso a paso
- **Builders concretos**: ProductoEstandarBuilder, ProductoConDescuentoBuilder, ProductoPromocionBuilder
- **ProductoDirector**: Coordina el proceso de construcción de productos
- **Producto** (clase abstracta base): Clase base para todos los tipos de productos
- **Clases concretas de productos**: ProductoEstandar, ProductoConDescuento, ProductoPromocion
- **Ctrl_Producto**: Controlador para operaciones de productos

Relaciones entre Clases

- **ProductoBuilder** define los métodos para establecer las propiedades de un producto y el método build() para crear la instancia final
- **Los builders concretos** implementan ProductoBuilder y conocen cómo construir cada tipo específico de producto
- **ProductoDirector** dirige el proceso de construcción utilizando los builders para crear productos con configuraciones predefinidas
- **Las clases de productos concretos** heredan de la clase abstracta Producto e implementan funcionalidades específicas
- **Ctrl_Producto** utiliza los builders y el director para crear productos y realizar operaciones sobre ellos

Componentes Principales

ProductoBuilder (Interfaz)

Define la interfaz común para todos los builders de productos.

```
public interface ProductoBuilder {
    ProductoBuilder setIdProducto(int idProducto);
    ProductoBuilder setNombre(String nombre);
    ProductoBuilder setCantidad(int cantidad);
    ProductoBuilder setPrecio(double precio);
    ProductoBuilder setDescripcion(String descripcion);
    ProductoBuilder setPorcentajeIva(int porcentajeIva);
    ProductoBuilder setIdCategoria(int idCategoria);
    ProductoBuilder setEstado(int estado);
    Producto build();
}
```

Métodos:

- setIdProducto(int idProducto): Establece el ID del producto
- setNombre(String nombre): Establece el nombre del producto
- setCantidad(int cantidad): Establece la cantidad disponible
- setPrecio(double precio): Establece el precio base
- setDescripcion(String descripcion): Establece la descripción
- setPorcentajeIva(int porcentajeIva): Establece el porcentaje de IVA
- setIdCategoria(int idCategoria): Establece la categoría
- setEstado(int estado): Establece el estado
- build(): Crea y devuelve la instancia del producto

Builders Concretos

ProductoEstandarBuilder

Implementa ProductoBuilder para construir productos estándar.

```java
public class ProductoEstandarBuilder implements ProductoBuilder {
    private int idProducto;
    private String nombre;
    private int cantidad;
    private double precio;
    private String descripcion;
    private int porcentajeIva;
    private int idCategoria;
    private int estado;

    @Override
    public ProductoBuilder setIdProducto(int idProducto) {
        this.idProducto = idProducto;
        return this;
    }

    @Override
    public ProductoBuilder setNombre(String nombre) {
        this.nombre = nombre;
        return this;
    }

    @Override
    public ProductoBuilder setCantidad(int cantidad) {
        this.cantidad = cantidad;
        return this;
    }

    @Override
    public ProductoBuilder setPrecio(double precio) {
        this.precio = precio;
        return this;
    }

    @Override
    public ProductoBuilder setDescripcion(String descripcion) {
        this.descripcion = descripcion;
        return this;
    }
```

```
    @Override
    public ProductoBuilder setDescripcion(String descripcion) {
        this.descripcion = descripcion;
        return this;
    }

    @Override
    public ProductoBuilder setPorcentajeIva(int porcentajeIva) {
        this.porcentajeIva = porcentajeIva;
        return this;
    }

    @Override
    public ProductoBuilder setIdCategoria(int idCategoria) {
        this.idCategoria = idCategoria;
        return this;
    }

    @Override
    public ProductoBuilder setEstado(int estado) {
        this.estado = estado;
        return this;
    }

    @Override
    public Producto build() {
        return new ProductoEstandar(idProducto, nombre, cantidad, precio,
                                    descripcion, porcentajeIva, idCategoria, estado);
    }
}
```

Atributos:

- Todos los atributos necesarios para construir un ProductoEstandar
- Métodos para establecer cada atributo que devuelven el builder para permitir encadenamiento
- build(): Crea una instancia de ProductoEstandar con los atributos configurados

