

# Programação Orientada a Objetos

1

**BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO**

**PROF. LUCIO AGOSTINHO ROCHA**

**AULA 7: HERANÇA**

**2º.SEMESTRE 2022**

Programação Orientada a Objetos - UTFPR Campus Apucarana

2

**Herança**

Programação Orientada a Objetos - UTFPR Campus Apucarana

# Herança

3

- Herança:

- Reusabilidade de Software
- Novas classes de objetos aproveitam características de classes já existentes.
  - ✦ Aproveitamento de Atributos e Métodos
  - ✦ Adição de novas funcionalidades
  - ✦ Exemplo: Classe Sapo herda da Classe Animal

# Herança

4

- Herança:

- Generalização x Especialização
- Subclasse herda da Superclasse
  - ✦ Subclasse: adiciona novas variáveis de instância e métodos
- Java não suporta herança múltipla, porém admite múltiplas Interfaces
- Herança: define um relacionamento do tipo “é um”:
  - ✦ Ex.: Sapo é um Animal
- Composição: define um relacionamento do tipo “tem um”:
  - ✦ Ex.: Empregado tem uma Data de Contratação.

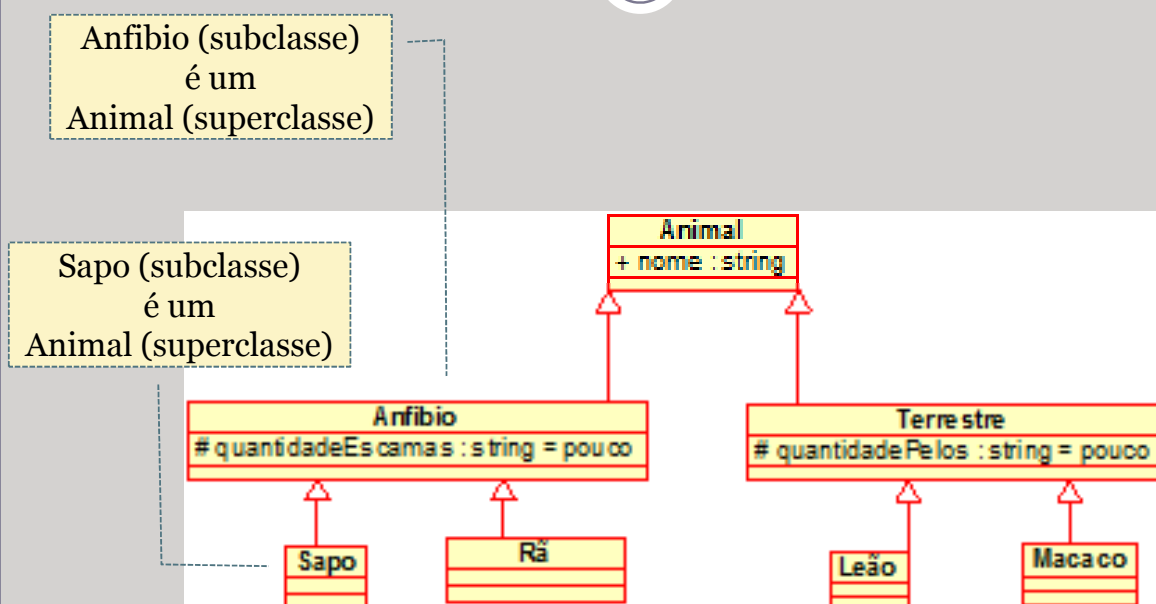
# Herança

5

- Herança: relacionamento “é um”:
  - Objeto “é um” objeto de outra classe
  - Herança define uma hierarquia de árvore.
    - ✦ Nota: Herança múltipla é fonte de problemas.
    - ✦ Nota: Hierarquia em árvore remove o problema de loops sintáticos e semânticos.

# Herança

6

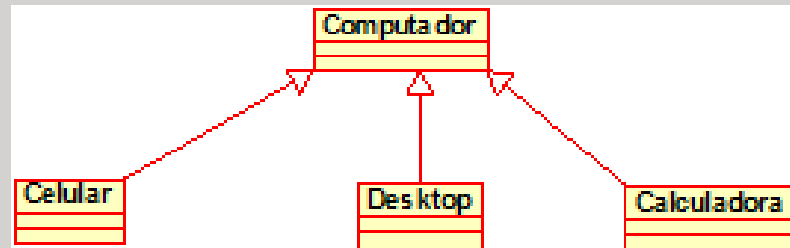


- Figura - Exemplo de Relacionamento de Herança.

