

Linguagem de Programação

MÉTODOS parte 2

Prof. Me. Humberto Zanetti



Métodos

- Passando métodos como argumentos
- Sobrecarga de métodos
- Métodos com vetores e matrizes
- Passagem de argumentos por referência

Passando argumentos entre métodos

```
static int elevaQuadrado(int v) {
    int quadrado;
    quadrado = v * v;
    return quadrado;
}
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, resultado;
    valor = entrada.nextInt();
    System.out.println("Valor ao quadrado: "+elevaQuadrado(valor));
    verificaPar(elevaQuadrado(valor));
}
```

Um método com retorno pode ser usado para passar esse retorno diretamente como argumento.

Sobrecarga de métodos

- Podemos declarar vários métodos com o mesmo nome, contanto que possuam listas de argumentos distintas.
- Os argumentos podem variar em número, tipo e ordem de declaração.
- Métodos que possuem o mesmo nome, mas argumentos diferentes são chamados de métodos sobrecarregados

Sobrecarga de métodos

```
static int somaValores(int v1, int v2){
    return v1 + v2;
}
static double somaValores(double v1, double v2){
    return v1 + v2;
}
static int somaValores(int v1, int v2, int v3){
    return v1 + v2 + v3;
}
public static void main(String[] args) {
    int resultadoInt;
    double resultadoDouble;
    resultadoInt = somaValores(2, 5);
    System.out.println("Soma de 2 inteiros: "+resultadoInt);
    resultadoDouble = somaValores(2.5, 5.75);
    System.out.println("Soma de 2 reais: "+resultadoDouble);
    resultadoInt = somaValores(2, 5, 3);
    System.out.println("Soma de 3 inteiros: "+resultadoInt);
}
```

Sobrecarga de métodos

- A ordem dos tipos de parâmetros é importante.
 - Caso haja tipos diferentes!
- As chamadas de método não podem ser distinguidas por tipo de retorno.
 - Somente pelos argumentos.
- Podem ter diferentes tipos de retorno se os métodos tiverem diferentes listas de parâmetro.
- Se diferenciam pela sua assinatura, que é uma combinação do **nome do método, tipos e ordem dos seus parâmetros**.

Exercício

1. Faça um programa que tenha um método potenciaNumero, que recebe um número qualquer inserido pelo usuário e eleva este número a um expoente, também definido pelo usuário pelo usuário. O usuário irá escolher antes se o número será número inteiro ou real para a base (ou seja, o número que será elevado). O expoente sempre será inteiro. Este método deverá ter retorno.

potenciaNumero (base, expoente)

↑
inteiro ou real


↑
inteiro

Métodos e vetores

- A diferença entre métodos com vetores está na **assinatura do método** (quando houver **retorno**) e no conceito de passagem de **argumentos por referência**.
 - A passagem por referência é indiferente se houver ou não retorno, está atrelado à passagem de argumentos.

Vetores como argumento

definimos um vetor sem dimensão como argumento de entrada



```
static void imprimeVetor(int[] v) {  
    for(int i=0; i<v.length; i++) {  
        System.out.println("Vetor["+i+"] : "+v[i]) ;  
    }  
}
```

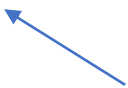
```
public static void main(String[] args) {  
    int[] numeros = {1,2,3,4,5,6,7,8,9,10};  
    imprimeVetor(numeros);  
}
```



passamos apenas o “nome” do vetor

Matrizes como argumento

definimos uma matriz sem dimensão como argumento de entrada



```
static void imprimeMatriz(int[] [] m) {  
    for(int i=0; i<m.length; i++){  
        for(int j=0; j<m[i].length; j++){  
            System.out.println("Matriz["+i+"] ["+j+"] : "+m[i][j]);  
        }  
    }  
}  
  
public static void main(String[] args) {  
    int[] [] numeros = {{1,2,3},{4,5,6},{7,8,9}};  
    imprimeMatriz(numeros);  
}
```



passamos apenas o “nome” da matriz

Vetores como retorno

```
static int[] preencheVetor(){
    Scanner entrada = new Scanner(System.in);
    int[] v = new int[5];
    for(int i=0; i<v.length; i++){
        v[i] = entrada.nextInt();
    }
    return v;
}

public static void main(String[] args) {
    int[] vetor;
    vetor = preencheVetor();
    for(int i=0; i<vetor.length; i++){
        System.out.println("Vetor["+i+"]:"+vetor[i]);
    }
}
```

