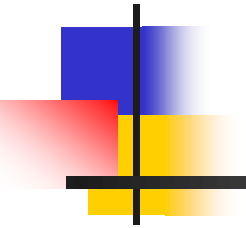
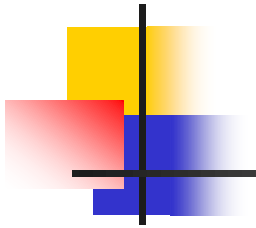


Arrays em Java

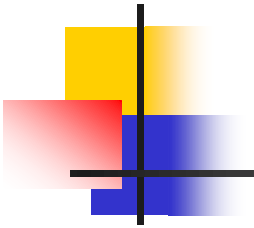
Profª. Rachel Reis

rachel@inf.ufpr.br



Array

- É um grupo de variáveis que contém valores que são todos do mesmo tipo.
- Arrays são objetos.
- Tipo do array:
  - Tipo primitivo: *int*, *char*, *double*, *float*, etc.
  - Tipo por referência (nome de classes):
    - Classes já existentes: *String*, *Math*, etc.
    - Classes novas: *Funcionario*, *Carro*, etc.



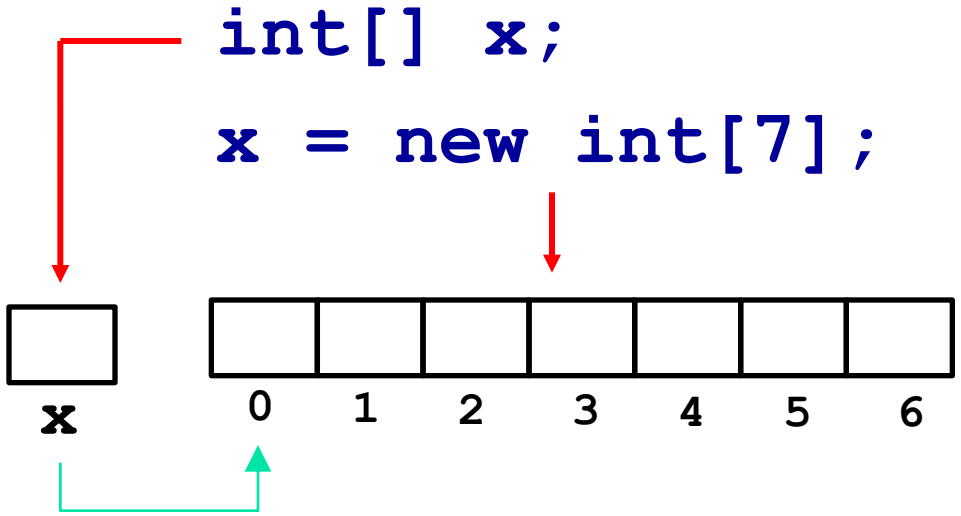
Array

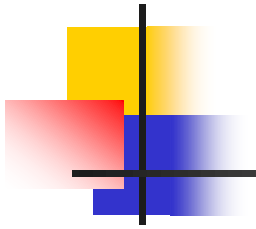
- Para criar arrays usa-se a palavra-chave ***new***, especificando o tipo dos elementos do array e o número de elementos.

- Tipos de array:
  - Unidimensional → vetores
  - Muldimensional → matrizes

Array Unidimensional

- Ex. 1 – tipo primitivo:





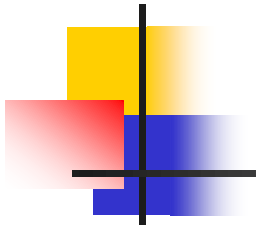
Array Unidimensional

- Ex. 1 – tipo primitivo:

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

- Ex. 2 – tipo primitivo:

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



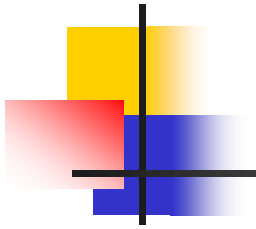
Array Unidimensional

- Ex. 3 – tipo por referência:

```
String[] y = new String[22];
```

- Ex. 4 – tipo por referência:

```
String[] y;
y = new String [22];
```



Array Unidimensional

- Um programa pode criar e inicializar um *array*
- Ex. 5:

```
int[] n = {10, 20, 30, 40};
```

