

TECNOLOGIA EM INFORMÁTICA

Estrutura de Dados

Prof. Glauco da Silva
contato: glaucoslv@gmail.com

Roteiro

- *Objetivos*
- *Herança*
- *Polimorfismo*
- *Herança e Polimorfismo*
- *Exercícios*

Roteiro

- ***Objetivos***
- *Herança*
- *Polimorfismo*
- *Herança e Polimorfismo*
- *Exercícios*

Objetivos

- *Rever os conceitos de Herança e Polimorfismo*
- *Apresentar exemplos e propor exercícios sobre o assunto*

Roteiro

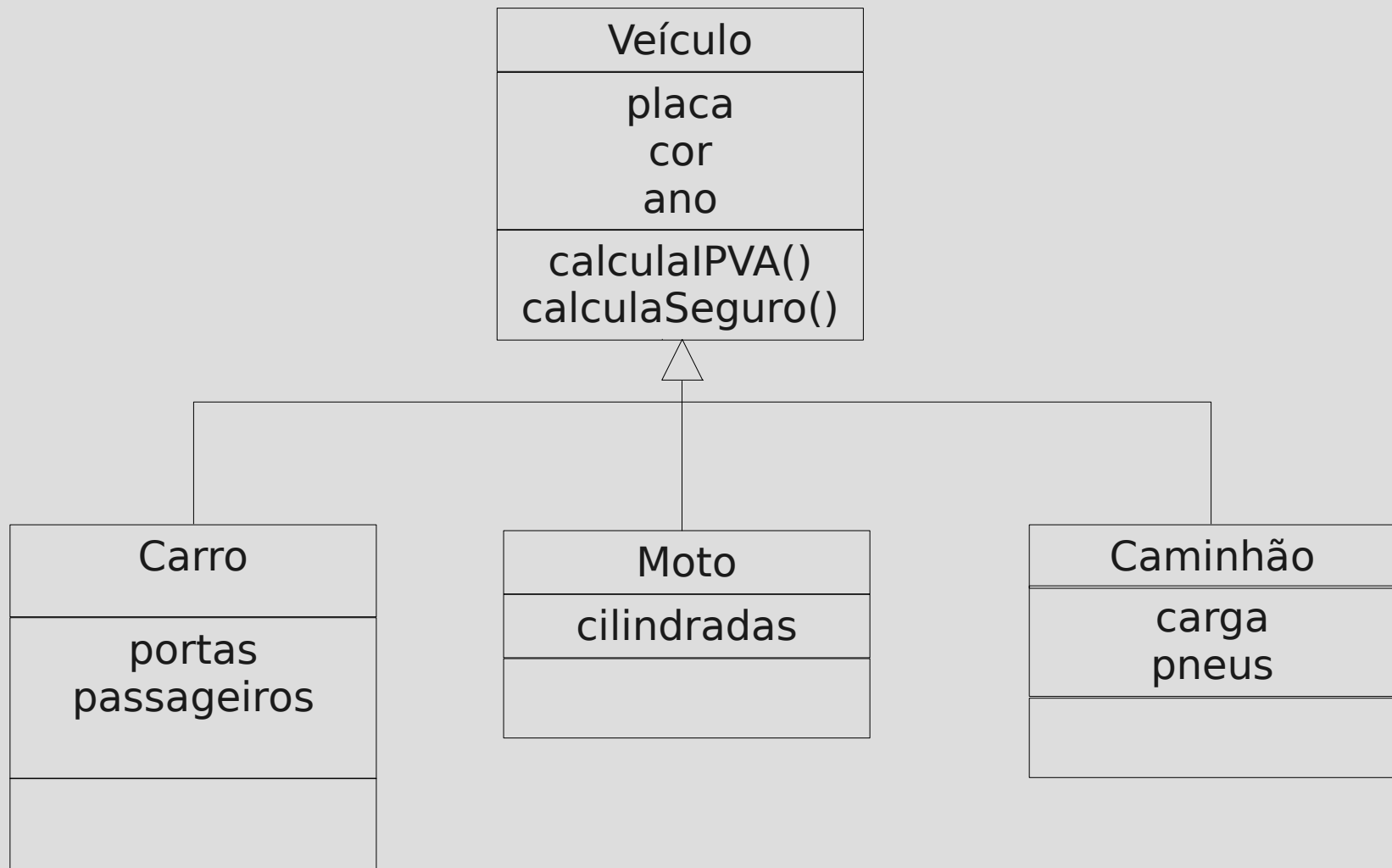
- *Objetivos*
- ***Herança***
- *Polimorfismo*
- *Herança e Polimorfismo*
- *Exercícios*

