AULA 7 – PADRÃO *TEMPLATE METHOD*

GSI020 – Programação Orientada a Objetos II

Prof. Dr. Murillo G. Carneiro
mgcarneiro@ufu.br

UFU

Material baseado nos slides cedidos pelo Prof. Rafael D. Araújo (FACOM/UFU)

Objetivo da aula

- Entender o funcionamento do padrão de projeto *Template Method* (Método-Template).

Template Method

- É um padrão de projeto de propósito **comportamental** com escopo de **classes**.
- Consiste na criação de um *template* ou gabarito para determinadas operações.

Intenção

- Definir o esqueleto de um algoritmo em uma operação, postergando alguns passos para as subclasses.
- Permite que subclasses redefinam certos passos de um algoritmo sem que sua estrutura seja modificada.
- Desta forma, um ou mais passos do algoritmo podem ser sobrescritos por subclasses, possibilitando diferentes comportamentos.

Quando usar

- Quando um algoritmo apresenta apenas partes variantes cujo comportamento é definido por subclasses.
- Quando o comportamento comum entre subclasses deve ser fatorado para evitar duplicação de código.
- Definir extensões de comportamento apenas em pontos específicos de algoritmos.

Operações gancho (*hook methods*)

- Define um ponto de extensão para que parte da operação (comportamento) seja especializada.
- No padrão método-template é importante especificar quais operações ganchos *podem* ser redefinidas e quais *devem* ser redefinidas (*abstract*).

Orientações

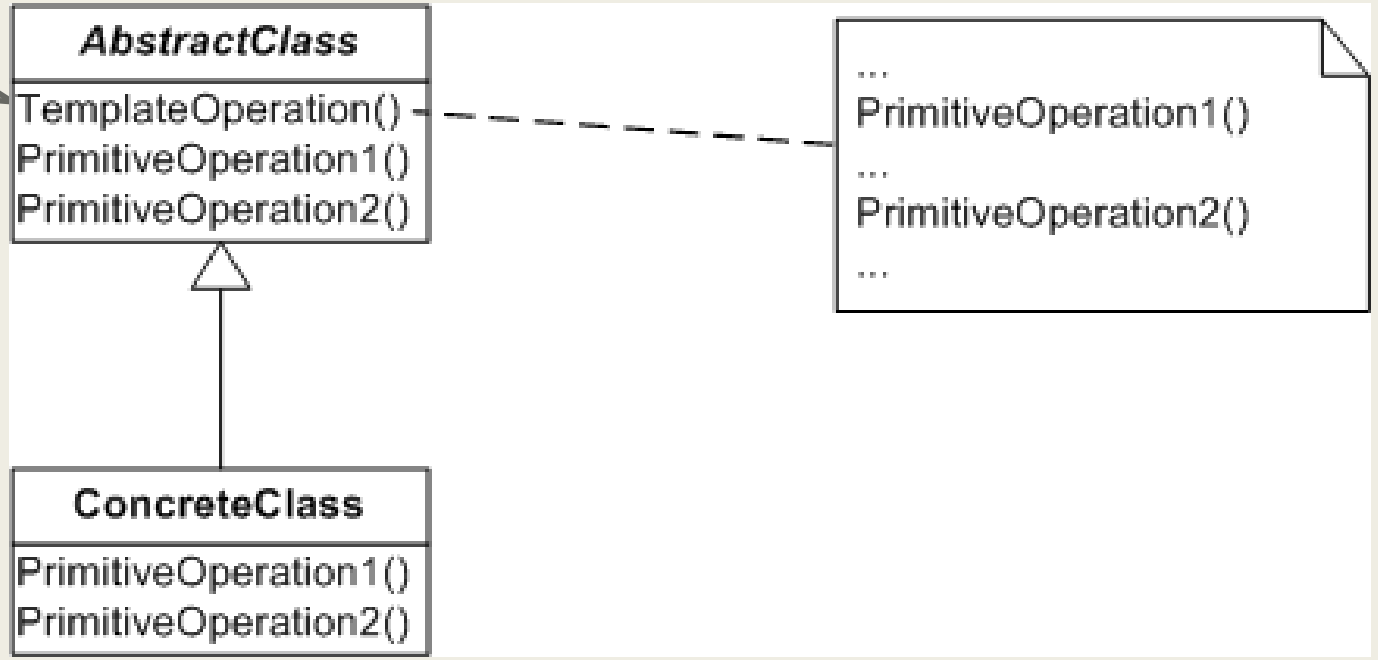
- Primeiro identifique as diferenças nos algoritmos
- Coloque as diferenças em novos métodos
- Substitua o código das diferenças por uma chamada a um dos novos métodos

Elementos participantes

- **AbstractClass**, define as operações primitivas abstratas (partes do algoritmo), que devem ser implementadas pelas subclasses e define o método-template, que invoca operações primitivas.
- **ConcreteClass**, implementa as operações primitivas para executarem passos específicos do algoritmo.

Estrutura

O método-template deve ser final para não permitir alteração.



Benefícios

- Técnica fundamental para reutilização de código
- Evita a replicação de código
- Os componentes de alto nível dizem aos componentes de baixo nível: “não nos telefone, nós telefonaremos pra você.” - Princípio de *Hollywood*.
- Muito utilizado na construção de *frameworks*

É uma aplicação semi-completa e reutilizável que pode ser especializada para produzir aplicações customizadas.

Desvantagens

- Entender a sequência do fluxo das chamadas de métodos pode ficar confuso.
- Métodos que não deveriam ser implementados acabam sendo implementados (ou vice-versa).
- Não é possível trocar o algoritmo inteiro por outra implementação (a seleção do algoritmo acontece em tempo de compilação).
- A manutenção pode se tornar difícil (pois cada pedaço dos algoritmos podem estar em classes diferentes).

Problema prático

- Implemente três tipos de funcionários comissionáveis: Atendente, Vendedor e Gerente.
- Atendente recebe 1% sobre as vendas, o Vendedor recebe 2% sobre as vendas e o Gerente recebe 3% sobre as vendas.
- Cada tipo de funcionário também possui uma meta mensal: Atendente (10.000), Vendedor (15.000), Gerente (20.000). Caso a meta não seja atingida, o valor da comissão é reduzido para 80% do valor combinado.
- Utilize o padrão *Template Method*.

Problema prático

1) Crie uma aplicação (main) que possui 3 funcionários, um de cada tipo.
2) Defina os seguintes valores de vendas: Atendente (5.000), Vendedor (16.000) e Gerente (18.000).
3) Imprima cada elemento da lista com o tipo, a meta mensal, o valor da venda e o valor da comissão.

Referências

- GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995. Capítulo 5.