INSTITUTO FEDERAL
Goiano

Padrões de Projeto: Decorator

Equipe

Giovani Paganini

Lucas Rezende

INSTITUTO FEDERAL
Goiano

Decorator

O Padrão Decorator complementa com responsabilidades adicionais um objeto, fazendo com que os decoradores trabalhem como alternativas de subclasses estendendo funcionalidades. Isso evita o uso de uma vasta hierarquia de subclasses.

INSTITUTO FEDERAL
Goiano

Características

- Adicionar funcionalidades a objetos em tempo de execução.
- Possui flexibilidade e permite aplicar somente as funcionalidades necessárias a um determinado objeto, evitando sobrecarregar uma classe.
- O decorador possui o mesmo supertipo que os objetos decorados.
- É possível usar mais de um decorador para englobar um objeto.
- É possível passar um objeto decorado no lugar do objeto original.
- Adiciona seu próprio comportamento antes e/ou depois de delegar o objeto a atividade.

INSTITUTO FEDERAL
Goiano

Problema:

- Uma casa especializada em massas em que existem diversos tipos de macarrões e ingredientes.
- Um prato é formado da combinação de um tipo de massa e de ingredientes que o complementam.

Macarrões:

- Spaghetti;
- Penne;
- Talharim;
- Farfalle;
- Campanelle;
- Bucatini;

Ingredientes adicionais

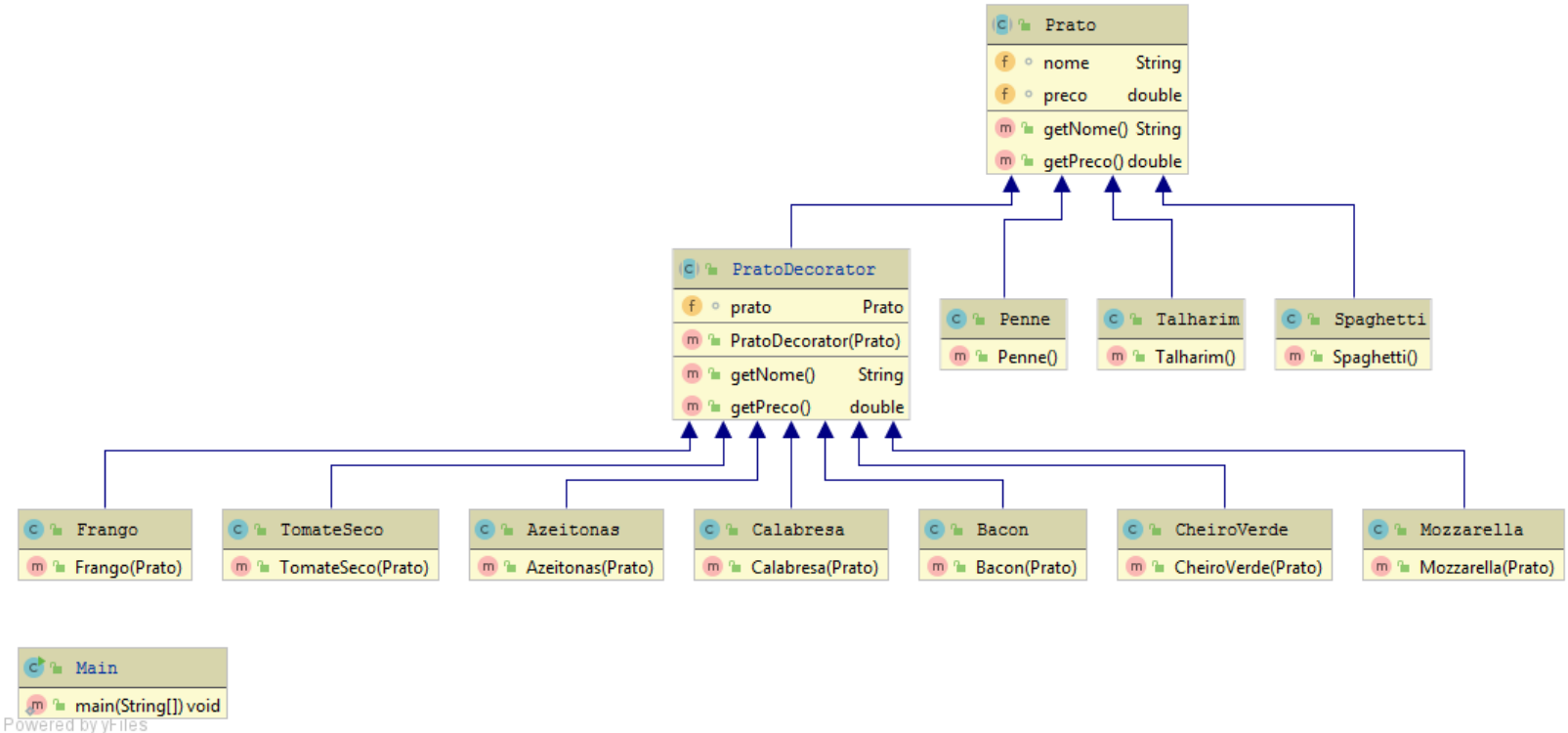
- Calabresa;
- Tomate seco;
- Mozzarella;
- Bacon;
- Frango;
- Cheiro verde;