- Neste caso não é necessário utilizar a palavra-chave *new*

Array Unidimensional

- Declaração do vetor de inteiros num1:

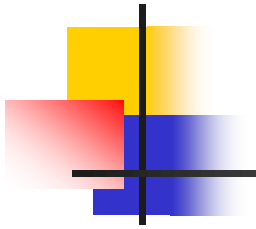
```
int num1[];
```

ou

```
int[] num1;
```

- Posição do abre e fecha colchetes não interfe:

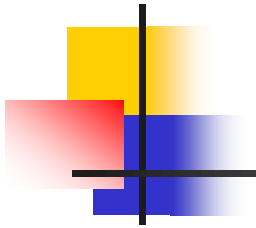
```
public static void main(String[] args){}
public static void main(String args[]){}
```



Atividade 1

- Existe diferença entre as declarações abaixo?

```
a) int[] num1, num2;

b) int num1, num2[];
```



Exemplo 1

- Crie um programa em Java que some os elementos de um array formado por cinco elementos do tipo inteiro.

```
public class PrincipalSomaVetor
{
    public static void main(String[ ] args)
    {
        int[ ] valores = {37, 42, 75, 13, 94};
        int i, soma = 0;

        for (i = 0; i < valores.length; i++)
        {
            soma = soma + valores[i];
        }
       System.out.println(“Soma: ” + soma);
    }
}
```

```
public class PrincipalSomaVetor
{
    public static void main(String[ ] args)
    {
```

/*

- A classe PrincipalSomaVetor contém o método main() que é responsável por iniciar o programa Java.

- Lembre que nesse caso, o código fonte deve ser salvo com o nome **PrincipalSomaVetor.java**

*/

```
    }
}
```

```
public class PrincipalSomaVetor
{
    public static void main(String[ ] args)
    {
        int[ ] valores = {37, 42, 75, 13, 94};

        /*
            Declaração e inicialização do vetor com cinco
            números inteiros.
        */



    }
}
```

```
public class PrincipalSomaVetor
{
    public static void main(String[ ] args)
    {
        int[ ] valores = {37, 42, 75, 13, 94};
        int i, soma = 0;

        /*
            - Declaração da variável de controle i.

            - Declaração e inicialização da variável soma.
        */

    }
}
```

```
public class PrincipalSomaVetor
{
    public static void main(String[ ] args)
    {
        int[ ] valores = {37, 42, 75, 13, 94};
        int i, soma = 0;

        for (i = 0; i < valores.length; i++)
        {

        }

    }
}
```

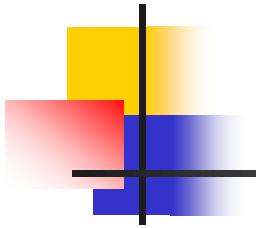
```
/*
    A instrução valores.length informa
    o número de elementos do array.
*/
```

```
public class PrincipalSomaVetor
{
    public static void main(String[ ] args)
    {
        int[ ] valores = {37, 42, 75, 13, 94};
        int i, soma = 0;

        for (i = 0; i < valores.length; i++)
        {
            soma = soma + valores[i];
        }

    }
}
```

```
/*
    Somando os elementos do array.
*/
```

```
public class PrincipalSomaVetor
{
    public static void main(String[ ] args)
    {
        int[ ] valores = {37, 42, 75, 13, 94};
        int i, soma = 0;

        for (i = 0; i < valores.length; i++)
        {
            soma = soma + valores[i];
        }
       System.out.println("Soma: " + soma);
    }
}
```

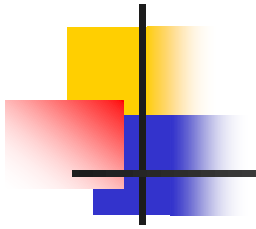
```
/*
    Exibindo o resultado da soma.
*/
```



Atividade 2

- Altere o código anterior para que o usuário do programa forneça os 5 números inteiros.

