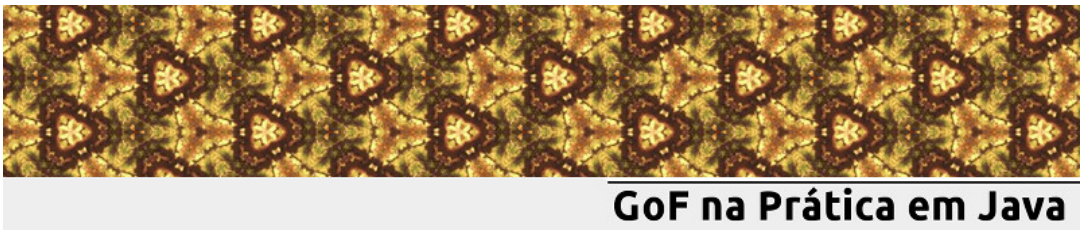
Adapter

Fernando Anselmo
GoF na Prática em Java

Função deste Padrão

Converte a interface da classe em outra interface esperada pelo cliente e permite que classes que não poderiam interagir devido a incompatibilidades das interfaces possam trabalhar em conjunto.



1 Ficha do Padrão

Tipo : Estrutural, diz respeito como classes e objetos podem ser combinados para formar grandes estruturas.

Conhecimentos : Interfaces e Classes Concretas.

Consequências : Permitir que a classe adaptadora reimplemente os comportamentos de uma outra classe. Introduz apenas um objeto, e não será necessário o uso de ponteiros adicionais para chegar a classe Adaptadora.

É usado quando : Usar uma classe já existente e sua interface não combina com a esperada pelo cliente. Uma classe reutilizável que coopera com classes não relacionadas ou não previstas, isto é, classes que não necessariamente tenham interfaces compatíveis. Usar várias subclasses existentes, mas é impraticável adaptar suas interfaces fazendo uma subclasse de cada uma.

2 Problema

Escreva uma classe que permita que um cliente da interface Vetor, que já usa a classe Figura, utilize outra classe CaixaArraste (e Coordenada) de modo a ficar transparente seu aprendizado.

3 Prévia Estrutura de Classes

Interface com a estrutura para organizar a família de figuras:

Listagem 1: *Interface Vetor*

```
1 interface Vetor {
2   int getPosX();
3   int getPosY();
4   int getLargura();
5   int getAltura();
6 }
```

Classe já utilizada pelo usuário:

Listagem 2: *Classe Figura*

```
class Figura implements Vetor {
  private int posX, posY, largura, altura;

  public int getPosX() {
    return posX;
  }
  public int getPosY() {
    return posY;
  }
  public int getLargura() {
    return largura;
  }
  public int getAltura() {
    return altura;
  }
}
```

Nova Classe “Caixa de Arrastar” a ser utilizada:

Listagem 3: *Classe CaixaArraste*

```
class CaixaArraste {
  private Coordenada topoEsq, rodapeDir;

  public Coordenada getTopoEsq() {
    return topoEsq;
  }
  public Coordenada getRodapeDir() {
    return rodapeDir;
  }
}
```

Classe auxiliar:

Listagem 4: *Classe Coordenada*

```
class Coordenada {
  private int x, y;

  public int getX() {
    return x;
  }
  public int getY() {
    return y;
  }
}
```

4 Aplicação do Padrão

Classe com a adaptação para a “Caixa de Arrastar”:

Listagem 5: *Classe CaixaArrasteAdapter*

```
class CaixaArrasteAdapter implements Vetor {
  private CaixaArraste caixaArraste = new CaixaArraste();

  public int getPosX() {
    Coordenada c = caixaArraste.getTopoEsq();
```

```
    return c.getX();
  }
  public int getPosY() {
    Coordenada c = caixaArraste.getTopoEsq();
    return c.getY();
  }
  public int getLargura() {
    Coordenada c1 = caixaArraste.getTopoEsq();
    Coordenada c2 = caixaArraste.getRodapeDir();
    return c2.getX() - c1.getX();
  }
  public int getAltura() {
    Coordenada c1 = caixaArraste.getTopoEsq();
    Coordenada c2 = caixaArraste.getRodapeDir();
    return c2.getY() - c1.getY();
  }
}
```

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.