ProductoConDescuentoBuilder

Implementa ProductoBuilder para construir productos con descuento.

```
public class ProductoConDescuentoBuilder implements ProductoBuilder {
    private int idProducto;
    private String nombre;
    private int cantidad;
    private double precio;
    private String descripcion;
    private int porcentajeIva;
    private int idCategoria;
    private int estado;
    private double porcentajeDescuento;

    @Override
    public ProductoBuilder setIdProducto(int idProducto) {
        this.idProducto = idProducto;
        return this;
    }

    @Override
    public ProductoBuilder setNombre(String nombre) {
        this.nombre = nombre;
        return this;
    }

    @Override
    public ProductoBuilder setCantidad(int cantidad) {
        this.cantidad = cantidad;
        return this;
    }

    @Override
    public ProductoBuilder setPrecio(double precio) {
        this.precio = precio;
        return this;
    }

    @Override
    public ProductoBuilder setDescripcion(String descripcion) {
        this.descripcion = descripcion;
        return this;
    }
```

```java
    @Override
    public ProductoBuilder setPorcentajeIva(int porcentajeIva) {
        this.porcentajeIva = porcentajeIva;
        return this;
    }

    @Override
    public ProductoBuilder setIdCategoria(int idCategoria) {
        this.idCategoria = idCategoria;
        return this;
    }

    @Override
    public ProductoBuilder setEstado(int estado) {
        this.estado = estado;
        return this;
    }

    public ProductoConDescuentoBuilder setPorcentajeDescuento(double porcentajeDescuento) {
        this.porcentajeDescuento = porcentajeDescuento;
        return this;
    }

    @Override
    public Producto build() {
        return new ProductoConDescuento(idProducto, nombre, cantidad, precio,
                                        descripcion, porcentajeIva, idCategoria, estado, porcentajeDescuento);
    }
}
```

Atributos Adicionales:

- porcentajeDescuento: Porcentaje de descuento a aplicar
- Método adicional setPorcentajeDescuento() que permite establecer este valor específico
- build(): Crea una instancia de ProductoConDescuento con los atributos configurados

ProductoDirector.java - Clase que encapsula lógicas de construcción comunes:

```java
public class ProductoDirector {

    // Construye un producto estándar básico
    public Producto construirProductoEstandarBasico(ProductoBuilder builder, int idProducto, String nombre) {
        return builder
            .setIdProducto(idProducto)
            .setNombre(nombre)
            .setCantidad(0)
            .setPrecio(0.0)
            .setDescripcion("")
            .setPorcentajeIva(19)  // IVA por defecto
            .setIdCategoria(1)     // Categoria por defecto
            .setEstado(1)          // Estado activo por defecto
            .build();
    }

    // Construye un producto con descuento básico
    public Producto construirProductoDescuentoBasico(ProductoConDescuentoBuilder builder,
                                                     int idProducto, String nombre, double porcentajeDescuento) {
        return builder
            .setIdProducto(idProducto)
            .setNombre(nombre)
            .setCantidad(0)
            .setPrecio(0.0)
            .setDescripcion("")
            .setPorcentajeIva(19)  // IVA por defecto
            .setIdCategoria(1)     // Categoria por defecto
            .setEstado(1)          // Estado activo por defecto
            //.setPorcentajeDescuento(porcentajeDescuento)
            .build();
    }
```

```
    // Construye un producto en promoción básico
    public Producto construirProductoPromocionBasico(ProductoPromocionBuilder builder,
                                                     int idProducto, String nombre, String descripcionPromocion) {
        return builder
            .setIdProducto(idProducto)
            .setNombre(nombre)
            .setCantidad(0)
            .setPrecio(0.0)
            .setDescripcion("")
            .setPorcentajeIva(19)  // IVA por defecto
            .setIdCategoria(1)     // Categoría por defecto
            .setEstado(1)          // Estado activo por defecto
           // .setDescripcionPromocion(descripcionPromocion)
            .build();
    }

    // Construye un producto completo con todos los parámetros
    public Producto construirProductoCompleto(ProductoBuilder builder,
                                              int idProducto, String nombre, int cantidad, double precio,
                                              String descripcion, int porcentajeIva, int idCategoria, int estado) {
        return builder
            .setIdProducto(idProducto)
            .setNombre(nombre)
            .setCantidad(cantidad)
            .setPrecio(precio)
            .setDescripcion(descripcion)
            .setPorcentajeIva(porcentajeIva)
            .setIdCategoria(idCategoria)
            .setEstado(estado)
            .build();
    }
}
```

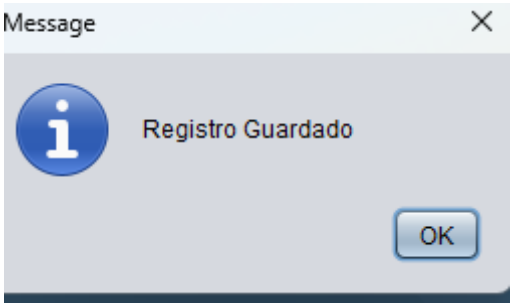
- Proporciona métodos predefinidos para construir productos con configuraciones comunes
- construirProductoEstandarBasico() - Crea producto estándar con valores por defecto
- construirProductoDescuentoBasico() - Crea producto con descuento con valores por defecto
- construirProductoCompleto() - Permite especificar todos los parámetros

Beneficios de esta Implementación

1. **Flexibilidad**: Permite crear diferentes tipos de productos con el mismo proceso de construcción
2. **Legibilidad**: El código de construcción es más claro y expresivo
3. **Inmutabilidad**: Los objetos se construyen completamente antes de ser usados
4. **Mantenibilidad**: Fácil de añadir nuevos tipos de productos o parámetros
5. **Valores por defecto**: El director proporciona configuraciones comunes

Resultados

Producto normal



Producto descuento