→ Dica: utilize a classe Scanner



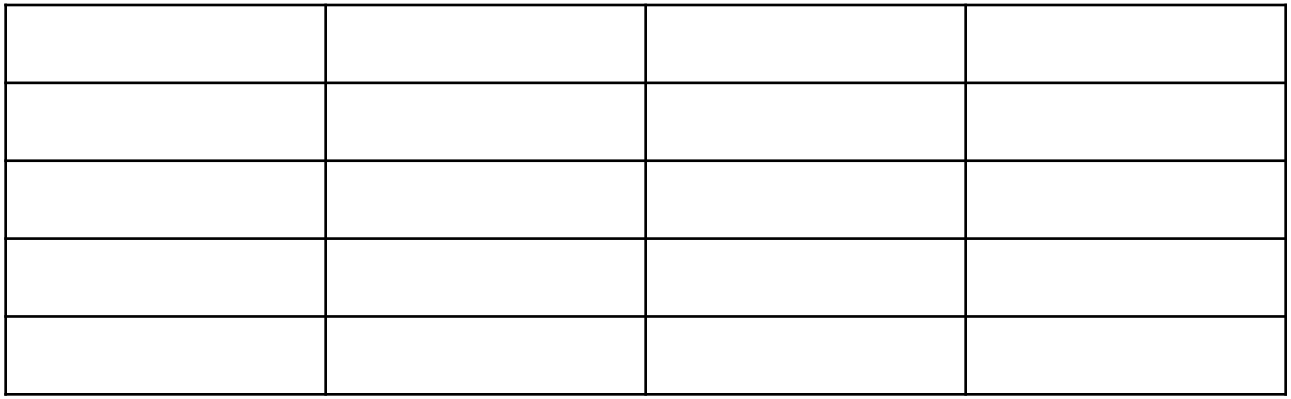
Array Multidimensional

- São estruturas de dados de duas ou mais dimensões.

- Exemplo:
  - Array bidimensional → matriz

Array Bidimensional

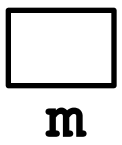
- Usados para representar tabelas (linhas e colunas)



Array Bidimensional

- Ex. 1:

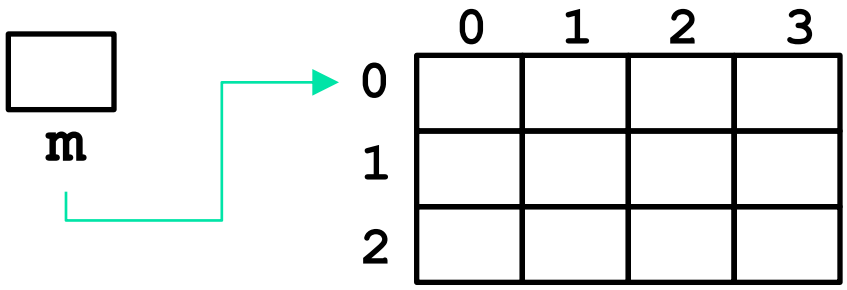
```
int[][] m;
m = new int [3][4];
```



Array Bidimensional

- Ex. 1:

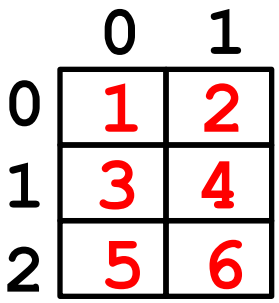
```
int[][] m;
m = new int [3][4];
```



Array Bidimensional

- Ex. 2:

```
int[][] m = {{1,2}, {3,4}, {5,6}};
```



Exemplo 2

- Imprimir os elementos de um array bidimensional inicializado com a instrução abaixo:

```
int[][] m = {{1,2}, {3,4}, {5, 6}};
```

```
public class PrincipalMatriz{
	public static void main(String[ ] args)
	{
		int[][] m = {{1, 2}, {3, 4}, {5, 6}};
		int lin, col;

		for (lin = 0; lin < m.length; lin++)
		{
			for(col = 0; col < m[lin].length; col++)
			{
				System.out.print(m[lin][col] + “ ”);
			 }
			System.out.print(“’\n ”);
		 }
	 }
}
```

```
public class PrincipalMatriz{
    public static void main(String[ ] args)
    {

/*
    - A classe PrincipalMatriz contém o método main() que é responsável por iniciar o programa Java.

    - Lembre que nesse caso, o código fonte deve ser salvo com o nome PrincipalMatriz.java
*/


    }
}
```

```
public class PrincipalMatriz{
      public static void main(String[ ] args)
      {
            int[ ][ ] m = {{1, 2}, {3, 4}, {5, 6}};
```

Declaração e inicialização da matriz m.

