DONCAN ALEJANDRO CASQUETE OLIVEROS

PATRONES DE DISEÑO DE SOFTWARE

PATRON ABSTRACT FACTORY

Patrón de Diseño: Abstract Factory

# Definición

El patrón Abstract Factory es un patrón creacional que proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas.

# Objetivo

Permitir que un sistema sea independiente de cómo se crean, componen y representan sus objetos.

# Cuándo Usarlo

- Cuando el sistema debe ser independiente de cómo se crean los productos.

- Cuando el sistema debe trabajar con varias familias de productos.

- Cuando se quiere garantizar que los productos de una familia se usen juntos.

# Estructura

1. AbstractFactory: declara un conjunto de métodos para crear cada uno de los productos abstractos.

2. ConcreteFactory: implementa los métodos para crear productos concretos.

3. AbstractProduct: declara una interfaz para un tipo de producto.

4. ConcreteProduct: implementa la interfaz del producto.

5. Client: utiliza solamente las interfaces declaradas por la AbstractFactory y AbstractProduct.

# Ventajas Ampliadas

1. Aislamiento del código cliente: el cliente no conoce las clases concretas que se están utilizando, lo que permite cambiar toda una familia de productos sin afectar el código del cliente.

2. Consistencia entre productos relacionados: garantiza que se usen productos de una misma familia, evitando errores como mezclar un botón de Windows con una checkbox de Mac.

3. Fácil intercambio entre familias de productos: solo se necesita cambiar la fábrica concreta para crear una nueva “piel” o “tema” completo de una aplicación.

4. Mejora la escalabilidad: puedes añadir nuevas familias de productos fácilmente creando una nueva fábrica concreta.

5. Promueve el principio de inversión de dependencias (DIP): el código depende de abstracciones, no de implementaciones concretas.

# Desventajas Ampliadas

1. Complejidad adicional: introduce muchas clases y capas de abstracción, lo que puede hacer el código más difícil de seguir si no se justifica su uso.

2. Dificultad para añadir nuevos productos: si se necesita añadir un nuevo tipo de producto (por ejemplo, un "Slider"), hay que modificar todas las fábricas existentes, lo cual rompe el principio abierto/cerrado.

3. Menos flexibilidad con productos individuales: está orientado a familias completas de objetos. Si solo necesitas cambiar un producto específico, este patrón puede resultar excesivo.

4. Sobrecoste para proyectos pequeños: en aplicaciones simples, usar Abstract Factory puede ser innecesario y complicar el diseño sin aportar beneficios reales.