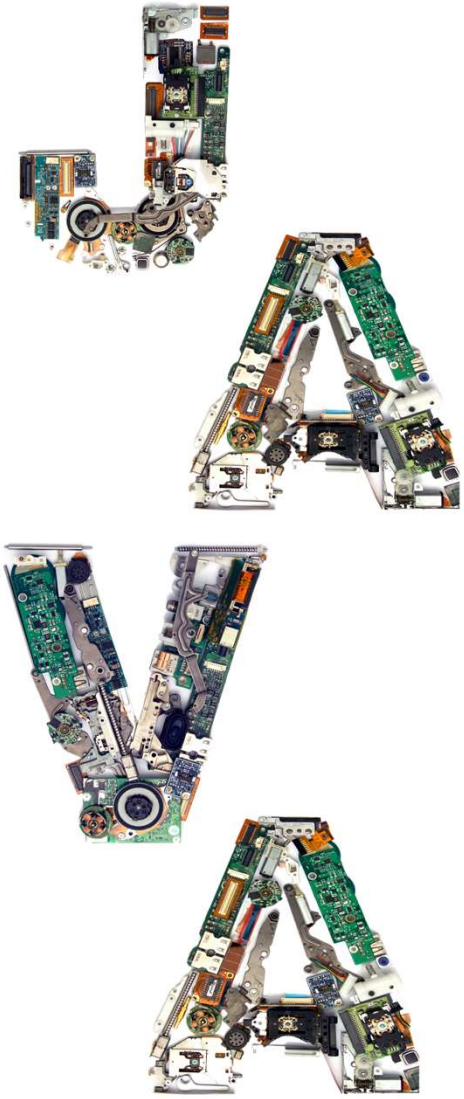
Programação Orientada a Objetos com Java e WEB



Arrays Estáticos

Introdução

São estruturas de dados que podem armazenar mais de um valor **do mesmo tipo de dado**.

Em java, os arrays são objetos. Tem comportamento de instâncias de classes. Eles precisam ser declarados, instanciados e inicializados.

Existem dois tipos de arrays: ***vetores*** (arrays unidimensionais) e ***matrizes*** (arrays multidimensionais).

Arrays unidimensionais – vetores

São arrays unidimensionais. Você pode imaginar como uma linha de uma tabela composta por várias células.

Declaração de um vetor:

```
<tipo>[] <nome>;
ou
<tipo> <nome>[];
```

Exemplo:

```
int[] x;
ou
int x[];
```

Arrays unidimensionais – vetores

Depois de declarar um vetor, é necessário instancia-lo. A instanciação é o processo pelo qual um endereço de memória é alocado para um objeto. A sintaxe para instanciação de vetores é:

```
<nome> = new <tipo_do_array>[<total_de_posições>];
```

Exemplo:

```
x = new int[40];
```

obrigatório definir o tamanho do array!!

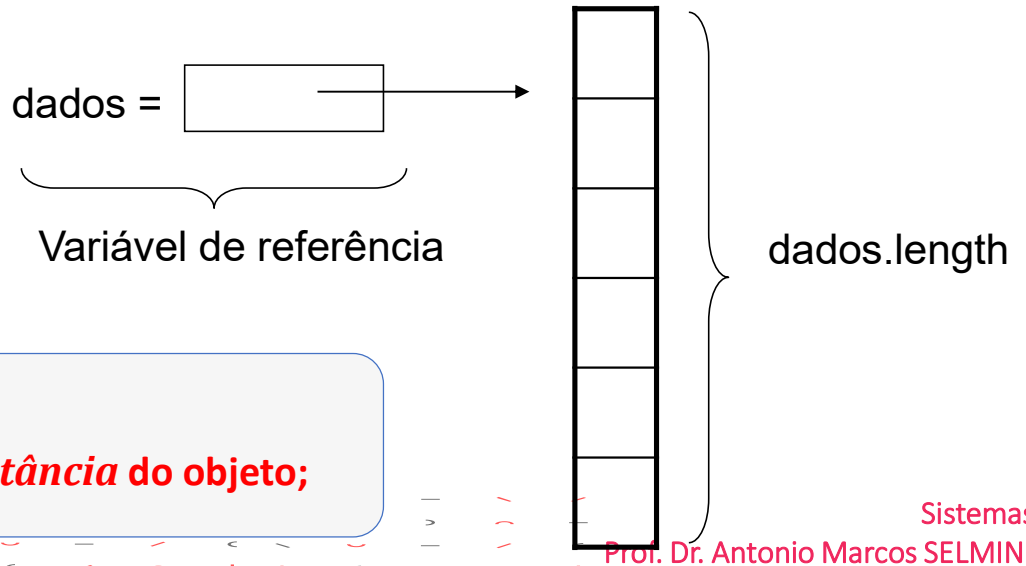
```
int[] x = new int[]
```

Não compila!!!
Falta o tamanho do array!

