



Universidade Federal de Uberlândia
Faculdade de Computação
Prof. Fabiano Azevedo Dorça
Programação Orientada a Objetos II

Padrões de Projeto

Padrão Decorator (Decorador)

Padrão Decorator (Decorador)

Problema:

- Em algumas situações é necessário adicionar responsabilidades à objetos específicos, e não a toda a classe.
- O padrão de projeto Decorator resolve o problema, permitindo que tais responsabilidades sejam adicionadas individualmente, em tempo de execução.
- Ao contrário da herança que aplica funcionalidades a todos os objetos da classe, o padrão decorator permite aplicar funcionalidades apenas a um objeto específico.

Padrão Decorator (Decorador)

- O padrão *Decorator* permite estender as funcionalidades de um objeto em tempo de execução usando uma forma de composição de objetos.
- A ideia central é modificar o comportamento de um método em um objeto, adicionando-lhe processamento adicional.

Padrão Decorator (Decorador)

- Adiciona responsabilidades de forma dinâmica a um objeto.
- Os Decoradores fornecem uma alternativa flexível à herança para estender funcionalidades.

Padrão Decorator (Decorador)

[GAMMA]

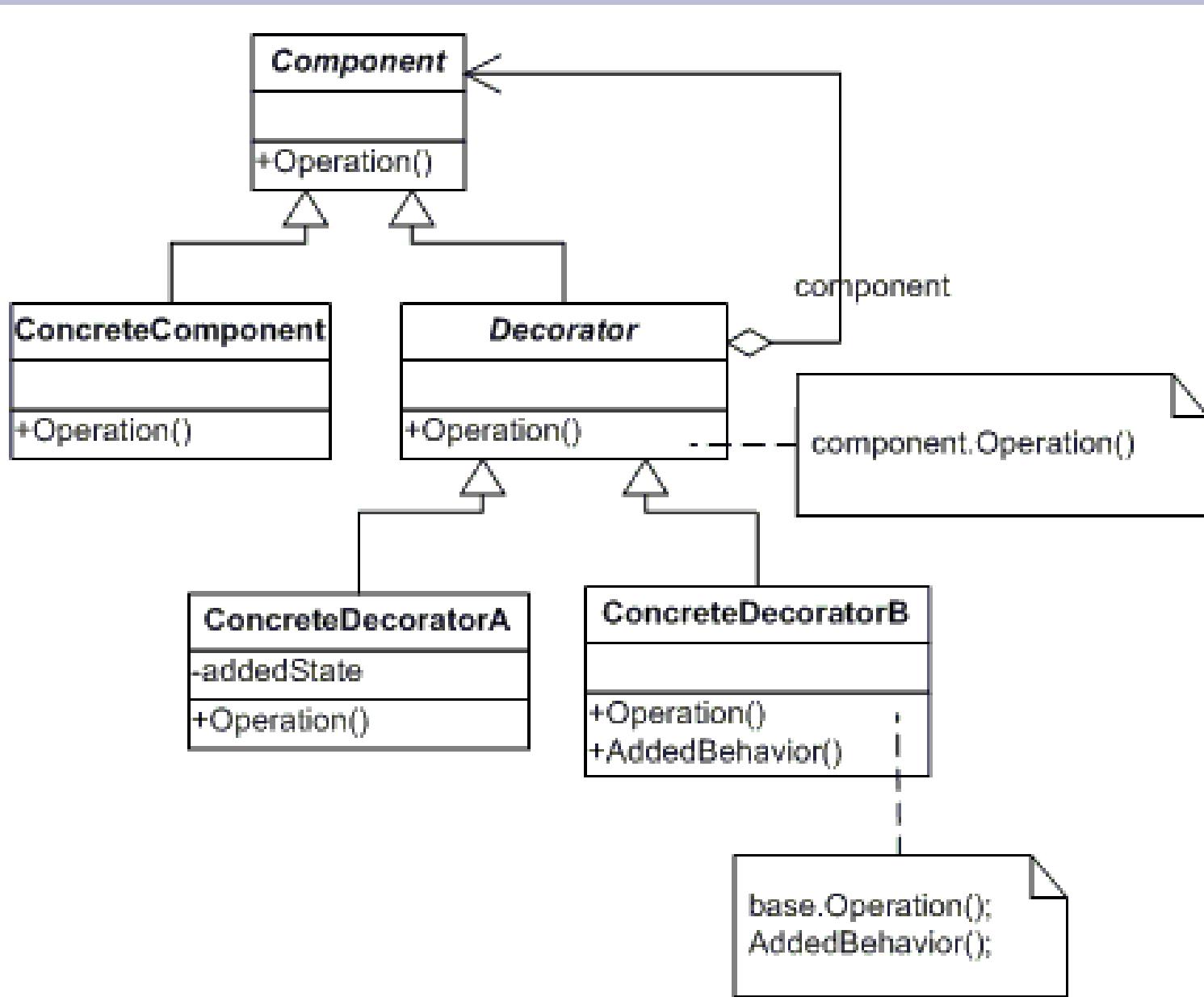
- Evita classes sobre carregadas de características na parte superior da hierarquia.
- Oferece uma abordagem do tipo 'use quando for necessário'.
- Pode-se definir uma classe simples e acrescentar funcionalidade de modo incremental.