Possíveis pratos:

- Penne + Calabresa + Cheiro verde;
- Spaghetti + Bacon + Tomates secos;
- Talharim + Frango + Mozzarella;
- etc.

INSTITUTO FEDERAL
Goiano

UML



INSTITUTO FEDERAL
Goiano

Exemplos

- Uma classe abstrata “Prato” genérica que serve como base às outras classes:

```
public abstract class Prato {
    String nome;
    double preco;

    public String getNome() { return nome; }

    public double getPreco() { return preco; }
}
```

INSTITUTO FEDERAL
Goiano

Todos os objetos possuem o tipo “Prato”, definindo o que todos os objetos possuem, por exemplo:

```
public class Penne extends Prato {
    public Penne(){
        nome = "Penne";
        preco = 20.0;
    }
}
```

INSTITUTO FEDERAL
Goiano

- Todas as classes de Pratos diferentes possuirão a mesma estrutura, apenas irão definir seus atributos. A classe Decorator que é abstrata, define que todos os decoradores devem ter um objeto “Prato”, que decoram, e um método que é aplicado ao objeto.

INSTITUTO FEDERAL
Goiano

```
public abstract class PratoDecorator extends Prato {
    Prato prato;

    public PratoDecorator(Prato umPrato) { prato = umPrato; }

    @Override
    public String getNome() { return prato.getNome() + " + " + nome; }

    public double getPreco() { return prato.getPreco() + preco; }
}
```

INSTITUTO FEDERAL
Goiano

- Como um decorador também é um Prato, ele herda os atributos nome e preço.
- Nas classes concretas são definidos os modificadores que serão aplicados as classes de pratos.
- Por exemplo, aplicando isso em um ingrediente “Calabresa”

INSTITUTO FEDERAL
Goiano

```java
public class Calabresa extends PratoDecorator {
    public Calabresa(Prato umPrato) {
        super(umPrato);
        nome = "Calabresa";
        preco = 1.50;
    }
}
```

INSTITUTO FEDERAL
Goiano

- No construtor do decorador é preciso passar um objeto Prato qualquer, podendo ser tanto um ingrediente quanto outro decorador.
- Esse é um conceito fundamental do Decorator.
- Desta forma, se quisermos montar um prato com um tipo de macarrão e diversos ingredientes diferentes, teremos o seguinte resultado:

INSTITUTO FEDERAL
Goiano

```
public class Main {
    public static void main(String[] args) {

        Prato meuPrato = new Penne();
        System.out.println(meuPrato.getNome() + " = "
                + meuPrato.getPreco());

        meuPrato = new Calabresa(meuPrato);
        System.out.println(meuPrato.getNome() + " = "
                + meuPrato.getPreco());
    }
}
```

INSTITUTO FEDERAL
Goiano

Note que, o prato final varia conforme o decorador que é aplicado. Quando os métodos getNome() e getPreco() são chamados, o primeiro método chamado é o método do último decorador a ser aplicado.

O decorador então chama a classe mãe e então chama o método do Prato ao qual ele está decorando.

Se fosse outro decorador, a solicitação percorre até chegar em um prato para responder a requisição, sem repassar a nenhum objeto.

INSTITUTO FEDERAL
Goiano

```
Penne = 20.0
Penne + Calabresa = 21.5

Process finished with exit code 0
```

INSTITUTO FEDERAL
Goiano

Desvantagens

- Não é possível verificar se um objeto possui um decorador específico. Ex: se o Prato possui um decorador chamado Molho.
- Se um decorador “Mozzarela” implementa um método “derreter()”, não é possível afirmar que um prato qualquer também possua esse método.
- Não é possível verificar o tipo do objeto depois de aplicado um determinado decorador, pois este modifica o tipo do mesmo.
- É necessário criar um objeto para cada novo decorador.