Manutenção de software

Wagner rodrigo da silva

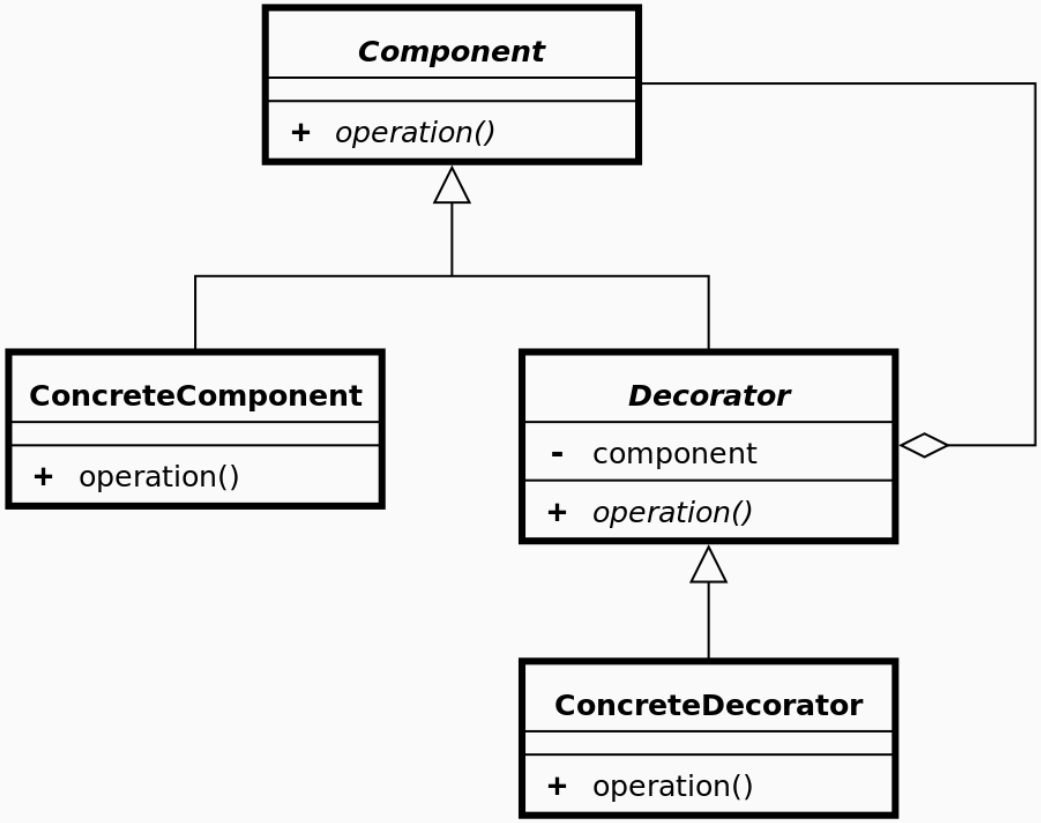
Decorator

O Padrão Decorator anexa responsabilidades adicionais a um objeto dinamicamente. Os decoradores fornecem uma alternativa flexível de subclasse para estender a funcionalidade.

