Patrones de diseño

Patrón de comportamiento

Template Method

Julián Guisao
Carolina Valdes

¿Qué es *Template Method?*

El **Template Method** es un patrón de diseño que establece la estructura general de un algoritmo en una clase base, pero permite que las clases hijas personalicen algunos pasos sin modificar el flujo principal.

Objetivo

Permitir la **reutilización de código** común en una jerarquía de clases, asegurando que la estructura del algoritmo sea consistente mientras permite personalización.

RECARGA SU KI

SUPERA LIMITES

SE TRANSFORMA

1- Aumenta su ki
2- Supera sus limites
3- No se transforma

💡 Problema

Cuando varias clases tienen algoritmos similares con pequeñas diferencias, se puede **duplicar código**, dificultando el mantenimiento.

```
internal class SinPatron
{
    0 references
    class Goku
    {
        0 references
        public void Transformarse()
        {
            Console.WriteLine("Goku está aumentando su Ki... ⚡🔥");
            Console.WriteLine("Goku supera sus límites con entrenamiento. 🏋️‍♂️");
            Console.WriteLine("¡Goku se transforma en Super Saiyajin! 🔥⚡");
            Console.WriteLine("Goku muestra su poder a todos. 💥");
        }
    }

    0 references
    class Vegeta
    {
        0 references
        public void Transformarse()
        {
            Console.WriteLine("Vegeta está aumentando su Ki... ⚡🔥");
            Console.WriteLine("Vegeta supera sus límites con orgullo. 💪");
            Console.WriteLine("¡Vegeta se transforma en Super Saiyajin Blue! 🔵⚡");
            Console.WriteLine("Vegeta muestra su poder a todos. 💥");
        }
    }
```

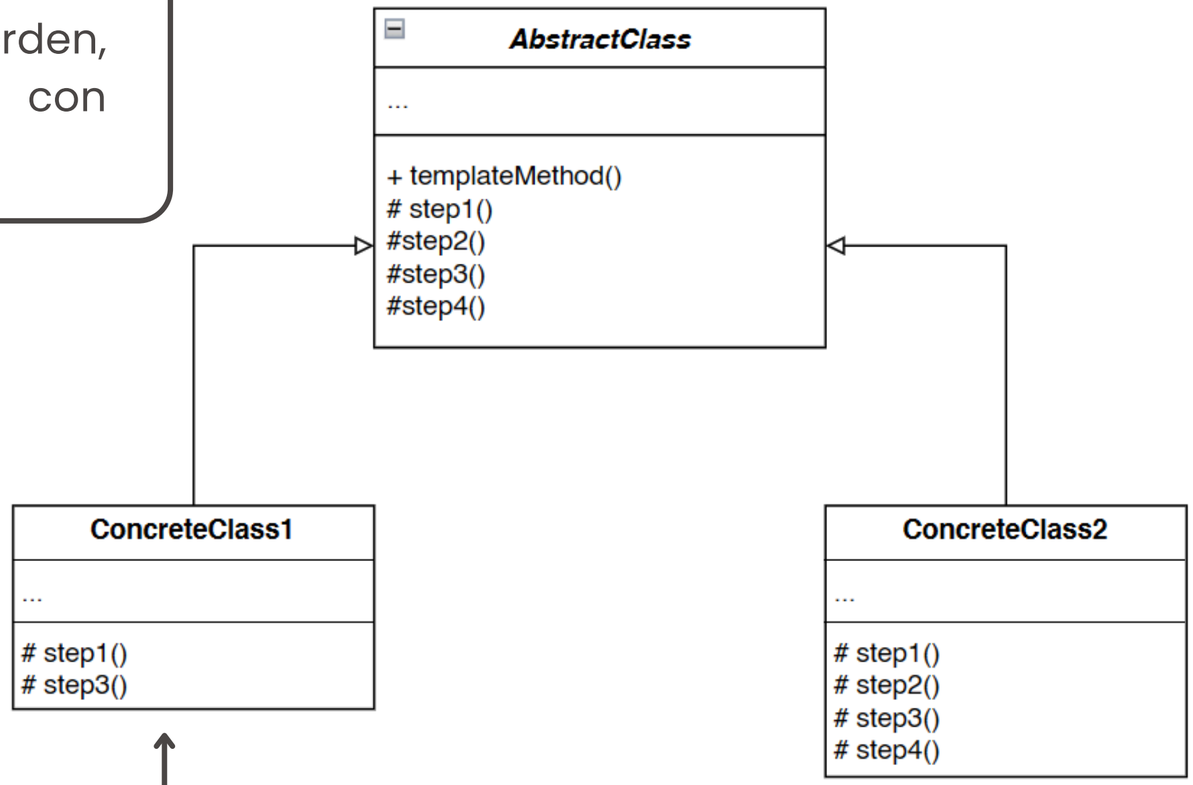
APLICAR

TEMPLATE METHOD

✅ Solución

Encapsular la estructura del algoritmo en una **clase base** y dejar que las subclases implementen solo las partes específicas que varían.

AbstractClass:
La **Clase Abstracta** define un método plantilla que ejecuta pasos de un algoritmo en orden, con métodos abstractos o con implementación por defecto.

