AULA 6 – Padrões de Projeto

Prof. Igor Justino Rodrigues
13/11/2024

INSTITUTO FEDERAL
Goiano
Campus Ceres

OBJETIVOS DA AULA

- Entender o conceito e a utilidade do padrão Decorator.
- Aprender a adicionar funcionalidades a objetos de forma dinâmica.
- Explorar exemplos práticos onde o Decorator é vantajoso.

- 13/11/2024

Introdução ao Padrão Decorator

O que é? Um padrão estrutural que permite adicionar comportamentos a objetos de forma flexível.

Quando usar?

•Quando queremos evitar subclasses para cada combinação de funcionalidades.

•Para adicionar recursos a objetos individualmente.

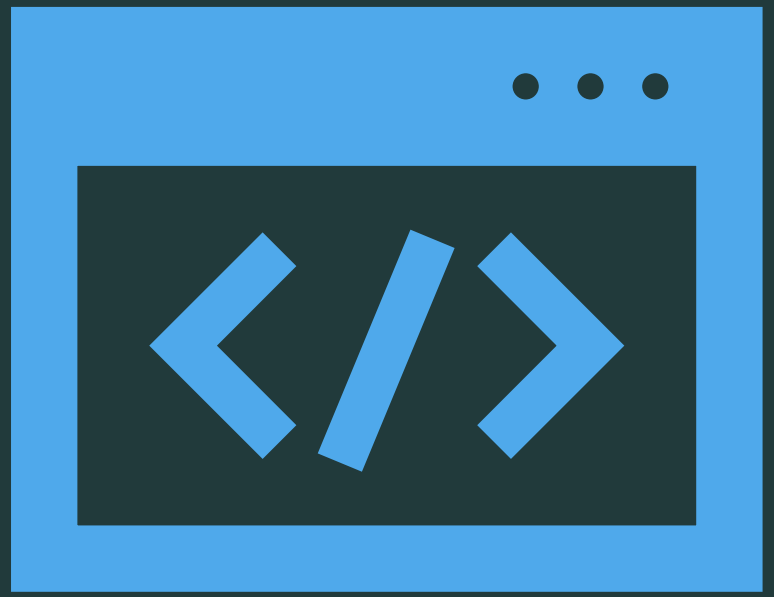
Diagrama Básico do Decorator

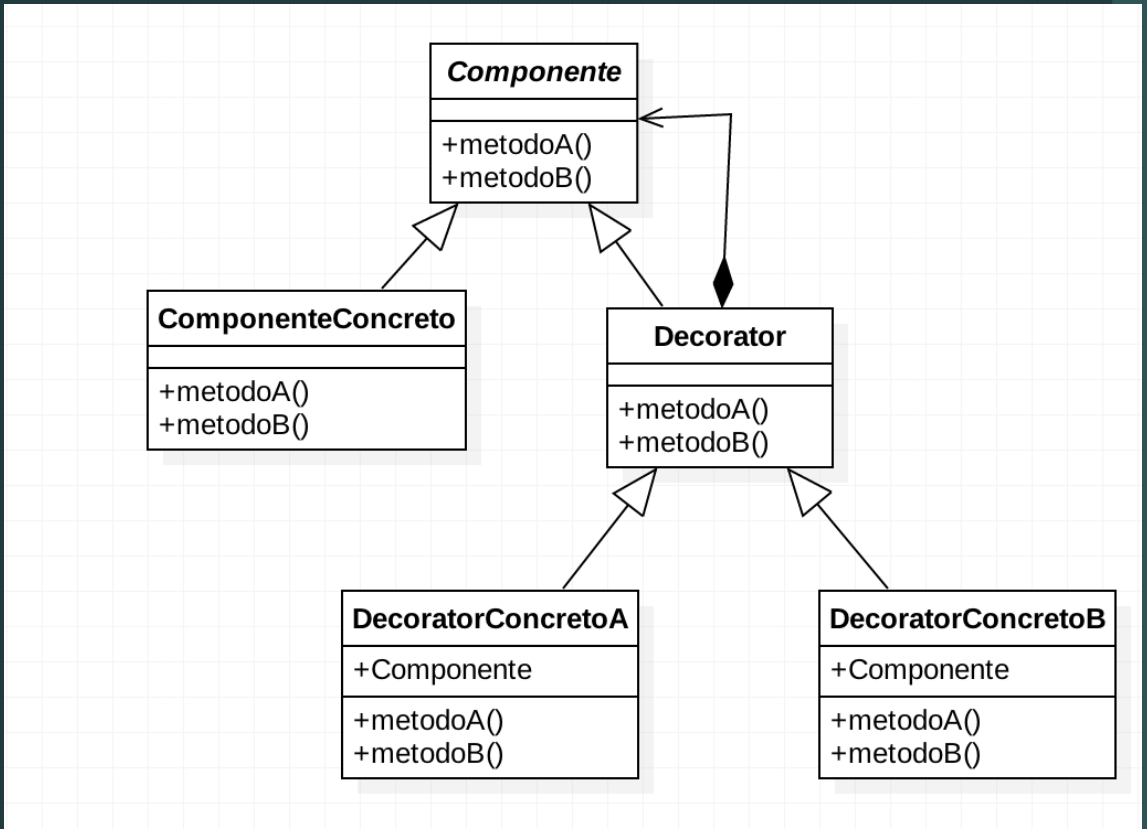
Componente: Interface ou classe abstrata.

