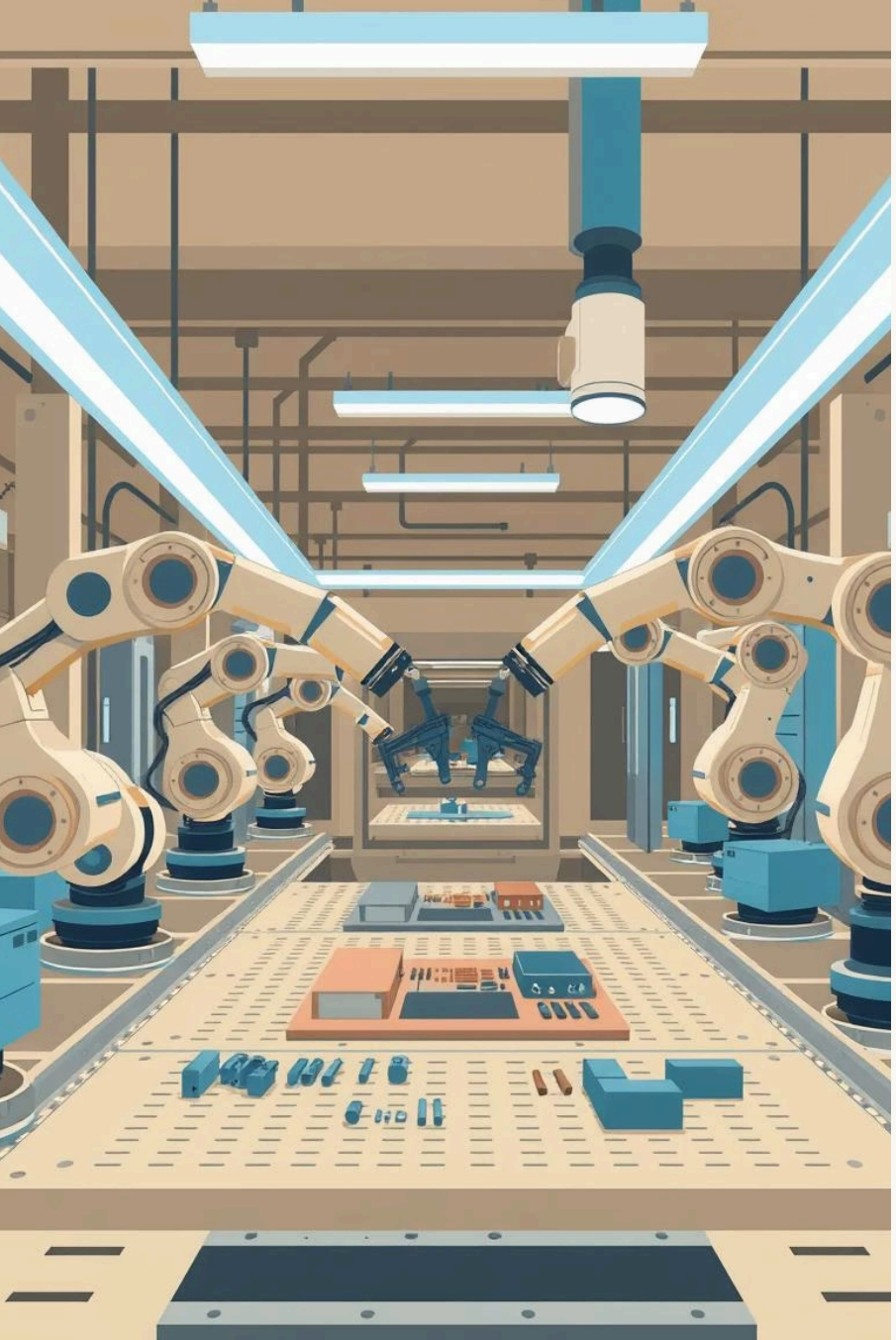

Patrón de Diseño Abstract Factory: Creación Flexible de Familias de Objetos

Descubra cómo el patrón Abstract Factory simplifica la creación de conjuntos de objetos relacionados, mejorando la flexibilidad y el mantenimiento de su software.

¿Qué es Abstract Factory?

1

Creación de Familias

Patrón creacional que permite crear familias de objetos relacionados sin especificar sus clases concretas.

2

Desacoplamiento

Facilita la escalabilidad y el mantenimiento al desacoplar la creación de objetos del código cliente.

3

Independencia del Sistema

Ideal cuando el sistema debe ser independiente de cómo se crean, componen o representan sus objetos.

Ventajas Clave:

- **Consistencia:** Asegura que los objetos creados son compatibles entre sí.
- **Flexibilidad:** Permite cambiar fácilmente entre diferentes familias de productos.
- **Escalabilidad:** Facilita la adición de nuevas familias sin modificar el código existente.

Ejemplo en Python: Vehículos Mercedes y BMW

Imaginemos que necesitamos fabricar diferentes tipos de vehículos (Sedán y SUV) de distintas marcas (Mercedes y BMW). El patrón Abstract Factory nos permite cambiar la marca de los vehículos que fabricamos de forma sencilla y sin modificar el código de la lógica de negocio.

```
class AbstractCar:
    def drive(self):
        pass

class AbstractSUV:
    def drive(self):
        pass

class MercedesSedan(AbstractCar):
    def drive(self):
        return "Conduciendo un Mercedes Sedán"

class MercedesSUV(AbstractSUV):
    def drive(self):
        return "Conduciendo un Mercedes SUV"

class BmwSedan(AbstractCar):
    def drive(self):
        return "Conduciendo un BMW Sedán"

class BmwSUV(AbstractSUV):
    def drive(self):
        return "Conduciendo un BMW SUV"

class CarFactory:
    def create_sedan(self):
        pass
    def create_suv(self):
        pass

class MercedesFactory(CarFactory):
    def create_sedan(self):
        return MercedesSedan()
    def create_suv(self):
        return MercedesSUV()

class BmwFactory(CarFactory):
    def create_sedan(self):
        return BmwSedan()
    def create_suv(self):
        return BmwSUV()

def client_code(factory: CarFactory):
    sedan = factory.create_sedan()
    suv = factory.create_suv()
    print(sedan.drive())
    print(suv.drive())

print("Cliente con MercedesFactory:")
client_code(MercedesFactory())
print("\nCliente con BmwFactory:")
client_code(BmwFactory())
```

Este código demuestra cómo, al cambiar la fábrica inyectada al `client_code`, se crean automáticamente las instancias correctas de los vehículos sin que el cliente necesite saber sus clases concretas.