

VEM SER

Módulo 01 - Java + OO

Aula 02 - Laços, matrizes e *Arrays*

Arrays



O que é um *Array*?

Também chamado de vetor, em programação, um *array* é uma estrutura de dados que permite **armazenar** um **conjunto** de elementos do **mesmo tipo**.

Ele é usado para agrupar valores relacionados sob um único nome e é acessado por meio de um índice numérico.

Arrays

O que isso quer dizer?

arrayDeComidas = [ ,  ,  , ]





Como nós podemos acessar a carne?

Arrays

arrayDeComidas[2] = 

Arrays

O que isso quer dizer?

arrayDeComidas = [ ,  ,  , ]

0 1 2 3



Índice ou posição do elemento

Lembrando: o tamanho do *array*
corresponde ao seu número de elementos

Portanto,

arrayDeComidas = [ ,  ,  , ]

O tamanho (*length*) desse array é 4.

Mas e no Java?

Vamos supor que temos o seguinte *array*:

```
int[] numeros = new int[3]
```

Mas e no Java?

Aqui temos o tipo do *array*, que no caso é um *array* de inteiros:

The image shows the Java array type for integers, 'int[]', displayed in white text on a blue rectangular background.

Mas e no Java?

Aqui temos o nome do *array*:

numeros

Mas e no Java?

Aqui temos o operador de atribuição:



Mas e no Java?

E por fim estamos alocando espaço na memória:

```
new int[3];
```

Agora vamos atribuir valores, certo?

numeros



{2, 3, 5}



Iniciamos um *array* **vazio**!

E se a gente quiser um *array* que não seja vazio?

```
int[] impares = {1, 3, 5, 7, 9}
```

Vamos praticar!



Crie um *array* com 3 valores inteiros (quaisquer) e:

- Calcule a soma dos valores;
- Calcule a média dos valores.

Matrices

O que são matrizes?

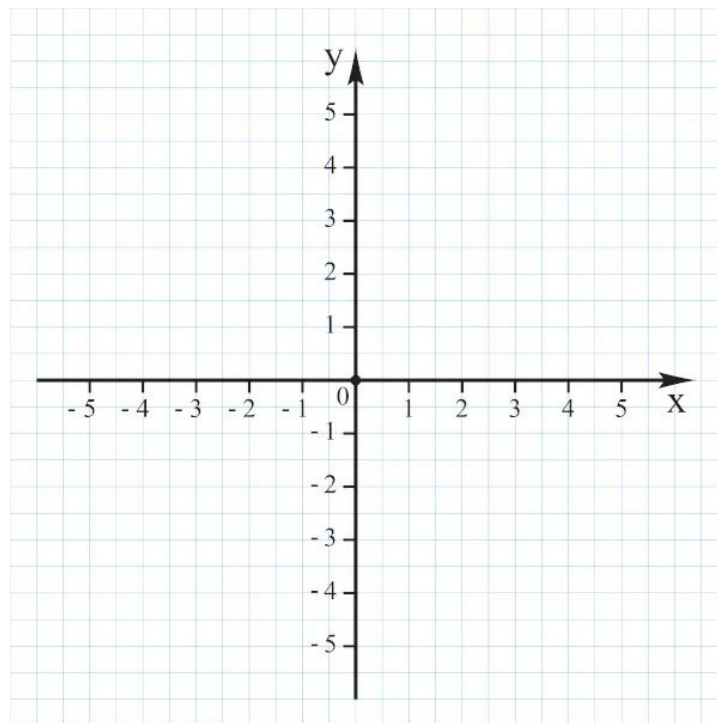
Em programação, matrizes são estruturas de dados **bidimensionais** que armazenam elementos em uma **grade** ou **tabela** retangular.

Uma matriz é essencialmente uma coleção de elementos de mesmo tipo organizados em linhas e colunas.

Ou seja, é um vetor com duas dimensões.

Vamos lembrar a matemática...

Lembram do plano cartesiano?



Vamos relembrar a matemática...

Lembram do plano cartesiano?

Um ponto era representado por uma coordenada no eixo X e outra coordenada no eixo Y

Mas o que isso tem a ver?

Voltando pra programação...

Imaginem uma matriz como sendo uma "tabela":

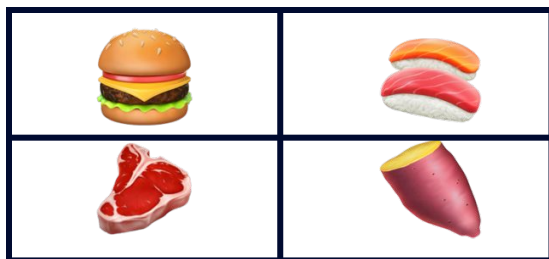


	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							
8							
9							

- No plano cartesiano, um ponto é representado por duas coordenadas, uma no eixo X e outra no eixo Y.
- Já numa tabela, uma célula é representada por uma linha (número) e por uma coluna (letra).
- E numa matriz, cada **elemento** é representado por uma linha (número) e por uma coluna (número).

