

---

# Sumário

- Classe em Java
- Métodos em Java
  - Organização
  - Tipos de modificadores
  - Tipos de Métodos
  - Métodos Construtores
    - Encadeamento
    - Sobrecarga
  - Métodos Operacionais
    - Métodos de Acesso
    - Sobrecarga
- Atributos em Java

# A Classe em Java

## ■ Corpo da classe

- O corpo de uma classe contém os atributos e métodos da classe
- O corpo de uma classe é determinado por um par de chaves

```
class aviao
{
    atributo1
    atributo2
    ....

    metodo1()
    metodo2()
    ....
}
```

---

# A Classe em Java

- Tipo de classe

- Classes privadas

- somente podem ser manipuladas por outras classes definidas no mesmo módulo de código;

- Classes públicas

- podem ser manipuladas por qualquer outra classe da aplicação.

# A Classe em Java

## ■ Tipo de classe

### □ Classes públicas

- O modificador **public** deve ser apresentado antes da palavra reservada **class**, nas declarações de classes públicas;
- Em Java, toda classe pública deve ser declarada em um arquivo com extensão “**.java**” com o mesmo nome da classe.

Ex:       arquivo: aviao.java  
          public class aviao {  
          ...  
          }

---

# Os Métodos em Java

- Organização de um Método {
  - cabeçalho (assinatura)
  - corpo
- Tipos de modificadores {
  - Visibilidade
  - Escopo
- Tipos de Métodos

# Os Métodos em Java

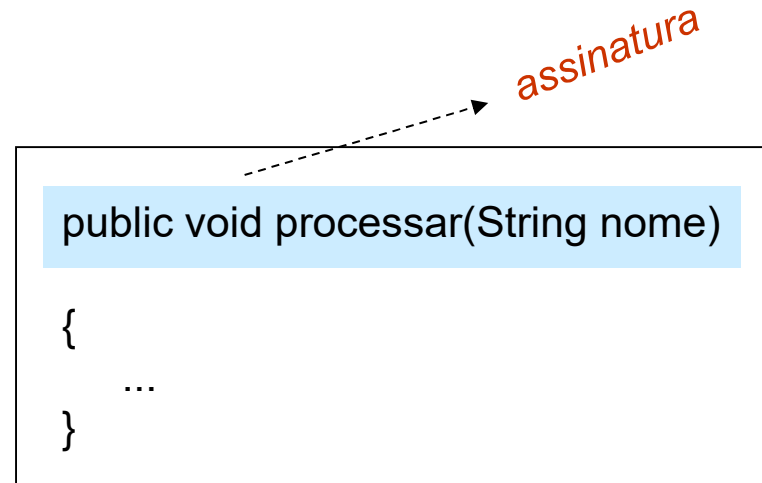
- Organização de um método

- Cabeçalho do método

- Modificadores de método;
    - Seu tipo de retorno;
    - O nome do método;
    - Seus parâmetros.

- Corpo do Método

- Declaração de variáveis locais
    - Código do método



```
public void processar(String nome)
{
    ...
}
```

*Obs.: As variáveis locais devem ser inicializadas.*

---

# Os Métodos em Java

## Modificador de Visibilidade

Indicam quais objetos podem acessar o método

- ❑ Modificador **public** indica que qualquer objeto pode acessar o método;
- ❑ O modificador **private** indica que somente o próprio objeto pode acessar o método;
- ❑ O modificador **protected** indica que somente o objeto e os descendentes de sua classe podem acessar o método.

# Os Métodos em Java

Modificador de Escopo: *static*

Indica a quem pertence o método

- ❑ Método de instância: manipula atributos de instância e classe;
- ❑ Método de classe: manipula somente atributos de classe

O método de classe pode ser chamado sem que se instancie um objeto dessa classe. Ex.:

```
Math.abs(-560);
```



---

# Os Métodos em Java

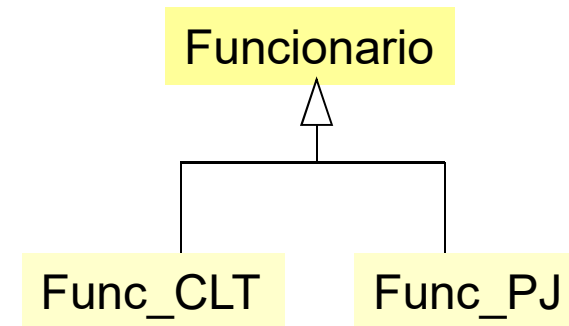
## **Classificação quanto a implementação de métodos:**