Arrays unidimensionais – vetores

A variável x declarada anteriormente é uma referência a um array de números inteiros.

- Os arrays têm comprimento fixo e não podem ser alterados (**são chamados de estáticos → não confundir com o modificador *static***).
- Os arrays têm elementos de um tipo específico.
- Exemplo: `double[] dados = new double[10];`



Observação:

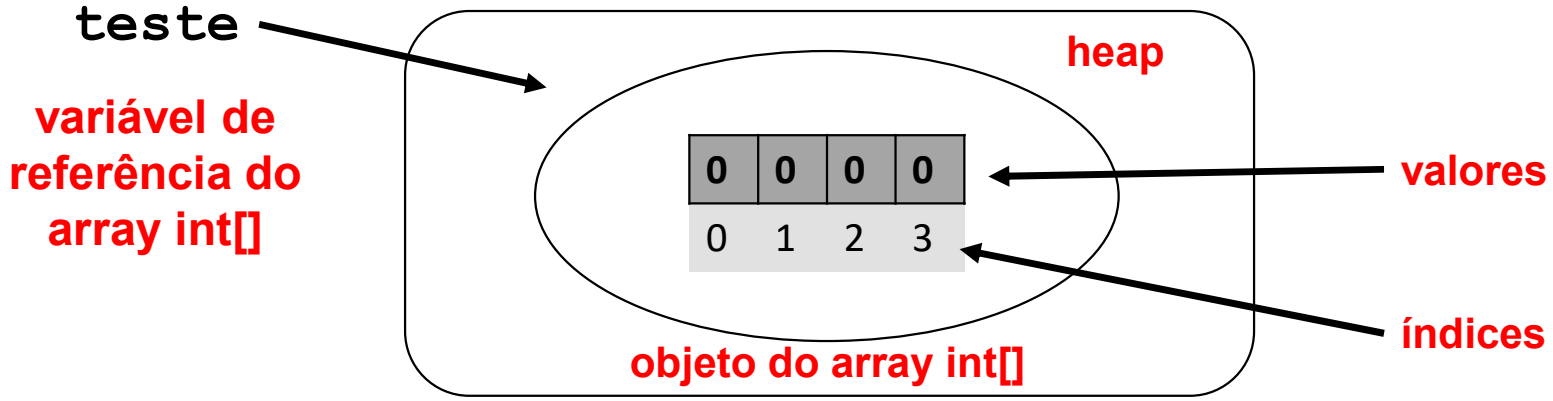
- ***length* não é um método;**
- ***length* é uma *variável de instância* do objeto;**

Arrays unidimensionais – vetores

```
int[] teste = new int[4];
```

mesmo efeito!!

```
int[] teste;
teste = new int[4];
```



Inicialização de vetores

1. `int[] primos = new int[5];`
   - primos[0] = 2;
   - primos[1] = 3;
   - primos[2] = 5;
   - primos[3] = 7;
   - primos[4] = 11;

Declaração e depois a inicialização das posições!!

Inicialização de vetores

```
2. int[] primos = {2, 3, 5, 7, 11};
```

Um objeto array de tamanho 5 é instanciado no heap e inicializado. O objeto é referenciado pela variável *primos*.

```
3. new int[] {2, 3, 5, 7, 11};
```

Mesma observação feita no exemplo 2. A diferença está no fato que não há nenhuma variável referenciado o objeto no heap. Normalmente é utilizado em retorno de métodos ou como argumento na chamada de métodos.

Inicialização de vetores

Exemplos de inicializações:

```
int[] x = new int[5];
x[4] = 2; //OK, o último elemento está no índice 4
X[5] = 3; //Não existe o índice 5!

int[] z = new int[2];
int y = -3;
z[y] = 4; //y é um número negativo!!
```

Um índice deve ser um valor inteiro e positivo!!

Inicialização de vetores – exemplo

```
public class Exemplo {
    public static void main(String[] args) {
        int[] x = new int[10];
        lerVetor(x);
        imprimirVetor(x);
    }

    public static void lerVetor(int[] x) {
        for(int i = 0; i < x.length; i++) {
            x[i] = 2*i;
        }
    }

    public static void imprimirVetor(int[] x) {
        for(int i = 0; i < x.length; i++) {
            System.out.print(x[i]+"\t");
        }
    }
}
```

o argumento é a referência para o array no *heap*.