Padrão Decorator (Decorador)

[GAMMA]

- A funcionalidade necessária pode ser composta a partir de peças simples.
- É fácil definir novas espécies de decorators independente das classes de objetos que eles estendem.
- Desta forma uma aplicação não precisa incorrer no custo de características e recursos que não usa.

Padrão Decorator (Decorador)



Padrão Decorator (Decorador)

Participantes:

As classes e/ou objetos que participam do padrão são:

Component

- Define a interface dos objetos que podem ter responsabilidades adicionadas de forma dinâmica.

ConcreteComponent

- Define um objeto ao qual mais responsabilidades podem ser adicionadas.

Padrão Decorator (Decorador)

Decorator

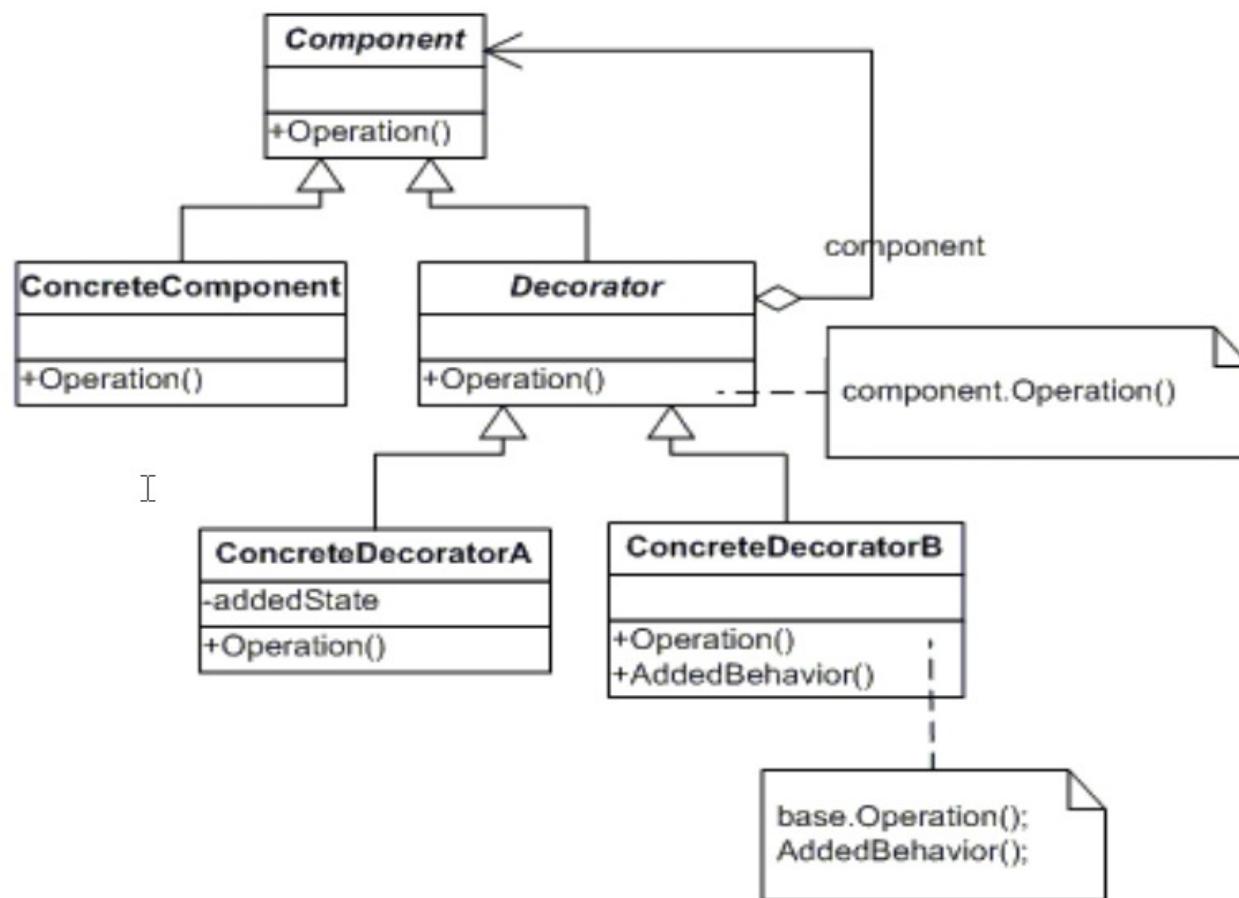
- Armazena a referência para um objeto Component, cujo comportamento será modificado.

ConcreteDecorator

- Adiciona responsabilidades ao componente.

