



# **METODOLOGIA E LINGUAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETO**

## **LINGUAGEM DE PROGRAMAÇÃO II**

**Dr<sup>a</sup>. Alana Morais**  
**[alanamm.prof@gmail.com](mailto:alanamm.prof@gmail.com)**



**VAMOS VOLTAR PARA AS  
CLASSES DAS AULAS  
ANTERIORES!!!**

# COLEÇÕES

- Coleções de tamanho fixo:
  - Vetor.
  - Matrizes.
- Coleções de tamanho indeterminado:
  - ArrayList
- Outros Tipos de estruturas de armazenamento
  - HashMap



# ROTEIRO

- **Coleções de tamanho fixo:**
  - **Vetor.**
  - **Matrizes.**
- Coleções de tamanho indeterminado:
  - ArrayList
- Outros Tipos de estruturas de armazenamento
  - HashMap





# VETORES E MATRIZES

# ARRAY

- Utilizado para armazenar e manipular uma lista de dados de forma mais eficiente em uma variável.
- Este tipo de variável é chamada de **Array**.
- Um **Array** armazena múltiplos itens de dados do mesmo tipo em um bloco contínuo de memória, dividido-o em certa quantidade de casas.



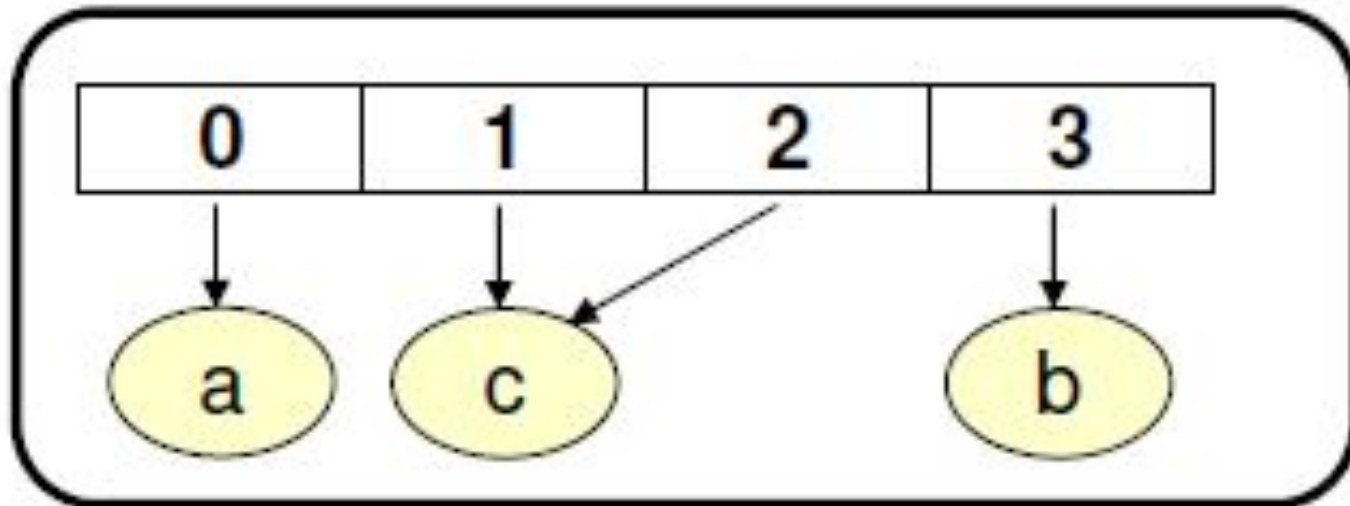
# ARRAY

- Vetores:
  - Em Java um vetor é um objeto, mesmo quando for composto por tipos primitivos.
  - Quando um vetor é criado, ele possui “métodos” e campos de dados como qualquer outro objeto.
- Matrizes:
  - Arrays multidimensionais.
  - Vetor de vetores.



