

# POO

## Array: vetor e matriz

Prof. Alcides Calsavara

PUCPR

# Vetores e Matrizes

Uma sequência finita de dados de certo tipo pode ser representada por uma única variável indexada, denominada **vetor** (ou **array**). Por exemplo, a sequência de números primos { **1, 2, 3, 5, 7, 11, 13** } pode ser armazenada em um vetor denominado **primo** com capacidade para armazenar **7** valores do tipo **int**, ou seja, o comprimento (ou tamanho) do vetor seria **7**. Em Java, esse vetor pode declarado e iniciado da seguinte forma:

```
int[ ] primo = {1, 2, 3, 5, 7, 11, 13};
```

O tipo da variável **primo** é **int[ ]**, o que significa que a variável é um vetor de valores do tipo **int**. A sequência { **1, 2, 3, 5, 7, 11, 13** } atribuída à variável **primo** não apenas define o seu valor inicial, mas também fixa o seu comprimento, nesse caso **7**, uma vez que a sequência possui **7** elementos.

primo

0	1	primo[0]
1	2	primo[1]
2	3	primo[2]
3	5	primo[3]
4	7	primo[4]
5	11	primo[5]
6	13	primo[6]

primo.length: 7

# Acesso aos elementos de um vetor

---

Cada um dos elementos da sequência pode ser acessado através do correspondente *índice*, sendo o primeiro elemento indexado por **0** (zero) e o último pelo comprimento da sequência menos 1.

```
int[ ] primo = {1, 2, 3, 5, 7, 11, 13};
```

```
System.out.println( primo[ 0 ] );
```

```
System.out.println( primo[ 3 ] );
```

```
System.out.println( primo[ 6 ] );
```

Execute o programa e confira os valores impressos.



# Impressão de um vetor

---

A impressão completa dos elementos de uma sequência pode ser feita por meio de um comando de repetição: o comando repetível seria o comando de impressão do *i*-ésimo elemento, variando-se *i* de 0 até o tamanho da sequência menos 1.

```
int[ ] primo = {1, 2, 3, 5, 7, 11, 13};
```

```
for (int i = 0; i < primo.length; i++)  
    System.out.println( primo[ i ] );
```

# Modificação de um elemento de um vetor

Qualquer elemento de um vetor pode ter o valor modificado por meio de um comando de atribuição.

```
int[ ] primo = {1, 2, 3, 5, 7, 11, 13};
```

```
primo[ 3 ] = 10 ; // modifica o quarto elemento do vetor
```

```
for (int i = 0; i < primo.length; i++)  
    System.out.println( primo[i] );
```

primo

0	1	primo[0]
1	2	primo[1]
2	3	primo[2]
3	5	primo[3]
4	7	primo[4]
5	11	primo[5]
6	13	primo[6]

primo.length: 7

primo

0	1	primo[0]
1	2	primo[1]
2	3	primo[2]
3	10	primo[3]
4	7	primo[4]
5	11	primo[5]
6	13	primo[6]

primo.length: 7



# Expressão com elementos de um vetor

Os elementos de uma sequência podem ser indexados e, assim, usados em qualquer expressão compatível com o tipo dos dados na sequência.

```
int[ ] primo = {1, 2, 3, 5, 7, 11, 13};
```

```
int soma = 0;
```

```
for (int i = 0; i < primo.length; i++)
```

```
    if ( primo[ i ] > 5 )
```

```
        soma = soma + primo[ i ] ;
```

```
System.out.println( soma );
```

# Criação de um vetor

Um vetor também pode ser iniciado por meio do comando **new**, deixando os seus elementos com o valor padrão do sistema, de acordo com o tipo definido.

```
int[ ] primo = new int[ 7 ] ;
```

```
primo[ 0 ] = 1;
```

```
primo[ 1 ] = 2;
```

```
primo[ 2 ] = 3;
```

```
primo[ 3 ] = 5;
```

```
for (int i = 0; i < primo.length ; i++)
```

```
System.out.println( primo[ i ] );
```

primo

0	1	primo[0]
1	2	primo[1]
2	3	primo[2]
3	5	primo[3]
4	0	primo[4]
5	0	primo[5]
6	0	primo[6]

primo.length: 7

# Criação de um vetor

O comprimento de um vetor pode ser definido de forma dinâmica, por meio do valor de um variável.

```
System.out.print( "Digite a quantidade de filósofos: " );
```

```
int n = teclado.nextInt();
```

```
String[ ] equipe = new String[ n ];
```

```
for (int i = 0; i < n; i++)  
{
```

