

# Linguagem de Programação

## MÉTODOS parte 1

Prof. Me. Humberto Zanetti



# Métodos

- O que são métodos?
- Descrevendo um método
- Tipos de métodos
- Chamadas em código

# Métodos

- **Sub-rotinas** ou **Subalgoritmos** são pequenos trechos de códigos (ou pequenos algoritmos) que possuem **uma função específica**, mas não resolvem o todo.
  - São uma parte da solução maior.
- Em algumas literaturas são conhecidos também por **Funções** ou **Módulos** (usa-se muito o termo “modularizar” o código)
- Em Java, chamamos esses trechos de código de **Métodos**.

# Métodos

- Nós usamos métodos:
  - Quando queremos que o código não fique muito complexo
  - Para proporcionar a reutilização de códigos e diminuição de re-trabalho
  - Para que através de uma análise *top-down*, possamos implementar a solução em partes, que podem ser testadas separadas e depois, unidas em uma solução única.

# Definindo um método

- A total compreensão do uso de métodos em Java só é possível com um estudo aprofundado de Orientação a Objetos.
- Aqui, vamos apenas aprender a definir, utilizar e concentrar em apenas uma classe.
- Por isso vamos apenas usar a definição de método **public** e **static**
  - Como estamos usando em **public static void main**

# Definindo um método

- O **public** e o **static** são definições ou assinaturas de métodos.
- **Public**
  - Define que o método é público, ou seja, acessível por qualquer objeto dentro do projeto.
- **Static**
  - São chamados de métodos da classe, que podem ser usados sem instanciar um objeto. Isso quer dizer, que podemos simplesmente chamar o método quando quiser, sem um objeto definido (apenas pelo nome do método)

# Definindo um método

- Basicamente, um método é definido da seguinte maneira:

```
public static tipo_retorno nome_do_método (argumentos) {  
    comando1;  
    comando2;  
    ...  
    comandoN;  
}
```

**DICA:** se quiser, podemos omitir o *public*, pois todo método por padrão é público!

# Retorno de um método

- Ao executar um método, é possível termos como um retorno, algum valor ao final da execução desse trecho.
  - Retorno significa “**retornar um valor à linha na qual a função foi chamada**”
- Temos 2 tipos: com ou sem retorno
  - **void**: quando definimos que o método não retornará valor.
  - **int, double, float, char, String...**: definimos qual é o tipo de valor poderá ser retornado.



# Exemplo: sem retorno

```
public static void teste() {  
    int x = 2;  
    System.out.println("Valor de x:" + x);  
}
```

Ao final da execução desse método, simplesmente mostrará o valor de x!

# Exemplo com retorno

```
public static int teste() {  
    int x = 2;  
    return x;  
}
```

**return** literalmente diz que o valor a sua frente será retornado!

Ao final da execução desse método, o valor de x será retornado à linha de código que a chamou!

# Chamando um método

```
public static void teste() {  
    int x = 2;  
    System.out.println("Valor de x: "+x) ;  
}
```

```
public static void main(String[] args) {  
    teste();  
}
```




**Chamar um método é essencialmente referenciá-lo pelo seu nome e argumentos, que vão dentro dos parênteses.**

**OBS.: Tanto o método teste() quanto o main() são declarados DENTRO da classe**


# Chamando um método

```
public static int teste() {  
    int x = 2;  
    return x;  
}
```



Tipo do valor a ser retornado

```
public static void main(String[] args) {  
    int valor;  
    valor = teste();  
    System.out.println("Valor de x: "+valor);  
}
```



Aqui, ao chamar o método, devemos ter em mente que ao final da execução um valor será retornado, com isso podemos armazenar em uma variável ou usar como parâmetro (para imprimir direto na tela, por exemplo)

# Utilizando argumentos

- Em muitos casos, o método, que tem uma função específica, manipula dados que são passados a ele no momento da chamada.
  - `nome_método (valor) ;`
- Esses dados podem ser vários e de tipos diferentes de dados.
  - `nome_método (valor1, valor2, ... ) ;`
- Para tais dados damos o nome de **argumentos** ou **parâmetros** ou **entradas do método**.

# Utilizando argumentos

```
static tipo_retorno nome_método (tipo arg1, tipo arg2, ...) {  
    comando1;  
    comando2;  
    ...  
    comandoN;  
}
```

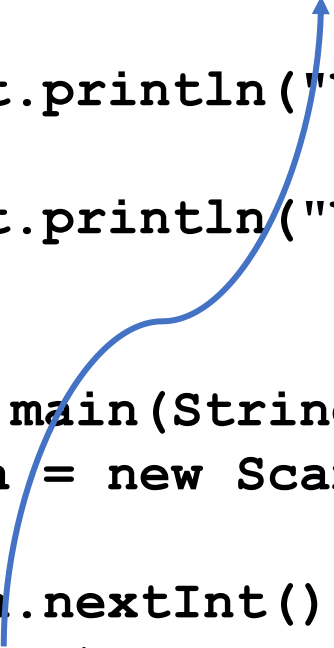
Aqui indicamos qual  
será o tipo de dado que  
deve ser recebido

Aqui definimos qual será o  
nome do argumento, para  
manipularmos no método.