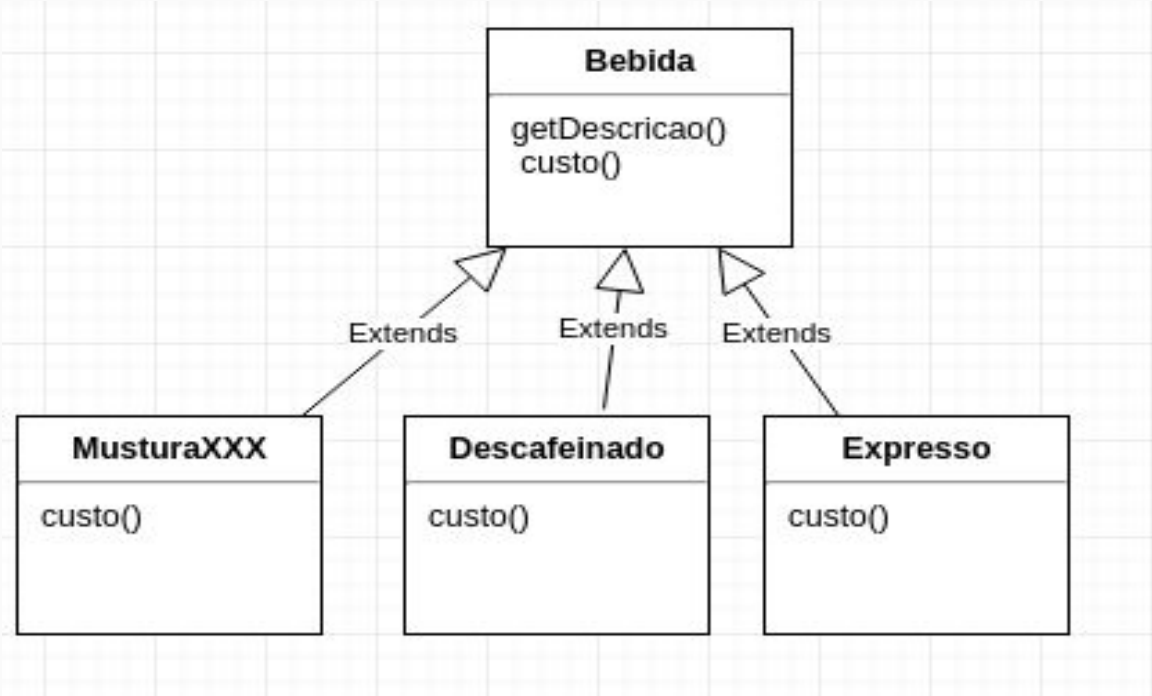
Características do Padrão Decorator

- Os decoradores têm o mesmo supertipo que os objetos que eles decoram;
- Você pode usar um ou mais decoradores para englobar um objeto;
- Uma vez que o decorador tem o mesmo supertipo que o objeto decorado, podemos passar um objeto decorado no lugar do objeto original (englobado);
- O decorador adiciona seu próprio comportamento antes e/ou depois de delegar o objeto que ele decora o resto do trabalho;
- Os objetos podem ser decorados a qualquer momento, então podemos decorar os objetos de maneira dinâmica no tempo de execução com quantos decoradores desejarmos

Diagrama Decorator



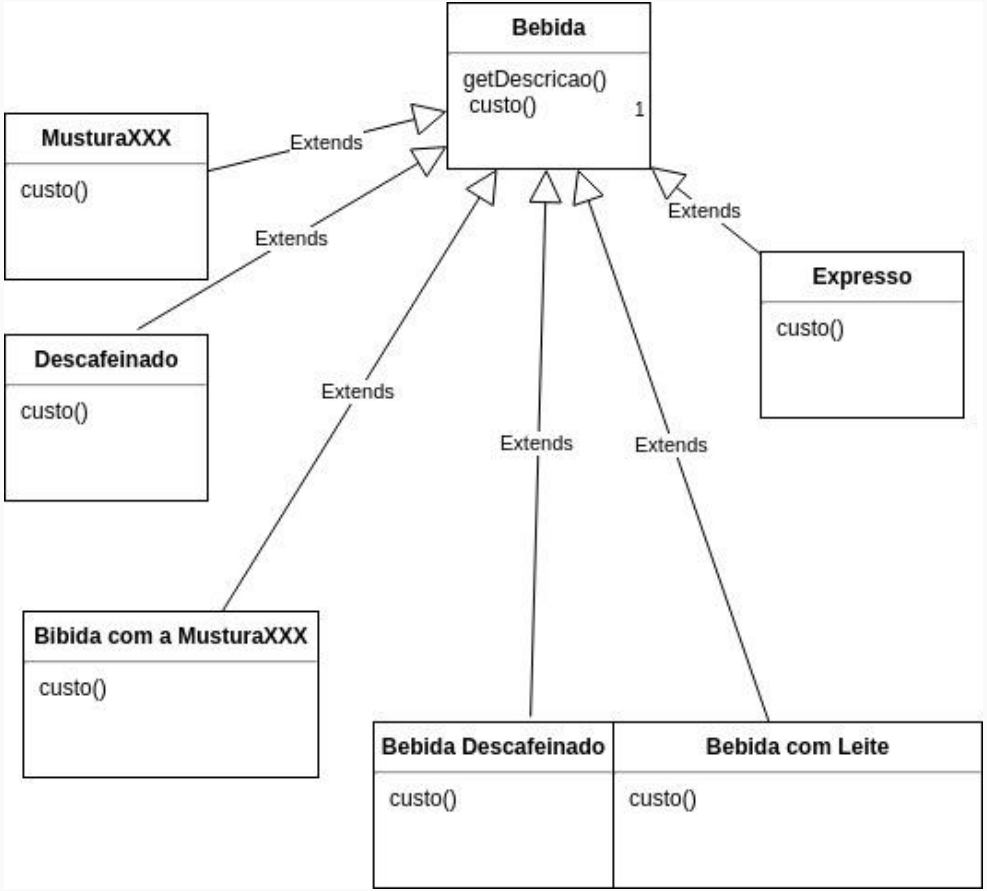
Sem o decorator


Bebida
getDescricao()
custo()
Extends
Extends
Extends
MusturaXXX
custo()
Descafeinado
custo()
Expresso
custo()

