



UNIFACS
LAUREATE INTERNATIONAL UNIVERSITIES

Programação Orientada a Objetos

Aulas nº X: Herança

Prof. Luis Gustavo Araujo
2018

Objetivo

Compreender os conceitos e aplicações de Herança, bem como suas Vantagens e Desvantagens.

Classes

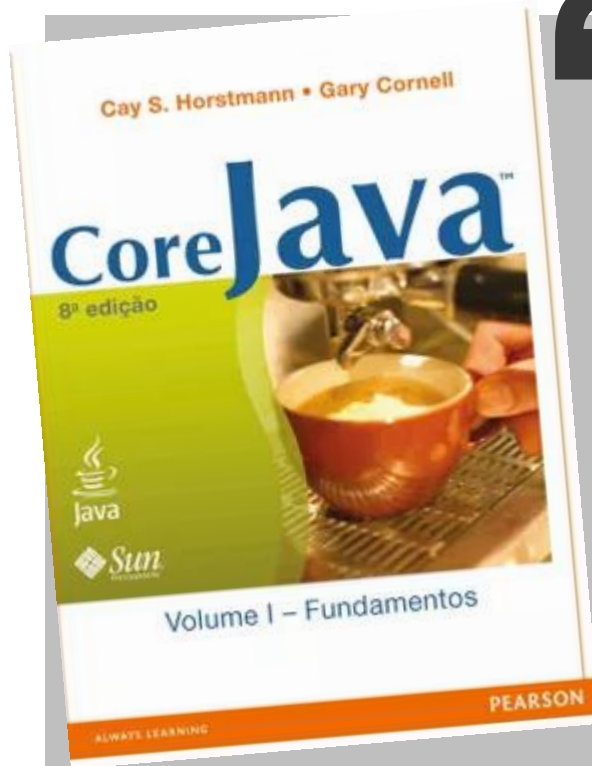
Associação

Herança

Relacionamento entre Classes

Herança é comumente tratada como um tipo de associação entre classes. Conhecido como relacionamento “É um”.
Ex: Funcionário é uma Pessoa.

Herança



“

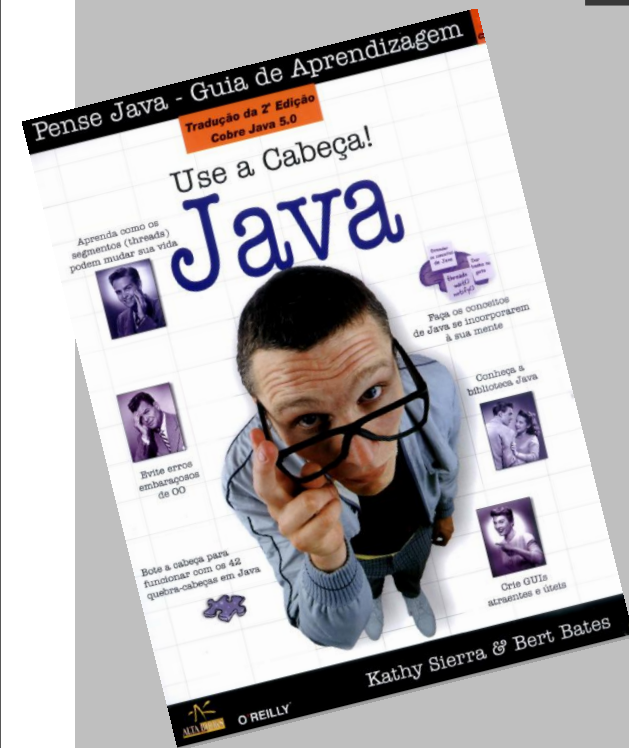
A ideia por trás da Herança é que você pode criar classes com base nas classes existentes. Ao herdar a partir de uma classe existente, reutiliza (ou herda) os métodos e campos e adiciona novos métodos e campos para adaptar sua nova classe a novas situações.

**Horstmann
& Cornell**

Core Java

Herança

“



Quando vocês projetar usando herança, inserirá código comum em uma classe e, em seguida, informará outras classes mais específicas que a classe comum (mais abstrata) é a sua superclasse. Quando uma classe herda de outra, a subclasse herda da superclasse.

