Programação Orientada a Objetos

Aulas nº X: SobreEscrita e SobreCarga

Prof. Luis Gustavo Araujo

2018

Objetivo

Compreender os conceitos e aplicações de Sobreescrita e Sobrecarga.

POO

Sobreescrita

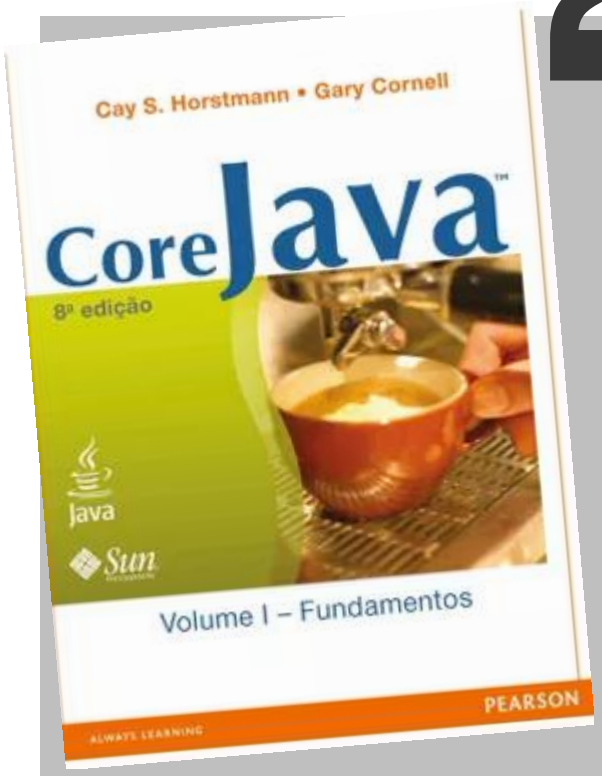
Sobrecarga

Sobrecarga e Sobrescita

São conceitos relacionados aos métodos. Sobreescita relaciona-se como *Herança* e *Interface*. Sobrecarga relaciona-se com *Classes*, de modo geral.

Sobrecarga

Sobrecarga



*Essa capacidade é chamada de **Sobrecarga** ou **Clonagem**, Sobrecarregamento ocorre se vários métodos tiverem o mesmo nome [...] e parâmetros diferentes.*

Hortmann & Cornell

Core Java

Sobrecarga

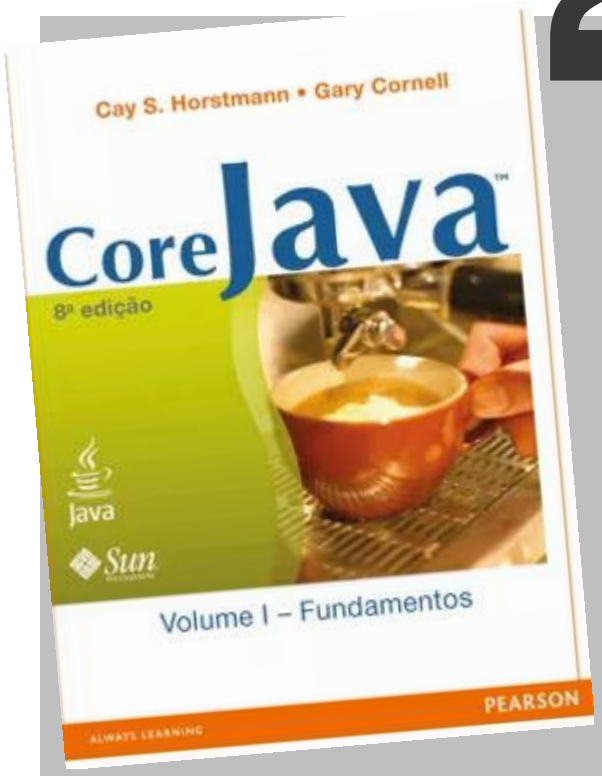
```
1.    public double getSalario(double adicional){
2.      return this.salario + adicional;
3.    }
4.
5.
6.    public double getSalario(){
7.      return this.salario;
8.    }
9.  }
```

Sobrecarga

```
1.    public double getSalario(double adicional){
2.      return this.salario + adicional;
3.    }
4.
5.
6.    public double getSalario(){
7.      return this.salario;
8.    }
9.  }
```

Como diferenciar, se eles têm o mesmo nome?