Componente Concreto: Implementação básica do componente.

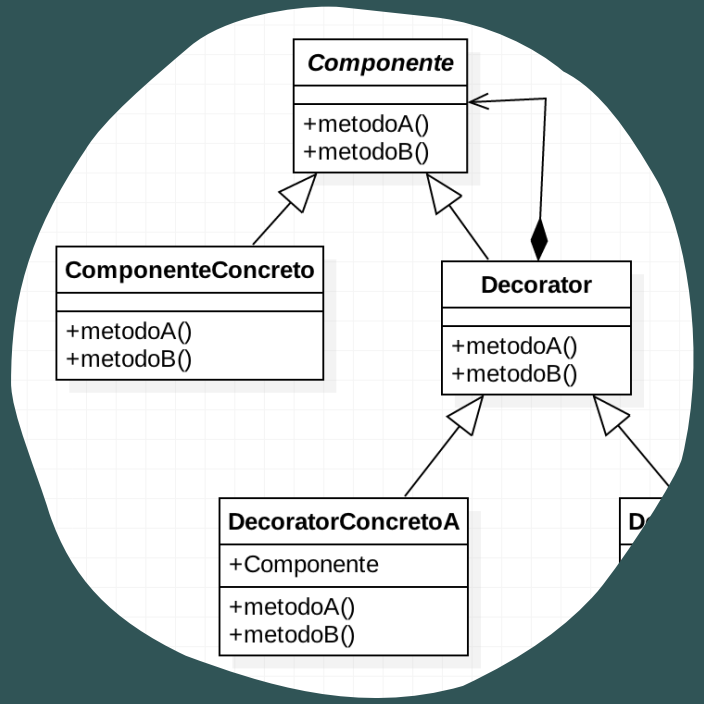
Decorator: Classe abstrata que contém um componente e adiciona funcionalidade.

Decoradores Concretos: Adicionam comportamentos específicos.