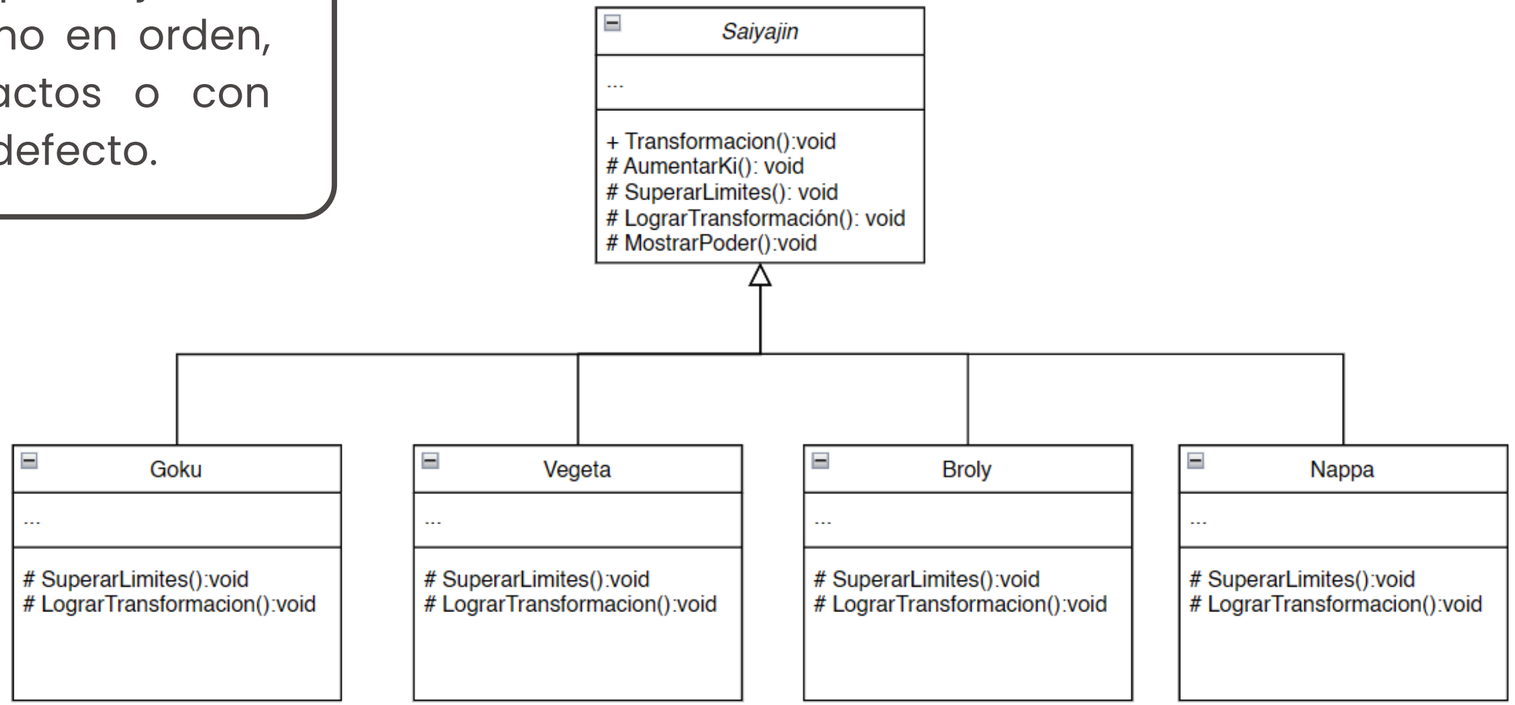


ConcreteClass:
Las **Clases Concretas** pueden sobrescribir todos los pasos, pero no el propio método plantilla.

Hook

Es un método que la clase base define, pero que las subclases pueden sobrescribir si lo necesitan.

AbstractClass:
La **Clase Abstracta** define un método plantilla que ejecuta pasos de un algoritmo en orden, con métodos abstractos o con implementación por defecto.



ConcreteClass:
Las **Clases Concretas** pueden sobrescribir todos los pasos, pero no el propio método plantilla.

Situaciones del día a día en las que es utilizado

El **Template Method** es útil cuando hay un **proceso estándar** con **pasos fijos**, pero algunos pasos deben ser personalizados según el contexto.

Pagos

1. Validar Pago
2. Cobrar
3. Enviar Confirmación

Autenticación

1. Solicitar credenciales
2. Validar Identidad
3. Redirigir usuario

Generación de reportes

1. Preparar datos
2. Formatear
3. Exportar

PROS

- **Reutilización de código:** estructura común en la superclase, menor duplicación.
- **Extensibilidad:** nuevas variantes sin modificar la base (Open/Closed).
- **Control del flujo:** orden predefinido, evitando alteraciones indebidas.
- **Flexibilidad:** métodos predeterminados sobrescribibles según necesidad.

Contras

- **Jerarquía rígida:** estructuras complejas y difíciles de modificar.
- **Curva de aprendizaje:** difícil de entender si hay muchas subclases.
- **Alto acoplamiento:** subclases dependen de la superclase.
- **Alternativa:** Strategy Pattern puede ser mejor para mayor flexibilidad.

Fuentes

https://refactoring.guru/es/design-patterns/template-method

Nicolas Battaglia - Patrón de diseño Template Method en C# VIDEO

DotTech-ES - ¿Cómo usar el patrón de diseño Template-Method? [TypeScript | JavaScript]

geeksforgeeks - Template Method Design Pattern