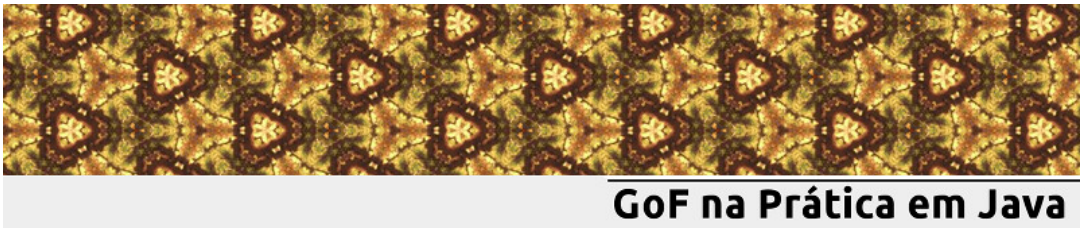
Memento

Fernando Anselmo
GoF na Prática em Java

Função deste Padrão

Capturar e expor o estado interno de um objeto (como uma foto), não violando o seu encapsulamento. Expor o estado para que o objeto possa ter este estado restaurado posteriormente.



1 Ficha do Padrão

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

Conhecimentos : Coleções de Objetos e Classes Concretas.

Consequências : Preservar o encapsulamento e simplificar o original.

É usado quando : O estado instantâneo de um objeto deve ser salvo até que possa ser restaurado mais tarde. Um acesso direto a interface para obter o estado do objeto expõe detalhes de implementação.

2 Problema

O usuário precisa implementar na classe que representa uma Ação de uma determinada Empresa como forma de preservar valores modificados para retorná-los em caso de mudanças indevidas.

3 Prévia Estrutura de Classes

Classe que representa a Ação de uma determinada Empresa e que deve ter o estado do valor salvo:

Listagem 1: *Classe Acao*

```
1  class Acao {
2    private String empresa;
3    private float valor;
4
5    public Acao(String empresa) {
6      this.empresa = empresa;
7    }
8    public String toString() {
9      return empresa + " - " + valor;
10   }
11   public void setValor(float valor) {
12     this.valor = valor;
13   }
14 }
```

4 Aplicação do Padrão

Classe para guardar o valor:

Listagem 2: *Classe AcaoMemento*

```
class AcaoMemento {
  private float valor;

  public AcaoMemento(float valor) {
     this.valor = valor;
  }
  public float getState() {
    return valor;
  }
}
```

Nova classe Acao com a aplicação do padrão:

Listagem 3: *Classe Acao*

```
class Acao {
  private String empresa;
  private float valor;
  private java.util.List<AcaoMemento> memoria = new java.util.ArrayList<AcaoMemento>();

  public Acao(String empresa) {
    this.empresa = empresa;
  }
  public String toString() {
    return empresa + " - " + valor;
  }
  public void setValor(float valor) {
    this.valor = valor;
    memoria.add(new AcaoMemento(valor));
  }
  public void undo() {
    if (memoria.size() > 1) {
      memoria.remove(memoria.size() - 1);
      this.valor = memoria.get(memoria.size() - 1).getState();
    }
  }
}
```

Classe exemplo para verificar o funcionamento dos retornos de valores:

Listagem 4: *Classe Bolsa*

```
public class Bolsa {
  public static void main(String [] args) {
    new Bolsa().movimentar();
  }
  public void movimentar() {
    Acao padrao = new Acao("Patterns Inc.");

    padrao.setValor(3.0f);
    System.out.println(padrao);

    padrao.setValor(8.0f);
    System.out.println(padrao);

```

```
14      padrao.setValor(25.0f);
15      System.out.println(padrao);
16
17      // Retornos
18      padrao.undo();
19      System.out.println(padrao);
20      padrao.undo();
21      System.out.println(padrao);
22      padrao.undo();
23      System.out.println(padrao);
24    }
25  }
```

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.