

UFSC-CTC-INE  
INE5384 - Estruturas de Dados

## Linguagem Algorítmica OO

Prof. Ronaldo S. Mello  
2002/2

## Linguagem Algorítmica

- Independente de linguagem de programação OO
- Utilizada para:
  - Exemplificar os algoritmos de manipulação das estruturas de dados
  - Desenvolver exercícios em sala de aula

## Definição de Classe

```
Classe nome_classe  
  [Subclasse de nome_classe]  
  [Implementa nome_interface]  
início  
  [definições_atributos;]  
  definições_métodos;  
fim;
```

## Definição de Atributos

```
Classe Pessoa  
início  
  privado nome string;  
           idade inteiro;  
           sexo caractere;  
  default* estadoCivil string ('solteiro', 'casado', ...);  
           temFilhos booleano;  
           residência Endereço;  
  ...  
fim;
```

*tipos de dados* →

↑  
*default\**

*\* outras possibilidades: protegido, público*

## Definição de Métodos

Classe Pessoa

início

...

*default*  
**público método** FazAniversário();

**início**

idade ← idade + 1; *comando de atribuição*

**fim;**

...

fim;

## Parâmetros

Classe Pessoa

início

...

*parâmetros*  
**método** AlteraEndereço(*novaRua string,*  
*novoNúmero inteiro, novaCidade string*);

**início**

residência.AlterarRua(novaRua);

residência.AlterarNúmero(novoNúmero);

residência.AlterarCidade(novaCidade);

**fim;**

...

fim;

## Método com Retorno

Classe Pessoa

início

...

método ehCrianca() *tipo retornado* retorna booleano;

início

se idade < 14 então retorna V

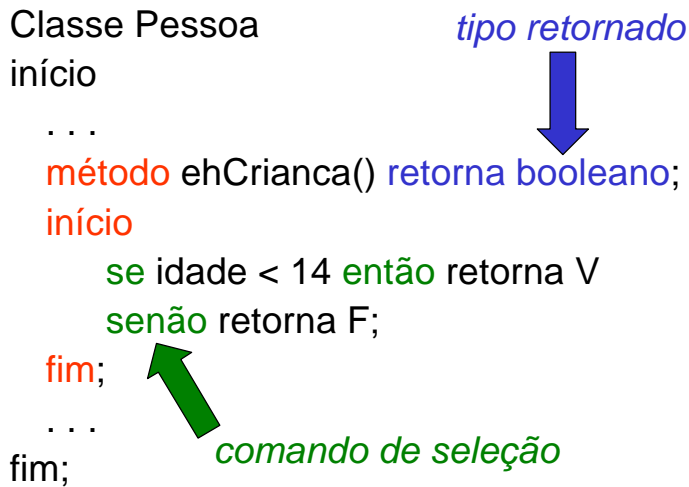
senão retorna F;

fim;

...

fim;

*comando de seleção*



## Método Construtor

construtor Pessoa(nome string, idade inteiro,  
sexo caractere, estadoCivil string,  
residência Endereço);

início

se idade < 18 E estadoCivil ≠ 'solteiro' então

Exceção estadoCivilInválido();

this.nome ← nome;

this.idade ← idade;

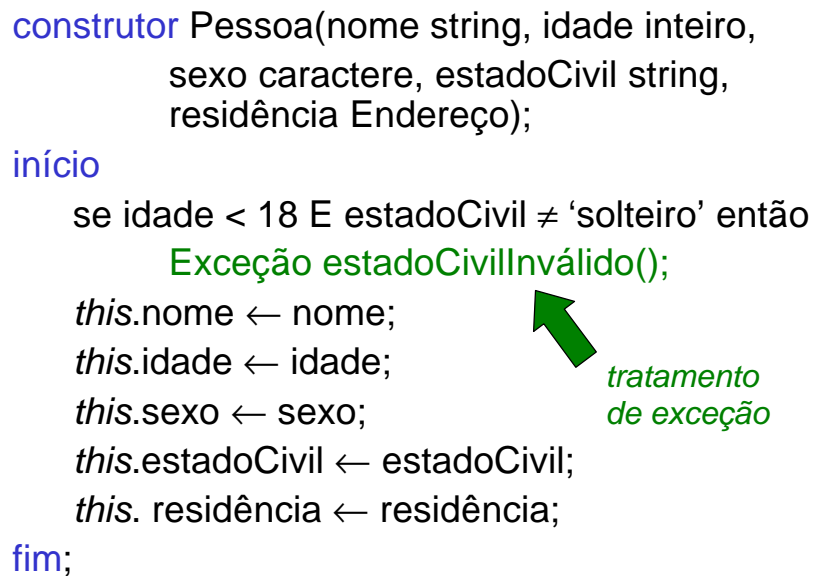
this.sexo ← sexo;

this.estadoCivil ← estadoCivil;

this.residência ← residência;

fim;