1	2
3	4
5	6

```
      }
}
```

```
public class PrincipalMatriz{
    public static void main(String[ ] args)
    {
        int[][] m = {{1, 2}, {3, 4}, {5, 6}};
        int lin, col;

        /*
            Declaração de duas variáveis de controle.
        */

    }
}
```

```
public class PrincipalMatriz{
    public static void main(String[ ] args)
    {
        int[][] m = {{1, 2}, {3, 4}, {5, 6}};
        int lin, col;

        for (lin = 0; lin < m.length; lin++)
        {
            /*
                Percorrendo as linhas da matriz.
            */


        }
    }
}
```

```
public class PrincipalMatriz{
    public static void main(String[ ] args)
    {
        int[][] m = {{1, 2}, {3, 4}, {5, 6}};
        int lin, col;

        for (lin = 0; lin < m.length; lin++)
        {
            for(col = 0; col < m[lin].length; col++)
            {

            }

        }
    }
}
```

```
/*
    - Percorrendo as colunas da matriz.
    - A instrução m[lin].length informa
      o número de colunas da linha lin.
*/
```

```
public class PrincipalMatriz{
      public static void main(String[ ] args)
      {
            int[][] m = {{1, 2}, {3, 4}, {5, 6}};
            int lin, col;

            for (lin = 0; lin < m.length; lin++)
            {
                  for(col = 0; col < m[lin].length; col++)
                  {
                        System.out.print(m[lin][col] + " ");
                   }

             }
       }
}
```

```
/*
      Exibe um elemento da matriz.
*/
```

```
public class PrincipalMatriz{
      public static void main(String[ ] args)
      {
           int[][] m = {{1, 2}, {3, 4}, {5, 6}};
           int lin, col;

           for (lin = 0; lin < m.length; lin++)
           {
                for(col = 0; col < m[lin].length; col++){
                     System.out.print(m[lin][col] + " ");
                 }
                System.out.print("'\n ");
            }
       }
}
```

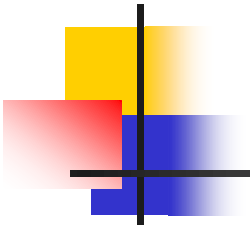
```
/*
     Pula a linha após exibir os
     elementos de uma linha da matriz.
*/
```

Passando arrays para Métodos

- Para passar um array (ex.: vetor, matriz) para um método, especifique o nome do array sem nenhum colchete.

- Ex. 1 – vetor:

```
int[] v = {1, 2, 3, 4, 5};
imprimirDadosVet(v);
```

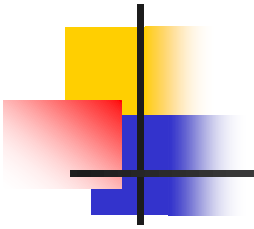


Passando arrays para Métodos

- Ex. 2 – matriz:

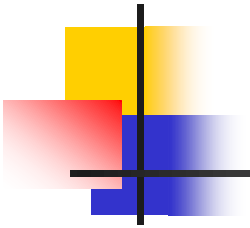
```
int[][] m = {{1,2}, {3,4}, {5, 6}};
imprimirDadosMat(m);
```

Em Java, quando passamos um array para um método, a passagem é por valor ou por referência??



Passando arrays para Métodos

- O Java não permite ao programador escolher entre passar por valor ou passar por referência
- Todos os argumentos são passados **por valor**
  - Cópias de valores primitivos
  - Cópias de referência para objetos



Referências

- Deitel, P. J.; Deitel, H. M. (2017). Java como programar. 10a edição. São Paulo: Pearson Prentice Hall.

- Barnes, D. J. (2009). Programação orientada a objetos com Java: uma introdução prática usando o BlueJ (4. ed.). São Paulo, SP: Prentice Hall.

- Boratti, I. C. (2007). Programação orientada a objetos em Java. Florianópolis, SC: Visual Books.