Herança

É a capacidade de se criar novos objetos que mantêm as propriedades e comportamentos dos objetos ancestrais.

Este conceito permite que se crie hierarquias de objetos que têm em um primeiro nível os objetos genéricos e nos próximos níveis os objetos que tem funcionalidades específicas. A vantagem da herança é a reutilização de código, uma vez que os objetos podem compartilhar o mesmo código.

Herança



Herança

- *Formas de utilização*

- *Especialização*

- *Refinamento da classe genérica em subclasses específicas*

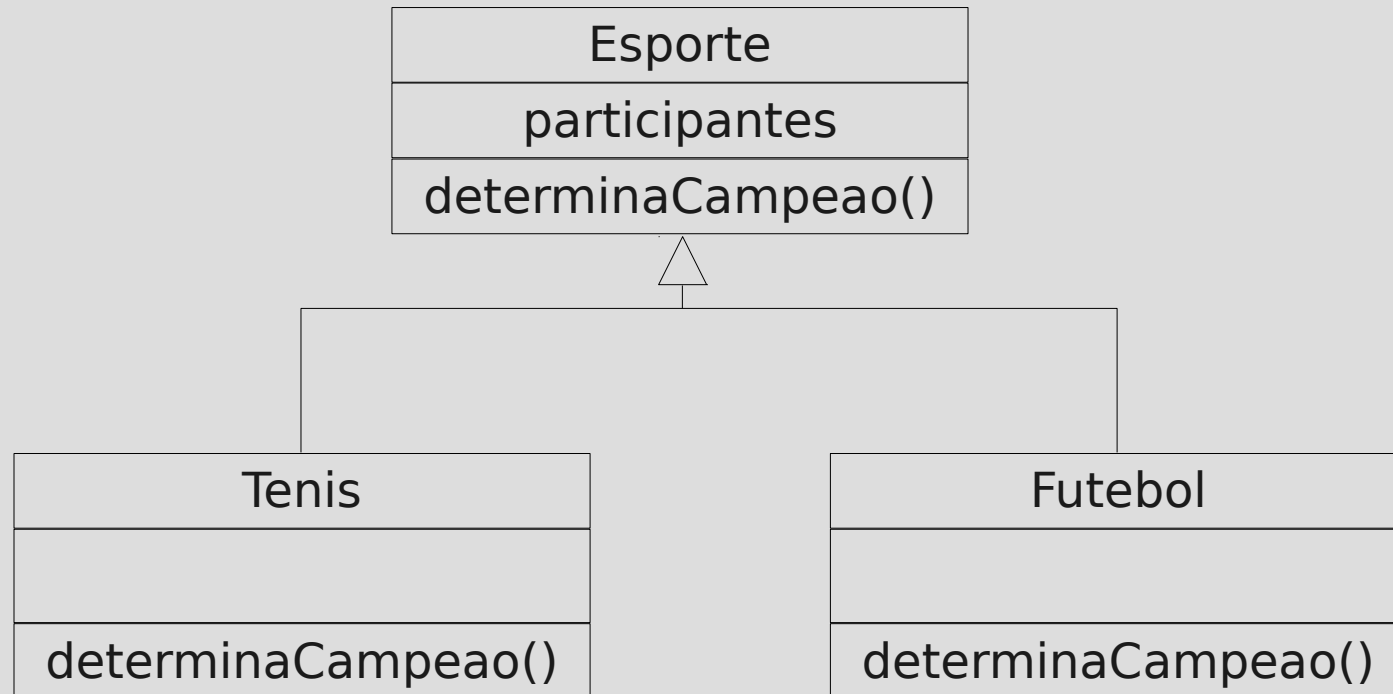
- *Subclasses possuem um relacionamento do tipo “é um” com a superclasse*

