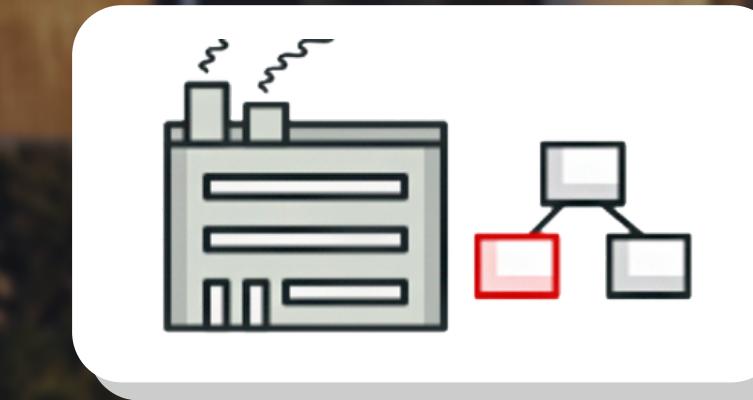


FACTORY METHOD

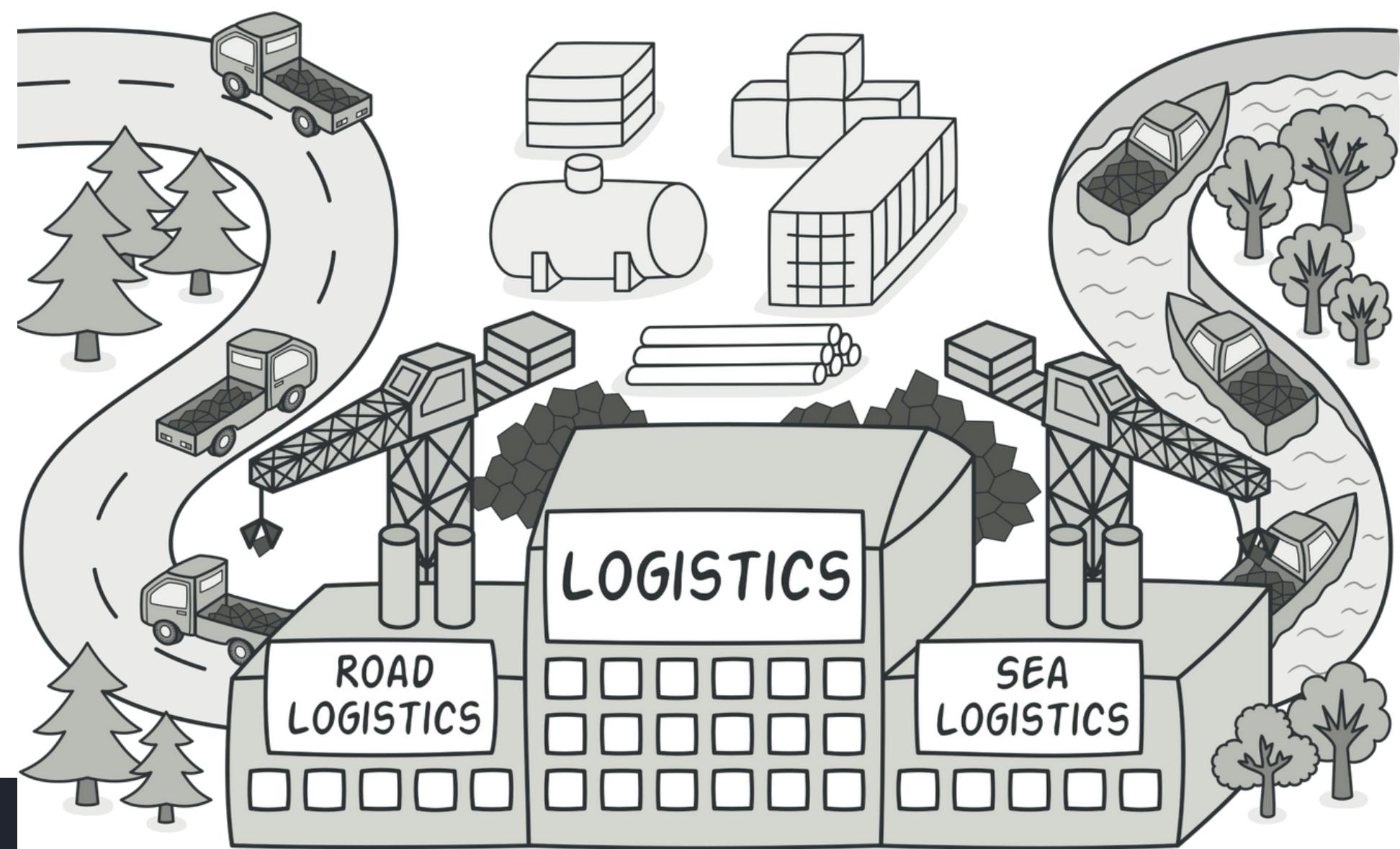
FABIÁN OROZCO | B95690



DEFINICIÓN

Provee una interfaz, donde las subclases deciden qué tipo de instancia crear basándose en la información que provee el cliente.

El método retorna una instancia, denominada "producto".