- **Os métodos são estáticos porque são chamados a partir de um contexto estático, sem objetos.**
- **Cada método recebe como parâmetro uma referência para o array instanciado no método *main*().**

Estrutura *for*() genérica

A partir da versão 5.0 do Java é possível utiliza a estrutura ***for()*** para iterar pelos elementos de um array sem utilizar um contador. Essa estrutura é chamada de ***for()*** genérico ou ***for()*** aprimorado.

A sintaxe para a estrutura for é:

```
for (parâmetro : nomeDoArray) {
    //instruções;
}
```

onde ***parâmetro*** tem duas partes: um tipo e um identificador e ***nomeDoArray*** é o array pelo qual o laço vai iterar.

Estrutura *for*() genérica

Como exemplo, considere a seguinte declaração para um vetor de números inteiros:

```
int[] x = new int[10];
```

```
for(int i : x) { }
```

Declare uma variável para armazenar um elemento do array. A variável deve ter o mesmo tipo do array.

O sinal de dois pontos (:) significa "de", ou seja, "para cada valor inteiro de *x*...".

Array que será percorrido pelo loop. A cada execução do loop o próximo elemento de *x* será atribuído para a variável *i*.

Estrutura *for*() genérica

Exemplos:

```
int[] x = new int[10];
```

```
for(int i : x) { }
```

```
double[] x = new double[10];
```

```
for(double i : x) { }
```

```
char[] x = new char[10];
```

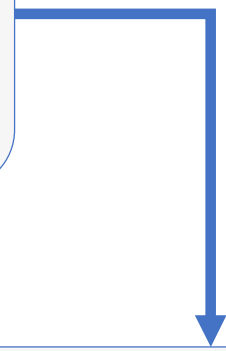
```
for(char i : x) { }
```

```
String[] x = new String[10];
```

```
for(String i : x) { }
```

Estrutura *for*() genérica

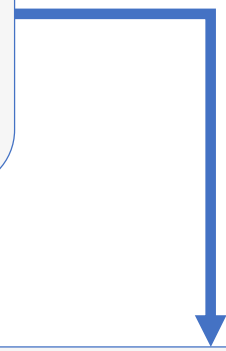
```
int[] x = new int[20];
for (int i = 0; i < x.length; i++) {
   System.out.print(x[i] + "\t");
}
```



```
int[] x = new int[20];
for (int i : x) {
   System.out.print(i + "\t");
}
```

Estrutura *for*() genérica

```
int[] x = new int[20];
for (int i = 0; i < x.length; i++) {
   System.out.print(x[i] + "\t");
}
```



```
int[] x = new int[20];
for (int i : x) {
   System.out.print(i + "\t");
}
```

Arrays de objetos – exemplos

```java
public class Aluno {
    String nome;
    double media;

    public Aluno(String n, double m) {
        nome = n;
        media = m;
    }

    public String getNome() {
        return nome;
    }
}
```

```java
public class TesteAluno {
     public static void main(String[] args) {
        int i;
        Aluno a[] = new Aluno[2];

        a[0] = new Aluno("Selmini", 8.5);
        a[1] = new Aluno("Maria", 10);

        for(i = 0; i < a.length; i++)
             System.out.println(a[i].getNome());
        }
}
```