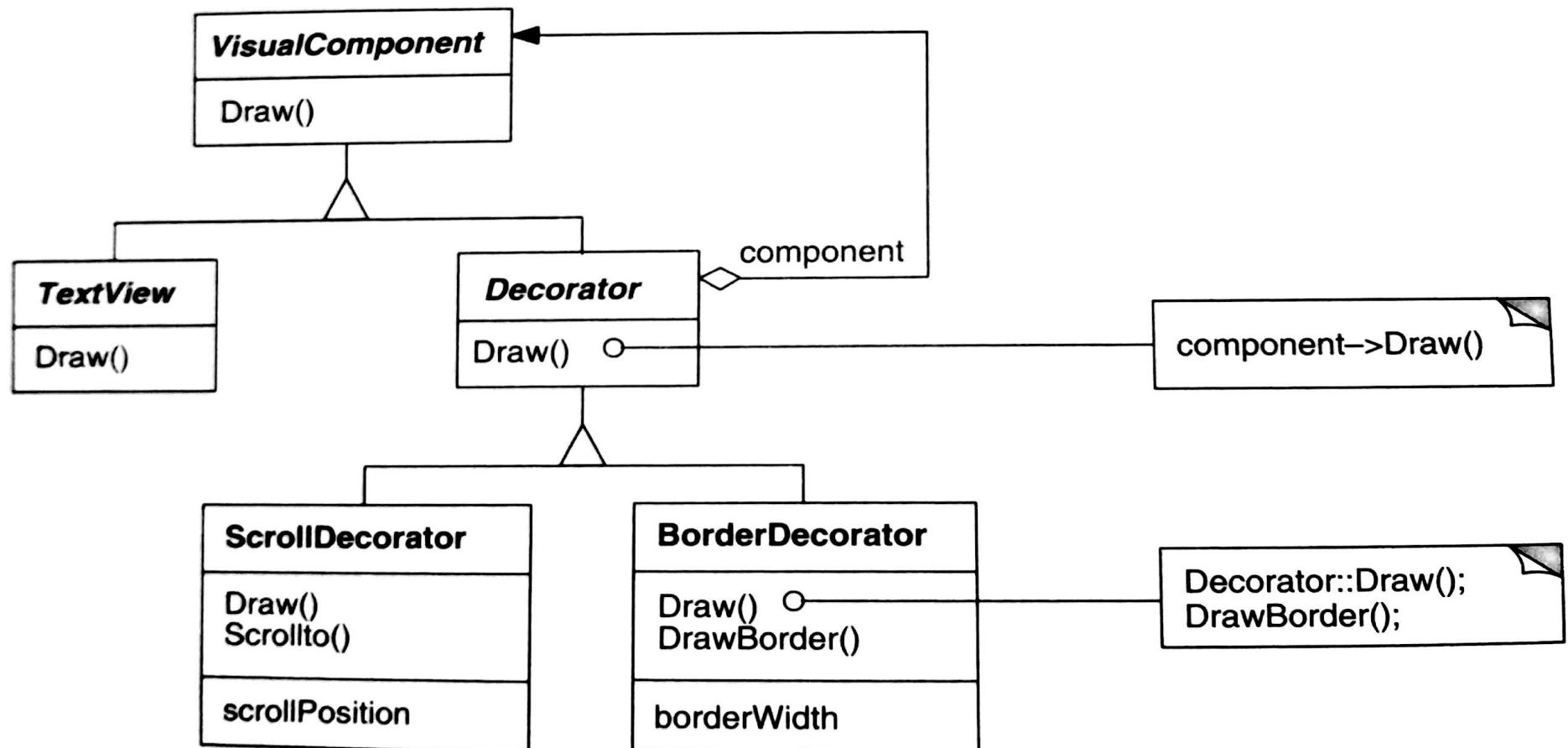
Padrão Decorator (Decorador)

Diagrama de classes



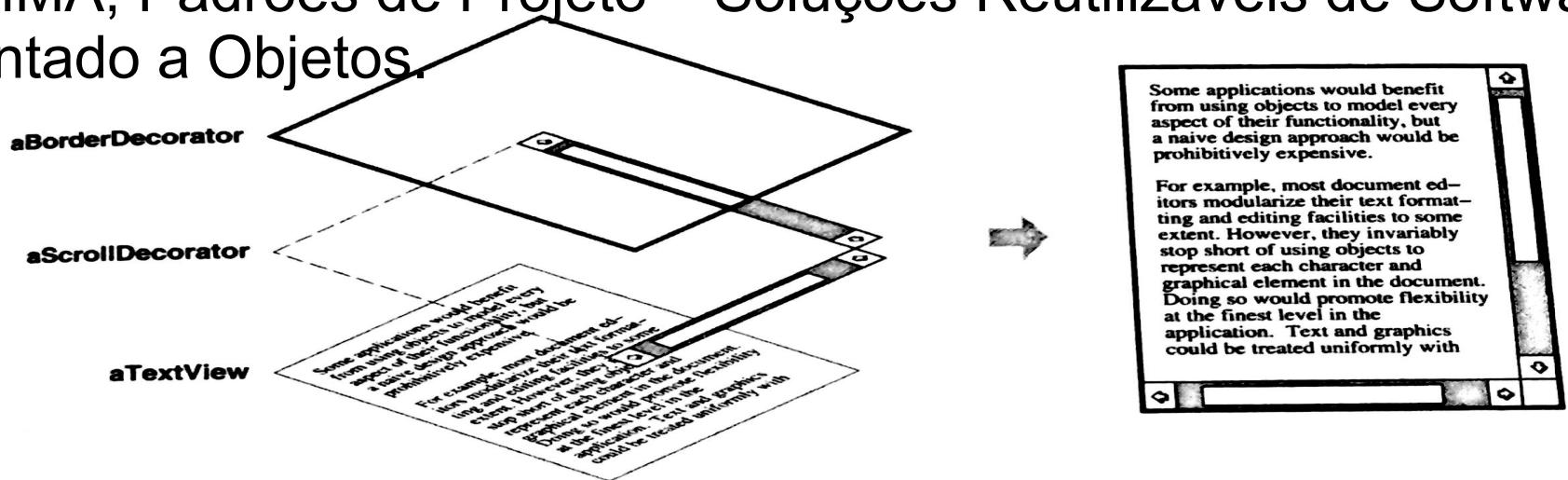
Padrão Decorator (Decorador)

GAMMA, Padrões de Projeto.