- Métodos abstratos
- Métodos concretos

# Os Métodos em Java

## Métodos Abstratos

Métodos sem implementação. Definem uma classe abstrata.



```
public abstract class Funcionario {
    private String nome;
    private String endereco;

    public Funcionario(String n, String end) {
        nome = n;
        endereco = end;
    }
    public void ImprimeNome() {
        System.out.println("Nome: " + nome);
    }
    public abstract double Calcula_Salario() ;
}
```

---

# Os Métodos em Java

## Métodos Concretos

Métodos com o objetivo de definir o comportamento de uma classe apresentado através do código implementado entre chaves.

```
public double calculaPreço(double quantidade )  
{  
    double total = quantidade * preçoUnit;  
    return total;  
}
```

---

# Os Métodos em Java

## **Tipos de Métodos**

- Métodos Construtores
  - Sobrecarga
  - Encadeamento
  
- Métodos Operacionais
  - Métodos de Acesso
  - Sobrecarga

# Métodos Construtores

Método chamado quando um objeto de uma classe é criado:

**new Ponto()**

Usado geralmente para inicializar os atributos da classe que foi instanciada. O método deve ter o mesmo nome da classe e pode ou não receber parâmetros. A classe pode ter um, nenhum ou mesmo muitos métodos construtores.

Ponto
Y : float
X : float
Imprime(): void

```
Ponto p1 = new Ponto();  
//p1 está em (0,0)
```

```
Ponto p2 = new Ponto(1,2);  
//P2 está em (1,2)
```

# Métodos Construtores

Nenhum Construtor definido pelo programador:

```
public class Ponto {  
    float X;  
    float Y;  
  
    public Ponto() {  
    }  
}
```

***O interpretador cria um construtor padrão***

Um Construtor definido pelo programador:

```
public class Ponto {  
    float X;  
    float Y;  
  
    public Ponto(float novoX, float novoY) {  
        X = novoX;  
        Y = novoY;  
    }  
}
```

```
public class Ponto {  
    float X;  
    float Y;  
  
    public Ponto(float X, float Y) {  
        this.X = X;  
        this.Y = Y;  
    }  
}
```

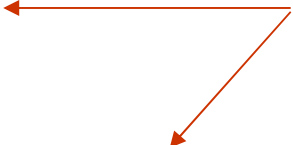
# Métodos Construtores

## Sobrecarga de Métodos Construtores

visa definir formas diferentes de criar um objeto

Exemplo:

```
class Ponto {  
    float x;  
    float y;  
    Ponto (float x) {  
        this.x = x;  
    }  
    Ponto (float x, float y) {  
        this.x = x;  
        this.y = y;  
    }  
}
```



Métodos Construtores  
com número de  
parâmetros diferentes

---

# Métodos Construtores

## Encadeamento de Métodos Construtores

- Um construtor pode chamar outro construtor, isto se chama “encadeamento de métodos construtores”
- Para isto é necessário usar a seguinte construção: `this(..)`. E esta chamada deverá ser a primeira linha do construtor.

***Obs.:** O `this` é uma palavra reservada que faz referência ao objeto dessa classe que está sendo manipulado no momento.*

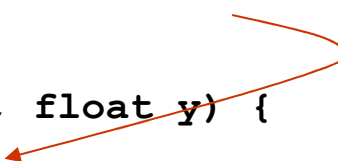


# Métodos Construtores

## Encadeamento de Métodos Construtores

Exemplo:

```
class Ponto {  
    float x;  
    float y;  
    Ponto () {  
        this(1,1);  
    }  
    Ponto (float x, float y) {  
        this.x=x;  
        this.y=y;  
    }  
}
```



# Métodos Operacionais

São todos os métodos exceto os métodos construtores da classe. Servem para definir o comportamento da classe.

## Funcionario

nome: String  
endereco: String

calculaSalario(): double  
ImprimeRel(): void

```
public class Funcionario {  
    String nome;  
    String endereco;  
  
    public void ImprimeRel() {  
        ...  
    }  
    public double calculaSalario() {  
        ...  
    }  
}
```

# Métodos Operacionais

## Métodos de Acesso

Dentro de métodos operacionais, encontramos um subgrupo chamada de métodos de acesso que servem para operar os atributos da classe que são privados.