Seirra & Bates

Use a Cabeça! Java

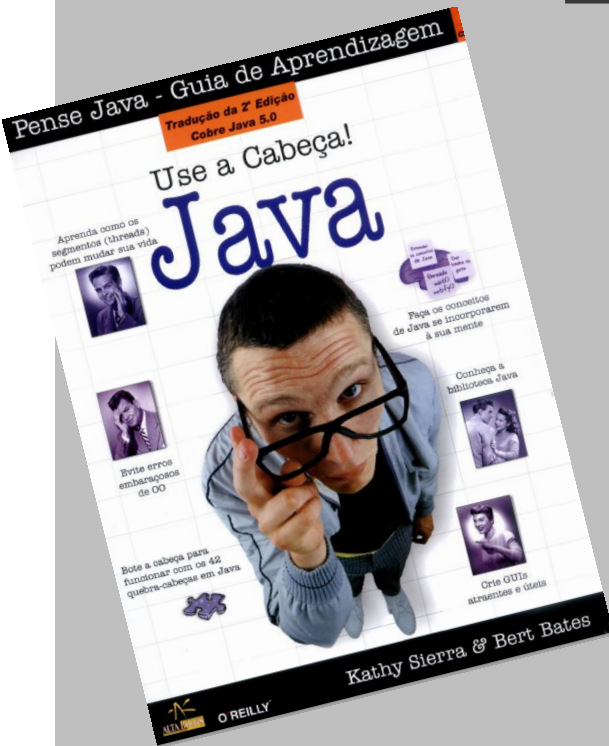
Herança

“

*Em Java, dizemos que uma **subclasse** **estende de uma superclasse**. Um relacionamento de herança significa que a subclasse herdará os membros da **superclasse**. Como termo “membros de uma classe” queremos nos referir às variáveis de instâncias e métodos.*

