Template Method.

Temas relacionados:Herança, classes abstratas e concretas, uso de interfaces, modificadores de acesso protected, public, private.

O Template Method é classificado como padrão comportamental, este padrão contribui com a implementação de algoritmos, reusando e organizando o código.E de que forma isto acontece, primeiramente vamos pensar em como ele pode contribuir. Quando falamos em algoritmos, logicamente falamos em problemas e então vamos lembrar que para resolvermos os problemas um algoritmos pode ser composto por vários passos, vários trechos, muitas vezes divididos em vários métodos. Ou até mesmo invocando outras classes para resolverem partes(passos) do algoritmo. E importante também lembrar que existe uma determinada sequência lógica para execução destes passos até que se chegue a uma solução para o algoritmo.

É aí que entra o Template Method, com ele podemos definir "fixar" a sequencia lógica para execução dos diversos passos de um algoritmo e proporcionar o reuso do código relativo à execução deste passos onde for necessário e também, forçar para que determinados passos sejam adaptados a cara utilização.

Para que fique mais claro vamos lembrar do conceito de Herança, a idéia deste padrão é basicamente que seja implementada uma classe que servira como base, nesta classe é definida a ordem de execução dos passos do algoritmo, bem como as chamadas, invocações entre eles, e também implementados os passos(métodos) que podem ser reutilizados em todas as diversas subclasses que herdarem desta classe Template. Além disso a classe Template é utilizada para conter a implementação concreta dos métodos que podem ser reutilizados, e a declaração de métodos abstratos, estes devem conter o trecho de código que deve ser implementado especificamente em cada subclasse, sua implementação é específica em cada subclasse pois devem ser personalizados a cada situação.

Portanto imagine uma situação onde você implementou um algoritmo composto de cinco passos, para realizar um processo relacionado à entidade Cliente do seu sistema. Após algum tempo surge a necessidade de disponibilizar este mesmo processo para a entidade Fornecedor. Neste momento você observa que 3 dos cinco passos podem ser totalmente reutilizados, ou seja você não precisaria reimplementar. Agora você encontrou uma situação onde o padrão Template Method se encaixa perfeitamente. A implementação deve consistir em uma classe base que contenha 3 métodos concretos e 2 métodos abstrados. Os métodos abstratos serão implementados de forma concreta nas superclasses ProcessoCliente e ProcessoFornecedor.

Observe o diagrama abaixo ilustrando uma estrutura de implementação classica do padrão Template Method.

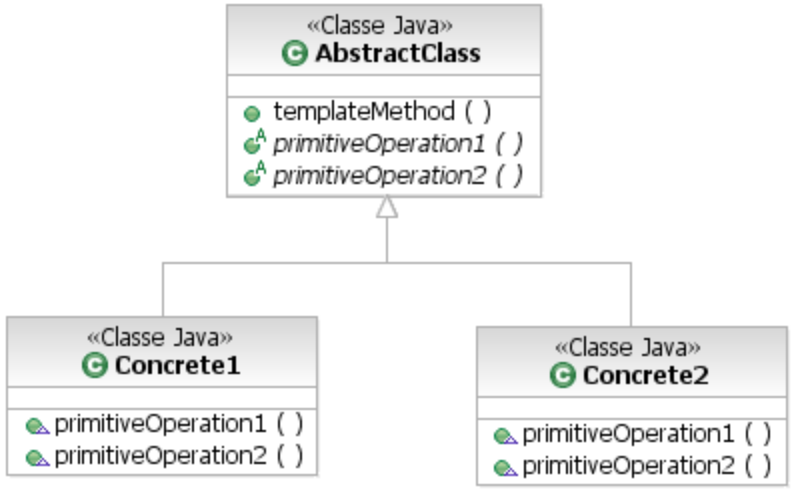

figura1:

Observe o diagrama agora relacionado ao nosso exemplo:

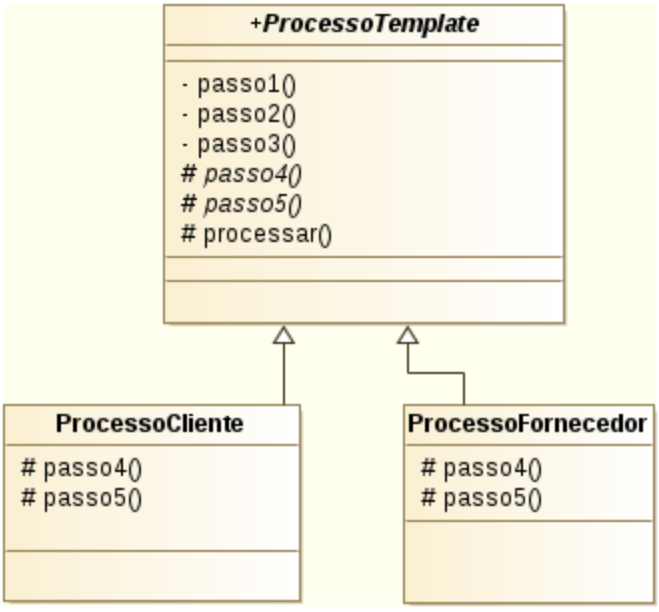

figura 2

Intenção:

Definir o esqueleto de um algoritmo em uma operação, postergando alguns passos para as subclasses. Template Method permite que suclasses redefinam certos passos de um algoritmo sem mudar sua estrutura.

Motivação:

Reuso de código, organização da execução dos algoritmos.

A soluções construídas com esse padrão se utilizam deu uma classe para definir a sequência lógica da execução de um algoritmo, mas permite que haja certa flexibilidade pois alguns passos podem ser definidos de forma diferente em cada subclasse se necessário.

Aplicabilidade:

O padrão Template Method pode ser usado:

- para implementar partes invariantes do código uma só vez e deixar para as subclasses a implementação do comportamento que pode variar.
- Quando o comportamento comun entre subclasses deve ser fatorado e concentrado numa classe comun para evitar duplicação de código. Primeiramente você identifica as diferenças no código existente e então separa as diferenças em novas operações . Por fim você substitui o código que apresentava diferenças por um método template que invoca uma dessas novas operações.

Estrutura:

Uma classe abstrata, e suas subclasses conforme o a figura 1.

Participantes:

- Abstract class
    - define operações primitivas abstratas que as sublcasses concretas definem para implementar o algoritmo.
    - implementa pelo menos um metodo template que define o esqueleto de um algoritmo. O método template invoca operações primitivas, bem como as operaçőes definidas em abstractclass ou ainda outros objetos.
- Concrete class
    - Implementa as operações primitivas para executarem os passos específicos do algoritmo da sublcasse.

Colaborações:

ConcreteClass depende de **AbstractClass** para implementar os passos invariantes do algoritmo.

Implementação:

- Aspectos dignos de nota
  - Utilize bem os modificadores de acesso para definir quais métodos devem , podem ou não podem ser sobrescritos nas subclasses.
  - Minimize operações primitivas
  - Use convenções de nomenclatura (doRead,doSave...)

Usos conhecidos:

Quase todas as classes abstratas.

Padrões relacionados:

- Factory Methods (invocação de métodos)
- Stategy (uso de herança para variar parte do algoritmo)

Referência:

Título	**Padrões de Projetos: Soluções Reutilizáveis**
Autores	Erich Gamma, Ralph Johnson, Richard Helm, John Vlissides
Editora	Bookman Companhia Ed, 2006
ISBN	8573076100, 9788573076103
Num. págs.	364 páginas