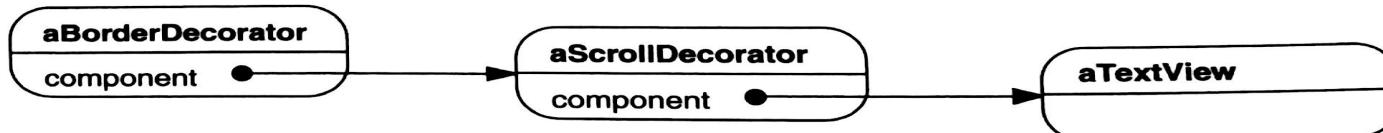
Padrão Decorator (Decorador)

GAMMA, Padrões de Projeto – Soluções Reutilizáveis de Software Orientado a Objetos.



Suponha, também, que queiramos acrescentar uma borda preta espessa ao redor do objeto **TextView**. Também podemos usar um objeto **BorderDecorator** para esta finalidade. Simplesmente compomos os decoradores com **TextView** para produzir o resultado desejado.

O diagrama de objetos abaixo mostra como compor um objeto **TextView** com objetos **BorderDecorator** e **ScrollDecorator** para produzir uma visão do texto cercada por bordas e rolável:



Padrão Decorator (Decorador)

É importante notar que:

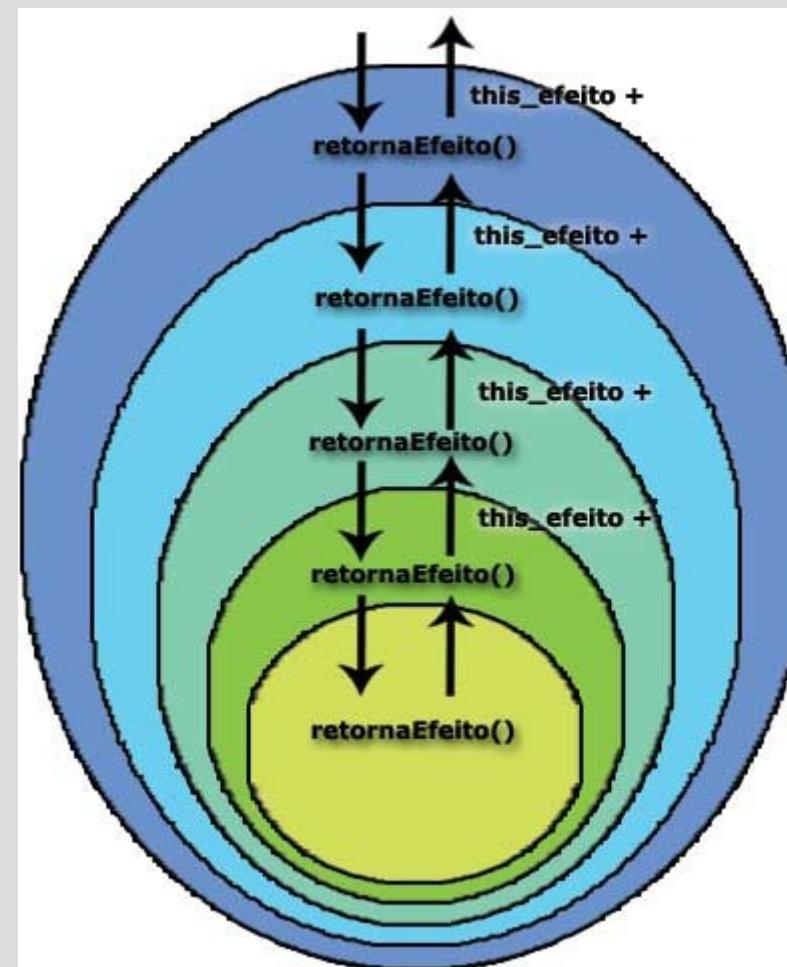
- Os decoradores têm o **mesmo super-tipo** que os objetos que eles decoram, **de modo que sua presença é transparente para os clientes do componentes que eles decoram.**
- Uma vez que o decorador tem o mesmo super-tipo que o objeto decorado, pode-se passar um objeto decorado no lugar do objeto original (englobado);
- Pode-se usar um ou mais decoradores para englobar um objeto;

Padrão Decorator (Decorador)

- O decorador adiciona seu próprio comportamento antes e/ou depois de delegar ao objeto que ele decora o resto do trabalho;
- Os objetos podem ser decorados a qualquer momento.
- Então pode-se decorar os objetos de maneira dinâmica no tempo de execução com quantos decoradores desejarmos.

