

Métodos

Thiago Leite e Carvalho

Engenheiro de Software, Professor, Escritor

Mais sobre mim

- Java, backend, docência, livros, artigos e cursos
- Mestre e Serpro
- O que me motiva?
- Pizzas e massas, cerveja e futebol



Mais sobre mim

- <https://www.linkedin.com/in/thiago-leite-e-carvalho-1b337b127/>
- <https://github.com/thiagoleitecarvalho>
- <https://github.com/tlcdio>

Objetivo do curso

Possibilitar que o aluno compreenda o que é um método, como criá-lo e utilizá-lo.

Percurso

Aula 1

Criação

Aula 2

Sobrecarga

Aula 3

Retornos

Requisitos

- ✓ Lógica de Programação
- ✓ Java
- ✓ IntelliJ

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (discord)

Aula 1: Criação

Métodos

Objetivos

1. Entender o que é um método
2. Saber como definir e utilizar métodos
3. Aplicar boas práticas em sua criação e uso



Conceituação

"É uma porção de código (sub-rotina) que é disponibilizada por uma classe. Este é executado quando é feita uma requisição a ele. São responsáveis por definir e realizar um determinado comportamento."

Criação

Padrão de definição:

**<?visibilidade?> <?tipo?> <?modificador?> retorno
nome (<?parâmetros?>) <?exceções?> corpo**

Criação

onde:

V: "public", "protected" ou "private"

T: concreto ou abstrato

M: "static" ou "final"

R: tipo de dado ou "void" // não retorna nada só

N: nome que é fornecido ao método// padrão

P: parâmetros que pode receber // se o método for usado definir dentro do parâmetro ou parâmetro vazio

E: exceções que pode lançar

C: código que possui ou vazio

Exemplos

```
public String getNome() { ... } // retorna um Nome  
public double calcularTotalNota() {...}  
public int verificarDistancia(int cordenada1, int cordenada2) {...}  
public abstract void executar() ; // corpo vazio do metodo  
public void alterarFabricante(Fabricante fabricante) { ... }  
public Relatorio gerarDadosAnaliticos(Cliente cliente,  
List<Compra> compras) {...} // como passar mais de um  
paramentro
```

```
public static R N(P) {...}
```

Utilização

Passa-se uma mensagem através de uma classe ou objeto.

`nome_da_classe.nome_do_metodo();` ou `nome_da_classe.nome_do_metodo(...);`

`nome_do_objeto.nome_do_metodo();` ou `nome_do_objeto.nome_do_metodo(...);`

`Math.random();` ou `Math.sqrt(4);`

`usuario.getEmail();` ou `usuario.alterarEndereco(
 endereco);`

Particularidades

- Assinatura: é a forma de identificar unicamente o método
Ass = nome + parâmetros

Método:

```
public double calcularTotalVenda(double  
precoItem1, double precoItem2, double precoItem3)  
{...}
```

Assinatura:

```
calcularTotalVenda(double precoItem1,  
double precoItem2, double precoItem3)
```

Particularidades

- Construtor e Destrutor: são métodos especiais usados na Orientação a Objetos.
- Mensagem: é o ato de solicitar ao método que o mesmo execute. Esta pode ser direcionada a um objeto ou a uma classe.

Boas práticas

- Nomes devem ser descritivos, mas curtos
- Notação camelo

```
verificarSaldo(); executarTransferencia(...); existeDebito();
```

- Deve possuir entre 80 e 120 linhas
- Evite lista de parâmetros longas
- Visibilidades adequadas

Exercitando

Cria uma aplicação que resolva as seguintes situações:

- Calcule as 4 operações básicas: soma, subtração, multiplicação e divisão. Sempre 2 valores devem ser passados.
- A partir da hora do dia, informe a mensagem adequada: Bom dia, Boa tarde e Boa noite.
- Calcule o valor final de um empréstimo, a partir do valor solicitado. Taxas e parcelas influenciam. Defina arbitrariamente as faixas que influenciam nos valores.