- *A subclasse especializa os métodos da superclasse*

Herança

- *Formas de utilização*

- *Especialização*



Herança

- *Formas de utilização*

- *Extensão*

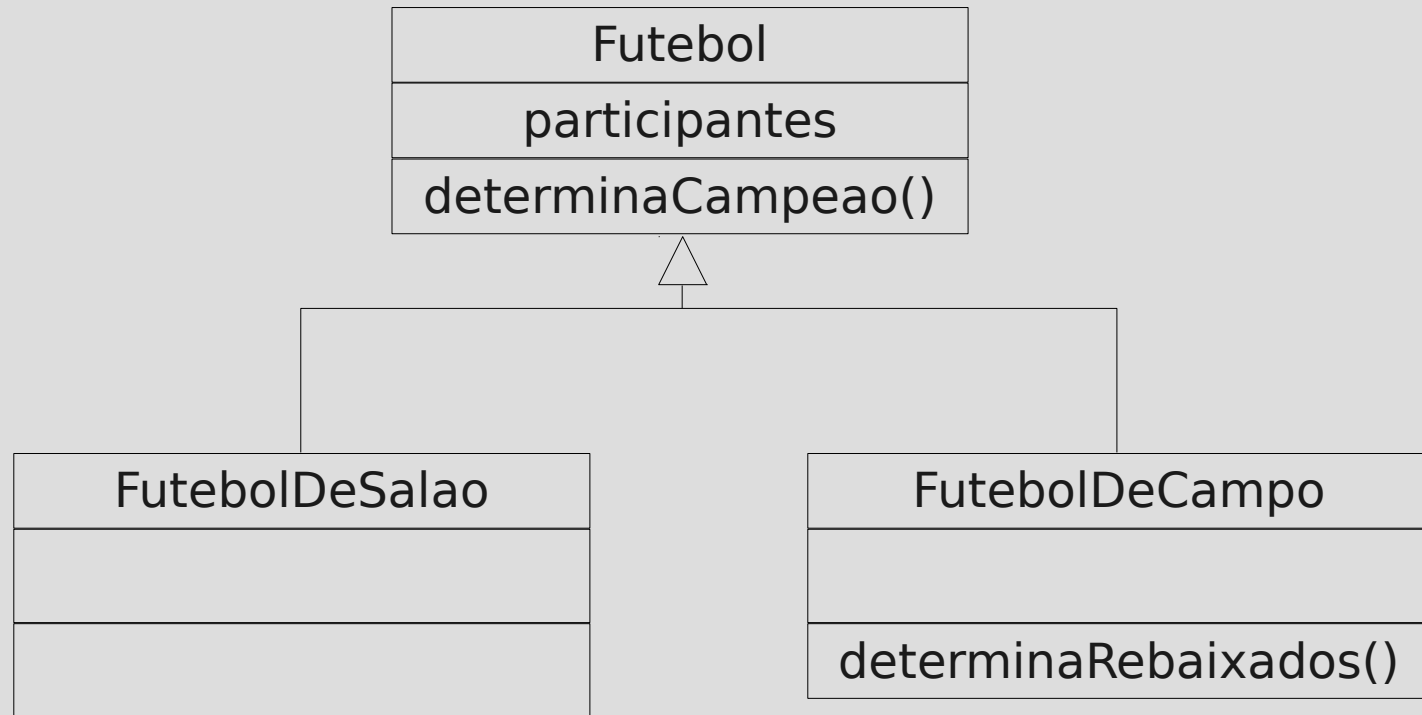
- *Reutilização do código da superclasse*

- *Adição de novos métodos, de forma a estender a funcionalidade da subclasse*

Herança

- *Formas de utilização*

- *Extensão*



Herança

- *A palavra reservada this*
 - *Utilizada para referenciar a instância corrente da classe*
 - *Utilizada para referenciar um atributo do objeto corrente que está em conflito com o nome de uma variável do bloco corrente*