Padrão Decorator (Decorador)

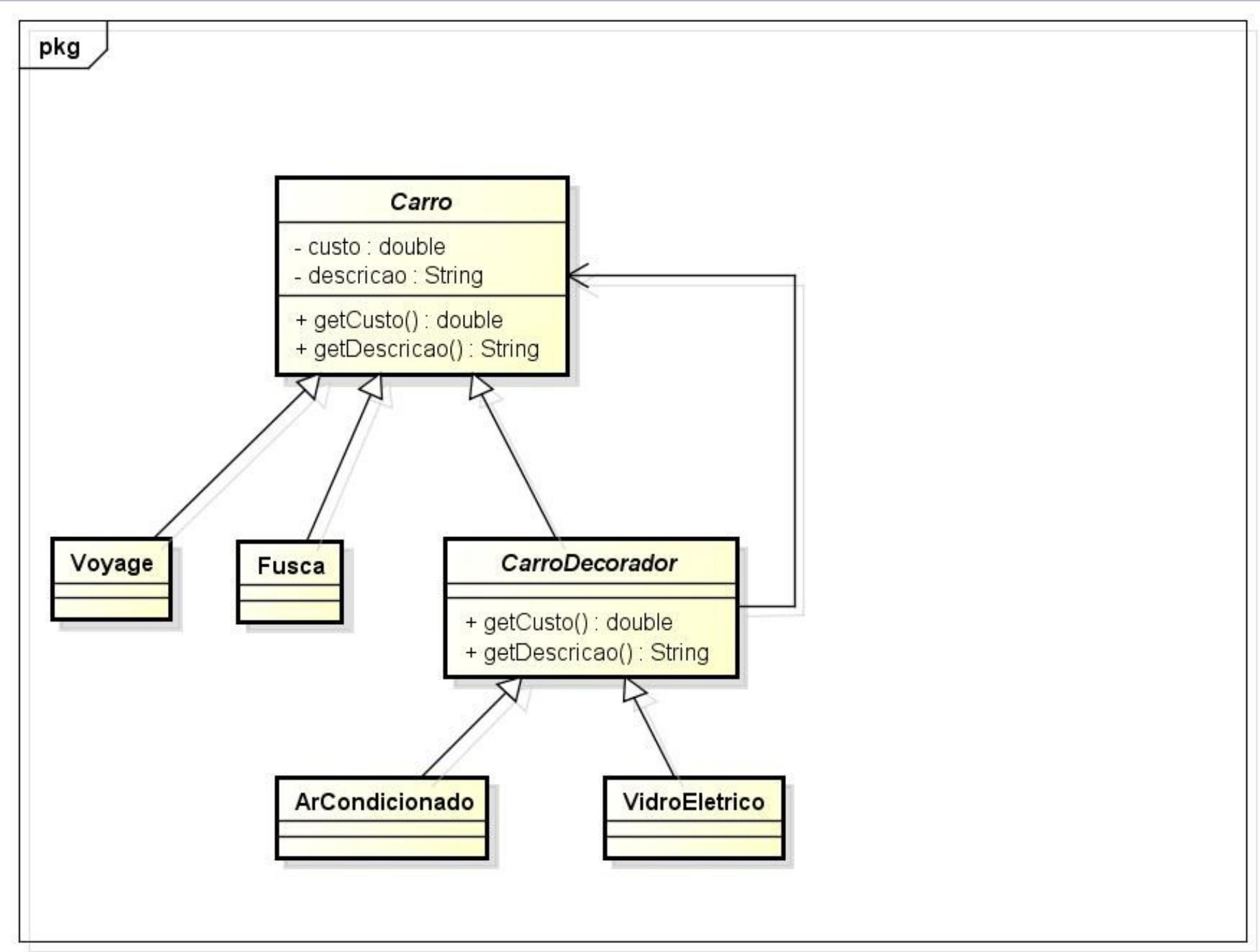
Efeito decorator...



Padrão Decorator (Decorador)

Exemplo:

Um *website* que auxilia a montagem de veículos pelo cliente, podendo então adicionar novos acessórios aos veículos, como ar condicionado, vidro elétrico, travas elétricas e direção hidráulica, alterando assim as características finais do produto.



Padrão Decorator (Decorador)

//Component

```
public abstract class Carro {  
    private double custo;  
    private String descricao;  
  
    public double getCusto() {  
        return this.custo;  
    }  
    public String getDescricao() {  
        return this.descricao;  
    }  
  
    public void setCusto(double custo){  
        this.custo = custo;  
    }  
    public void setDescricao(String descricao){  
        this.descricao = descricao;  
    }  
}
```

Padrão Decorator (Decorador)

```
//ConcreteComponent
public class Voyage extends Carro {
    Voyage() {
        this.setCusto(60.000);
        this.setDescricao("Voyage");
    }
}
```

Padrão Decorator (Decorador)

```
//Decorador
public abstract class CarroDecorador extends Carro {
    private Carro carroDecorado;

    public CarroDecorador(Carro carroDecorado) {
        this.carroDecorado = carroDecorado;
    }

    public double getCusto() {
        return carroDecorado.getCusto() + super.getCusto();
    }

    public String getDescricao() {
        return carroDecorado.getDescricao() + ", " + super.getDescricao();
    }
}
```

Padrão Decorator (Decorador)

```
//ConcreteDecorator
public class VidroEletrico extends CarroDecorador {
    public VidroEletrico(Carro carroDecorado) {
        super(carroDecorado);
        setCusto(600.00);
        setDescricao("Vidro Eletrico");
    }
}
//ConcreteDecorator
public class RodaLigaLeve extends CarroDecorador {
    public RodaLigaLeve(Carro carroDecorado) {
        super(carroDecorado);
        setCusto(200.00);
        setDescricao("Roda Liga Leve");
    }
}
//ConcreteDecorator
public class ArCondicionado extends CarroDecorador {
    public ArCondicionado(Carro carroDecorado) {
        super(carroDecorado);
        setCusto(900.00);
        setDescricao("Ar Condicionado");
    }
}
```

