FIAP

Lógica de Programação - Java

Aula 11 – Métodos e Funções com retornos

Alexandre Russi Jr.

profalexandre.junior@fiap.com.br

Motivação dos retornos de um método

FIAP

- As variáveis definidas no interior de uma função **não podem ser acessadas de nenhum lugar fora da função**, porque a variável está definida apenas no escopo da função.
- Portanto, no exemplo da função abaixo, ela irá somar os números, apresentar na tela, mas eu não conseguirei acessar esse resultado de nenhum lugar fora da função.
- Logo, além da exibição, esse valor se torna "inútil" para mim, pois eu não consigo armazená-lo em nenhum outro lugar ou fazer qualquer outro cálculo com ele.

```
static void somar(int a, int b) {
    int soma = a + b;
    System.out.println(soma);
}
```

Exemplo

- Caso queira que ao final da execução a função "**devolva**" um valor, como um resultado, podemos usar a instrução `return`.

```
public class ExemploFunc {

    static int somar(int a, int b) {
        int soma = a + b;
        return soma;
    }

    public static void main(String[] args) {

        int valorRetorno = somar(5, 3);
        System.out.println(valorRetorno);

    }

}
```

Retorna o valor inteiro 8

Armazena o valor inteiro 8 retornado pela função `somar`

Método com argumentos e com retornos

- Sintaxe:

```
<tipoRetorno> <nomeFuncao>(<tipoArg> <nomeArg>) {
     <instruções>
     return <dadoRetorno>;
}
```

- `<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.
- `<tipoArg>:` é o tipo do dado que receberá como parâmetro/argumento.
- `<nomeArg>:` nome utilizado para referenciar o dado que será recebido como parâmetro.
- `<tipoRetorno>:` é o tipo de dado que será retornado pela função.
- `<dadoRetorno>:` é a informação que será retornado pelo método. Deve ser do mesmo tipo declarado e também pode estar armazenado em uma variável local.
- Obs.: **Não** é obrigatório uma função com retorno ter argumentos. Uma função com retorno pode NÃO ter parâmetros.

Extra: Atividade 1

FIAP

- Crie uma classe com o nome “ProgramaTop” com o método main() padrão.
- Crie uma classe com o nome “Operacoes” sem método main por padrão.
- Crie essas duas classes no **mesmo** pacote.
- Na classe Operacoes, crie um método de somar dois números.
- Na classe ProgramaTop tente chamar o método criado dentro da classe Operacoes

Extra: Atividade 1 – Resolução

```
public class Operacoes {

    public static int soma(int a, int b) {
        int s = a + b;
        return s;
    }

}
```

Extra: Atividade 1 – Resolução

```
public class ProgramaTop {

    public static void main(String[] args) {

        System.out.println("Soma: ");
        int soma = Operacoes.soma(12, 3);
        System.out.println(soma);

    }

}
```

Perceba...

- Criamos o nosso método **somar** como **public**, pois não é possível acessar um método que não seja público em outras classes do seu programa.
- Caso você troque **public** por **private** no seu método, você irá notar um erro ao chamar esse método na classe ProgramaTop.
- Agora conseguimos entender tudo sobre o método **main**:

```
public static void main(String[] args)
```

  - **public**: torna um método público, qualquer classe pode acessá-lo de fora.
  - **static**: torna o método estático, não é necessário transformar em objeto para utilizá-lo.
  - **void**: meu método não retorna valor.
  - **String[] args**: indica que o meu parâmetro **args** é um vetor de strings.

Desafios e exercícios

Métodos e Funções com retornos

Exercício 1

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

Exercício 2

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

Exercício 3

- 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.
- Retorne o resultado.

Exercício 4

- 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.
- O método deve exibir a temperatura nas 3 escalas.
- Também deve perguntar qual valor deseja retornar (Celsius, Fahrenheit ou Kelvin).

Exercício 5 – Desafio

- Crie uma classe com um nome qualquer com o método main() padrão.
- Crie uma classe com o nome "OperacoesMatematicas" sem método main por padrão.
- Crie essas duas classes no **mesmo** pacote.
- Na classe OperacoesMatematicas, crie um método que receba como parâmetro a operação (+, -, *, /) e dois números. Esse método deve chamar a função correspondente a sua operação e retornar o resultado.
- Além disso, na classe OperacoesMatematicas, você deve criar os métodos para cada operação. Esses métodos não podem ser acessados fora da classe OperacoesMatematicas
- Na classe com o método main(), chame o método da classe OperacoesMatematicas passando como parâmetro a operação desejada e dois números. Exiba o resultado.

Alguma Dúvida?

Entre em contato por e-mail ou via LinkedIn

/alexandrerussi