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 Prototype para Sistema de Productos

Descripción General

Esta implementación utiliza el patrón de diseño Prototype para crear diferentes tipos de productos (estándar, con descuento y promocionales) mediante clonación de objetos prototipo. El patrón Prototype permite crear nuevos objetos duplicando instancias existentes (prototipos), lo que reduce la necesidad de crear subclases específicas para cada variante de producto.

Estructura del Sistema

Diagrama de Clases

El sistema está compuesto por las siguientes clases principales:

- **ProductoPrototype** (interfaz): Define la operación de clonación
- **Producto** (clase abstracta base que implementa ProductoPrototype)
- **Clases concretas de productos**: ProductoEstandar, ProductoConDescuento, ProductoPromocion
- **RegistroProductos**: Registro centralizado de prototipos
- **Ctrl_Producto**: Controlador para operaciones de productos

Relaciones entre Clases

- **ProductoPrototype** define el método `clonar()` para la creación de productos por clonación
- Las clases de productos concretos heredan de la clase abstracta **Producto** e implementan el método `clonar()`
- **RegistroProductos** mantiene una colección de prototipos y proporciona acceso a ellos
- **Ctrl_Producto** utiliza el registro de prototipos para crear nuevas instancias por clonación

Componentes Principales

ProductoPrototype (Interfaz)

Define la operación de clonación para todos los productos.

Métodos:

- **clonar()**: Crea y devuelve una copia del objeto

Producto (Clase Abstracta)

Representa la interfaz común para todos los tipos de productos e implementa ProductoPrototype.

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
- **clonar()**: Método abstracto heredado de ProductoPrototype
- **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
- **clonar()**: Implementación que crea una nueva instancia de ProductoEstandar con los mismos valores

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
- **clonar()**: Implementación que crea una nueva instancia de ProductoConDescuento con los mismos valores
- **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
- **clonar()**: Implementación que crea una nueva instancia de ProductoPromocion con los mismos valores
- **Getters y setters** para descripcionPromocion

RegistroProductos (Prototype Registry)

Mantiene un registro centralizado de prototipos para todos los tipos de productos.

Atributos:

- **prototipos**: Map que almacena los prototipos disponibles, con el tipo de producto como clave

Métodos:

- **getPrototipo()**: Retorna un clon del prototipo solicitado
- **registrarPrototipo()**: Añade o actualiza un prototipo en el registro

Ctrl_Producto

Controlador que gestiona las operaciones relacionadas con productos.

Métodos:

- **crearProducto(String tipoProducto)**: Crea un nuevo producto básico del tipo especificado
- **crearProductoEstandar()**: Crea un producto estándar personalizado
- **crearProductoConDescuento()**: Crea un producto con descuento personalizado
- **crearProductoPromocion()**: Crea un producto promocional personalizado
- **crearProducto(String tipoProducto, ...)**: Método generalizado para crear cualquier tipo de producto
- **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 Prototype en Detalle

Propósito

El patrón Prototype permite crear nuevos objetos duplicando prototipos existentes, lo que ofrece varias ventajas:

1. **Reducción de subclases**: Evita crear numerosas subclases para distintas variantes de objetos

2. **Clonación dinámica**: Permite crear nuevos objetos copiando objetos existentes en tiempo de ejecución
3. **Configuración en tiempo de ejecución**: Los objetos pueden ser clonados con estados específicos
4. **Aislamiento del código cliente**: El cliente no necesita conocer las clases concretas de productos

Implementación

En este sistema:

- La interfaz **ProductoPrototype** define el método `clonar()` que todas las clases de productos implementan
- Cada producto concreto implementa `clonar()` para crear una copia profunda de sí mismo
- **RegistroProductos** almacena los prototipos iniciales y proporciona acceso a clones
- **Ctrl_Producto** utiliza los prototipos para crear nuevos productos personalizados

Diferencias con Factory Method

- **Prototype** se basa en la clonación de objetos existentes
- **Factory Method** delega la creación a subclases específicas
- **Prototype** reduce la necesidad de clases creador separadas
- **Prototype** permite crear objetos sin conocer sus clases concretas

Flujo de Trabajo

Inicialización

1. El sistema carga prototipos predefinidos en el **RegistroProductos**
2. Cada prototipo representa una variante básica de producto (estándar, con descuento, promocional)

Creación de un Producto

1. El cliente solicita la creación de un producto a través del controlador **Ctrl_Producto**
2. El controlador obtiene un clon del prototipo adecuado del **RegistroProductos**
3. El controlador personaliza el clon con los datos específicos proporcionados
4. El producto clonado y personalizado se devuelve al cliente
5. El controlador puede guardar el producto en la base de datos si es necesario

Clonación Directa

1. Un producto existente puede ser clonado directamente usando su método `clonar()`
2. El clon obtenido puede ser modificado para crear una variante del producto original
3. Esto facilita la creación de productos similares con pequeñas variaciones

Ventajas del Patrón Prototype

1. **Menor acoplamiento**: Reduce la dependencia entre el código cliente y las clases de productos
2. **Flexibilidad**: Permite añadir o eliminar productos en tiempo de ejecución
3. **Economía de recursos**: Evita la creación repetitiva de objetos similares
4. **Facilidad de extensión**: Simplifica la adición de nuevos tipos de productos
5. **Personalización eficiente**: Permite crear variantes de productos más fácilmente