

Lógica de Programação - Java

Aula 10 – Métodos e Funções sem e com argumentos



- É um conjunto de instruções que executa uma tarefa ou calcula um valor. Para usar uma função, você deve defini-la em algum lugar no escopo do qual você quiser chamá-la.
- É como se ensinássemos para o computador uma série de comandos que queremos executar chamando apenas por um nome. Isso é muito útil quando precisamos executar um certo código várias vezes durante a execução do nosso programa, então ao invés de repetir várias vezes esses códigos, criamos uma função para facilitar.
- No **java** chamamos essas funções de **métodos**
- Esses métodos são declarados dentro das **classes**

Por que métodos e funções?

- Simplificação do código
- Possibilidade de reaproveitamento
- Facilidade de manutenção

Dicas:

- Um método faz apenas uma coisa e bem
- Escreva pouco erre pouco: métodos com poucas linhas são fáceis de corrigir erros ou eles não possuem erros.
- Use nomes de métodos, parâmetros e variáveis significativos

Métodos sem argumentos e sem retorno

- Sintaxe:

```
void <nomeFuncao>() {  
    <instruções>  
}
```

- **<nomeFuncao>**: declara o nome do método
- **<instruções>**: código que será executado ao se chamar o método. Pode ser qualquer estrutura de código. Por exemplo: estrutura condicional, repetição etc.
- Exemplo:

```
static void saudacao() {  
    System.out.println("Seja bem vindo ao programa!!");  
    System.out.println("Essa é uma função que só executa alguma coisa");  
}
```

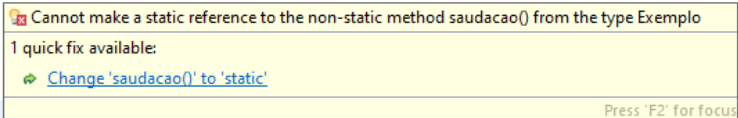
Chamar um método ou função

- Observe que nada é executado nesse exemplo, pois o método foi apenas declarado, ou seja, nós apenas ensinamos ao computador o que ele deve fazer.
- Para que ele execute os comandos, é necessário chamar o método, e fazemos isso apenas pelo nome do método seguido de parênteses.

```
public class Exemplo {  
  
    void saudacao() {  
        System.out.println("Seja bem vindo ao programa!!");  
        System.out.println("Essa é uma função que só executa alguma coisa");  
    }  
  
    public static void main(String[] args) {  
  
        saudacao();  
  
    }  
  
}
```

Opss....

```
public class Exemplo {  
    void saudacao() {  
        System.out.println("Seja bem vindo ao programa!!");  
        System.out.println("Essa é uma função que só executa alguma coisa");  
    }  
  
    public static void main(String[] args) {  
        saudacao();  
    }  
}
```



- Percebemos que o código dá um erro.
- O método **main()** é estático. O método estático não faz parte de um instanciamento de objeto, ele é funcional somente dentro da classe.
- Não podemos chamar um método que não é estático dentro de um método estático.
- Ou seja, colocaremos nosso método saudação como estático inserindo a palavra **static** antes de **void**.

Nosso código do exemplo ficará assim:

```
public class Exemplo {  
  
    static void saudacao() {  
        System.out.println("Seja bem vindo ao programa!!");  
        System.out.println("Essa é uma função que só executa alguma coisa");  
    }  
  
    public static void main(String[] args) {  
  
        saudacao();  
  
    }  
  
}
```

Incrementando o exemplo...

```
import java.util.Scanner;

public class Exemplo {

    static void saudacao() {
        System.out.println("Seja bem vindo ao programa!!");

        Scanner tec = new Scanner(System.in);

        System.out.print("Digite seu nome: ");
        String nome = tec.next();

        System.out.print("Digite sua idade: ");
        int idade = tec.nextInt();

        if (idade >= 18) {
            System.out.printf("Olá, %s! Você é maior de idade. \n", nome);
        } else {
            System.out.printf("Olá, %s! Você é menor de idade. \n", nome);
        }
    }

    public static void main(String[] args) {

        saudacao();

    }
}
```


Métodos com argumentos e sem retorno


- Sintaxe:

```
void <nomeFuncao>(<tipo> <nomeArg>, <tipo> <nomeArg2>) {  
    <instruções>  
}
```
- **<nomeFuncao>**: declara o nome do método
- **<instruções>**: código que será executado ao se chamar o método. Pode ser qualquer estrutura de código. Por exemplo: estrutura condicional, repetição etc.
- **<tipo>**: é o tipo do dado que receberá como parâmetro/argumento.
- **<nomeArg>**: nome utilizado para referenciar o dado que será recebido como parâmetro.

Métodos com argumentos e sem retorno

- Exemplo:

```
public class Exemplo {  
  
    static void saudacaoComArgs(String nome, int idade) {  
        System.out.println(nome);  
        System.out.println(idade);  
    }  
  
    public static void main(String[] args) {  
  
        saudacaoComArgs("Alexandre", 22);  
                        String nome    int idade  
    }  
}
```



Passamos os dados como parâmetros dentro dos parênteses, na ordem correta do método.

Melhorando o método **com** parâmetros

```
import java.util.Scanner;

public class Exemplo {

    static void saudacaoComArgs(String nome, int idade) {
        if (idade >= 18) {
            System.out.printf("Olá, %s! Você é maior de idade. \n", nome);
        } else {
            System.out.printf("Olá, %s! Você é menor de idade. \n", nome);
        }
    }

    public static void main(String[] args) {

        Scanner tec = new Scanner(System.in);

        System.out.print("Digite seu nome: ");
        String nomeDig = tec.next();

        System.out.print("Digite sua idade: ");
        int idadeDig = tec.nextInt();

        saudacaoComArgs(nomeDig, idadeDig);

    }

}
```

Desafios e exercícios

Métodos sem e com argumentos



Exercício 1

- Faça uma função com o nome **msgErro** que, quando chamada, exibe a seguinte mensagem na tela;
 - “Essa é uma mensagem padronizada de erro. Cuidado com o que está fazendo!!”

Exercício 2

- Faça um programa que tenha um método chamado `calcularArea()`, que receba as dimensões de um terreno retangular como argumento (largura e comprimento) e mostre a área do terreno.

Exercício 3

- Faça um método que receba como parâmetro a idade de uma pessoa.
- A função deve exibir se o voto da pessoa é:
 - Obrigatório
 - Proibido
 - Opcional

Exercício 4

- Faça um programa, com um método que necessite de três argumentos:
 - dois números e um sinal de operador matemático (+, -, * ou /).
- Ela deve fazer o cálculo indicado pelo operador usando os dois número passados.

Exercício 5

- Escreva um método para conversão de temperatura.
- Ela deve receber 2 argumentos:
 - um número
 - um caractere 'C', 'F' ou 'K', indicando que a temperatura está em Celsius, Fahrenheit ou Kelvin.
- A função deve exibir a temperatura nas 3 escalas.



Alguma Dúvida?

Entre em contato por e-mail ou via LinkedIn



[/alexandrerussi](#)