*tratamento de exceção*



## Classe Especializada

Classe Empregado

Subclasse de Pessoa

início

salário real;

empresa string;

horasTrabalhoDia inteiro[0..5];

. . .

fim;

## Comandos de Repetição

método médiaHorasTrabalhadas() retorna real;

início

somaHoras real;

i inteiro;

somaHoras  $\leftarrow$  0;

**para** i **de** 0 **até** 5 **faça**

somaHoras  $\leftarrow$  somaHoras + horasTrabalhoDia[i];

retorna somaHoras / 6;

fim;

## Comandos de Repetição

```
método diaVago() retorna booleano;  
início  
    dia inteiro;  
  
    dia ← 0;  
    enquanto dia <= 5 faça  
        início  
            se horasTrabalhoDia[dia] = 0 então retorna V;  
            dia ← dia + 1;  
        fim;  
    retorna F;  
fim;
```

## Comando Caso

```
método ehPobre() retorna booleano;  
início  
    se salário < 200.00 então retorna V;  
    caso estadoCivil  
        'solteiro': se temFilhos = V E salário < 500.00  
            então retorna V;  
        'casado',  
        'divorciado': se salário < 500.00 OU  
            (salário < 800.00 E temFilhos = V)  
            então retorna V;  
        'viúvo': ...  
    fim;  
    retorna F;  
fim;
```

## Leitura de Dados

```
Classe Empregado
Especialização de Pessoa
início
    construtor Empregado();
    início
        . . .
        Ler(nome, idade, ...);
        this.nome ← nome;
        . . .
    fim;
fim;
```

## Escrita de Dados

```
Classe Empregado
Especialização de Pessoa
início
    método ImprimeSalário();
    início
        Escrever('Valor do salário: ', salário);
    fim;
fim;
```

## Tradução para Delphi

Classe Empregado  
Subclasse de Pessoa

início

```
salário real;  
empresa string;  
horasTrabalhoDia  
    inteiro[0..5];  
...
```

fim;

Type Empregado = class (Pessoa)

private

```
salário: real;  
empresa: string;  
horasTrabalhoDia:  
    array[0..5] of integer;
```

public . . .

implementation . . .

end.



## Tradução para Java

Classe Empregado  
Subclasse de Pessoa

início

```
salário real;  
empresa string;  
horasTrabalhoDia  
    inteiro[0..5];  
...
```

fim;

public class Empregado

extends Pessoa

{

```
protected float salário,  
protected empresa string;  
protected int[ ] horasTrabalhoDia;  
...
```

...

public Empregado(...)

{ ...

*this*.horasTrabalhoDia ← new int[5]; ...

}

}





## Tradução para Delphi

Classe Pessoa

início

...

método AlteraEndereço(novaRua string, ...);

início

residência.AlterarRua(novaRua); ...

fim;

fim.



Type Pessoa = class

...

public

procedure AlteraEndereço(novaRua: string, ...);

...

implementation

procedure Pessoa.AlterarEndereço(novaRua: string, ...);

begin

residência.AlterarRua(novaRua); ...

end;

end.

## Tradução para Java

Classe Pessoa

início

...

método AlteraEndereço(novaRua string, ...);

início

residência.AlterarRua(novaRua); ...

fim;

fim.



public class Pessoa

{

...

public void AlterarEndereço(novaRua string, ...);

{

residência.AlterarRua(novaRua); ...

}

...

}

## Exemplo de Função

```
método diaVago() retorna booleano;  
início  
    dia inteiro;  
  
    dia ← 0;  
    enquanto dia <= 5 faça  
    início  
        se horasTrabalhoDia[dia] = 0 então retorna V;  
        dia ← dia + 1;  
    fim;  
    retorna F;  
fim;
```

## Tradução para Delphi

```
type Empregado = class (Pessoa)  
...  
public  
    function diaVago : boolean;  
...  
implementation  
    function Empregado.diaVago() : boolean;  
    var dia: integer;  
    begin  
        dia := 0;  
        while dia <= 5 do  
            begin  
                if horasTrabalhoDia[dia] = 0 then result := true;  
                dia := dia + 1;  
            end;  
        fim;  
        result := false;  
    end;
```

## Tradução para Java

```
public class Empregado
    extends Pessoa
{
    public boolean diaVago();
    {
        int dia;

        dia = 0;
        while dia <= 5
        {
            if horasTrabalhoDia[dia] = 0 return true;
            ++dia;
        }
        return false;
    }
}
```