

# 08.Vetores e Matrizes

Prof. Alexandre Krohn



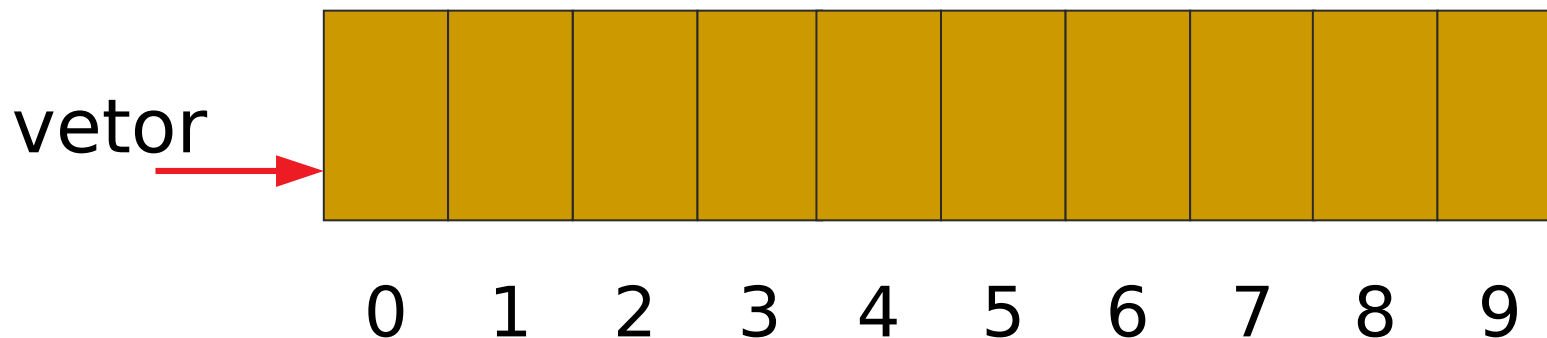


# Roteiro

- Vetores
- Matrizes
- Exercícios

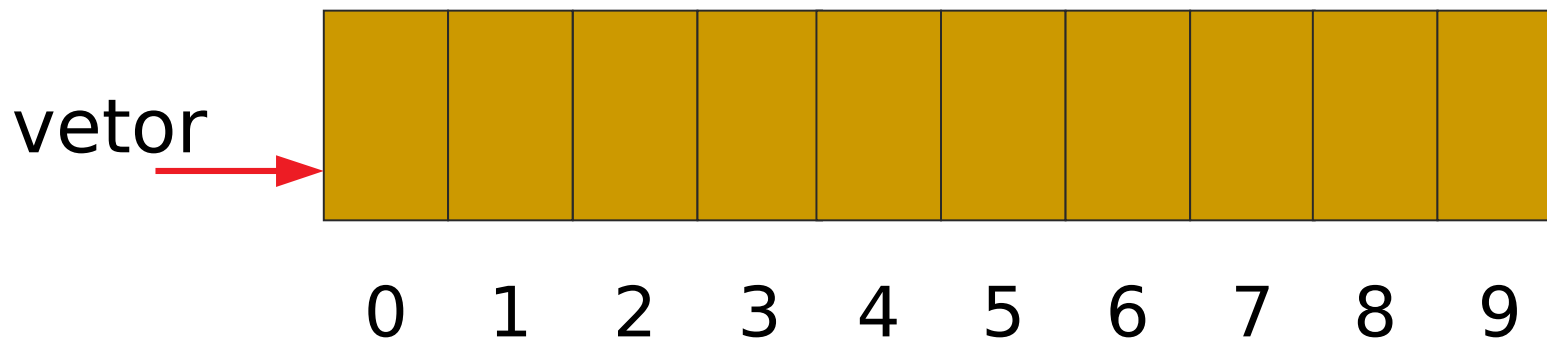
# Vetores (Arrays)

- Um vetor, ou array, é uma estrutura de dados composta por um número finito de componentes (ou elementos), todos eles do mesmo tipo, armazenado de forma contígua na memória, sendo que cada componente é identificado por um índice.



