



# Sobrecarga

---

Prof<sup>a</sup>. Rachel Reis  
rachel@inf.ufpr.br



# Sobrecarga

---

- Tipos:
  - Sobrecarga de constructores
  - Sobrecarga de métodos



# Sobrecarga de Construtores

---

- Uma classe pode ter vários construtores sobrecarregados permitindo que objetos dessa classe sejam inicializados de diferentes maneiras.
- Para sobrecarregar construtores, basta fornecer múltiplas declarações de construtor com assinaturas diferentes.
- As regras para sobrecarga de construtores são:
  - A lista de parâmetros **tem** que mudar
  - O modificador de acesso **pode** mudar

## Quatro construtores

```
public class ContaBancaria{
    private String nomeT;
    private double saldo;

    // Construtor 1 - padrão
    public ContaBancaria(){}

    // Construtor 2
    public ContaBancaria(double saldo){
        this.setSaldo(saldo);
    }

    // Construtor 3
    public ContaBancaria(String nomeT){
        this.setNomeT(nomeT);
    }

    // Construtor 4 - completo
    public ContaBancaria(String nomeT, double saldo){
        this.setNomeT(nomeT);
        this.setSaldo(saldo);
    }
}
```

## Lista de parâmetros tem que mudar

```
public class ContaBancaria{
    private String nomeT;
    private double saldo;

    // Construtor 1 - padrão
    public ContaBancaria() {}

    // Construtor 2
    public ContaBancaria(double saldo) {
        this.setSaldo(saldo);
    }

    // Construtor 3
    public ContaBancaria(String nomeT) {
        this.setNomeT(nomeT);
    }

    // Construtor 4 - completo
    public ContaBancaria(String nomeT, double saldo) {
        this.setNomeT(nomeT);
        this.setSaldo(saldo);
    }
}
```

**Modificador de acesso  
pode mudar**

```
public class ContaBancaria{
    private String nomeT;
    private double saldo;

    // Construtor 1 - padrão
    public ContaBancaria(){}

    // Construtor 2
    public ContaBancaria(double saldo){
        this.setSaldo(saldo);
    }

    // Construtor 3
    public ContaBancaria(String nomeT){
        this.setNomeT(nomeT);
    }

    // Construtor 4 - completo
    public ContaBancaria(String nomeT, double saldo){
        this.setNomeT(nomeT);
        this.setSaldo(saldo);
    }
}
```

# Sobrecarga válida?

```
// Construtor 4 - completo
public ContaBancaria(String nomeT, double saldo) {
    this.setNomeT(nomeT);
    this.setSaldo(saldo);
}

// Construtor 5
public ContaBancaria(double saldo, String nomeT) {
    this.setSaldo(saldo);
    this.setNomeT(nomeT);
}
```

- A lista de parâmetros **tem** que mudar
- O modificador de acesso **pode** mudar

```
// Construtor 1 - padrão  
public ContaBancaria() {}
```

```
// Construtor 2  
public ContaBancaria(double saldo) {  
    this.setSaldo(saldo);  
}
```

```
// Construtor 3  
public ContaBancaria(String nomeT) {  
    this.setNomeT(nomeT);  
}
```

```
// Construtor 4 - completo  
public ContaBancaria(String nomeT, double saldo) {  
    this.setNomeT(nomeT);  
    this.setSaldo(saldo);  
}
```






# Construtores - referência **this**


---

- Um construtor pode ser chamado dentro de outro construtor utilizando a palavra-chave **this**.
- A instrução **this** deve vir sempre na primeira linha do construtor.

**Solução:**  
uso da referência this

```
// Construtor 1 - padrão  
public ContaBancaria() {}
```

```
// Construtor 2  
public ContaBancaria(double saldo) {  
 this("", saldo);  
}
```

```
// Construtor 3  
public ContaBancaria(String nomeT) {  
 this(nomeT, 0.0);  
}
```

```
// Construtor 4 - completo  
public ContaBancaria(String nomeT, double saldo) {  
    this.setNomeT(nomeT);  
    this.setSaldo(saldo);  
}
```

```
public class Funcionario{
```

```
...
```

```
public Funcionario(String nome, int mat, Data dataC,  
                    boolean estrangeiro){
```

```
...
```

```
}
```

```
}
```

```
public class FuncionarioH extends Funcionario{
```

```
...
```

```
// Construtor 1 - completo
```

```
public FuncionarioH(String nome, int mat, Data dataC,  
                    boolean estrangeiro, int valorH, int horasT){
```

```
    super(nome, mat, dataC, estrangeiro);
```

```
    this.setValorH(valorH);
```


```
    this.setHorasT(horasT);
```

```
}
```

```
}
```

```
public class FuncionarioH extends Funcionario{
    ...
    // Construtor 1 - completo
    public FuncionarioH(String nome, int mat, Data dataC,
        boolean estrangeiro, int valorH, int horasT)
    {
        super(nome, mat, dataC, estrangeiro);
        this.setValorH(valorH);
        this.setHorasT(horasT);
    }
    // Construtor 2
    public FuncionarioH(String nome, int mat, int valorH,
        int horasT)
    {
        super(nome, mat, NULL, False);
        this.setValorH(valorH);
        this.setHorasT(horasT);
    }
}
```