# ARRAY - VETORES

- Uma coleção de objetos *indexada por  $0, 1, \dots, N-1$  com tamanho fixo.*





# ARRAY - VETORES

- Ações:
  - **Declaração.**
  - **Especificar Tamanho (Criar Array).**
  - **Adicionar valores.**



# ARRAY - VETORES

- Ações:
  - **Declaração.**
  - **Especificar Tamanho (Criar Array).**
  - **Adicionar valores.**



# ARRAY - VETORES

- **Declaração**

- Escreve-se o tipo de dado seguido por colchetes e por um identificador

`int [ ] ages; ou int ages[ ];`

- Pode declarar arrays de todos os tipos, primitivos ou objetos.

- Exemplo:

ages



int [ ]



# ARRAY - VETORES

- Ações:
  - Declaração.
  - **Especificar Tamanho (Criar Array).**
  - Adicionar valores.



# CRIAR ARRAY - VETORES

- Criar o array e especificar seu tamanho com um parâmetro no construtor
  - Escrever a palavra-chave new, definir o tipo de dado seguido por colchetes contendo a quantidade de elementos do array:

```
int ages[ ]; // declaração
```

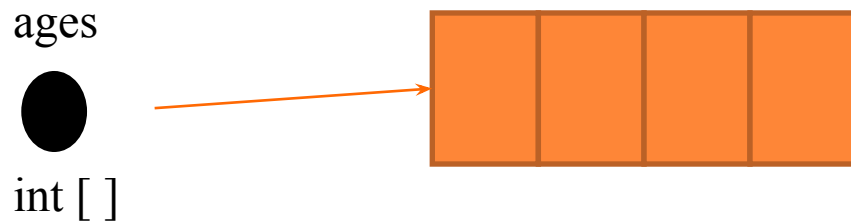
```
ages = new int[100]; //construindo um objeto
```

```
int ages[ ] = new int[100]; // declarando e construindo  
um objeto
```



# CRIAR ARRAY - VETORES

- Exemplo:
  - `int ages[ ] = new int[4];`



# ARRAY - VETORES

- Ações:
  - Declaração.
  - Especificar Tamanho (Criar Array).
  - **Adicionar valores.**



# ADICIONANDO VALORES AO ARRAY

- `int [ ] arr = new int [3];`  
    `arr [0] = 1;`  
    `arr [1] = 23;`  
    `arr [2] = 3;`
- `int arr[ ] = {1, 23, 3};`

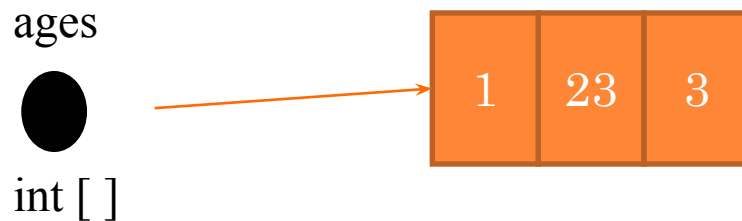




# CRIAR ARRAY - VETORES

□ Exemplo:

● `int ages[ ] = {1, 23, 3};`



# EXEMPLO

```
int [ ] i;
```

```
Pessoa [ ] p;
```

```
i = new int [20];
```

```
p= new Pessoa[100];
```

**ou**

```
Pessoa [ ] p = new Pessoa[100];
```

```
int [ ] i = new int[20];
```



# EXEMPLO

```
Pessoa [ ] p = new Pessoa[100];
```

```
int [ ] i = new int[20];
```

```
for (i = 0; i < 20; i++)
```

```
{
```

```
    p[i] = new Pessoa( "NomePessoa");
```

```
    //instanciando cada um dos objeto
```

```
}
```



# EXERCÍCIO

- Modele um **funcionário**. Ele deve ter um identificador (int), identificador do setor (int), salario(int), RG (int) e um valor (boolean) que indique se o **funcionário** ainda está na empresa no momento ou se já foi mandado embora.
  - Crie o método **bonifica** que aumenta o salário do funcionário de acordo com o parâmetro passado.
  - Crie o método **demite**, que não recebe parâmetro algum, só modifica o valor boolean indicado que o funcionário não trabalha mais aqui.
- Crie uma classe **Teste** para testar e armazenar 30 funcionários.