Padrão Decorator (Decorador)

```
public class Principal
{
    public static void main(){
        Carro carro = new Voyage();
        carro = new ArCondicionado(carro);
        carro = new VidroEletrico(carro);
        carro = new RodaLigaLeve(carro);

        //--cliente-----
        System.out.println(carro.getDescricao());
        System.out.println(carro.getCusto());
    }
}
```

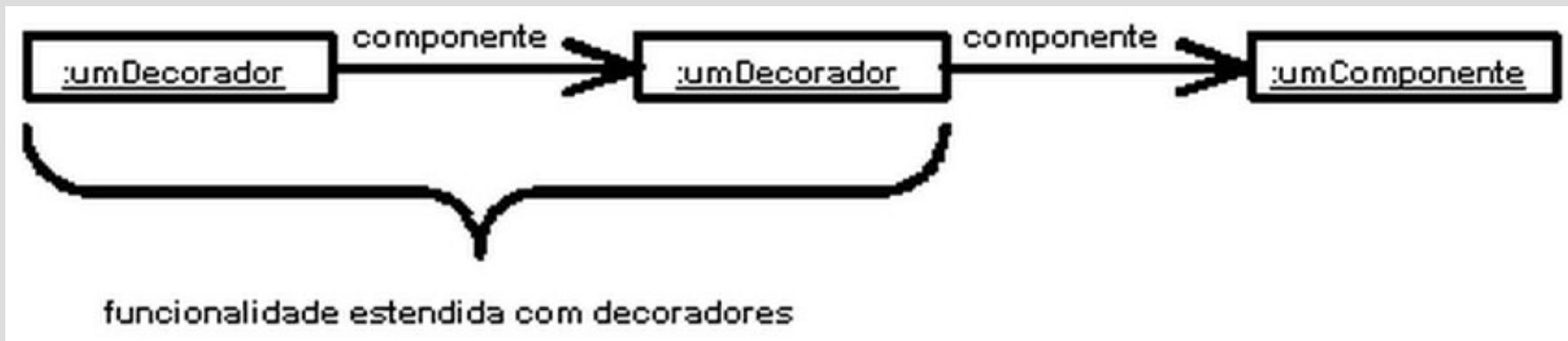
Padrão Decorator (Decorador)

testando...

```
Carro carro;  
carro = new Voyage();  
carro = new ArCondicionado(carro);  
carro = new VidroEletrico(carro);  
System.out.println(carro.getCosto());  
System.out.println(carro.getProduto());
```

Padrão Decorator (Decorador)

- Os decoradores estendem a funcionalidade do componente através da montagem de uma lista encadeada de decoradores e componente.



Padrão Decorator (Decorador)

Exemplo:

O Swing utiliza este padrão de projeto.

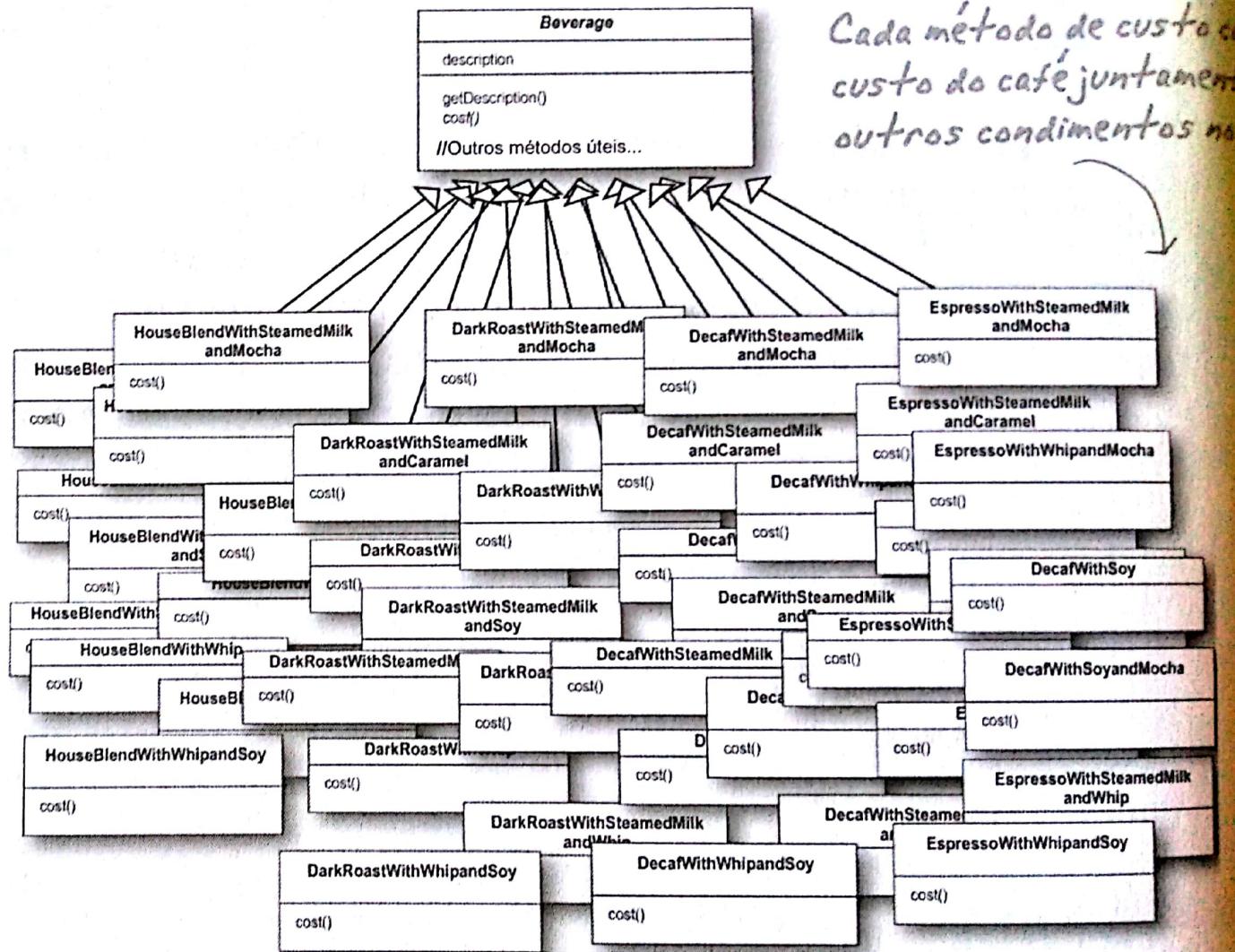
Por exemplo, criar uma caixa de texto com suporte a barras de rolagem, basta criar um JScrollPane (barra de rolagem) "em volta" de um JTextArea (caixa de texto), sub-classes de Component:

```
//a caixa de texto é o componente "decorado"  
JTextArea txt = new JTextArea();
```

```
//"decora" a caixa de texto com barras de rolagem  
JSScrollPane comp = new JSScrollPane(txt);
```