Matrizes

Vamos criar uma matriz de comidas?



Lembrando que, assim como nos *arrays*,
nas matrizes todos os elementos deve ser
do mesmo tipo.

Mas e no Java?

```
int[][] numeros = new int[2][2];
```

Usamos dois [] pois são duas dimensões



Dizemos que essa matriz é 2×2 (dois por dois)



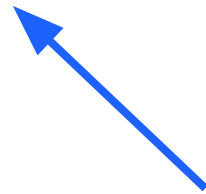
Mas e no Java?

```
int[][] numeros = new int[2][2];
```

Lembrando que assim estamos declarando uma matriz vazia (todos os elementos serão zero)!

E se quisermos uma matriz que não seja vazia?

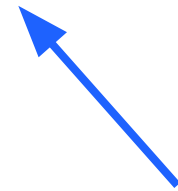
```
int[][] numeros = {{2, 4}, {8, 16}};
```



Cada {} representa uma linha

E se quisermos uma matriz que não seja vazia?

```
int[][] numeros = {{2, 4}, {8, 16}};
```



Cada elemento representa uma coluna

Vamos praticar!



Crie uma matriz 2x2, atribua valores e faça as seguintes operações:

- Calcule a soma de todos os valores;
- Calcule a soma dos valores da primeira linha;
- Calcule a soma dos valores da segunda linha;
- Calcule (soma dos valores da primeira linha - soma dos valores da segunda linha).

Estruturas de repetição (ou laços, ou
loops)

Estruturas de repetição

“São estruturas que permitem a repetição controlada de comandos.” SOUZA et al (2019, p. 160)

Existem dois tipos principais:

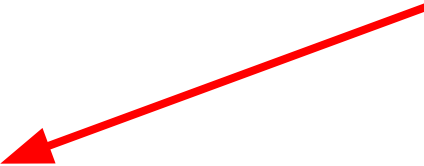
- REPITA-ATÉ (*for*);
- ENQUANTO-FAÇA (*while*).

For

```
for (int i = 0; i < 3; i++) {  
    <comandos>;  
}
```


For

Variável de controle



```
for (int i = 0; i < 3; i++) {  
    <comandos>;  
}
```

For

Condição de parada



```
for (int i = 0; i < 3; i++) {  
    <comandos>;  
}
```

For

Incremento da variável de controle

```
for (int i = 0; i < 3; i++) {  
    <comandos>;  
}
```

For

```
int[] valores = { 5, 10, 15, 20, 25, 30 };
```

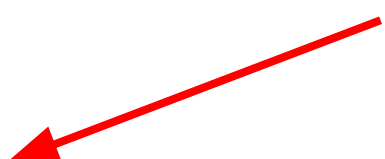
```
for (int i = 0; i < valores.length; i++) {  
    System.out.println(valores[i]);  
}
```

Variação do *for*: *foreach*

```
for (int valor : valores) {  
    <comandos>;  
}
```

Variação do *for*: *foreach*

Variável com o elemento



```
for (int valor : valores) {  
    <comandos>;  
}
```

Variação do *for*: *foreach*

Array / lista que será percorrida



```
for (int valor : valores) {  
    <comandos>;  
}
```

Variação do *for*: *foreach*

```
int[] valores = { 5, 10, 15, 20, 25, 30 };
```

```
for (int valor : valores) {  
    System.out.println(valor);  
}
```


Vamos praticar!



Crie um vetor de inteiros com quantidade X de elementos inteiros (solicitada ao usuário) e para cada posição desse vetor, preencha com entradas do usuário.

- Solicite ao usuário a quantidade de elementos desejada;
- Preencha o vetor;
- Calcule a média dos valores do vetor.

While

```
while (<condição>){  
    <comandos>;  
}
```

While

Ir  se repetir enquanto a condi  o for verdadeira



```
while (<condi  o>){  
    <comandos>;  
}
```

While

```
int numero = 0;
```

```
while (numero < 3) {  
    System.out.println(numero);  
    numero++;  
}
```

Vamos praticar!



Crie um programa que solicite e imprima palavras ao usuário, até que ele digite "fim"

- Peça uma palavra
- Imprima a palavra
- Se a palavra for = "fim" pare de pedir as palavras e encerre o programa



Let's *Tech Up Together*