

# Linguagem de Programação Orientada a Objetos I

Arrays em Java

Prof. Tales Bitelo Viegas

<https://fb.com/ProfessorTalesViegas>

The bottom of the slide features a decorative graphic consisting of several overlapping, wavy lines in shades of gray and light blue, creating a modern, abstract background.

# Introdução

## ▶ Arrays

- São estruturas de dados homogêneas
- Permitem agrupar diversas variáveis ou objetos do mesmo tipo

## ▶ Propriedades

- Um array inicia na posição 0 (igual a linguagem C)
- Podem ser unidimensionais ou multidimensionais

# Introdução

## ► Propriedades

- Apresentam limitações em função da estrutura estática
  - ▢ Não se muda o seu tamanho facilmente
  - ▢ Remoção no meio do array exige uma rotina para reorganização da estrutura

# Arrays Unidimensionais

## ► Criação de um Array

- `<tipo> <nome>[] = new <tipo>[<tamanho>]`
- `<tipo>[] <nome> = new <tipo>[<tamanho>]`

## ► Declaração e instanciação

- `int valores[]; // apenas declaração`
- `valores = new int[5]; // Instanciação do objeto`
- `int valores[] = new int[5]; // Declaração e instanciação`
- `char vogais[] = new char[5];`
- `char[] vogais = new char[5];`

# Introdução

- ▶ Arrays também são objetos
  - ▢ Necessitam ser instanciados
- ▶ Arrays como argumentos
  - ▢ Podem ser utilizados como argumentos na passagem de parâmetros de métodos de uma classe
- ▶ Arrays como valores de retorno
  - ▢ Assim como métodos retornam valores do tipo int, double, etc, eles também podem retonar arrays

# Introdução

- ▶ Arrays como argumentos
  - Podem ser utilizados como argumentos na passagem de parâmetros de métodos de uma classe (passados por referência)
  - Na chamada (especificar o nome sem colchetes):
    - ❏ `int vetor[] = new int[5];`
    - ❏ `modificaVetor(vetor);`
  - Na declaração do método
    - ❏ Declarar como uma referência ao array
    - ❏ `public void modificaVetor(int v []) { ... }`

# Introdução

- ▶ Arrays como valores de retorno
  - Assim como métodos retornam valores do tipo int, double, etc, eles também podem retonar arrays

```
public int[] getArray(){  
    int vet[] = new int[10];  
    return vet;  
}
```

# Arrays Unidimensionais

- ▶ É possível inicializar um array com valores determinados
- ▶ Exemplos
  - `char vogais[] = {'a', 'e', 'i', 'o', 'u'}; // já inicializado`
  - `vogais[0]='a'; vogais[1]='e'; ... // mesmo efeito`
- ▶ Como acessar os elementos:
  - `valores[1]`
  - `valores[3]`
  - `vogais[0]`
- ▶ Tamanho de um array
  - Atributo *length*: determina o tamanho do array (não é um método)



# Arrays Unidimensionais

## ► Exemplo

```
public class ArrayTest {  
    public static void main(String[] args) {  
        char vogais[] = new char[5];  
        vogais[0]='a'; vogais[1]='e'; vogais[2]='i';  
        vogais[3]='o'; vogais[4]='u';  
        // uma outra opcao seria:  
        // char vogais[] = {'a', 'e', 'i', 'o', 'u'};  
        // impressao das vogais  
        for(int i=0; i<vogais.length; i++){  
            System.out.println(vogais[i]);  
        }  
    }  
}
```

# Arrays Unidimensionais

## ► Exemplo

```
public class ArrayTest {  
    public static void main(String[] args) {  
        int valores[] = {10, 20, 30, 40, 50, 60};  
        System.out.println("O tamanho: " + valores.length);  
        valores[1]= 25;  
        valores[3]= 45;  
        // imprimindo valores  
        for(int i=0; i<valores.length; i++){  
            System.out.println(valores[i]);  
        }  
    }  
}
```

# Arrays de Instâncias de Classe

- ▶ Arrays de Instâncias de Classe (objetos)
  - Podem ser declarados da mesma forma
  - Exemplo:

```
public class Carro {  
    private String modelo;  
    private String placa;  
    ...  
    public Carro(String modelo, String placa){  
        this.modelo = modelo;  
        this.placa = placa;  
    }  
    public void mostrarCarro(){  
        System.out.println(this.modelo + " " + this.placa);  
    }  
}
```

