



Polimorfismo



**Certified
Developer**
The Ultimate Tech Degree

DigitalHouse >
Coding School



Temas

1

Ligação
dinâmica

2

Polimorfismo

3

Casting





Antes de ver o que é polimorfismo,
devemos entender o que é **ligação
dinâmica.**



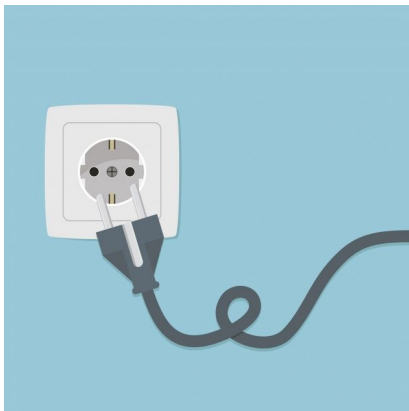
1

**Ligação dinâmica
(dynamic binding)**



Ligação dinâmica (dynamic binding)

A vinculação dinâmica de uma referência funciona como um plugue. Em uma tomada você pode conectar coisas diferentes: uma TV, uma geladeira, um notebook.



Veremos que em uma referência podemos apontar para diferentes tipos de objetos.



Aqui, tanto a **referência** quanto o **objeto** referenciado são do mesmo tipo: Doberman. No entanto, a referência e o objeto referenciado podem ser de tipos diferentes.

Objeto

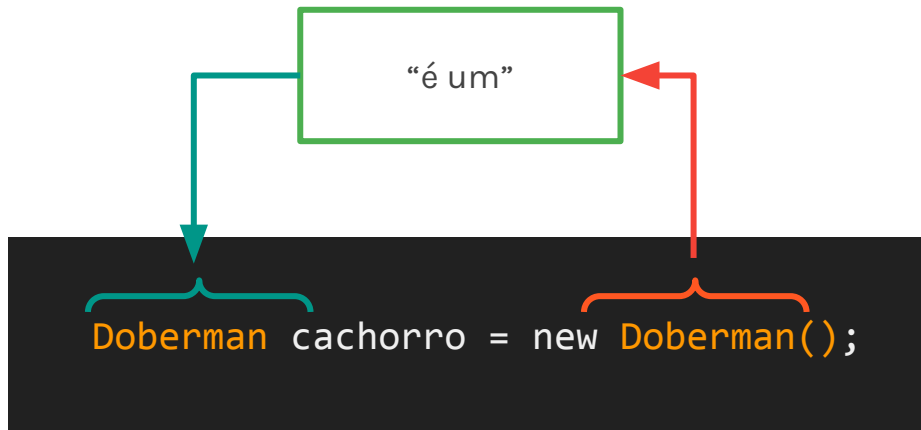
```
Doberman cachorro = new Doberman();
```

Referência/Variável



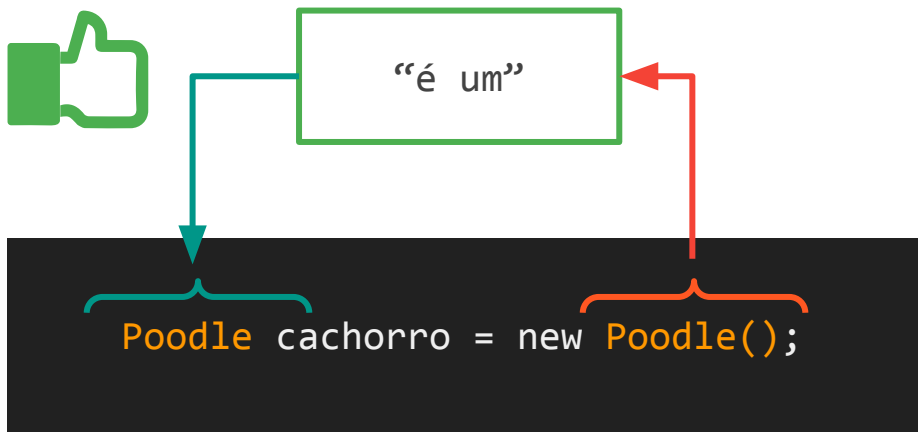


Em linguagens que não são tipadas, a **referência** e o **objeto** podem ser de qualquer tipo, em linguagens fortemente tipadas como Java, o objeto deve ser de uma classe que tenha um relacionamento do tipo "**é um**" em relação ao referência.