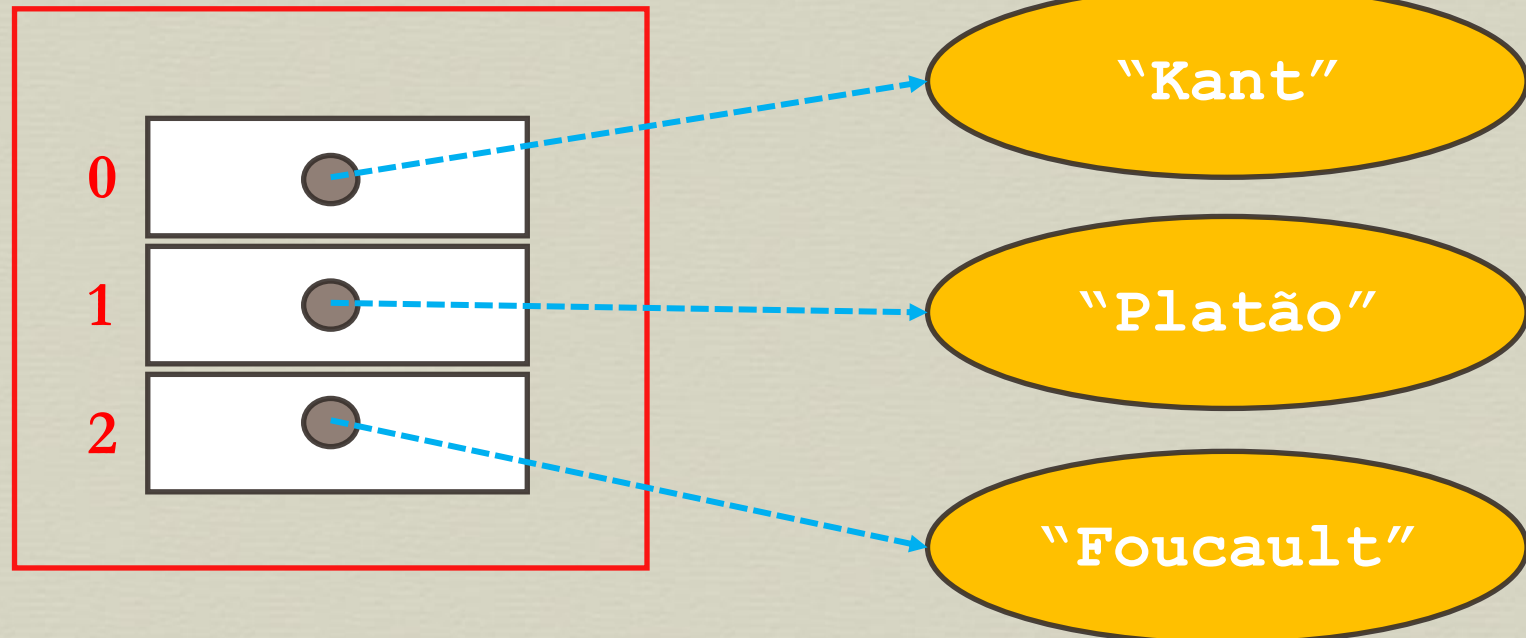
```
    System.out.print( "Digite o nome do filósofo " + i + ": " );
```

```
    equipe[ i ] = teclado.next() ;
```

```
}
```

Digite a quantidade de filósofos: 3  
Digite o nome do filósofo 0: Kant  
Digite o nome do filósofo 1: Platão  
Digite o nome do filósofo 2: Foucault

equipe





# Matriz

Os elementos de uma sequência também podem ser sequências, ou seja, o tipo dos elementos de um vetor pode ser um vetor de dados de certo tipo. Nesse caso, diz-se que a variável é um *vetor bidimensional* ou, simplesmente, que a variável é uma *matriz*.

```
int [ ] [ ] vendas = { { 40, 32, 30 }, { 20, 26, 38 } };
```

```
int [ ] [ ] vendas = { { 40, 32, 30 }, { 20, 26, 38 } } ;
```

**vendas**

	0	1	2
0	40	32	30
1	20	26	38

`vendas [0] [0] :` 40

`vendas [1] [0] :` 20

`vendas [0] [1] :` 32

`vendas [1] [1] :` 26

`vendas [0] [2] :` 30

`vendas [1] [2] :` 38

# Impressão de uma matriz

```
int [ ] [ ] vendas = { {40, 32, 30}, {20, 26, 38} };
```

```
for (int i = 0; i < 2; i++)  
{  
    System.out.print( "Vendedor " + i + ": ");
```

```
    for ( int j = 0; j < 3; j++)  
    {  
        System.out.print( vendas[ i ] [ j ] );  
        System.out.print( " " );  
    }
```

```
    System.out.println();
```

```
}
```

# Dimensões variáveis

Os comprimentos das sequências em uma sequência de sequências não precisam ser iguais.

```
int[ ][ ] vendas = { {40, 32, 30} , {20, 26} , {10, 4, 18, 40} };
```

3

2

4

```
int[ ][ ] vendas = { {40, 32, 30}, {20, 26}, {10, 4, 18, 40} };
```

	0	1	2	3
vendas [0]	40	32	30	
vendas [1]	20	26		
vendas [2]	10	4	18	40

Comprimento de cada linha da matriz:

```
vendas [0] .length: 3
```

```
vendas [1] .length: 2
```

```
vendas [2] .length: 4
```