O default dos programadores Java é utilizar os prefixos

get<nomeAtributo> para recuperar o valor do atributo e  
set<nomeAtributo> para alterar o valor do atributo.

### Funcionario

nome: String  
endereco: String

calculaSalario(): double

# Métodos Operacionais

## Métodos de Acesso

### Funcionario

nome: String  
endereco: String

calculaSalario(): double

```
public class Funcionario {  
    String nome;  
    String endereco;  
  
    public String getNome() {  
        return this.nome;  
    }  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
    public String getEndereco() {  
        return this.endereco;  
    }  
    public void setEndereco (String endereco) {  
        this. endereco = endereco;  
    }  
}
```

# Métodos Operacionais

## Sobrecarga de Métodos Operacionais

- Sobrecarregar um método significa definir dois ou mais métodos com o mesmo nome, porém com assinaturas diferentes.
- Assinatura difere normalmente no tipo parâmetros recebido.

```
public class Funcionario {  
    String nome;  
    String endereco;  
  
    public double calculaSalario (int numHoras) {  
        ...  
    }  
    public double calculaSalario () {  
        ...  
    }  
}
```

---

# Métodos Operacionais

## Sobrecarga de Métodos Operacionais

- A sobrecarga pode ser feita igualmente aos métodos operacionais.
- Uma boa prática é usar a sobrecarga somente em métodos que possuam a mesma funcionalidade.

# Métodos Operacionais

## Sobrecarga de Métodos Operacionais

Exemplo:

```
public class Ponto {  
    ...  
    public void mover (float dx, float dy) {  
        x += dx;  
        y += dy;  
    }  
    public void mover (int raio, float ang) {  
        x += raio*Math.cos(ang);  
        y += raio*Math.sin(ang);  
    }  
}
```

*A assinatura dos  
metodos é diferente*



---

# Os Atributos em Java

- Declaração do atributo
- Modificadores
- Tipos de Atributos
- Nome do Atributo
- Inicialização do atributo



---

# Os Atributos em Java

Declaração de um atributo é composta de:

- ❑ Modificadores de atributo
- ❑ Tipo do atributo
- ❑ Nome do Atributo
- ❑ Valor inicial

Exemplo:

```
private double velocidade = 0.0;
```

# Os Atributos em Java

## Modificadores de atributo: Visibilidade

Indicam quais objetos poderão manipular o atributo

- ❑ Modificador **public** indica que qualquer objeto pode manipular o atributo (fere a regra do encapsulamento);
- ❑ O modificador **private** indica que somente o próprio objeto pode manipular o atributo;
- ❑ O modificador **protected** indica que somente o objeto e descendentes de sua classe podem manipular o atributo.

Exemplo:

```
private double velocidade = 0.0;
```

# Os Atributos em Java

## Modificadores de atributo: Escopo

Indica se o atributo pertence ao objeto ou a classe

- ❑ Um atributo de objeto possui um valor distinto para cada objeto da classe;
- ❑ Um atributo de classe possui um valor único para todos os objetos da classe;
- ❑ Sempre que um objeto altera o valor de um atributo de classe, a alteração se reflete em todos os objetos da classe;
- ❑ O modificador **static** indica que o atributo pertence à classe;
- ❑ Por padrão, o atributo pertence ao objeto

Exemplo:

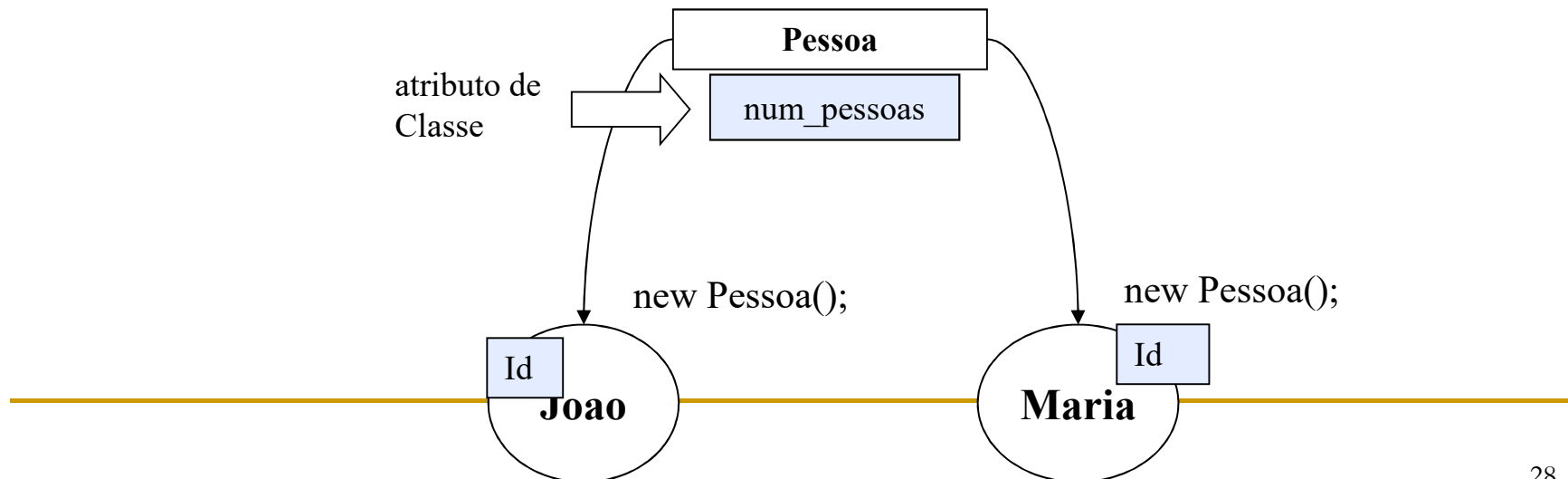
```
public static double taxa = 0.038;
```

# Os Atributos em Java

## Modificadores de atributo: Escopo

```
class Pessoa {  
    static int num_pessoa = 0;  
    int id;  
    Pessoa () {  
        id = num_pessoa;  
        num_pessoa++;  
    }  
    ...  
}
```

```
....  
Pessoa joao = new Pessoa ();  
joao.id = 0;  
System.out.println("Num. de pessoas: " + joao.num_pessoa);  
  
Pessoa ana = new Pessoa();  
ana.id = 1;  
System.out.println("Num. de pessoas: " + ana.num_pessoa);  
...
```



# Os Atributos em Java

## Modificadores de atributo: Redefinição

Indica se o valor do atributo pode ser alterado

- ❑ O modificador ***final*** indica que o valor do atributo não pode ser alterado após a primeira atribuição;
- ❑ Utilizado na definição de constantes;
- ❑ Por padrão, o valor do atributo sempre poderá ser alterado.

Exemplo:

```
public final double velocidade = 0.0;
```

---


# Os Atributos em Java

## Tipos do Atributo

- Os tipos da linguagem Java são utilizados:
  - ❑ Na declaração de atributos;
  - ❑ No tipo de retorno de um método;
  - ❑ Na lista de parâmetros de um método;
  - ❑ Nas variáveis locais de um método.
  
- A linguagem Java suporta
  - ❑ Tipos primitivos
  - ❑ Classes
  - ❑ TAD's Classes declaradas pelo desenvolvedor

# Os Atributos em Java

## Tipos do Atributo

- Primitivos
  - Classes
  - TAD
- 
- Inteiros
    - **byte**: 8 bits
    - **short**: 16 bits
    - **int**: 32 bits
    - **long**: 64 bit
  - Números reais
    - **float**: precisão simples 32 bits (IEEE 754 SPFP)
    - **double**: precisão dupla 64 bits (IEEE 754 DPFP)
  - Outros
    - **char**: caractere 16 bits (Unicode)
    - **boolean**: pode receber dois valores (true ou false)

---

# Os Atributos em Java

## Nome do Atributo

- Características do nome
  - ❑ O nome do atributo deve ser um identificador válido;
  - ❑ Uma classe não pode ter dois atributos com o mesmo nome;
  - ❑ Um atributo não pode ter o mesmo nome de sua classe.
  
- Java é case-sensitive
  - ❑ Letras minúsculas e maiúsculas são consideradas diferentes;
  - ❑ Normalmente, os nomes dos atributos são escritos com letras minúsculas. Se um nome possui diversas palavras, a primeira letra de cada palavra, a partir da segunda, é maiúscula (padrão mundialmente adotado).



---

# Os Atributos em Java

## Inicialização do Atributo

- Atributos podem ser opcionalmente inicializados em sua declaração;
- O valor do atributo é indicado ao lado de seu nome, precedido de um sinal de igual;
- Atributos finais
  - Somente recebem valores uma vez;
  - É comum a inicialização durante a declaração.