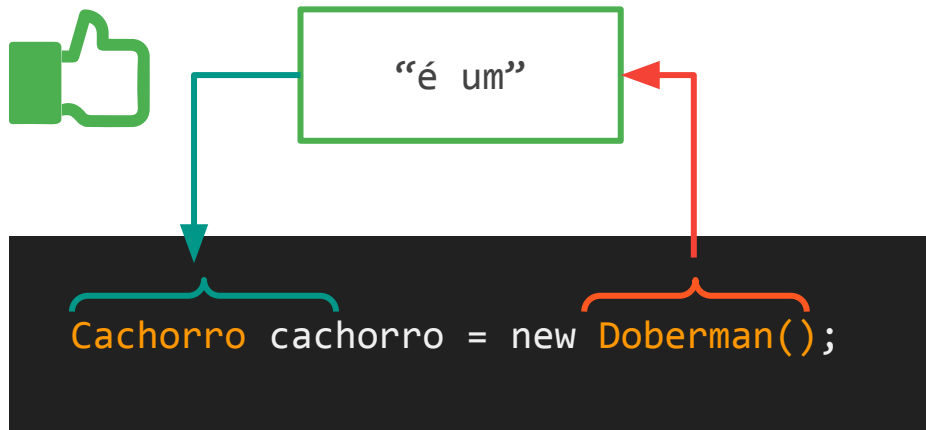


Veamos o seguinte exemplo:



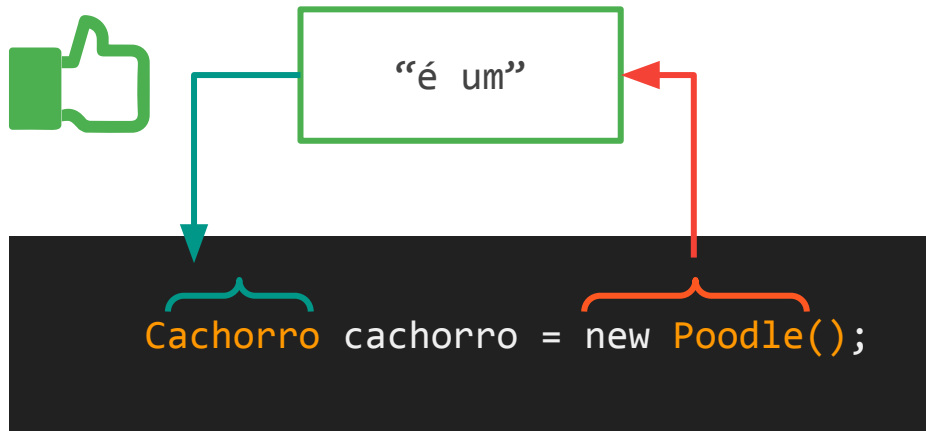


Vemos que, neste caso, a **referência** tem um tipo diferente do **objeto** referenciado, mas atende à condição "é um".



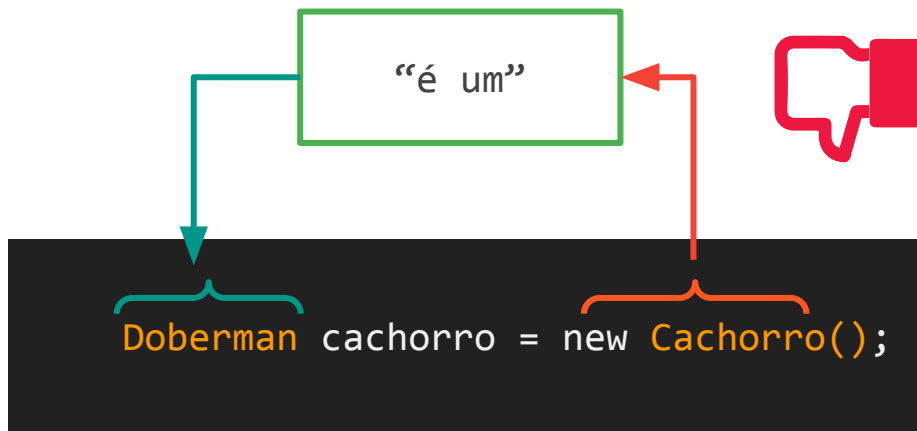


Vemos que, neste caso, a **referência** também tem um tipo diferente do **objeto** referenciado, mas atende à condição "é um".



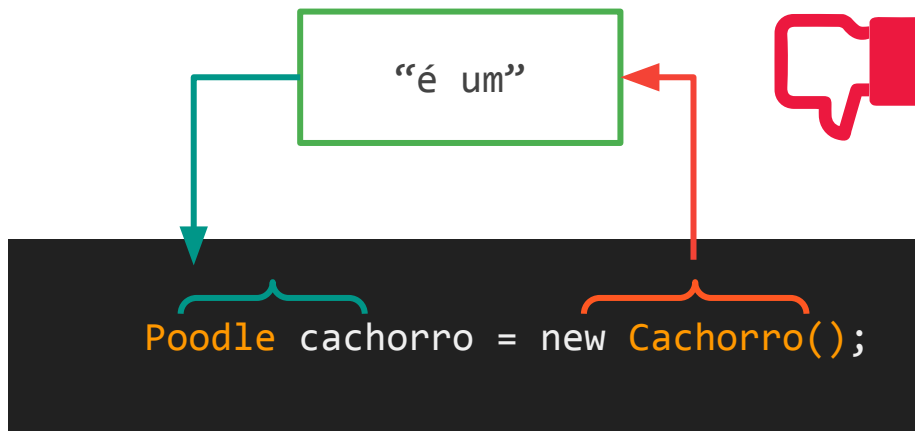


Um cachorro não é necessariamente sempre um Doberman. Isso não é permitido.



