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

Julia Marleny

Vera Acevedo Héctor Aramís

Grupo:

7SB

Fecha de entrega:

18/03/2025

Documentación del Patrón Factory Method para Sistema de Productos

Descripción General

Esta implementación utiliza el patrón de diseño Factory Method para crear diferentes tipos de productos (estándar, con descuento y promocionales) sin exponer la lógica de creación al cliente. El patrón Factory Method define una interfaz para crear un objeto, pero permite que las subclases decidan qué clase instanciar.

Estructura del Sistema

Diagrama de Clases

El sistema está compuesto por las siguientes clases principales:

- `Producto` (clase abstracta base)
- Clases concretas de productos: `ProductoEstandar`, `ProductoConDescuento`, `ProductoPromocion`
- `ProductoCreator` (creador abstracto)
- Creadores concretos: `ProductoEstandarCreator`, `ProductoConDescuentoCreator`, `ProductoPromocionCreator`
- `Ctrl_Producto`: Controlador para operaciones de productos

Relaciones entre Clases

- Las clases de productos concretos heredan de la clase abstracta `Producto`
- Los creadores concretos heredan de `ProductoCreator`
- El controlador `Ctrl_Producto` utiliza los creadores para instanciar productos

Componentes Principales

Producto (Clase Abstracta)

Representa la interfaz común para todos los tipos de productos.

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.

Métodos

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

ProductoConDescuento

Implementación para productos con descuento porcentual.

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`

ProductoCreator (Creador Abstracto)

Define la interfaz para crear productos.

Métodos

- `crearProducto()`: Método abstracto Factory Method que crea un producto

- `prepararProducto()`: Método que utiliza el Factory Method y puede realizar operaciones adicionales

Creadores Concretos

ProductoEstandarCreator

Creador para productos estándar.

Atributos

- Todos los atributos necesarios para crear un producto estándar

Métodos

- `crearProducto()`: Implementación del Factory Method que crea un `ProductoEstandar`

ProductoConDescuentoCreator

Creador para productos con descuento.

Atributos

- Todos los atributos necesarios para crear un producto con descuento
- `porcentajeDescuento`: Porcentaje de descuento para aplicar

Métodos

- `crearProducto()`: Implementación del Factory Method que crea un `ProductoConDescuento`

ProductoPromocionCreator

Creador para productos en promoción.

Atributos

- Todos los atributos necesarios para crear un producto en promoción
- `descripcionPromocion`: Descripción de la promoción

Métodos

- `crearProducto()`: Implementación del Factory Method que crea un `ProductoPromocion`

Ctrl_Producto

Controlador que gestiona las operaciones relacionadas con productos.

Métodos

- `getCreator()`: Retorna el creador apropiado según el tipo de producto
- `crearProducto()`: Crea un nuevo producto utilizando el creador correspondiente
- `guardar()`: Guarda un producto en la base de datos
- `existeProducto()`: Verifica si existe un producto con un nombre dado
- `actualizar()`: Actualiza la información de un producto existente
- `eliminar()`: Elimina un producto del sistema
- `actualizarStock()`: Actualiza la cantidad disponible de un producto

El Patrón Factory Method en Detalle

Propósito

El patrón Factory Method encapsula la lógica de creación de objetos en clases separadas de las clases que usan esos objetos. Esto permite:

1. Desacoplar el código que crea objetos del código que los usa
2. Facilitar la adición de nuevos tipos de productos sin cambiar el código existente
3. Permitir que las subclases modifiquen el tipo de objetos que se crean

Implementación

En este sistema:

- `ProductoCreator` define el método abstracto `crearProducto()` (Factory Method)
- Cada creador concreto (`ProductoEstandarCreator`, `ProductoConDescuentoCreator`, `ProductoPromocionCreator`) implementa este método para crear un tipo específico de producto
- El método `prepararProducto()` en `ProductoCreator` utiliza el Factory Method para crear productos y puede realizar operaciones adicionales si es necesario

Diferencias con Abstract Factory

- Factory Method utiliza herencia para crear productos (a través de subclases)
- Abstract Factory utiliza composición para crear familias de productos relacionados

Flujo de Trabajo

Creación de un Producto

1. El cliente solicita la creación de un producto a través del controlador `Ctrl_Producto`
2. El controlador obtiene el creador apropiado mediante `getCreator()`
3. El creador instancia el producto utilizando `prepararProducto()`, que a su vez llama al Factory Method `crearProducto()`
4. El producto creado se devuelve al controlador
5. El controlador guarda el producto en la base de datos si es necesario.