# Herança

7

Celular (subclasse)  
é um  
Computador (superclasse)



- Figura - Exemplo de Relacionamento de Herança.

# Herança

8

- Herança:
  - Ocultamento de informação:
    - ✦ Objetos não sabem como outros objetos são implementados.
      - Ex.: é possível usar um computador sem saber como ele calcula os números.
  - Abstração:
    - ✦ Pensar em termos de propriedades comuns a vários objetos.
      - Celular (Classe) é um tipo de Computador (Classe)
      - Desktop (Classe) é um tipo de Computador (Classe)
      - Calculadora (Classe) é um tipo de Computador (Classe)

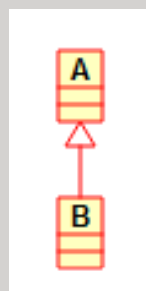
# Herança

9

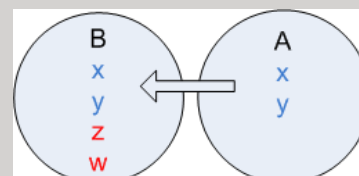
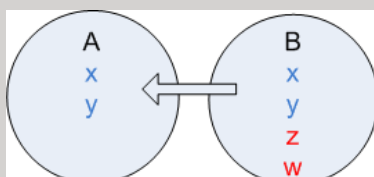
- Objeto da Subclasse (mais especializado) pode ser tratado como um objeto da Superclasse (menos especializado)
  - Subclasse contém todos os Membros da Superclasse
  - Ex.: Sapo é um Anfíbio E Sapo é um Animal
- Objeto da Superclasse (menos especializado) não podem ser diretamente tratados como um objeto da Subclasse (mais especializado).
  - Há perda de informação.
  - Ex.:
    - ✖ Animal é um Anfíbio X
    - ✖ Animal é um Sapo X

# Herança

10



- B é um A

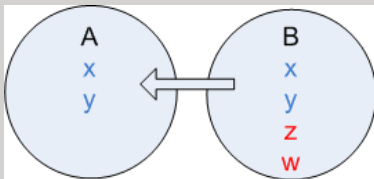


- Faltam campos

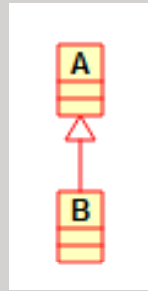
# Herança

11

```
public class A {  
    public A() { }  
}  
public class B extends A {  
    public B() { }  
    public void metodoB() {  
        return "B";  
    }  
}
```

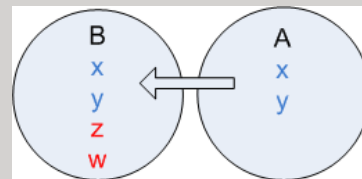


- Lado direito deve, pelo menos, possuir todos os campos do lado esquerdo.



- B é um A

```
public class C {  
    A a = new A ();  
    B b = new B ();  
    a = b;    //'a' continua sendo 'A'  
             ((B) a).metodoB();  
  
    b = (B) a; //downcast  
              //(necessário a=b)  
}
```



- Faltam campos

- Lado direito deve, pelo menos, possuir todos os campos do lado esquerdo.

# Herança

12

- Construtor:
  - Construtores nunca são herdados.
  - É a primeira instrução invocada pela classe.
  - Há chamadas sucessivas na hierarquia para inicializar as variáveis de instância de cada classe

# Herança

13

- **Garbage Collector:**
  - Método finalize da subclasse deve invocar o método finalize da superclasse com sobrecarga.
  - Caso contrário, apenas a subclasse será indicada para Garbage Collector.
  - Portanto, é uma boa prática de programação sempre incluir um método finalize na superclasse.

14



## Revisão

## Revisão

15

- Herança:
  - É um mecanismo nativo de linguagens de Programação Orientadas a Objetos que permite o reaproveitamento de código.
  - Herança melhora a legibilidade do código e auxilia na organização da estrutura do projeto.

## Exercícios

16

<Ver conteúdo na Plataforma de Ensino>





# Referências

17

- Referências bibliográficas da disciplina.