Exercitando

Observações:

- Tente ao máximo criar métodos que trabalhem sozinhos ou em conjunto
- Pode chamar um método dentro de outro
- Pode passar como parâmetro, a chamada de um outro método

Aula 2: Sobrecarga

Métodos

Objetivos

1. Entender o que é sobrecarregar um método
2. Saber como criar sobrecargas

Conceituação

"É a capacidade de definir métodos para diferentes contextos, mas preservando seu nome. "

Criação

Alterar a assinatura do método:

Ass = nome + parâmetros

```
converterParaInteiro(float f);  
converterParaInteiro(double d);  
converterParaInteiro(String s);  
converterParaInteiro(float f, RoundType rd);  
converterParaInteiro(double d, RoundType rd);  
converterParaInteiro(String s, RoundType rd);  
  
converterParaInteiro(RoundType rd, String s);  
converterParaInteiro();
```



Exemplos

<https://docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html>

void	<code>println()</code> Terminates the current line by writing the line separator string.
void	<code>println(boolean x)</code> Prints a boolean and then terminate the line.
void	<code>println(char x)</code> Prints a character and then terminate the line.
void	<code>println(char[] x)</code> Prints an array of characters and then terminate the line.
void	<code>println(double x)</code> Prints a double and then terminate the line.
void	<code>println(float x)</code> Prints a float and then terminate the line.
void	<code>println(int x)</code> Prints an integer and then terminate the line.
void	<code>println(long x)</code> Prints a long and then terminate the line.
void	<code>println(Object x)</code> Prints an Object and then terminate the line.
void	<code>println(String x)</code> Prints a String and then terminate the line.



Exemplos

<https://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

<code>static String</code>	<code>valueOf(boolean b)</code> Returns the string representation of the boolean argument.
<code>static String</code>	<code>valueOf(char c)</code> Returns the string representation of the char argument.
<code>static String</code>	<code>valueOf(char[] data)</code> Returns the string representation of the char array argument.
<code>static String</code>	<code>valueOf(char[] data, int offset, int count)</code> Returns the string representation of a specific subarray of the char array argument.
<code>static String</code>	<code>valueOf(double d)</code> Returns the string representation of the double argument.
<code>static String</code>	<code>valueOf(float f)</code> Returns the string representation of the float argument.
<code>static String</code>	<code>valueOf(int i)</code> Returns the string representation of the int argument.
<code>static String</code>	<code>valueOf(long l)</code> Returns the string representation of the long argument.
<code>static String</code>	<code>valueOf(Object obj)</code> Returns the string representation of the Object argument.

Curiosidade

Sobrecarga x Sobrescrita

Exercitando

Cria uma aplicação que calcula a área dos 3 quadriláteros notáveis: quadrado, retângulo e trapézio.

Obs: Use sobrecarga.

Aula 3: Retornos

Métodos

Objetivos

1. Entender como funcionam

Relembrando

- É uma instrução de interrupção
- Simbologia: return

Funcionamento

O método executa seu retorno quando:

- Completa todas suas instruções internas
- Chega a uma declaração explícita de retorno
- Lança uma exceção

Considerações

- O tipo de retorno do método é definido na sua criação e pode ser um tipo primitivo ou objeto;
- O tipo de dado do return deve ser compatível com o do método;
- Se o método for sem retorno(void), pode ou não ter um "return" para encerrar sua execução.

Exemplos

```
public String getMensagem() {  
    return "Ola!";  
}  
  
public double getJuros() {  
    return 2.36;  
}  
  
public int getParcelas() {  
    return 1.36f;  
}
```

```
public void setIdade(...) {  
    return 10;  
}  
  
public void executar() {  
    ...  
    return;  
    ....  
}
```

Exercitando

Recrie a aplicação que calcula a área dos 3 quadriláteros notáveis. Agora faça os métodos retornarem valores.

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Para saber mais

- <https://www.casadocodigo.com.br/products/livro-oo-conceitos>
- <https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html>

Para saber mais

- <https://docs.oracle.com/javase/tutorial/java/javaOO/returnvalue.html>
- <https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html>