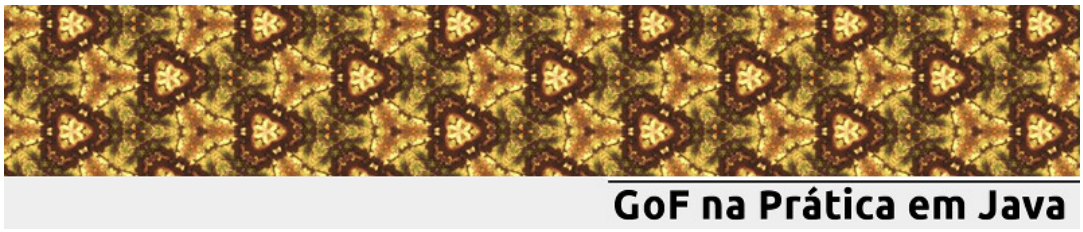
Mediator

Fernando Anselmo
GoF na Prática em Java

Função deste Padrão

Desligar e gerenciar as colaborações entre um grupo de objetos. Definir um objeto que encapsula as interações neste grupo.



1 Ficha do Padrão

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

Conhecimentos : Classes Concretas.

Consequências : Prover um baixo acoplamento e simplificar os protocolos de objetos. Abstrair a maneira como os objetos cooperam e centralizar o controle.

É usado quando : Para um conjunto de objetos que se comunicam de uma maneira bem definida porém complexa, o que resulta em interdependências desestruturadas e difíceis de entender. O reuso de um objeto é difícil por causa de suas referências e comunicação com outros objetos. Um comportamento que é distribuído entre várias classes deve ser customizado sem um conjunto de subclasses.

2 Problema

O usuário deseja construir simulaações de salas de bate-papo, para isso conseguiu definir a classe Usuário mais não sabe como avançar.

3 Prévia Estrutura de Classes

Classe básica para implementação do Usuário:

Listagem 1: *Classe Usuario*

```
1  class Usuario {
2    private String nome;
3
4    public Usuario(String nome){
5      this.nome = nome;
6    }
7    public String toString() {
8      return nome;
9    }
10 }
```

4 Aplicação do Padrão

Classe para implementação das salas de bate-papo:

Listagem 2: *Classe SalaChat*

```
class SalaChat {
  public static void show(Usuario usuario, String mens){
    System.out.println(usuario + ": " + mens);
  }
}
```

Modificação na classe Usuario após a implementação da sala de bate-papo:

Listagem 3: *Classe ExpressaoE*

```
class Usuario {
  private String nome;

  public Usuario(String nome){
    this.nome = nome;
  }
  public String toString() {
    return nome;
  }
  public void enviarMens(String mens){
    SalaChat.show(this, mens);
  }
}
```

Classe exemplo para uma sala de bate-papo:

Listagem 4: *Classe Analise*

```
public class Mirc {
  public static void main(String[] args) {
    new Mirc().abrirSala();
  }
  public void abrirSala() {
    Usuario fernando = new Usuario("Fernando");
    Usuario anselmo = new Usuario("Anselmo");
    fernando.enviarMens("Anselmo que Pattern estamos?");
    anselmo.enviarMens("Acho que no 17o.");
  }
}
```

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.