Sobrecarga



O compilador precisa classificar qual método chamar. Ele seleciona o método correto correspondendo os tipos de parâmetro nos cabeçalhos dos vários métodos com os tipos dos valores utilizados na chamada de método específica.

Hortmann & Cornell

Core Java

Sobrecarga

```
1.    public double getSalario(double adicional){
2.      return this.salario + adicional;
3.    }
4.
5.
6.    public double getSalario(){
7.      return this.salario;
8.    }
9.  }
```

Há.... !

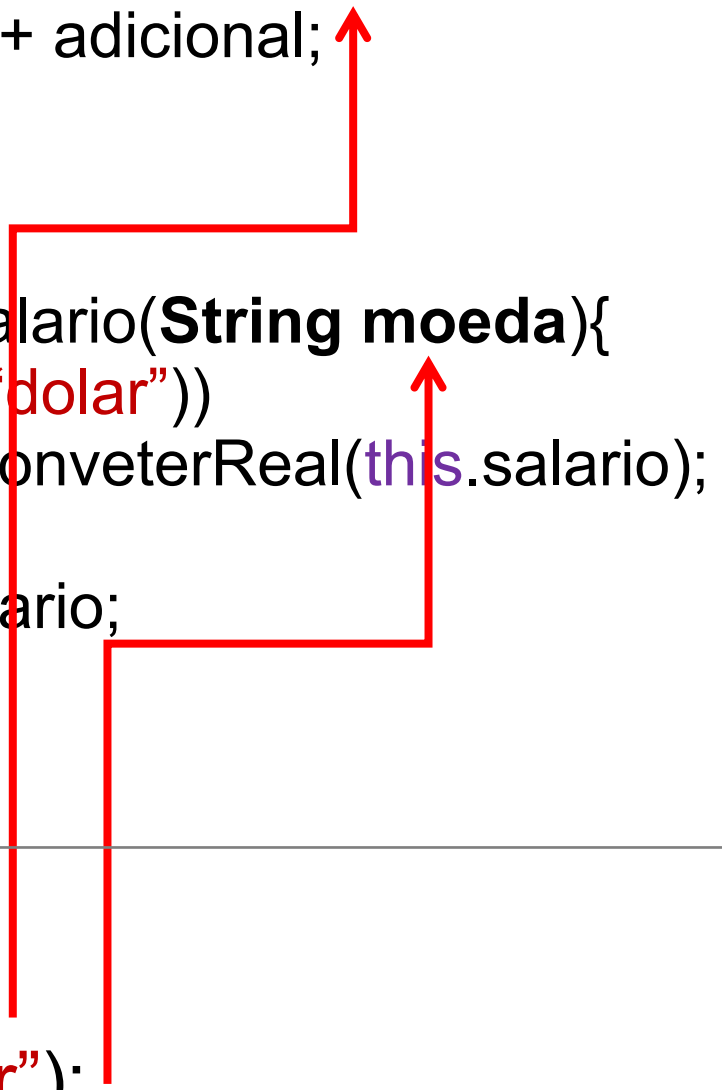
Sobrecarga

```
1.     public double getSalario(double adicional){
2.       return this.salario + adicional;
3.     }
4.
5.
6.     public double getSalario(String moeda){
7.       if( moeda.equals("dolar"))
8.             return Dolar.conveterReal(this.salario);
9.       else
10.            return this.salario;
11.    }
12. }
```

```
1. f.getSalario(2.0);
2. f.getSalario("dolar");
```