Torna-se um processo  
similar ao de criar uma  
variável

# Exemplo


```
static void verificaPar(int v) {  
    if(v % 2 == 0)  
        System.out.println("Valor é par!");  
    else  
        System.out.println("Valor é ímpar!");  
}  
  
public static void main(String[] args) {  
    Scanner entrada = new Scanner(System.in);  
    int valor;  
    valor = entrada.nextInt();  
    verificaPar(valor);  
}
```



**Nesse contexto, o argumento “v” recebe o valor da variável “valor”.  
Seria similar ao processo de “valor = v”.**

# Exemplo

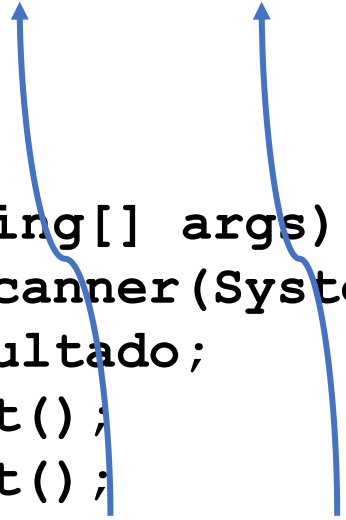
```
static double elevaQuadrado(double v) {  
    double quadrado;  
    quadrado = v * v;  
    return quadrado;  
}  
  
public static void main(String[] args) {  
    Scanner entrada = new Scanner(System.in);  
    double valor, resultado;  
    valor = entrada.nextDouble();  
    resultado = elevaQuadrado(valor);  
    System.out.println("Valor ao quadrado: "+resultado);  
}
```





# Vários argumentos

```
static int somaValores(int v1, int v2) {  
    return v1 + v2;  
}  
  
public static void main(String[] args) {  
    Scanner entrada = new Scanner(System.in);  
    int valor1, valor2, resultado;  
    valor1 = entrada.nextInt();  
    valor2 = entrada.nextInt();  
    resultado = somaValores(valor1, valor2);  
    System.out.println("Valor ao quadrado: "+resultado);  
}
```

Two blue arrows originate from the arguments 'valor1' and 'valor2' in the 'somaValores' call within the 'main' method. The arrow from 'valor1' points to the 'v1' parameter in the 'somaValores' method signature. The arrow from 'valor2' points to the 'v2' parameter in the 'somaValores' method signature.

É importante notar que a sequência de atribuição de argumentos depende da sequência dos dados passados na chamada. Nesse caso ficaria “**v1** = **valor1**” e “**v2** = **valor2**”.

Se houver tipos diferentes de argumentos, o cuidado deve ser maior!

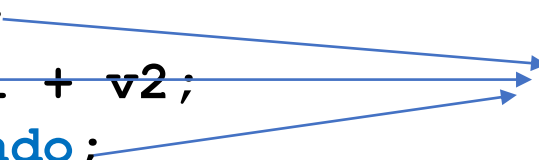
# Variáveis locais e globais

- Variáveis declaradas dentro de qualquer método (isso inclui também os argumentos) são chamadas de **variáveis locais**.
  - Local indica que a variável só é acessada pelo método, nunca fora dele.
- Já as variáveis globais, são variáveis que são declaradas para a classe, podendo ser acessada em qualquer método.

# Variáveis locais e globais

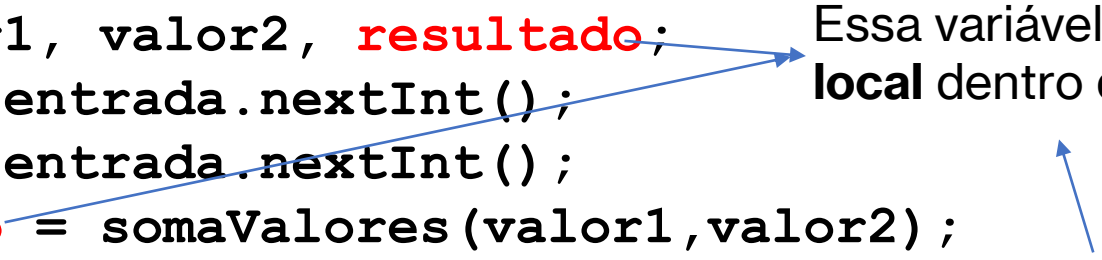
```
static int somaValores(int v1, int v2){  
    int resultado;  
    resultado = v1 + v2;  
    return resultado;  
}
```

Essa variável “resultado” é **local** dentro de somaValores()



```
public static void main(String[] args) {  
    Scanner entrada = new Scanner(System.in);  
    int valor1, valor2, resultado;  
    valor1 = entrada.nextInt();  
    valor2 = entrada.nextInt();  
    resultado = somaValores(valor1, valor2);  
    System.out.println("Valor ao quadrado: "+resultado);  
}
```

Essa variável “resultado” é **local** dentro do main()



# Variáveis locais e globais

```
static int resultado;
static void somaValores(int v1, int v2){
    resultado = v1+v2;
}
public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
    int valor1, valor2;
    valor1 = entrada.nextInt();
    valor2 = entrada.nextInt();
    somaValores(valor1, valor2);
    System.out.println("Soma dos valores: "+resultado);
}
```

Declaração de uma variável **global**

Essa variável global “resultado” pode ser acessada por qualquer método.

**CUIDADO!:** Tome o cuidado de não criar variáveis locais e globais com o mesmo nome!

# Exercício

1. Faça um programa que tenha um método para o cálculo do Índice de Massa Corpórea (IMC) de uma pessoa. O método deve receber o peso da pessoa (valor inteiro) e a altura (valor real). O método não terá retorno.

$$\text{IMC} = \text{peso} / (\text{altura} \times \text{altura}).$$

2. Refaça o exercícios anterior, agora com o método tendo retorno.

# Dúvidas!?

---

[humberto.zanetti@fatec.sp.gov.br](mailto:humberto.zanetti@fatec.sp.gov.br)