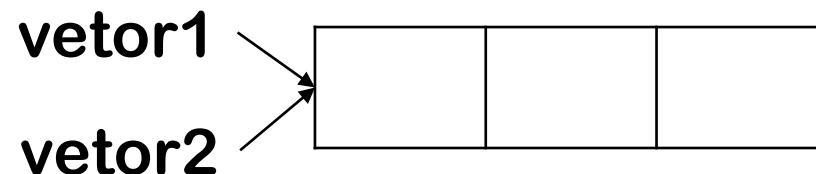
Matrizes como retorno

```
static int[][] preencheMatriz() {
    Scanner entrada = new Scanner(System.in);
    int[][] m = new int[2][2];
    for(int i=0; i<m.length; i++){
        for(int j=0; j<m[i].length; j++){
            m[i][j] = entrada.nextInt();
        }
    }
    return m;
}

public static void main(String[] args) {
    int[][] mat;
    mat = preencheMatriz();
    for(int i=0; i<mat.length; i++){
        for(int j=0; j<mat[i].length; j++){
            System.out.println("Matriz["+i+"]["+j+"]:"+mat[i][j]);
        }
    }
}
```

Passagem por referência

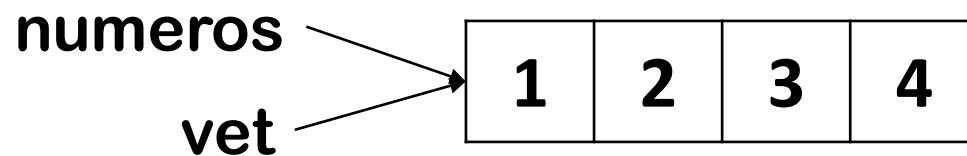
- Passagem por referência indica que não há uma cópia dos valores, mas sim uma passagem de uma nova referência para o mesmo conjunto de memória.
- Ou seja, podemos ter 2 nomes (ou mais) de vetores (ou matrizes) que apontam para o mesmo grupo de endereços de memórias reais.



Passagem por referência

```
static void imprimeVetor(int[] vet) {  
    for(int i=0; i<vet.length; i++) {  
        System.out.println("Vetor["+i+"]:"+vet[i]);  
    }  
}
```

```
public static void main(String[] args) {  
    int[] numeros = {1,2,3,4};  
    imprimeVetor(numeros);  
}
```



Exemplo – alterando na função

```
static void modificaVetor(int[] v){  
    for(int i=0; i<v.length; i++){  
        v[i] *= 2;  
    }  
}
```

Modifica o valor do vetor aqui!



```
public static void main(String[] args) {  
    int[] numeros = {1,2,3,4};  
    modificaVetor(numeros);  
    for(int i=0; i<numeros.length; i++){  
        System.out.println("Vetor["+i+"]:"+numeros[i]);  
    }  
}
```

É impresso aqui!



Saída:

Vetor[0]:2

Vetor[1]:4

Vetor[2]:6

Vetor[3]:8

Não acontece com valores simples...

```
static void mudaValor(int v) {  
    v = 0;  
    System.out.println("Valor na função:"+v);  
}  
  
public static void main(String[] args) {  
    Scanner entrada = new Scanner(System.in);  
    int valor;  
    System.out.println("Entre com o valor:");  
    valor = entrada.nextInt();  
    mudaValor(valor);  
    System.out.println("Valor no main:"+valor);  
}  
}
```

Saída:

Entre com o valor:2
Valor na função:0
Valor no main:2

... ou com Strings!

```
static void mudaString(String str) {  
    str = str.toUpperCase();  
    System.out.println("String na função:"+str);  
}  
  
public static void main(String[] args) {  
    Scanner entrada = new Scanner(System.in);  
    String palavra;  
    System.out.println("Entre com a string:");  
    palavra = entrada.nextLine();  
    mudaString(palavra);  
    System.out.println("String no main: "+palavra);  
}
```

Saída:

Entre com a string: Java
String na função: JAVA
String no main: Java

Exercício

2. Faça um método `ordenaVetor()` que implementa a solução de ordenação *BubbleSort*. Pode utilizar como exemplo a implementação disponível [aqui](#).

Dúvidas!?

humberto.zanetti@fatec.sp.gov.br