Taller 5 DPOO

Santiago Arenas Rojas

202220359

Abstract Factory

<https://github.com/kiat/OOP-Design-Patterns/tree/master/Java/AbstractFactory-Example-Taxes>

**Información general de Abstract Factory:**

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

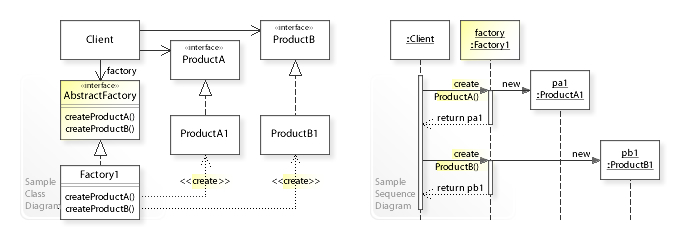
Se utiliza para crear objetos que están relacionados entre sí y están diseñados para trabajar juntos. El patrón es similar al patrón Factory Method, pero está diseñado para crear familias de objetos relacionados, mientras que el patrón Factory Method está diseñado para crear solo un tipo de objeto

El patrón de Abstract Factory se suele utilizar en situaciones en las que un sistema necesita ser independiente de cómo se crean, componen y representan sus objetos. También se utiliza cuando es necesario configurar un sistema con una de varias familias de objetos

El patrón promueve el acoplamiento flexible entre objetos, ya que el código de cliente solo interactúa con la interfaz de Abstract Factory y no necesita conocer las clases específicas de los objetos que crea

En resumen, el patrón Abstract Factory es un patrón de diseño creativo que proporciona una interfaz para crear familias de objetos relacionados sin especificar sus clases concretas. Se utiliza para crear objetos que están relacionados entre sí y están diseñados para trabajar juntos, y promueve el acoplamiento flexible entre los objetos.

**UML Estructura de Diseño:**



**Información general del proyecto:**

El proyecto consiste en diseñar un sistema para un banco que pueda manejar la creación de diferentes tipos de cuentas bancarias, incluidas cuentas corrientes y de ahorro personales y corporativas, así como cuentas de paquete que combinen ambos tipos de cuentas. El sistema debe desencadenar diferentes procesos en segundo plano en el sistema de financiación bancaria en función del tipo de cuenta que se cree, como añadir datos a diferentes bases de datos o informar a los corredores sobre la creación de una nueva cuenta. El objetivo es proporcionar un método simple para que los clientes soliciten una cuenta corriente o de ahorros sin necesidad de conocer los procesos internos del banco. Los principales desafíos de diseño para este proyecto incluyen agilizar el proceso de apertura de cuentas para que sea rápido y fácil de completar, diseñar el sistema para que sea lo más digital posible e implementar un sistema de diseño para construir con confianza desde el principio

Además, el sistema debe ser capaz de manejar los requisitos normativos y de seguridad, y garantizar que se activen los subprocesos correctos en función del tipo de cuenta que se está creando

**Patrón Implementado en el proyecto:**

El patrón Abstract Factory se utiliza para crear familias de objetos relacionados sin especificar sus clases concretas. En el proyecto Impuestos, el patrón Fábrica abstracta se utiliza para crear diferentes tipos de calculadoras de impuestos. Así es como se usa el patrón de diseño Abstract Factory en el proyecto Taxes:

1. **Interfaz AbstractFactory: El proyecto define una**  interfaz AbstractFactory que contiene métodos para crear diferentes tipos de calculadoras de impuestos. Esta interfaz es la clave del patrón Abstract Factory, ya que permite la creación de familias de objetos relacionados sin especificar sus clases concretas.
2. **Clases ConcreteFactory:** el proyecto implementa la interfaz **AbstractFactory** con las clases **ConcreteFactory**. Cada clase ConcreteFactory es responsable de crear tipos específicos de calculadoras de impuestos. Estas clases son responsables de implementar los métodos definidos en la  **interfaz AbstractFactory**.
3. **Clases de cliente:** Las clases de cliente del proyecto utilizan la  **interfaz AbstractFactory** para crear calculadoras de impuestos sin conocer sus clases específicas. Esto permite que el código de cliente permanezca independiente del tipo específico de calculadora de impuestos que se está creando.
4. **Ventajas:** El patrón Abstract Factory proporciona varias ventajas en el proyecto Taxes. Permite la creación de familias de objetos relacionados sin especificar sus clases concretas, lo que hace que el código sea más flexible y fácil de mantener. También promueve el acoplamiento flexible entre objetos, ya que el código de cliente solo interactúa con la  **interfaz AbstractFactory** y no necesita conocer las clases específicas de los objetos que crea.
5. **Desventajas:** Una posible desventaja de usar el patrón Fábrica abstracta en el proyecto Taxes es que puede dar lugar a un aumento en el número de clases del sistema, ya que cada familia de objetos relacionados requiere su propio conjunto de interfaces y clases. Esto puede hacer que el código sea más complejo y difícil de entender.

En conclusión, el patrón de diseño Abstract Factory se está utilizando en el proyecto Taxes para crear diferentes tipos de calculadoras de impuestos, manteniendo la flexibilidad y promoviendo el acoplamiento flexible entre objetos. Sin embargo, el proyecto también podría considerar el uso del patrón Método de fábrica como una solución alternativa, ya que ambos patrones proporcionan beneficios similares.