TIPO
Creacional

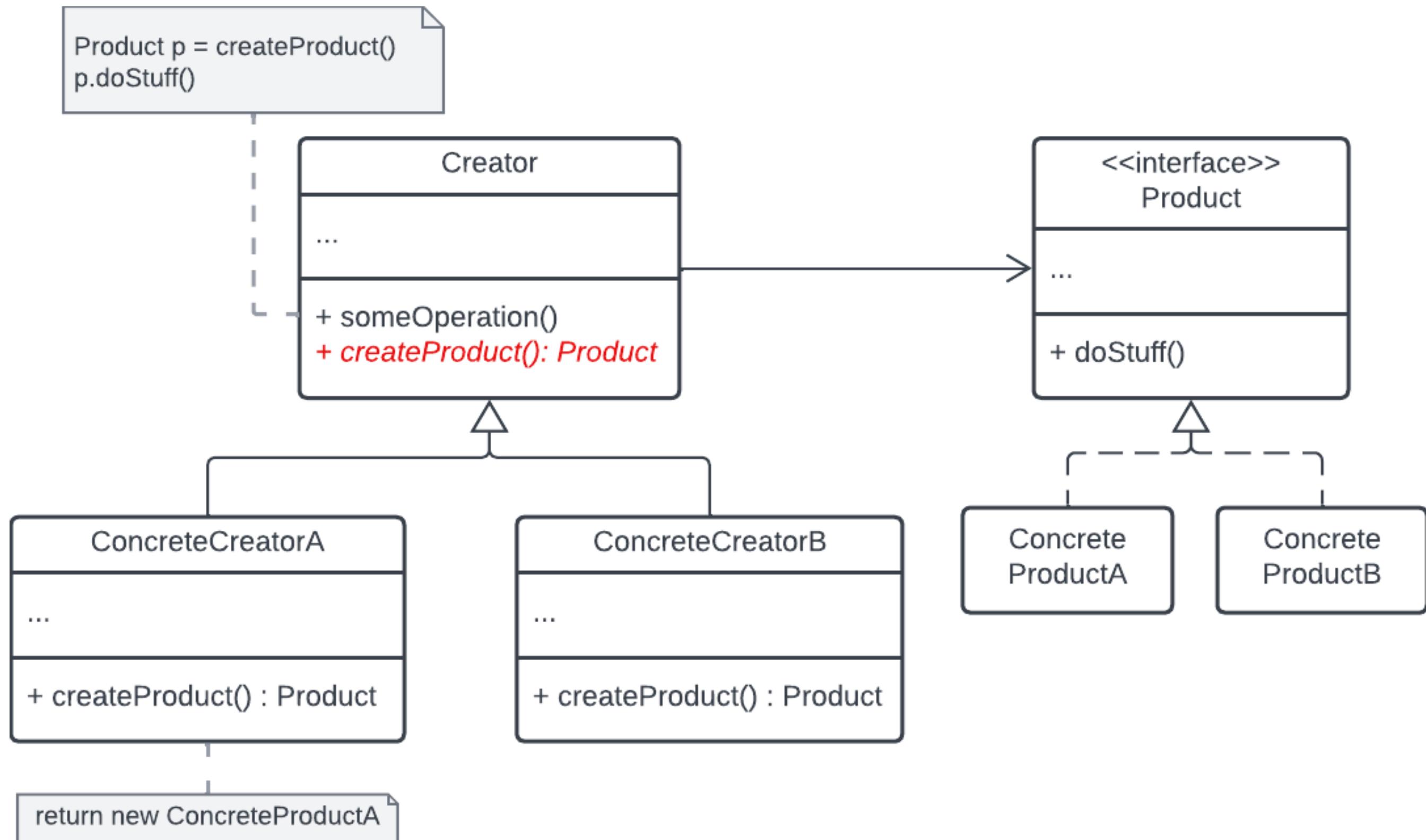
PROBLEMA

Manejo de la creación de objetos que cuentan con una interfaz común sin plagiar la aplicación de condicionales.



El cliente conoce que todos los productos realizan una **acción en común**, pero no conoce ni le interesa cómo lo hace cada uno.

SOLUCIÓN



EJEMPLO CASO PROBLEMA

```
class Avion {  
    public string Operation() {  
        return "{Soy un avion}";  
    }  
  
class Carro {  
    public string Operation() {  
        return "{Soy un carro}";  
    }  
}
```

```
class Program {  
    static void Main(string[] args) {  
        Avion Avion = new Avion();  
        Carro Carro = new Carro();  
        Console.WriteLine(Avion.Operation());  
        Console.WriteLine(Carro.Operation());  
    }  
}
```

EJEMPLO SOLUCIÓN

```
abstract class Creator {
    public abstract IVehiculo FactoryMethod();

    public string SomeOperation() {
        var product = FactoryMethod();
        var result = "Creador: el mismo código sirve para crear " + product.Operation();
        return result;
    }
}

class CreatorAvion: Creator {
    public override IVehiculo FactoryMethod() {
        return new Avion();
    }
}

class CreatorCarro : Creator {
    public override IVehiculo FactoryMethod() {
        return new Carro();
    }
}
```

EJEMPLO SOLUCIÓN

```
public interface IVehiculo {  
    string Operation();  
}  
  
class Avion : IVehiculo {  
    public string Operation() {  
        return "{Soy un avion}";  
    }  
}  
  
class Carro : IVehiculo {  
    public string Operation() {  
        return "{Soy un carro}";  
    }  
}
```

```
class Program {  
    static void Main(string[] args) {  
        List<Creator> creators = new List<Creator>();  
        creators.Add(new CreatorAvion());  
        creators.Add(new CreatorCarro());  
  
        foreach (Creator creator in creators) {  
            Console.WriteLine(creator.SomeOperation());  
        }  
    }  
}
```

¿CUÁNDΟ UTILIZARLO?

Cuando no se sabe de antemano los tipos y dependencias de los objetos.

Cuando quiera dar al usuario de su biblioteca o framework una forma de extender los componentes internos.

VENTAJAS DE UTILIZARLO

01

Evita acoplamiento

Creación y productos

02

Facilita el soporte

Un solo lugar.

03

Extensión

Nuevos productos.

DRY

SRP

OCP

DESVENTAJA

DE UTILIZARLO

01

Código complicado

Muchas subclases

¡GRACIAS!