Adicionar novos sabores

Aplicação básica está preparada receber esse novos tipos de ingredientes?

bem ... mas funciona ...



Código

```
abstract public class Bebida {
    protected String descricao;
    protected ArrayList<Condimento> condimentos;

    protected Bebida(){
        descricao = "";
        condimentos = new ArrayList<Condimento>();
    }
    public String getDescricao(){
        return descricao;
    }
    public void addCondimento( Condimento condimento ){
        condimentos.add( condimento );
    }
    public double getPreco(){
        double preco = 0;
        for( Condimento c : condimentos ){
            preco += c.getPreco();
        }
        return preco;
```

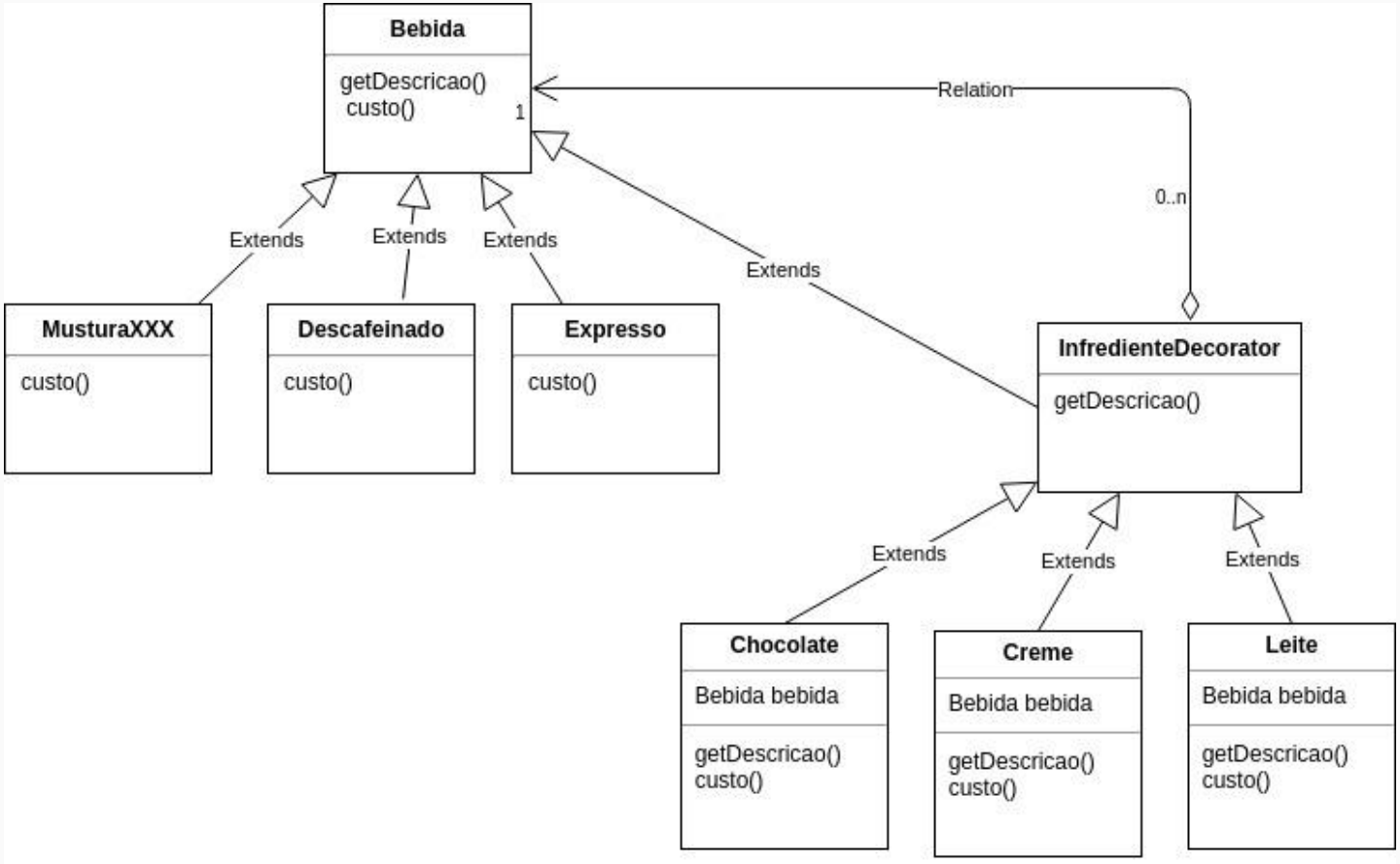
Pontos a ser destacados

Herança nem sempre leva a designs flexíveis e fáceis de manter

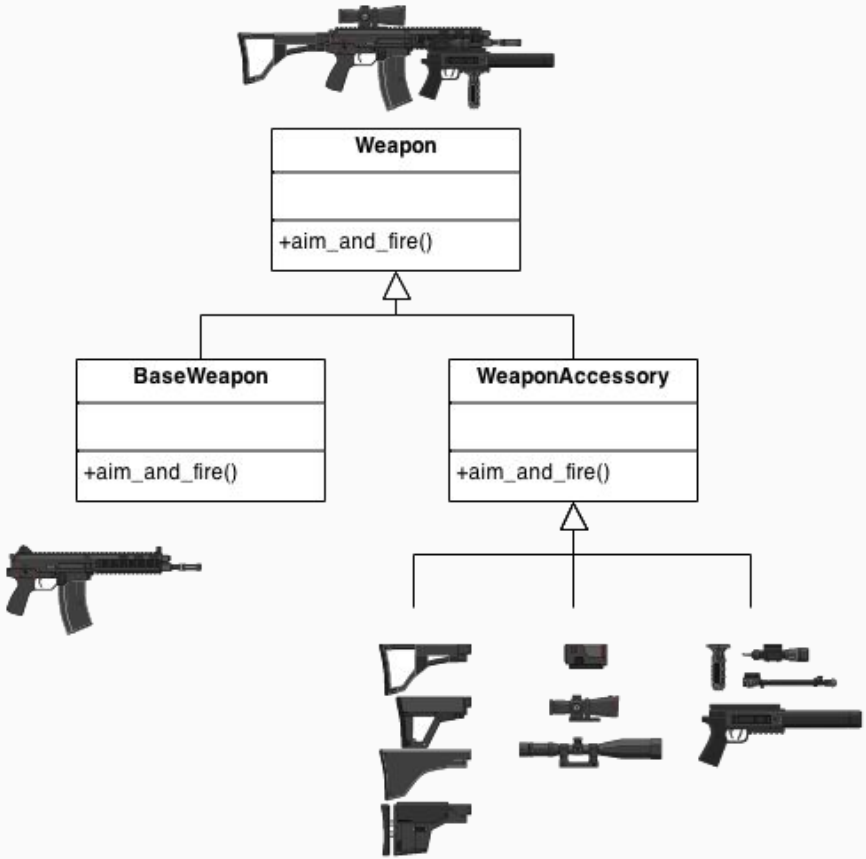
Existe outras formas de utilizar o comportamento de herança.

Criando objetos de forma dinâmica é possível adicionar novas funcionalidades através da criação de um código novo, **ao invés de alterar o já existente.**

Diagrama final



Um outro exemplo



Use o padrão Decorator quando:

- Acrescentar responsabilidades a objetos individuais de forma dinâmica e transparente (sem afetar outros objetos)
- Para responsabilidades que podem ser removidas
- Quando a extensão através de subclasses não é prática

Maior flexibilidade do que a herança estática

- Responsabilidades removidas e acrescentadas em tempo de execução através de associação e dissociação
- Enquanto o mecanismo da herança cria uma nova subclasses para cada funcionalidade
- Decorators permitem que propriedades sejam adicionadas 2 ou mais vezes

Evita classes sobrecarregadas de características na parte superior da hierarquia

Use quando for necessário

NetBeans.

Referências

https://sourcemaking.com/design_patterns/decorator

https://www.ateomomento.com.br/acoplamento-e-coesao/

https://www.thiengo.com.br/padrao-de-projeto-decorator-decorador

https://www.devmedia.com.br/padrao-de-projeto-decorator-em-java/26238

https://www.tutorialspoint.com/design_pattern/decorator_pattern.htm

Obrigado