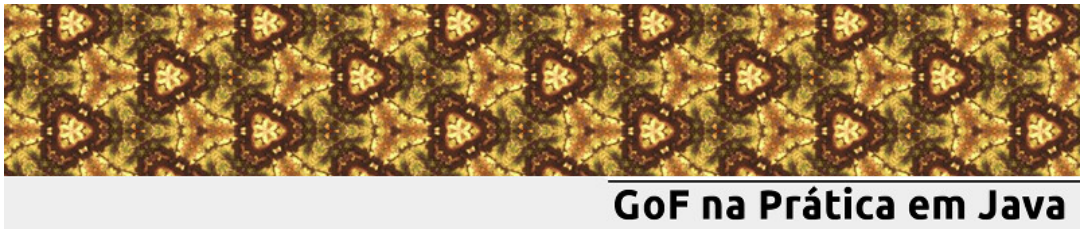
Template Method

Fernando Anselmo
GoF na Prática em Java

Função deste Padrão

Definir um esqueleto de um algoritmo em uma operação, permitindo que sub classes componham o algoritmo e tenham a possibilidade de redefinir certos passos a serem tomados no processo, sem contudo mudá-lo.



1 Ficha do Padrão

Tipo : Comportamental, especificamente voltados para a comunicação entre objetos.

Conhecimentos : Classes Abstratas e Classes Concretas.

Consequências : Técnicas fundamental de reuso de código. Importante em bibliotecas de classes, por ser uma maneira para refatorar o comportamento comum fora de seu código.

É usado quando : Implementar partes invariáveis de um algoritmo e deixar que as subclasses implementem os comportamentos variáveis. Comportamentos comuns entre subclasses devem ser fatorados e localizados em uma classe comum para evitar a duplicação de código.

2 Problema

O usuário deseja trabalhar com dois tipos de item: Item Geral que possui um estoque na loja e outro na matriz e Item Raro que só pode ser retirado uma determinada quantidade por vez. Porém essas classes devem possuir os mesmos métodos descritos na classe Item de modo a padronizar seu uso.

3 Prévia Estrutura de Classes

Classe Abstrata para implementação dos itens:

Listagem 1: *Classe Item*

```
1 abstract class Item {
2   public final boolean remover(int qtd) {
3     if (!validarQtd(qtd)) {
4       System.out.println("Impedir remover.");
5       return false;
6     }
7     if (getEstoque() >= qtd) {
8       decEstoque(qtd);
9       System.out.println("Itens removidos.");
```

```
      return true;
    }
    System.out.println("Impedir remover.");
    return false;
  }
  protected abstract boolean validarQtd(int qtd);
  protected abstract int getEstoque();
  protected abstract void decEstoque(int qtd);
}
```

Classe Final para o teste das classes:

Listagem 2: *Classe Loja*

```
public class Loja {
  public static void main(String[] arg) {
    new Loja().negociar();
  }
  public void negociar() {
    System.out.println("Exemplo de Item Geral");
    Item normal = new ItemGeral("Sal", 20, 10);
    System.out.println(normal);
    System.out.println("Obter 6 Unidades:");
    normal.remover(6);
    System.out.println(normal);
    System.out.println("Obter 17 Unidades:");
    normal.remover(17);
    System.out.println(normal);

    System.out.println("Exemplo de Item Raro");
    Item raro = new ItemRaro("Plutonio", 5, 20);
    System.out.println(raro);
    System.out.println("Obter 6 Unidades:");
    raro.remover(6);
    System.out.println(raro);
    System.out.println("Obter 3 Unidades:");
    raro.remover(3);
    System.out.println(raro);
  }
}
```

4 Aplicação do Padrão

Classe para a implementação do Item Geral:

Listagem 3: *Classe ItemGeral*

```
class ItemGeral extends Item {
  private String nome;
  private int qtd;
  private int qtdMatriz;

  public ItemGeral(String nome, int qtd, int qtdMatriz) {
    this.nome = nome;
    this.qtd = qtd;
    this.qtdMatriz = qtdMatriz;
  }
  public String toString() {
    return nome + ". Na Loja: " + qtd + " Na Matriz: " + qtdMatriz;
```

```
  }
  protected boolean validarQtd(int qtd) {
    return qtd >= 0;
  }
  protected int getEstoque() {
    return qtd + qtdMatriz;
  }
  protected void decEstoque(int qtdDec) {
    if (qtdDec > this.qtd) {
      qtdDec -= this.qtd;
      this.qtd = 0;
      qtdMatriz -= qtdDec;
    } else {
      this.qtd -= qtdDec;
    }
  }
}
```

Classe para a implementação do Item Raro:

Listagem 4: *Classe ItemRaro*

```
class ItemRaro extends Item {
  private String nome;
  private int qtd;
  private int maxRetirada;

  public ItemRaro(String nome, int maxRetirada, int qtd) {
    this.nome = nome;
    this.maxRetirada = maxRetirada;
    this.qtd = qtd;
  }
  public String toString() {
    return nome + " - Qtd: " + qtd + " (Max.Retirada: " + maxRetirada + ")";
  }
  protected int getEstoque() {
    return qtd;
  }
  protected void decEstoque(int qtd) {
    this.qtd -= qtd;
  }
  protected boolean validarQtd(int qtd) {
    return qtd >= 0 && qtd <= Math.min(qtd, maxRetirada);
  }
}
```

Referências

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides *Design Patterns. Elements of Reusable Object-Oriented Software 1 ed.* Estados Unidos, Addison-Wesley, 1995, ISBN 0-201-63361-2.