**Seirra
& Bates**

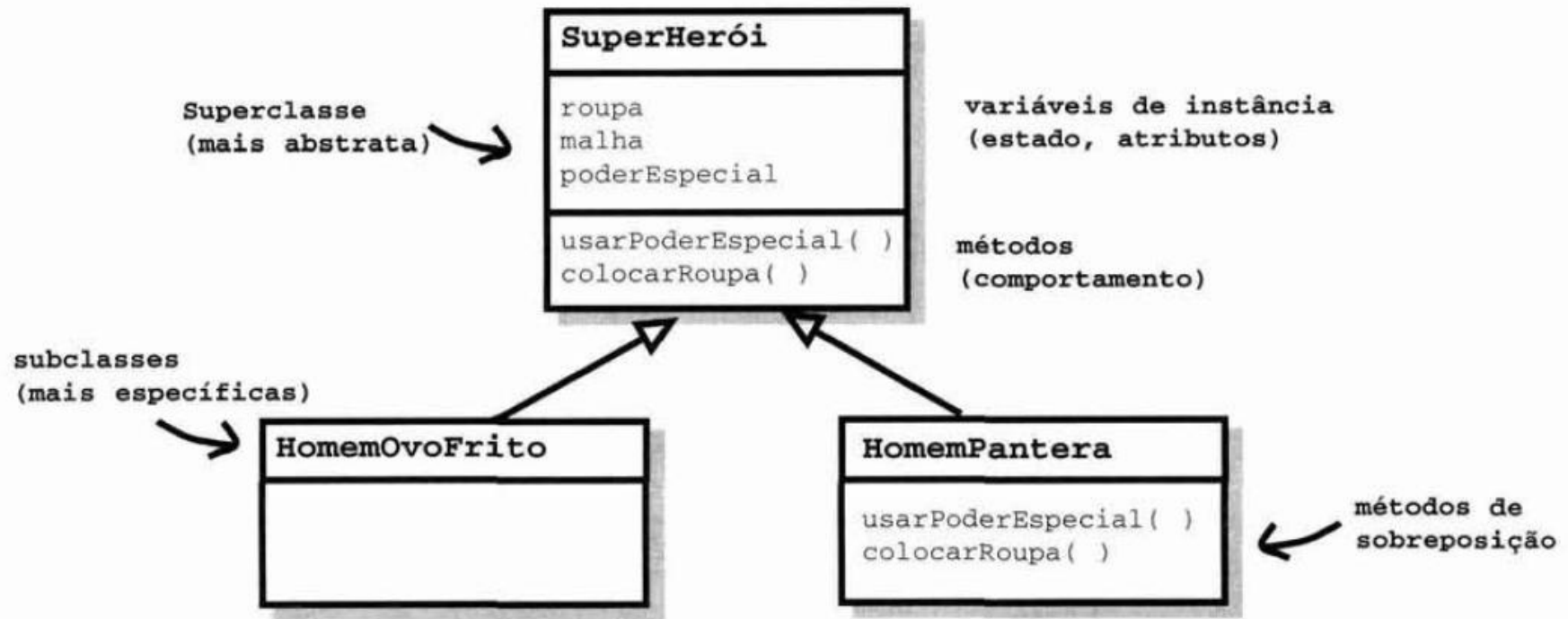
Use a Cabeça! Java



Termos utilizados na Herança

Na Herança existe a superclasse (classe básica ou classe pai) e a subclasse (classe derivada ou classe filha)

Herança (Diagrama)



Herança em Java

```
1. public class HomemOvoFrito extends SuperHeroi
2. {
3.     /** metodos e atributos */
4. }
```

Herança (Características)

A superclasse não pode acessar métodos específicos da subclasse, mas o contrário é verdadeiro : subclasses acessam métodos da superclasse.

Herança (Características)

A superclasse não pode acessar atributos específicos da subclasse, mas o contrário é verdadeiro : subclasses acessam atributos da superclasse.

Identificando Heranças

Em uma empresa, há funcionários que recebem o seu salário. Dentre os funcionários, há os Gerentes, esses recebem o salários + um bonus por serviços prestados à empresa.

Herança em Java

```
1. public class Funcionario{
2.     String nome;
3.     double salario;
4.     /** construtor */
5.     public Funcionario(String nome, double salario){
6.         this.nome = nome;
7.         this.salario = salario;
8.     }
9.     public String getNome(){
10.        return this.nome;
11.    }
12.    public double getSalario(){
13.        return this.salario;
14.    }
15.}
```

Herança em Java

```
1. public class Funcionario{
2.     String nome;
3.     double salario;
4.     /** construtor **/
5.     public Funcionario(String nome, double salario){
6.         this.nome = nome;
7.         this.salario = salario;
8.     }
9.     public String getNome(){
10.        return this.nome;
11.    }
12.    public double getSalario(){
13.        return this.salario;
14.    }
15.}
```

Não é válido para
Gerente!


Herança (Exemplo)

Como funcionário e Gerente possuem coisas em comum, podemos criar a classe Gerente como subclasse de Funcionário e adicionar o atributo bonus.

Herança em Java

```
1. public class Gerente class Funcionario {  
2.     double bonus;  
3.     /** construtor **/  
4.     public Gerente(String nome, double salario){  
5.         this.nome = nome;  
6.         this.salario = salario;  
7.         this.bonus = 0;  
8.     }  
9. }
```

Funcionário já
faz isso no construtor



Como chamar o construtor de
Funcionário?

Acessando a Superclasse

Assim como a palavra *this*, que referencia a própria classe, podemos, na herança, usar palavras especiais: *super* para referenciar a superclasse.

Herança em Java

```
1. public class Gerente class Funcionario {  
2.     double bonus;  
3.     /** construtor **/  
4.     public Gerente(String nome, double salario){  
5.         super(nome, salario);  
6.         this.bonus = 0;  
7.     }  
8. }
```

Mas ainda assim, quando eu usar
getSalario, só irei receber o salário, pois é
como está em Funcionário (superclasse).

Sobrescrita de métodos

veremos com mais detalhes depois, mas permite rescrever o método apenas para a classe Gerente, basta declararmos com o mesmo nome.