Sobrecarga

```
1.    public double getSalario(double adicional){
2.      return this.salario + adicional;
3.    }
4.
5.
6.    public double getSalario(String moeda){
7.      if( moeda.equals("dolar"))
8.            return Dolar.conveterReal(this.salario);
9.      else
10.           return this.salario;
11.   }
12. }
```



```
1. f.getSalario(2.0);
2. f.getSalario("dolar");
```

Sobrecarga (Erro)

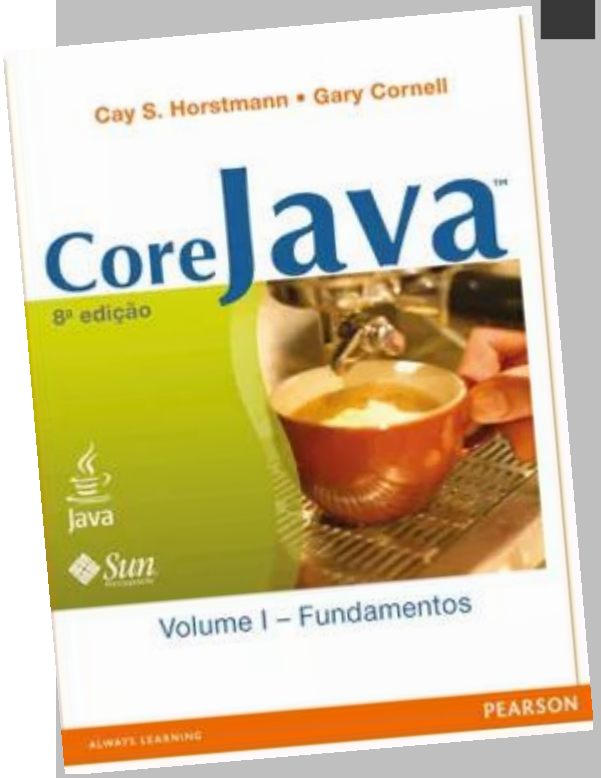
```
1.   public double getSalario(String bonus){
2.      if( bonus.equals("sim"))
3.          return this.salario + this.bonus;
4.      else
5.          return this.salario + this.bonus;
6.    }
7.
8.
9.    public double getSalario(String moeda){
10.     if( moeda.equals("dolar"))
11.           return Dolar.conveterReal(this.salario);
12.     else
13.           return this.salario;
14.   }
15.}
```

Sobrecarga (*vale para Construtores*)

```
1.  public class Funcionario{
2.     String nome;
3.     double salario;
4.
5.     public Funcionario(String nome, double salario){
6.        this.nome = nome;
7.        this.salario = salario;
8.     }
9.
10.    public Funcionario(String nome){
11.       this.nome = nome;
12.    }
13. }
```

```
1. Funcionario f1 = new Funcionario("Luis", 1000);
2. Funcionario f2 = new Funcionario("Luis");
```

Sobrecarga em Construtores



Um construtor padrão é um construtor sem parâmetro. [..] Se você escrever uma classe sem nenhum construtor, um construtor será então fornecido por padrão para você. [..] Se uma classe fornece pelo menos um construtor e não fornece um construtor padrão, é inválido criar objetos sem parâmetros de construção.

Hortmann & Cornell

Core Java

Sobrecarga em Construtores

```
1.    public class Funcionario{
2.       String nome;
3.       double salario;
4.
5.       public Funcionario(String nome, double salario){
6.          this.nome = nome;
7.          this.salario = salario;
8.       }
9.
10.      public Funcionario(String nome){
11.         this.nome = nome;
12.      }
13.   }
```

```
1.  Funcionario f =  new Funcionario();
```