**Problema:**  
código repetido

```
public class FuncionarioH extends Funcionario{
    ...
    // Construtor 1 - completo
    public FuncionarioH(String nome, int mat, Data dataC,
        boolean estrangeiro, int valorH, int horasT)
    {
        super(nome, mat, dataC, estrangeiro);
        this.setValorH(valorH);
        this.setHorasT(horasT);
    }
    // construtor 2
    public FuncionarioH(String nome, int mat, int valorH,
        int horasT)
    {
 this(nome, mat, NULL, False, valorH, horasT);
    }
}
```

**Solução:**  
**uso da referência this**



# Palavra chave `this` - revisão

---

- Usada para invocar um construtor da classe.

```
public Carro(int numPortas, double valor)
{
    this.setNumPortas(numPortas);
    this.setValor(valor);
}
```

```
public Carro(double valor)
{
    ➡ this(0, valor);
}
```




# Palavra chave `this` - revisão

---

- Usada para referenciar atributos da classe.

```
public char getTipoCombustivel()  
{  
    return this.tipoCombustivel;  
}
```





# Palavra chave `this` - revisão

---

- Usada para referenciar métodos da classe.

```
public Carro(int numPortas)
{
    this.setNumPortas(numPortas);
}
```







# Sobrecarga de Métodos

---

- É a capacidade de possuir métodos diferentes com mesmo nome, mas com parâmetros diferentes.
- O interpretador determinará qual método deve ser invocado pelo tipo de parâmetro passado.



# Sobrecarga de Métodos

---

- As regras para sobrecarga de métodos são:
  - 1) O nome do método **tem** que ser o mesmo.
  - 2) A lista de parâmetros **tem** que mudar.
  - 3) O tipo de retorno **pode** mudar.
  - 4) O modificador de acesso **pode** mudar.



# Sobrecarga de Métodos

---

- Exemplo 1

```
public void imprimir(int i) { ... }  
  
public void imprimir(float f) { ... }  
  
public void imprimir(String s) { ... }
```

**nome do método:** tem que ser o mesmo.



# Sobrecarga de Métodos

---

- Exemplo 1

```
public void imprimir(int i) { ... }  
  
public void imprimir(float f) { ... }  
  
public void imprimir(String s) { ... }
```

**lista de parâmetros:** tem que mudar



# Sobrecarga de Métodos

---

- Exemplo 1

```
public void imprimir(int i) { ... }  
  
public void imprimir(float f) { ... }  
  
public void imprimir(String s) { ... }
```

**tipo de retorno:** pode mudar



# Sobrecarga de Métodos

---

- Exemplo 1

```
public void imprimir(int i) { ... }  
  
public void imprimir(float f) { ... }  
  
public void imprimir(String s) { ... }
```

**modificador de acesso:** pode mudar

## ■ Exemplo 2

```
public int soma(int x, int y){  
    return x + y;  
}
```

**nome do método**  
**tem que ser o mesmo**

```
public double soma(double x, double y){  
    return x + y;  
}
```

```
public String soma(String x, String y){  
    return x + y;  
}
```

## ■ Exemplo 2

```
public int soma(int x, int y){  
    return x + y;  
}
```

**lista de parâmetros  
tem que mudar**

```
public double soma(double x, double y){  
    return x + y;  
}
```

```
public String soma(String x, String y){  
    return x + y;  
}
```



## ■ Exemplo 2

```
public int soma(int x, int y){  
    return x + y;  
}
```

**tipo de retorno  
pode mudar**

```
public double soma(double x, double y){  
    return x + y;  
}
```

```
public String soma(String x, String y){  
    return x + y;  
}
```

## ■ Exemplo 2

```
public int soma(int x, int y){  
    return x + y;  
}
```

**modificador de acesso  
pode mudar**

```
public double soma(double x, double y){  
    return x + y;  
}
```

```
public String soma(String x, String y){  
    return x + y;  
}
```



# Referências

---

- Deitel, P. J.; Deitel, H. M. (2017). Java como programar. 10a edição. São Paulo: Pearson Prentice Hall.
- Barnes, D. J. (2009). Programação orientada a objetos com Java: uma introdução prática usando o BlueJ (4. ed.). São Paulo, SP: Prentice Hall.
- Boratti, I. C. (2007). Programação orientada a objetos em Java. Florianópolis, SC: Visual Books.