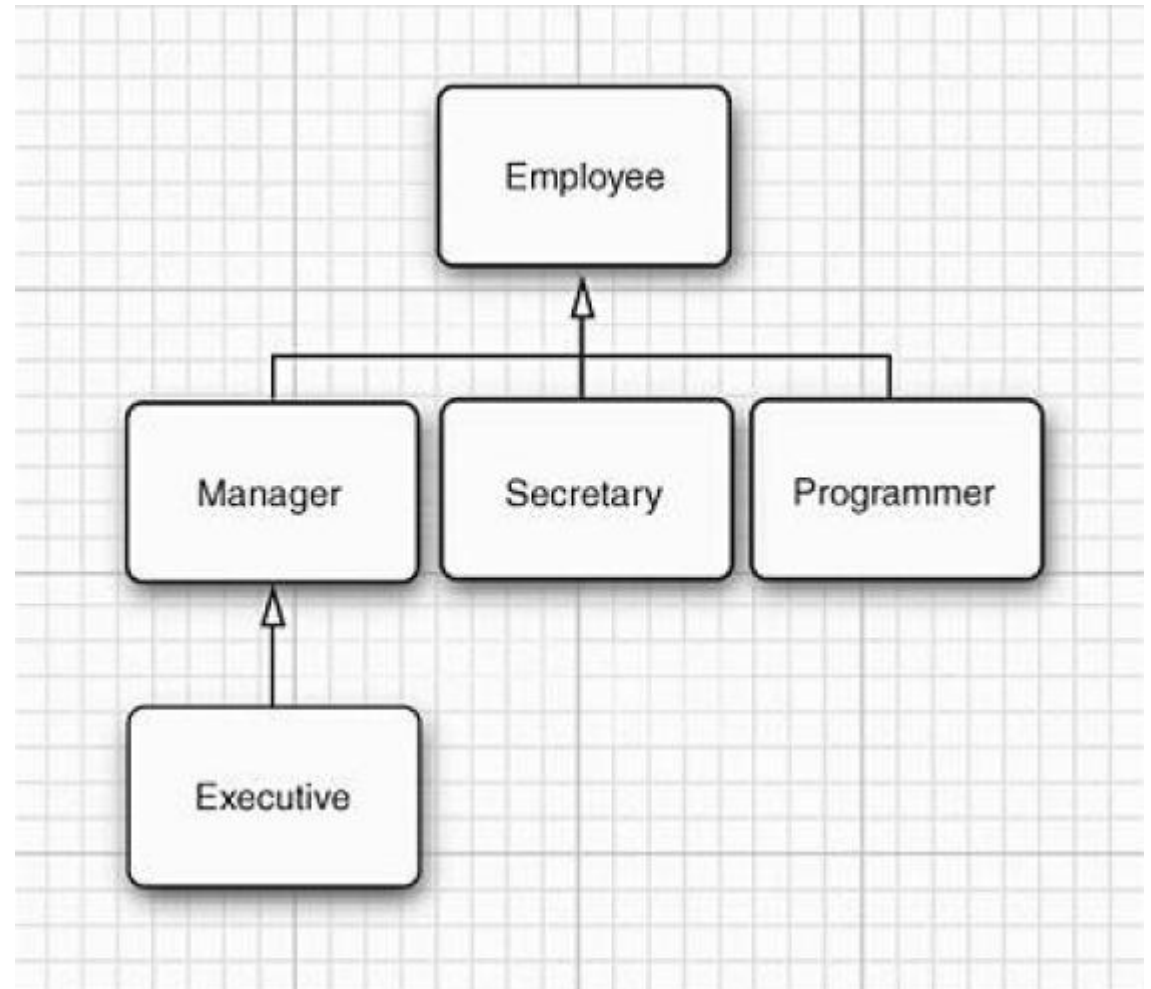
Herança em Java

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

Hierarquia de Herança

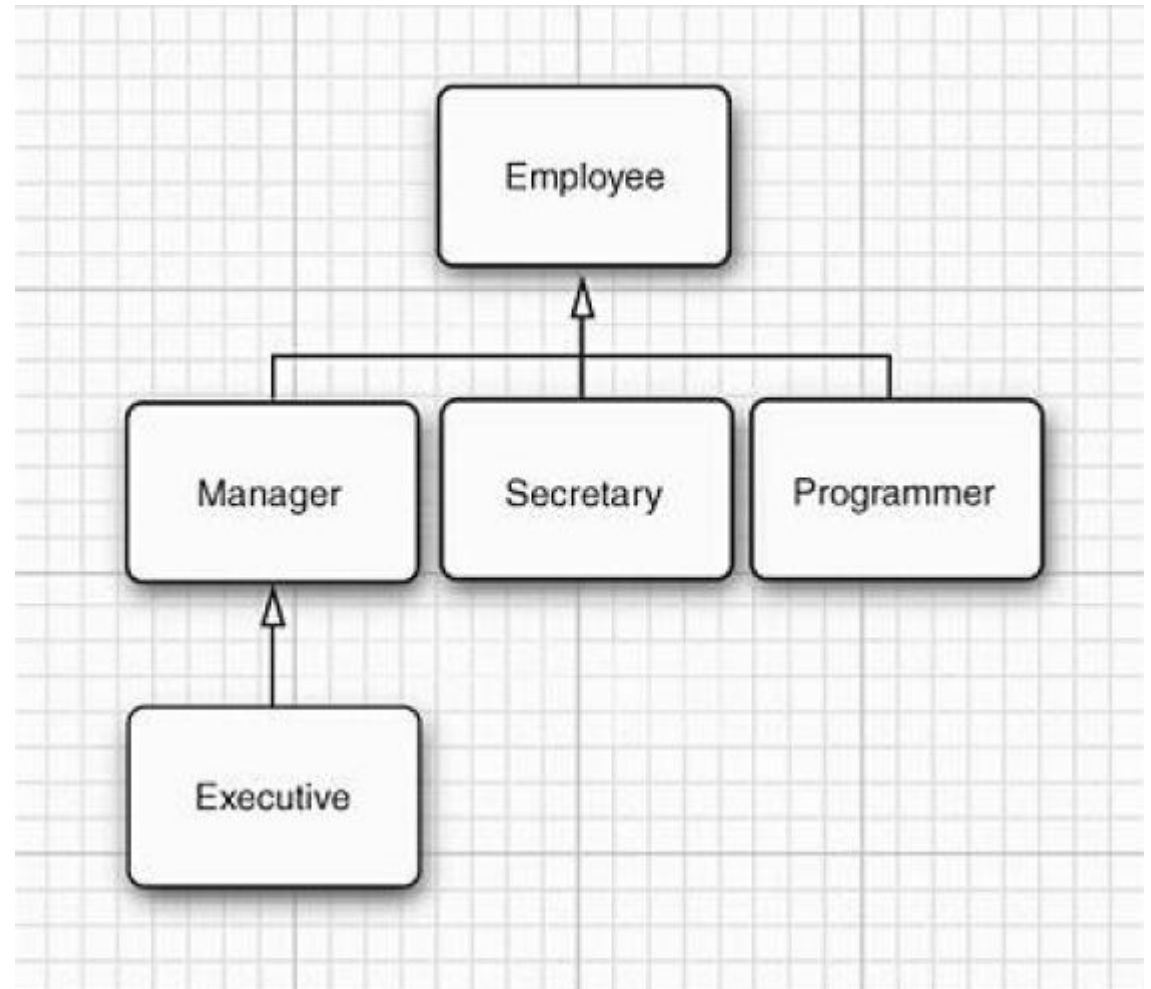
Você pode usar quantas heranças forem necessárias. Criando especializações das classes.

A coleção de todas as classes que se estendem a partir de uma superclasse, chama-se hierarquia de Herança.



Hierarquia de Herança

O Caminho entre uma classe específica e seus antepassados na hierarquia de herança é sua **cadeia de herança**.



Restrição de Acesso em Herança

```
1. public class Funcionario{  
2.     private String nome;  
3.     private double salario;  
4. }
```

```
1. public class Gerente class Funcionario {  
2.  
3.     private double bonus;  
4.     public void showNome(){  
5.         system.out.print(this.nome);  
6.     }  
7. }
```

Restrição de Acesso em Herança

```
1. public class Funcionario{  
2.     protected String nome;  
3.     protected double salario;  
4. }
```

```
1. public class Gerente class Funcionario {  
2.  
3.     private double bonus;  
4.     public void showNome(){  
5.         system.out.print(this.nome);  
6.     }  
7. }
```


Atividade

1 - Identifique no Projeto de Gestão Escolar, a presença de Heranças (Modelo a Classes, Atributos e Métodos).

Vantagens da Herança

- a) Elimina código duplicado, generalizando um comportamento de um grupo;
- a) Possibilita manutenção em cascata;
- a) Estabelece um protocolo para as subclasses (contrato)

Desvantagens da Herança

- a) Gera alto acoplamento entre as classes;
- b) Fraco encapsulamento;

Herança ou Composição?

Dada as desvantagens da Herança, recomenda-se que ela seja evitada. Quando possível, utiliza-se Composição!

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.



UNIFACS
LAUREATE INTERNATIONAL UNIVERSITIES

Programação Orientada a Objetos

Aulas nº X: Herança

Prof. Luis Gustavo Araujo
2018