Sobreescrita

Sobreescrita

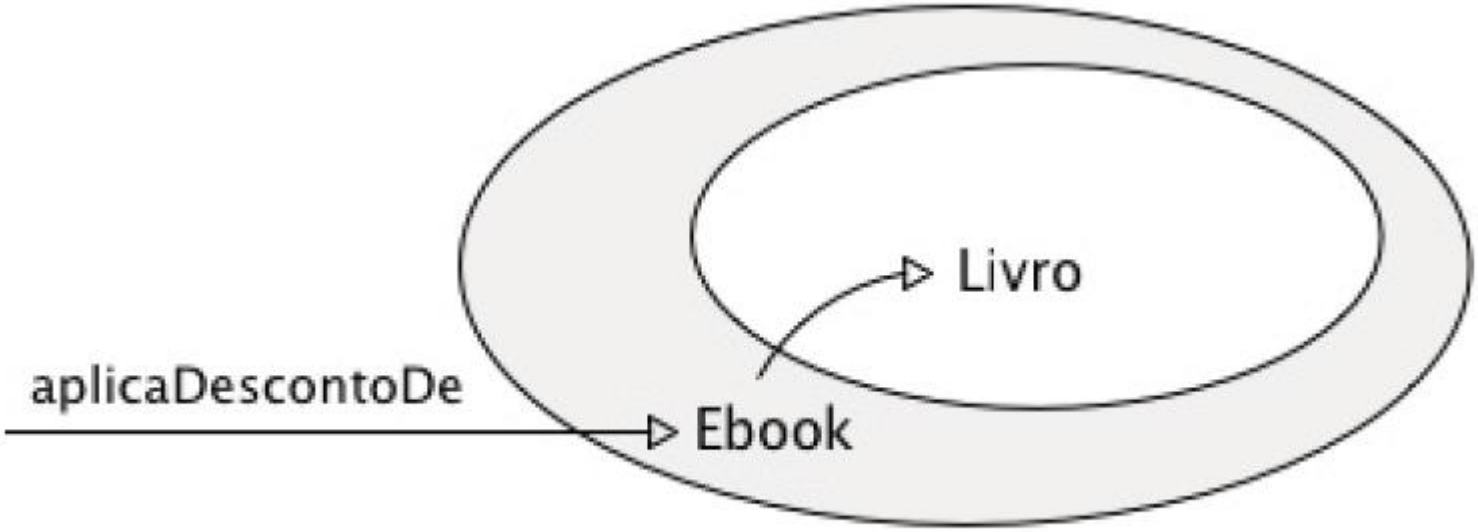
> ***Algumas vezes precisamos de comportamentos específicos para as classes filhas***. *Um modo de implementar isso é usar Sobreescrita. A sobrescrita ocorre quando criamos, na classe filha, métodos com o mesmo nome e os mesmos tipos de parâmetros de métodos da classe pai.*