Exemplo

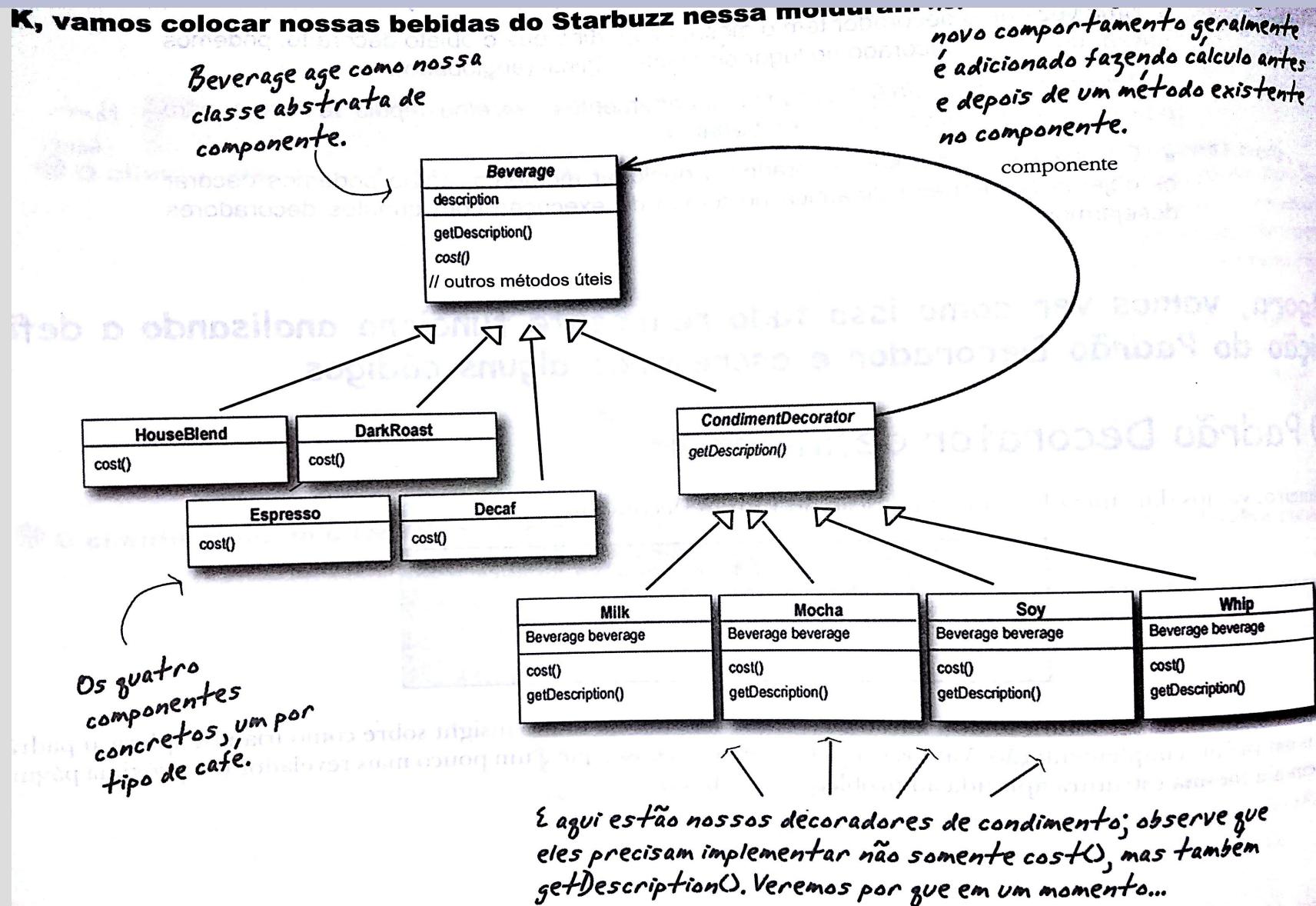
Esta é a primeira tentativa...