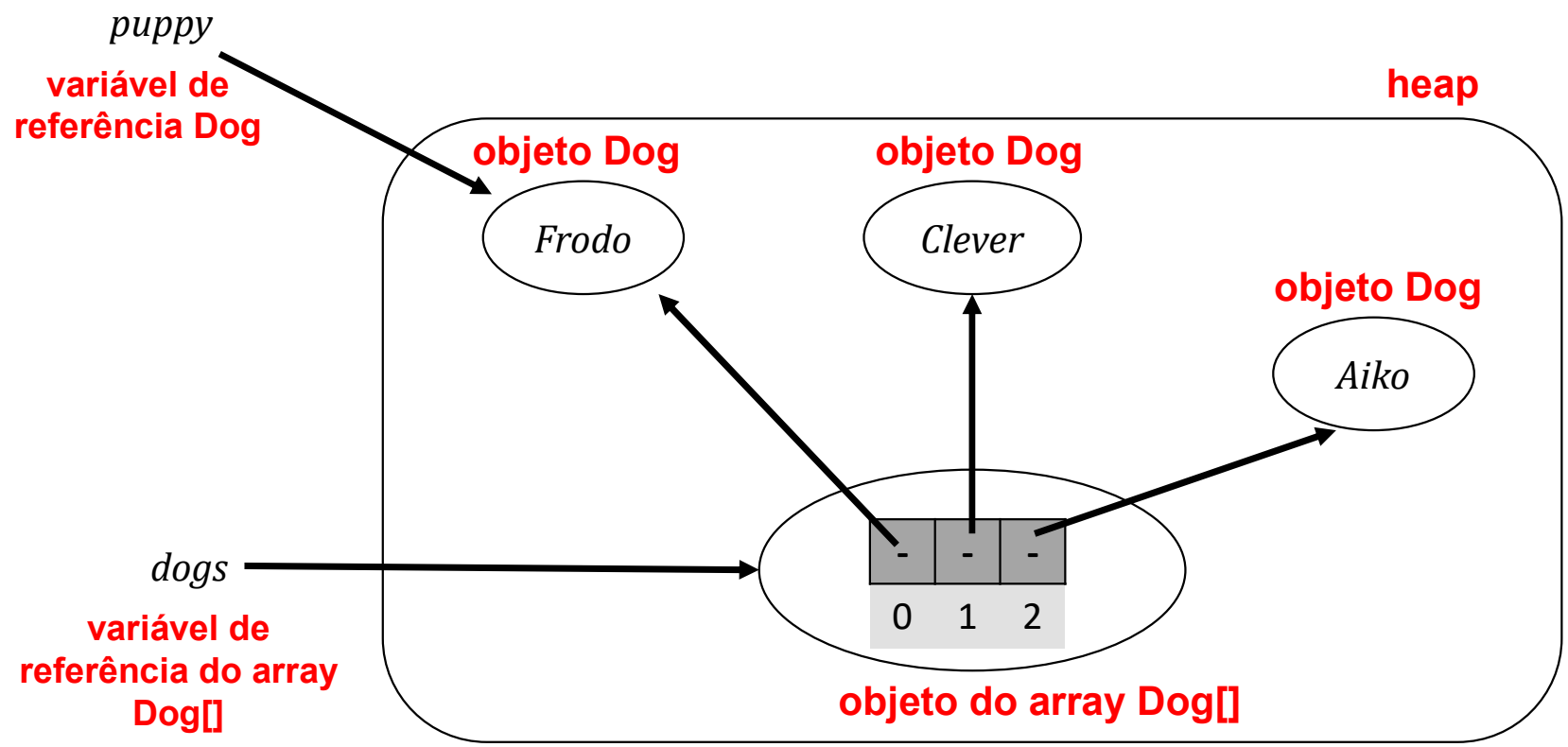
Arrays de objetos – exemplos

```
public class Dog {
	String nome;

	public Dog(String nome) {
		this.nome = nome;
	}
}
```

```
public class TestaDog {
	public static void main(String[] args) {
		Dog puppy = new Dog("Frodo");
		Dog[] dogs = {puppy, new Dog("Clever"), new Dog("Aiko")};
	}
}
```

Arrays de objetos – *heap*



Arrays multidimensionais

São arrays multidimensionais (podem ter duas ou mais dimensões). O uso mais comum é a bidimensional. Declaração:

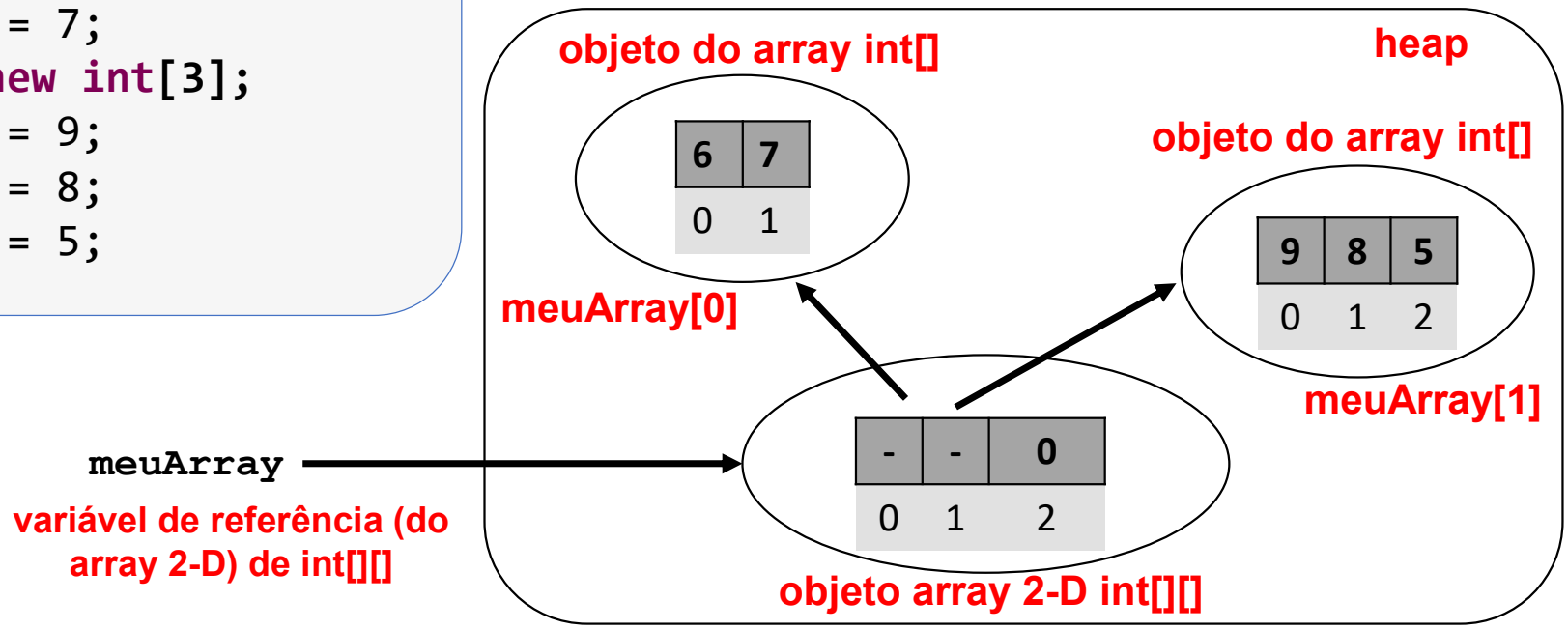
```
<tipo>[][] <nome>;
ou
<tipo> <nome>[][];
```

Exemplo:

```
int x[][] = new int[5][5];
```

Arrays multidimensionais – inicialização

```
int[][] meuArray = new int[3][];
meuArray[0] = new int[2];
meuArray[0][0] = 6;
meuArray[0][1] = 7;
meuArray[1] = new int[3];
meuArray[1][0] = 9;
meuArray[1][1] = 8;
meuArray[1][2] = 5;
```



Arrays multidimensionais – exemplo

```
import java.util.Random;

public class Exemplo {
   public static void main(String[] args) {
      Random gerador = new Random();
         int[][] m = new int[5][5];
         for(int linha = 0; linha < m.length; linha++) {
            for(int coluna = 0; coluna < m[linha].length; coluna++) {
               m[linha][coluna] = gerador.nextInt(101);
            }
         }
   }
}
```

Arrays como parâmetros

Ao passar um array para um método, apenas indicamos a variável de referência. Por exemplo:

```
int[] x = new int[10];
lerVetor(x);
```

O método que recebe o array deve declarar uma variável do mesmo tipo para receber a referência. Exemplo:

```
public void lerVetor(int[] x) {}
```

Arrays como parâmetros

Para passar um array bidimensional como argumento para um método:

```
int[][] x = new int[10][10];
lerMatriz(x);
```

Para receber um array bidimensional como parâmetro:

```
public void lerMatriz(int[][] x) {}
```

Lista de argumento de comprimento variável

A partir da versão 5.0 do Java é possível criar métodos com lista de argumentos de comprimento variável.

Um tipo seguido por reticências (...) indica que o método recebe um número variável de argumentos para aquele tipo.

Isso só pode ocorrer uma única vez e, quando ocorrer deve ser colocado no final da lista de parâmetros.

O Java trata a lista de argumentos de comprimento variável como um vetor para o tipo especificado.

Lista de argumento de comprimento variável

```
public class Exemplo {
    public static void main(String args[]) {
        System.out.printf("%.2f\n", media(2.5, 3.0));
        System.out.printf("%.2f\n", media(2.2, 3.3, 4.1, 5.05));
    }

    public static double media(double... x) {
        double total = 0;
        for(double i : x ) {
            total += i;
        }
        return total / x.length;
    }
}
```

Bibliografia

- DEITEL, H. M., DEITEL, P. J. **JAVA como programar**. 10ª edição. São Paulo: Prentice-Hall, 2010.
- SCHILDT, H. **Java para Iniciantes – Crie, Compile e Execute Programas Java Rapidamente**. 6ª Edição, Editora Bookman, Porto Alegre, RS, 2015.