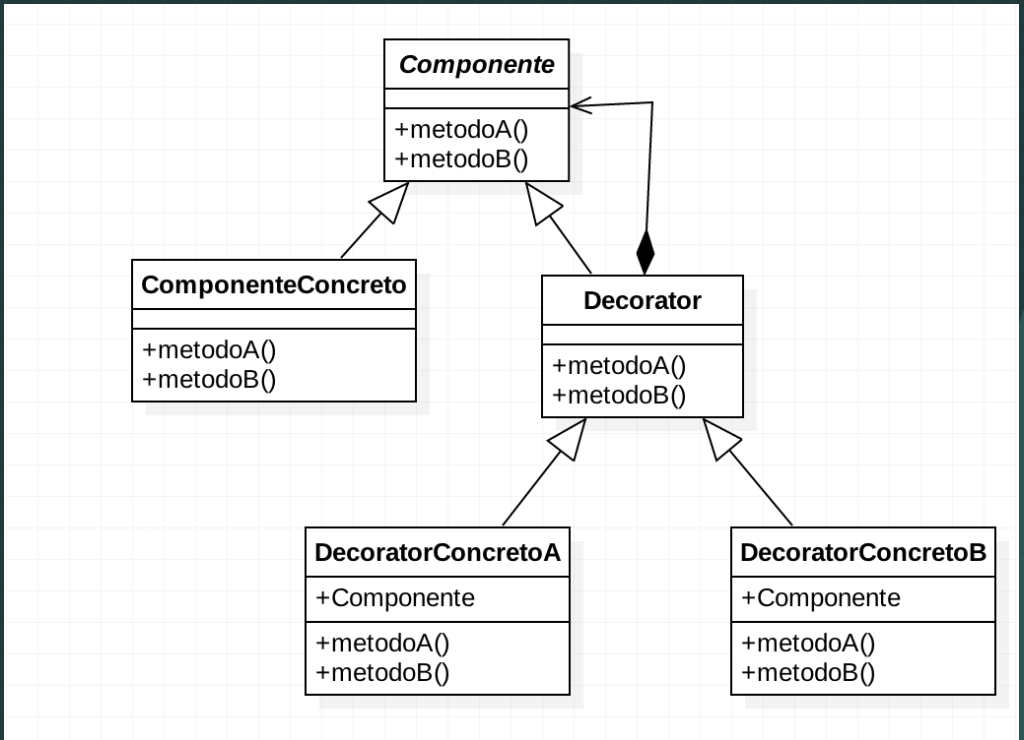


- **Componente**:
- Esta é uma interface ou classe abstrata que define o contrato comum para todos os objetos que podem ter funcionalidades adicionais.
- No diagrama, o **Componente** possui dois métodos: metodoA() e metodoB().
- Todos os elementos do diagrama devem implementar ou estender essa interface/classe para garantir que possuem os mesmos métodos.



ComponenteConcreto:

- Esta classe representa o objeto principal que será "decorado" com funcionalidades adicionais.
- **ComponenteConcreto** implementa ou herda de **Componente** e fornece a implementação padrão dos métodos metodoA() e metodoB().
- É o objeto base que, sem decorações, executa as operações básicas.



Decorator:

- A classe **Decorator** é uma classe abstrata que também implementa a interface **Componente**.
- Ela contém uma referência a um objeto **Componente** que será "envolvido" (ou seja, decorado).
- Assim, o **Decorator** pode receber qualquer instância de **Componente** (seja a implementação concreta ou outros decoradores), e adicionar ou modificar funcionalidades.
- O Decorator delega as operações para o **Componente** interno e pode adicionar comportamentos adicionais antes ou depois da chamada dos métodos.