# ROTEIRO

- **Coleções de tamanho fixo:**
  - **Vetor.**
  - **Matrizes.**
- Coleções de tamanho indeterminado:
  - ArrayList
- Outros Tipos de estruturas de armazenamento
  - HashMap



# ROTEIRO

- **Coleções de tamanho fixo:**
  - **Vetor.**
  - **Matrizes.**
- Coleções de tamanho indeterminado:
  - ArrayList
- Outros Tipos de estruturas de armazenamento
  - HashMap



# ARRAY - MATRIZES

- São implementados como arrays dentro de arrays.

30	1	4	-4
1	10	43	5
0	1	74	9



30	1	4	-4
----	---	---	----

1	10	43	5
---	----	----	---

0	1	74	9
---	---	----	---



# ARRAY - MATRIZES

- Ações:
  - **Declaração.**
  - **Especificar Tamanho (Criar Array).**
  - **Adicionar valores.**





# ARRAY - MATRIZES

- Ações:
  - **Declaração.**
  - **Especificar Tamanho (Criar Array).**
  - **Adicionar valores.**



# ARRAY - MATRIZES

- **Declaração**

- Semelhante ao vetor:

**int[ ][ ] twoD;** ou

**int twoD [ ][ ];**

- Pode declarar arrays de todos os tipos, primitivos ou objetos.

- **Exemplo:**

twoD



int [ ][ ]



# ARRAY - MATRIZES

- Ações:
  - Declaração.
  - **Especificar Tamanho (Criar Array).**
  - Adicionar valores.



# CRIAR ARRAY - MATRIZES

- Criar o array e especificar seu tamanho com um parâmetro no construtor

```
int twoD[ ][ ]; // declaração
```

```
twoD = new int[2][2]; //construindo um objeto
```

ou

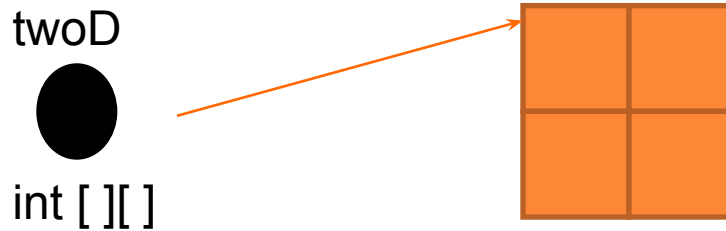
```
int twoD[ ][ ] = new int[2][2]; // declarando e  
construindo um objeto
```



# CRIAR ARRAY - MATRIZES

- Exemplo:

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



# ARRAY - MATRIZES

- Ações:
  - Declaração.
  - Especificar Tamanho (Criar Array).
  - **Adicionar valores.**