Número de linhas na matriz:

```
vendas .length: 3
```



# Impressão de matriz com dimensões variáveis

---

```
int[ ][ ] vendas = { {40, 32, 30}, {20, 26}, {10, 4, 18, 40} };

for (int i = 0; i < vendas.length ; i++)
{
    System.out.print( "Vendedor " + i + ": ");

    for ( int j = 0; j < vendas[ i ].length ; j++)
    {
        System.out.print( vendas[ i ][ j ] );
        System.out.print( " ");
    }

    System.out.println();
}
```

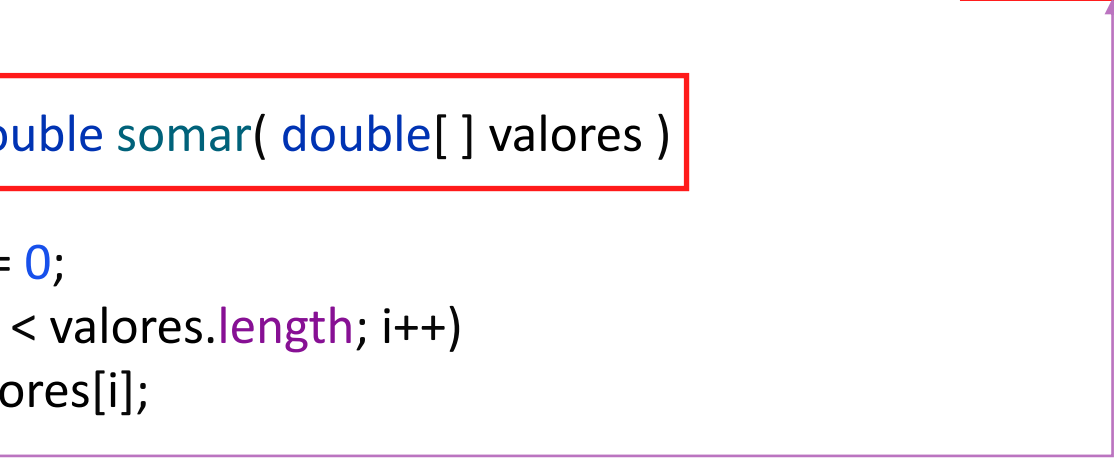
# Dimensões dinâmicas

As dimensões de uma matriz também podem ser definidas de forma dinâmica. No exemplo a seguir, a matriz **T** de valores do tipo **double** é criada com **m** linhas e **n** colunas, sendo que **m** e **n** são variáveis do tipo **int**.

```
int m = 4;  
int n = 5;  
double [ ] [ ] T = new double [m] [n] ;  
  
for (int i = 0; i < m ; i++)  
    for (int j = 0; j < n ; j++)  
  
        T[ i ] [ j ] = 1.0;
```

# Vetor como parâmetro

```
public class Parametro {  
    public static void main(String[] args)  
    {  
        double[] notas = {6.5, 8.3, 9.0, 5.6, 7.7, 5.5, 6.8, 9.5, 4.0, 7.5};  
  
        System.out.println( String.format("Quantidade de notas: %d", notas.length) );  
  
        System.out.println( String.format("Soma das notas: %.2f", somar(notas)) );  
    }  
  
    private static double somar( double[] valores )  
    {  
        double total = 0;  
        for (int i = 0; i < valores.length; i++)  
            total += valores[i];  
        return total;  
    }  
}
```



A diagram consisting of two red rectangular boxes. The first box is located around the `somar(notas)` expression in the `main` method. The second box is located around the `return total;` statement in the `somar` method. A purple arrow originates from the bottom of the second box and points upwards to the bottom of the first box, illustrating the flow of control and data from the `somar` method back to the `main` method.

```
public class Parametro {  
    public static void main(String[] args)  
    {  
        double[ ] notas = {6.5, 8.3, 9.0, 5.6, 7.7, 5.5, 6.8, 9.5, 4.0, 7.5};  
        double[ ] bonus = {1.0, 0.5, 0.0, 2.0, 0.0, 0.5, 1.0, 0.0, 1.0, 0.5};  
  
        adicionar(notas, bonus);  
  
        for (int i = 0; i < notas.length; i++)  
            System.out.print(" " + notas[i]);  
    }  
    private static void adicionar(double[] valores, double[] extra)  
    {  
        // Pré-condição: valores.length == extra.length  
        for (int i = 0; i < valores.length; i++)  
            valores[i] += extra[i];  
    }  
}
```

# Vetor como retorno

```
public class Retorno {  
    public static void main(String[ ] args)  
    {  
        int[ ] idades = { 7, 12, 8, 23, 18, 40};  
        int[ ] inverso = inverter(idades);  
        for (int i = 0; i < inverso.length; i++)  
            System.out.print(" " + inverso[i]);  
    }  
    private static int[ ] inverter(int[ ] valores)  
    {  
        int comprimento = valores.length;  
        int[ ] invertido = new int[ comprimento ];  
        for (int i = 0, j = comprimento-1; i < comprimento; i++, j--)  
            invertido[j] = valores[i];  
        return invertido;  
    }  
}
```