# Arrays de Instâncias de Classe

## ► Exemplo (continuação):

```
public class TestaCarro {  
    public static void main(String[] args) {  
        Carro vetorCarros[] = new Carro[3];  
        vetorCarros[0] = new Carro("Brasilia", "III8564");  
        vetorCarros[1] = new Carro("Kombi", "JJJ7675");  
        vetorCarros[2] = new Carro("Chevete", "KKK5643");  
        for(int i=0; i<vetorCarros.length; i++){  
            vetorCarros[i].mostrarCarro();  
        }  
    }  
}
```

# Arrays de Instâncias de Classe

## ▶ IMPORTANTE:

- Cada elemento de um array de objetos, inicialmente aponta para null.

```
Carro vetorCarros[] = new vetorCarros[3];
```

```
vetorCarros[0].mostrarCarro();
```

- Gera um NullPointerException, pois vetorCarros[0] não possui um objeto Carro associado

# Arrays de Instâncias de Classe

- ▶ Uma forma correta poderia ser:

```
Carro vetorCarros[] = new vetorCarros[3];
```

```
vetorCarros[0] = new Carro("Variant", "PPP9999");  
vetorCarros[0].mostrarCarro();
```


# Arrays Multidimensionais

- ▶ Pode-se criar arrays multidimensionais
- ▶ Criação de um ArrayMultidimensional
  - `<tipo> <nome>[][] = new <tipo>[<tamanho>][<tamanho>]`
  - `<tipo>[][] <nome> = new <tipo>[<tamanho>][<tamanho>]`
- ▶ Exemplos
  - `int matriz[][] = new int[5][5];`
  - `// matriz de 2 linhas e 3 colunas`
  - `int matriz[][] = {{1,2,3}, {4,5,6}}`
  - `Xadrez tabuleiro[][] = new Xadrez[8][8]`

# Arrays Multidimensionais

## ► Exemplo

```
public class ArrayTest {  
    public static void main(String[] args) {  
        int matriz1[][] = new int[3][5];  
        // matriz de 4 linhas e 3 colunas  
        int matriz2[][] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};  
        // imprimindo matriz  
        System.out.println("Tamanho: " + matriz2.length);  
        for(int i=0; i<matriz2.length; i++){  
            System.out.println("Tamanho: " + matriz2[i].length);  
            for(int j=0; j<matriz2[i].length; j++){  
                System.out.println(matriz2[i][j]);  
            }  
        }  
    }  
}
```





# Cópia de Arrays

## ► Cópia de Arrays

- Como os arrays são objetos, ao fazer uma atribuição entre duas variáveis que representam arrays é fazer que referenciem o mesmo array
- Exemplo: `vetor1 = vetor2;`
  - ▢ Se alterar `vetor1` o `vetor2` também sentirá esta mudança
- Para copiar:
  - ▢ Podemos escrever um código
  - ▢ Utilizar a API Java (método `arraycopy` da classe `System`)

# Cópia de Arrays

- ▶ Exemplo: copiando “a mão”

```
public class ArrayTest {  
    public static void main(String[] args) {  
        int vetor1[] = {1, 3, 5, 7, 9, 11};  
        int vetor2[] = {21, 23, 25, 27, 29, 31};  
        // copia a mao  
        for(int i=0; i<vetor1.length; i++){  
            vetor2[i] = vetor1[i];  
        }  
        ...  
    }  
}
```

# Cópia de Arrays

- ▶ Exemplo: copiando com arraycopy

```
public class ArrayTest {  
    public static void main(String[] args) {  
        int vetor1[] = {1, 3, 5, 7, 9, 11};  
        int vetor2[] = {21, 23, 25, 27, 29, 31};  
        // copiando com arraycopy  
        System.arraycopy(vetor1, 0, vetor2, 0, 6);  
        ...  
    }  
}
```

fonte

posição  
inicial  
fonte

destino

posição  
inicial  
destino

tamanho

# Classe Arrays

## ▶ Classe Arrays

- Uma classe que pertence ao pacote `java.util`
- Permite a manipulação de arrays
- Possui métodos para busca ordenação de dados entre outros
- Olhar na API

## ▶ Método para ordenação

- É um quicksort melhorado

# Classe Arrays

## ► Ordenação de Arrays

```
import java.util.Arrays;
public class ArrayTest {
    public static void main(String[] args) {
        int valores[] = {65, 12, 90, 15, 21, 45};
        Arrays.sort(valores);
        // imprimindo valores
        for(int i=0; i<valores.length; i++){
            System.out.println(valores[i]);
        }
    }
}
```

# Percorrendo o Array

- ▶ Pode ser percorrido com o comando for da posição 0 até o tamanho do array

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

# Percorrendo o Array

- ▶ Pode ser percorrido utilizando o for na sintaxe “foreach”

```
// imprimindo valores  
for(int inteiro : valores){  
    System.out.println(inteiro);  
}
```