Universidade Tecnológica Federal do Paraná

Engenharia de Software - Curso de Arquitetura de Software (AS27S)

INSTRUTOR: Prof. Dr. Gustavo Santos

Kamila Antunes de Souza Neves, 2204690

CCH - Design Patterns Template Method – Spider-Man

Problema

O padrão de projeto Template Method é um padrão comportamental que define o esqueleto de um algoritmo em uma classe base, delegando a implementação de partes específicas desse algoritmo para as classes derivadas. Ele permite que as subclases forneçam implementações diferentes para partes específicas do algoritmo, enquanto mantém a estrutura geral inalterada.

Descrição da Solução

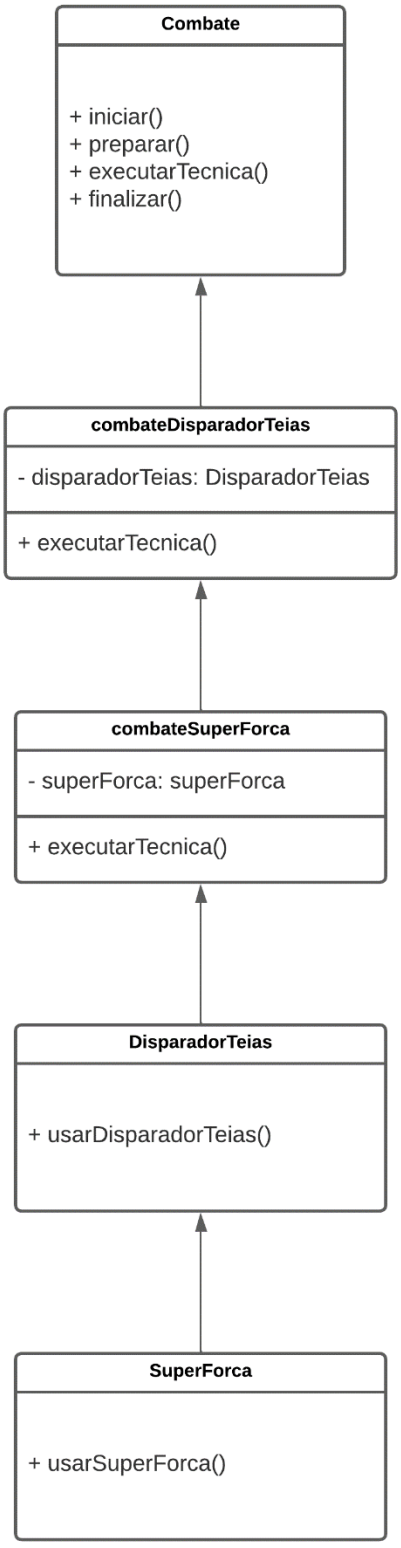
No contexto Spider-Man, pode-se considerar um cenário em que o Spider-Man possui diferentes técnicas de combate. Cada técnica segue uma sequência de passos semelhante, mas com implementações específicas para cada técnica. No entanto, a estrutura geral do combate é a mesma para todas as técnicas. Nesse caso, o padrão Template Method pode ser útil para definir a estrutura geral de combate e permitir que as subclasses implementem as técnicas de combate específicas.

O objetivo desse código é demonstrar o uso do padrão de projeto comportamental Template Method relacionado ao Spider-Man utilizando a linguagem de programação JavaScript.

No exemplo, a classe **Combate** represente a classe base que define o esqueleto do algoritmo de combate do Spider-Man. Ela possui um método **iniciar()** que define a sequência geral do combate, chamando os métodos **preparar()**, **executar()** e **finalizar()**.

As classes **CombateDisparadorTeias** e **CombateSuperForca** são subclasses de **Combate**. Elas herdam a estrutura geral o combate e implementam o método **executarTecnica()** de acordo com as técnicas específicas do Spider-Man.

Visão Geral [exemplo]



Temos a classe **Combate** com os métodos **iniciar()**, **preparar()**, **executarTecnica()** e **finalizar()**. As subclasses **CombateDisparadorTeias** e **CombateSuperForca**

herdam de **Combate** e fornecem suas próprias implementações para o método **executarTecnica()**. As classes **DisparadorTeias** e **SuperForca**, são utilizadas pelas subclasses para realizar ações específicas.

- **Combate** – define o esqueleto do algoritmo e as subclasses e fornecem implementações específicas para a técnica e combate usando disparador de teias e super força.
- **Iniciar()** – método principal que coordena a sequência geral do combate. Ela chama os métodos **preparar()**, **executarTecnica()** e **finalizar()**. Fornece uma estrutura comum para o combate, definindo a ordem em que as etapas devem ser executadas.
- **Preparar()** – método responsável por realizar as ações de preparação antes do combate, permite que as subclasses possam adicionar comportamentos específicos de preparação.
- **ExecutarTecnica()** – método abstrato que deve implementado pelas subclasses. Representa a técnica de combate específica que será executada.
- **Finalizar()** – método responsável por finalizar o combate.
- **usarDisparadorTeias()** – método que representa a ação de disparar teias.
- **usarSuperForca()** – método que representa a ação de usar super força.

Exemplo de Código JavaScript

O código pode ser acessado através do repositório disponibilizado através do link – CCH 3 Padrões de projetos comportamentais.

Consequências [vantagens e desvantagens]

- Vantagens
    - Reutilização de código: o Template Method permite definir uma estrutura comum e reutilizável para algoritmos similares.

- Extensibilidade: permite que as subclasses forneçam implementações específicas para as partes do algoritmo que variam, mantendo a estrutura geral intacta.
- Manutenção simplificada: ao definir o algoritmo em uma classe base, as alterações ou melhorias podem ser aplicadas centralmente e refletidas em todas as subclasses;

- Desvantagens
  - Rigidez – a estrutura geral do algoritmo é fixa, tornando difícil a modificação da sequência de passos sem alterar a classe base.
  - Complexidade adicional – o uso do padrão Template Method pode adicionar complexidade ao código, especialmente quando há muitas etapas ou partes de algoritmo.