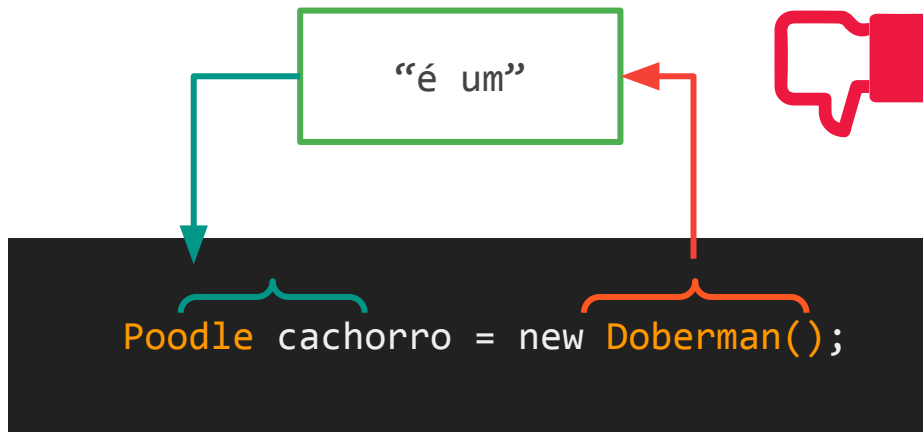


Da mesma forma, um cão não é necessariamente sempre um Poodle. Isso não é permitido.



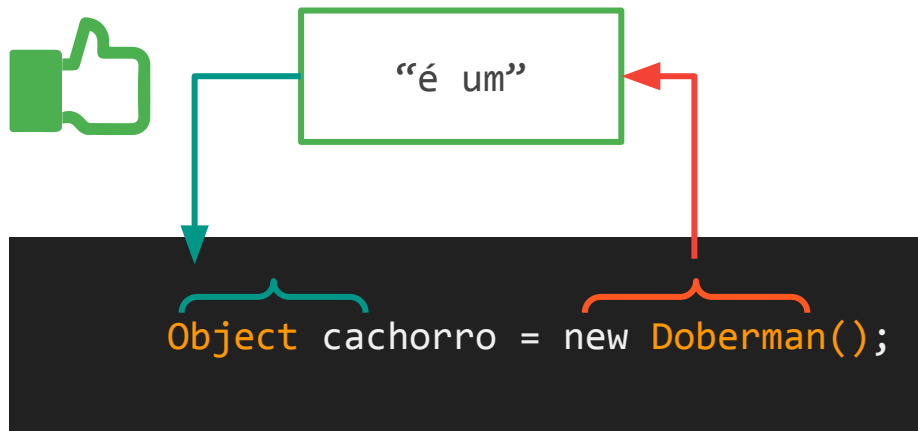


Um Doberman pode ter coisas semelhantes a um Poodle, mas não é.





Em Java, todas as classes, por definição, herdam de Object, portanto, um Doberman é um Object.



2 | Polimorfismo



Polimorfismo

É a capacidade do mesmo objeto de se comportar como outro. Em outras palavras, é a capacidade de um objeto funcionar de várias maneiras.

Vamos ver com os exemplos anteriores:

```
Cachorro c;  
  
c = new Doberman();  
c.latir();  
  
c = new Poodle();  
c.latir();
```

A referência c pode se comportar e **latir** como um **Doberman**.

E a mesma referência c ao vincular dinamicamente a um **Poodle**.

Se comporta de maneira diferente e **late** como um **Poodle**.



Se usarmos o polimorfismo, podemos ter certeza de que modificações futuras, que adicionam novas subclasses, não afetarão o código ou seu funcionamento.



Se o seu código usa cachorro(ou seja, qualquer objeto que "É um" cachorro) desde que as novas raças de cachorros introduzidas no sistema herdadas do Cachorro funcionem corretamente.



3 | Casting



Casting

Suponha que **Doberman** tenha um método chamado `morderComoDoberman()`, mas a referência ou variável é do tipo `Cachorro`. Para forçar um cachorro a ser um Doberman, usamos o casting. Dessa forma, podemos invocar os próprios métodos do Doberman.

```
Cachorro cachorro = new Doberman();  
cachorro.latir();  
  
((Doberman)cachorro).morderComoDoberman()  
                            
      Casting
```





O mesmo acontece se nosso objeto de referência for do tipo Object. Nesse caso, como a classe Object não possui o método latir(), devemos convertê-lo para Cachorro que possui o referido método ou para Doberman.

```
Object cachorro = new Doberman();  
((Cachorro)cachorro).latir()  
  
((Doberman)cachorro).morderComoDoberman()  
  
Casting
```



DigitalHouse>
Coding School