Luis Araujo

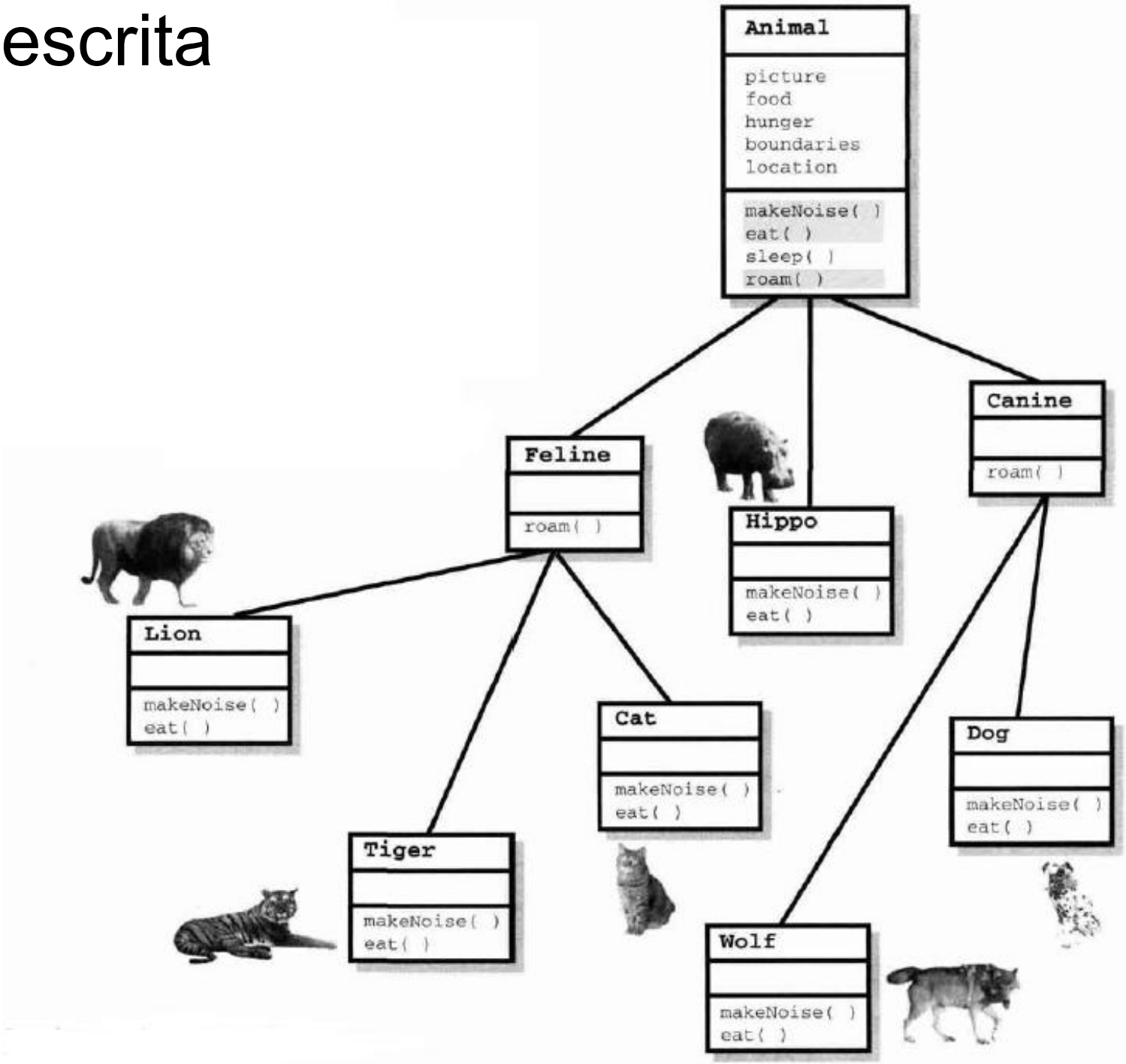
não achei uma boa definição nos livros...

Sobreescrita

> *AJVM sempre vai procurar o método primeiro no objeto que foi instanciado e apenas quando não encontrar procurará em sua superclasse. Você pode ver um exemplo aqui:*



Sobreescrita



Sobreescrita (*aula sobre herança*)

```
1.  public class Gerente class  Funcionario {
2.
3.   double bonus;
4.   /** construtor **/
5.   public Gerente(String nome, double salario){
6.      super(nome, salario);
7.      this.bonus = 0;
8.    }
9.    @Overrride
10.  public double getSalario(){
11.    return this.salario + this.bonus;
12.  }
13.}
```

FUNCIONÁRIO POSSUI O MESMO MÉTODO!

Referências Técnicas

Sierra, Kathy, and Bert Bates. **Use a cabeça!: java**. Alta Books, 2007.

HORSTMANN, S.; CORNELL, G.; **Core JAVA**. 1. vol. São Paulo: Pearson Education, 2010.

TURNI, R. **Desbravando Java e Orientação a Objetos**: um guia para o iniciante da linguagem. São Paulo: Casa do Código, 2016.

Programação Orientada a Objetos

Aulas nº X: SobreEscrita e SobreCarga

Prof. Luis Gustavo Araujo

2018