Herança

- *A palavra reservada this*

```
public class TesteThis {
    public int num=2;

    public void numero(){
        int num=5;
        System.out.println("A variavel local num é = " + num);
        System.out.println("O atributo num é = " + this.num);
    }

    public static void main(String[] args){
        TesteThis t = new TesteThis();
        t.numero();
    }
}
```

Roteiro

- *Objetivos*
- *Herança*
- ***Polimorfismo***
- *Herança e Polimorfismo*
- *Exercícios*

Polimorfismo

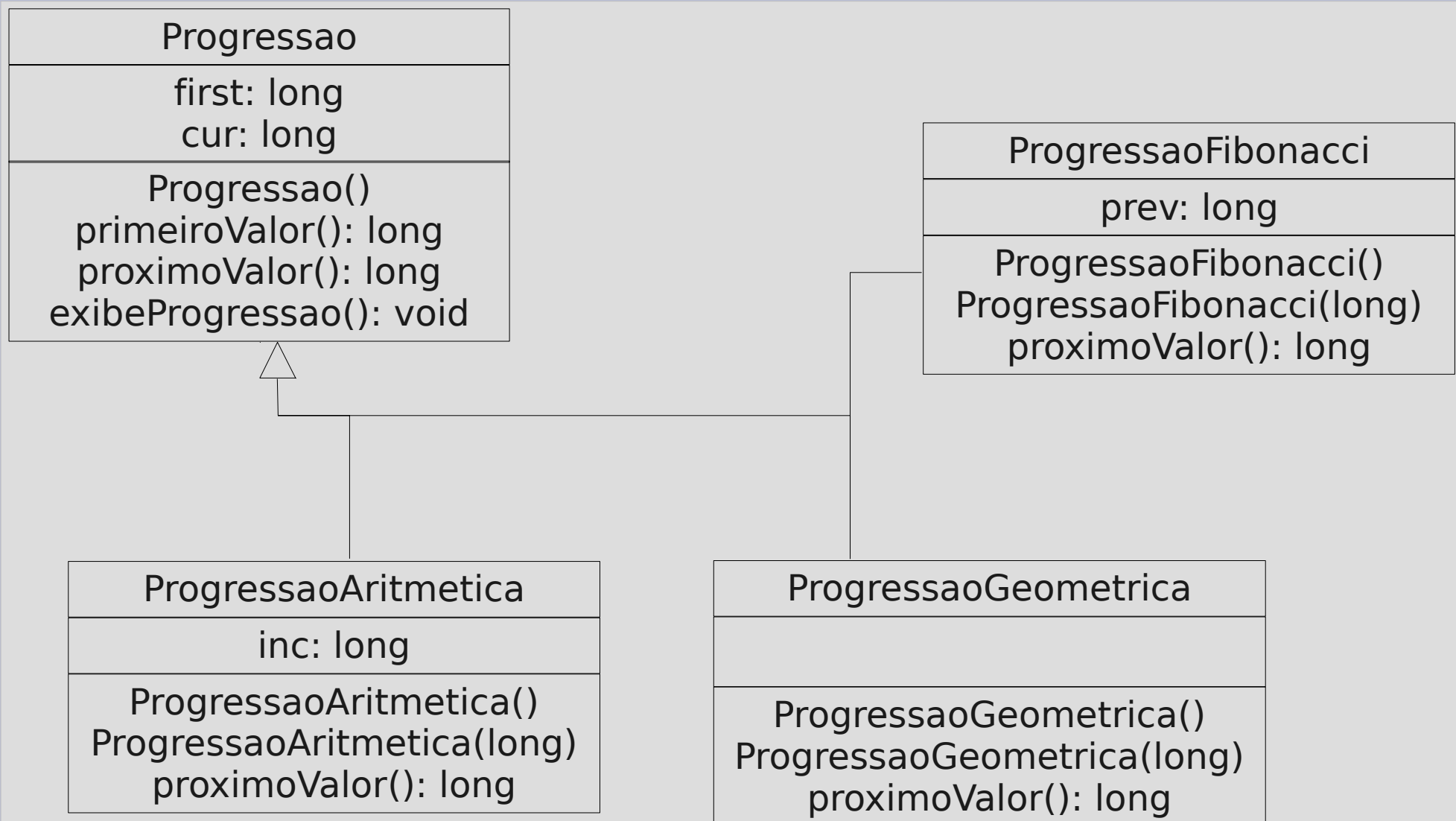
- *Literalmente, polimorfismo significa "muitas formas". Em linguagens orientadas a objeto, esse conceito refere-se à habilidade de uma variável de objeto assumir formas diferentes.*