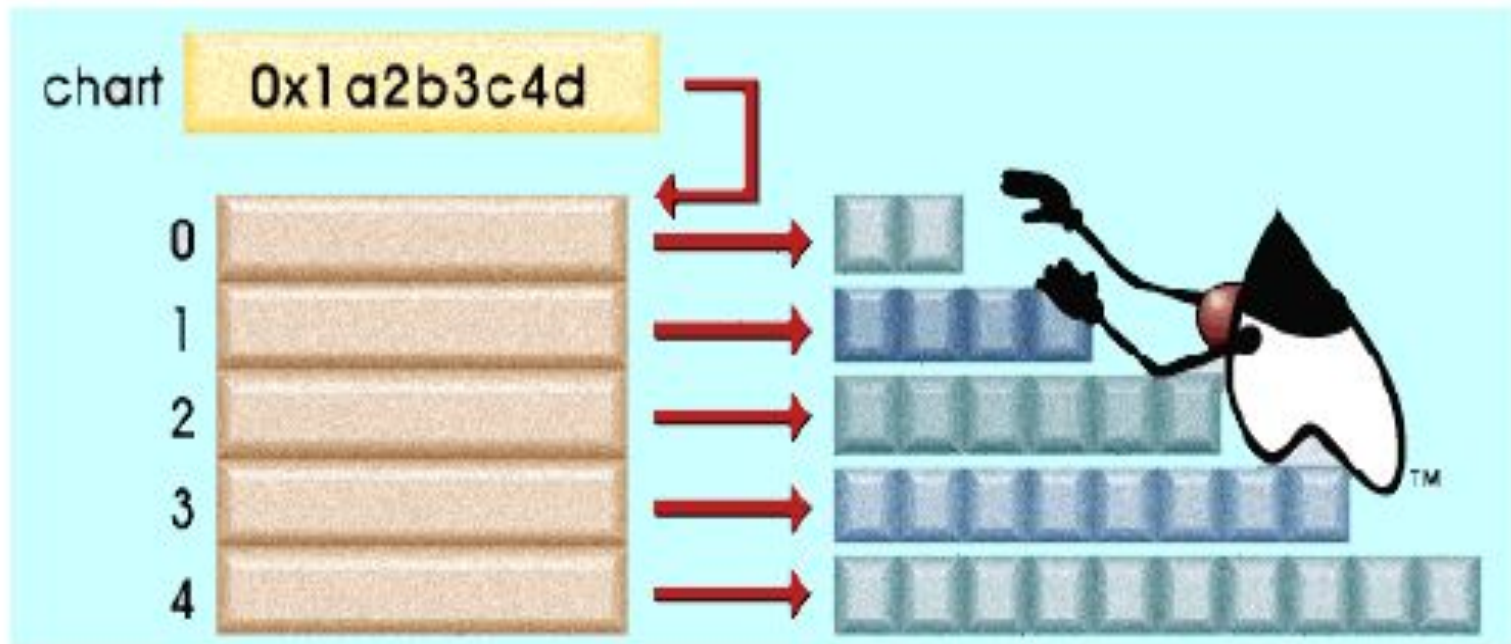
# ADICIONANDO VALORES AO ARRAY

- `String[ ][ ] dogs = {  
    { "terry", "brown" },  
    { "Kristin", "white" },  
    { "toby", "gray"},  
    { "fido", "black"}  
};`
- Como este array ao final das inserções?



# ADICIONANDO VALORES AO ARRAY

- Java permite criar matrizes não retangulares.





# ADICIONANDO VALORES AO ARRAY

- `String[ ][ ] dogs = {  
    { "terry"},  
    { "Kristin", "white" },  
    { "toby", "gray", "larry"},  
    { "fido", "black"}  
};`



# EXEMPLO

```
int [ ] i;
```

```
Pessoa [ ] p;
```

```
i = new int [20];
```

```
p= new Pessoa[100];
```

**ou**

```
Pessoa [ ] p = new Pessoa[100];
```

```
int [ ] i = new int[20];
```

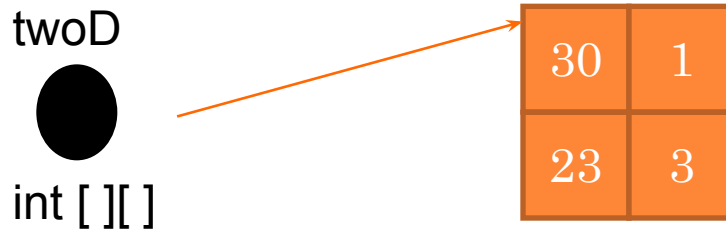


# CRIAR ARRAY - MATRIZES

- Exemplo:

```
int [ ][ ] twoD = {  
    {30,1},  
    {23,3}  
};
```

twoD  
●  
int [ ][ ]



A diagram illustrating the memory layout of the 2D array. A black circle representing the variable 'twoD' has an orange arrow pointing to the top-left cell of a 2x2 orange matrix. The matrix contains the values 30, 1, 23, and 3.

30	1
23	3



# EXERCÍCIO

- Implemente uma exemplo de matriz identidade.



DÚVIDAS ?