Freeman, Padrões de Projeto - Use a Cabeça.

Exemplo

Freeman, Padrões de Projeto - Use a Cabeça.



Padrão Decorator (Decorador)

- **Observações:**

- ◆ Os decoradores têm o mesmo supertipo que os objetos que eles decoram.
- ◆ Pode-se usar um ou mais decoradores em um objeto.
- ◆ Pode-se passar um objeto decorado no lugar de um objeto original, pois o supertipo é o mesmo.
- ◆ O decorador adiciona seu próprio comportamento antes ou depois do comportamento do objeto que decora.
- ◆ Os objetos originais podem ser decorados a qualquer momento em tempo de execução.

Padrão Decorator (Decorador)

Conclusão:

Permite, dinamicamente, agregar responsabilidades adicionais a objetos. Os Decorators fornecem uma alternativa flexível ao uso de subclasses para extensão de funcionalidades.

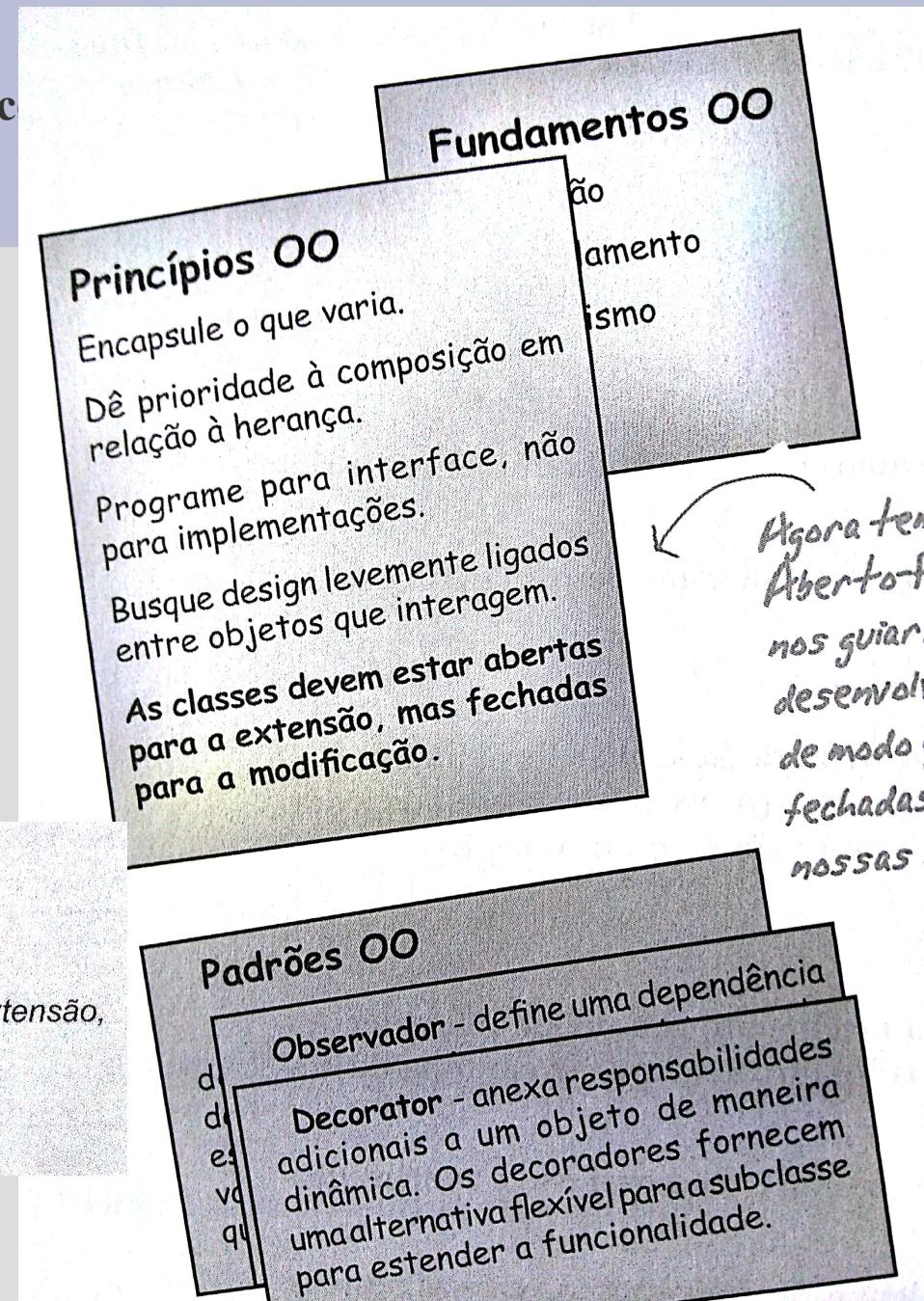
Permite a extensão fácil da aplicação, na medida em que novos decoradores podem ser adicionados sem modificação do código existente.



Princípio de projeto

As classes devem estar abertas para extensão, mas fechadas para modificação.

Padrão Dec



Freeman, Padrões de Projeto - Use a Cabeça.

Padrão Decorator (Decorador)

- Fim.