Roteiro

- *Objetivos*
- *Herança*
- *Polimorfismo*
- *Herança e Polimorfismo*
- *Exercícios*

Herança e Polimorfismo

- *Exemplo: Progressões matemáticas*



Herança e Polimorfismo

Progressao.java

```
public class Progressao {
    protected long first;
    protected long cur;

    Progressao(){
        cur = first = 0;
    }

    protected long primeiroValor(){
        cur = first;
        return cur;
    }

    protected long proximoValor(){
        return ++cur;
    }

    public void exhibeProgressao(int n){
        System.out.print(primeiroValor());
        for (int i = 2; i<=n; i++)
            System.out.print(" " + proximoValor());
        System.out.println();
    }
}
```

Herança e Polimorfismo

ProgressaoAritmetica.java

```
public class ProgressaoAritmetica extends Progressao{
    protected long inc;

    ProgressaoAritmetica(){
        inc = 1;
    }

    ProgressaoAritmetica(long increment){
        inc = increment;
    }

    protected long proximoValor(){
        cur += inc;
        return cur;
    }
}
```

Herança e Polimorfismo

ProgressaoGeometrica.java

```
public class ProgressaoGeometrica extends Progressao{
    protected long base;

    ProgressaoGeometrica(){
        base = 2;
        first = 1;
        cur = first;
    }

    ProgressaoGeometrica(long b){
        base = b;
        first = 1;
        cur = first;
    }

    protected long proximoValor(){
        cur *= base;
        return cur;
    }
}
```

Herança e Polimorfismo

ProgressaoFibonacci.java

```
public class ProgressaoFibonacci extends Progressao{
    long prev;

    ProgressaoFibonacci(){
        first = 0;
        prev = 1;
    }

    ProgressaoFibonacci(long valor1, long valor2){
        first = valor1;
        prev = valor2 - valor1;
    }

    protected long proximoValor(){
        long temp = prev;
        prev = cur;
        cur += temp;
        return cur;
    }
}
```

Herança e Polimorfismo

TesteProgressao.java

```
public class TesteProgressao {
    public static void main(String[] args){
        Progressao prog;
        System.out.println("Progressao aritmetica com incremento padrao:");
        prog = new ProgressaoAritmetica();
        prog.exibeProgressao(10);
        System.out.println("Progressao aritmetica com incremento de 5:");
        prog = new ProgressaoAritmetica(5);
        prog.exibeProgressao(10);
        System.out.println("Progressao geometrica com base padrao:");
        prog = new ProgressaoGeometrica();
        prog.exibeProgressao(10);
        System.out.println("Progressao geometrica com base 3:");
        prog = new ProgressaoGeometrica(3);
        prog.exibeProgressao(10);
        System.out.println("Sequencia de Fibonacci com valores iniciais padrao:");
        prog = new ProgressaoFibonacci();
        prog.exibeProgressao(10);
        System.out.println("Sequencia de Fibonacci com valores iniciais 4 e 6:");
        prog = new ProgressaoFibonacci(4,6);
        prog.exibeProgressao(10);
    }
}
```

Roteiro

- *Objetivos*
- *Herança*
- *Polimorfismo*
- *Herança e Polimorfismo*
- *Exercícios*

Exercícios

- *Escreva o código para o seguinte esquema de heranças. Em seguida escreva uma classe executável, crie um objeto de cada classe e chame o método `Habilidade()` a partir dos objetos criados.*