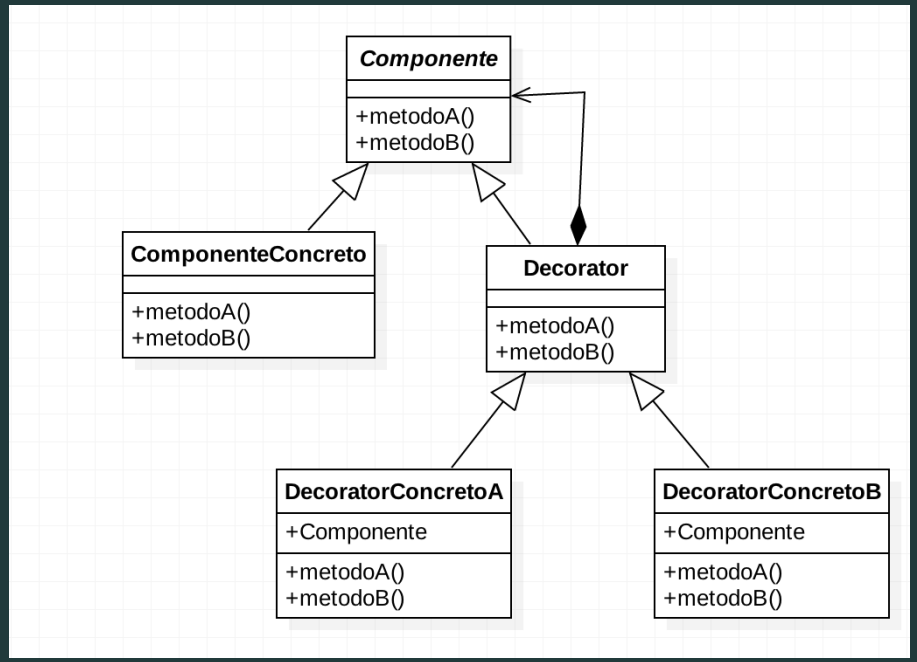
DecoratorConcretoA e DecoratorConcretoB:

•Essas são classes concretas que estendem o **Decorator**.

•Elas representam os "decoradores" reais que adicionam ou modificam funcionalidades do **Componente**.

•Cada decorador concreto (DecoratorConcretoA e DecoratorConcretoB) possui a implementação de metodoA() e metodoB(), e pode fazer algo antes ou depois de delegar a chamada ao **Componente** "envolvido".

•Por exemplo, DecoratorConcretoA pode adicionar uma funcionalidade extra em metodoA() enquanto mantém o comportamento básico de Componente.



Como o Padrão Funciona no Diagrama

O **ComponenteConcreto** é o objeto principal que realiza as operações básicas.

O **Decorator** e seus decoradores concretos (DecoratorConcretoA e DecoratorConcretoB) são usados para "envolver" o **ComponenteConcreto**, permitindo que novas funcionalidades sejam adicionadas.

Os decoradores podem ser combinados de forma flexível. Por exemplo, DecoratorConcretoA pode envolver um ComponenteConcreto, e DecoratorConcretoB pode envolver DecoratorConcretoA, criando uma cadeia de decorações.

Esse design permite que os objetos sejam compostos de forma dinâmica, adicionando ou alterando funcionalidades sem modificar diretamente a classe base (ComponenteConcreto), promovendo flexibilidade e modularidade no código.

Exemplo do Mundo Real

- **Exemplo**: Café com ingredientes
  - Café básico: $5
  - Adição de leite: +$1.5
  - Adição de chocolate: +$2.0
  - Adição de canela: +$0.75

- Como usar o Decorator para combinar diferentes ingredientes?

Código de Exemplo
Interface Cafe

```php
interface Cafe {
    public function getDescricao();
    public function getCusto();
}
```

Código de Exemplo
Classe CafeBasico

```php
class CafeBasico implements Cafe {
    public function getDescricao() { return "Café básico"; }
    public function getCusto() { return 5.0; }
}
```

Implementação de Decoradores
Decorator Abstrato CafeDecorator

```php
abstract class CafeDecorator implements Cafe {
    protected $cafe;
    public function __construct(Cafe $cafe) { $this->cafe = $cafe; }
}
```

Implementação de Decoradores
Decorador Concreto Leite

```php
class Leite extends CafeDecorator {
    public function getDescricao() { return $this->cafe->getDescricao() . ", Leite"; }
    public function getCusto() { return $this->cafe->getCusto() + 1.5; }
}
```

Código que combina vários ingredientes:

```php
$cafe = new CafeBasico();
$cafeComLeite = new Leite($cafe);
$cafeCompleto = new Canela(new Chocolate($cafeComLeite));
echo $cafeCompleto->getDescricao();
```