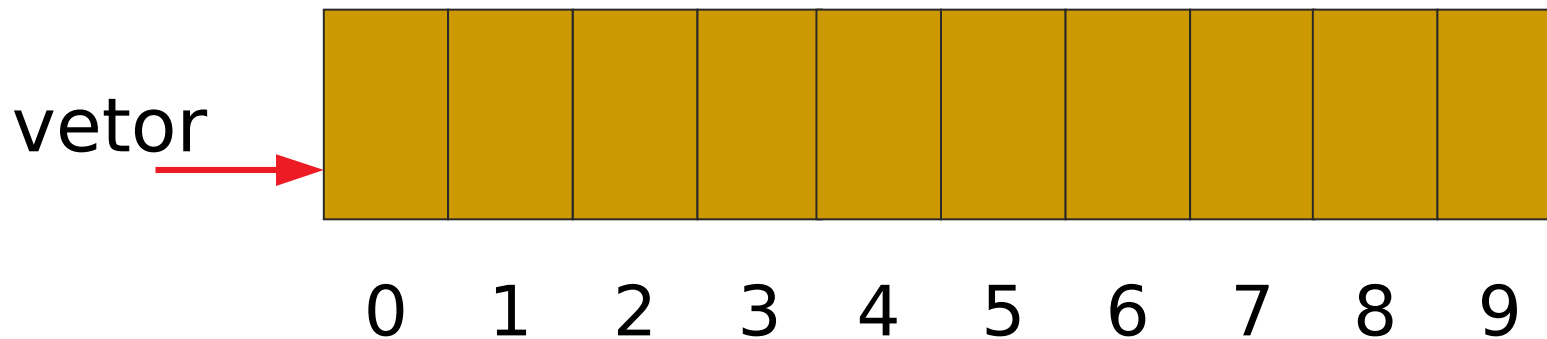
# Vetores (Arrays)

- Seus elementos podem ser acessados diretamente sem a necessidade de acessar elementos anteriores ao desejado.
- O tamanho de um array é correspondente à quantidade de componentes que possui.



# Vetores (Arrays)

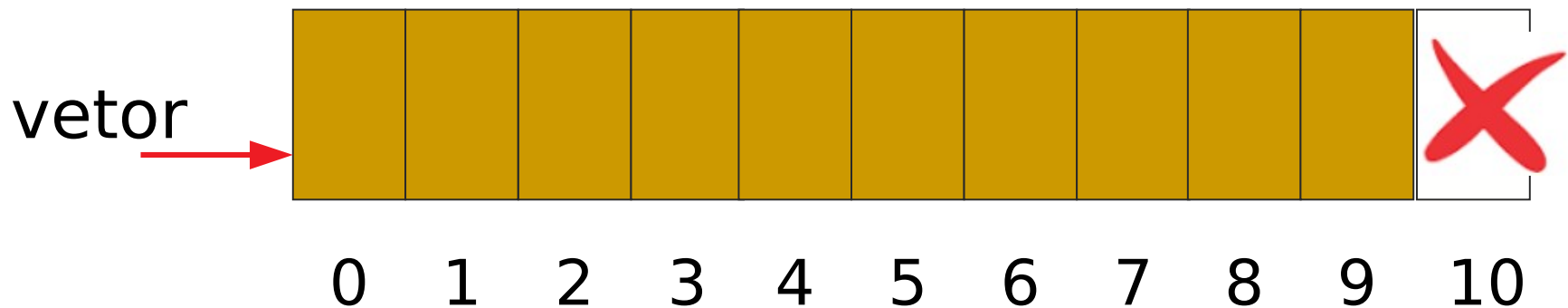
- Vetores podem conter tipos primitivos e/ou objetos. Uma vez declarado, só pode conter um tipo
- Vetores tem seu tamanho declarado na instanciação, e depois disso seu tamanho não pode mais mudar



# Vetores (Arrays)

- Os índices dos vetores começam em **0**
- Escrever além do limite do vetor gera uma exceção

***ArrayIndexOutOfBoundsException***



# Vetores : Declaração

- Vetores são declarados utilizando colchetes:

```
int v[];
```

```
double valores[];
```

```
String[] telefones;
```

```
Contato[] contatos;
```

# Vetores são objetos

- Vetores são objetos, por isso precisam ser alocados

```
String[] placas = new String[100];
```

```
int[] a = new int[30];
```

```
float[] precos = new float[10];
```

É na alocação  
que se define  
o **tamanho**  
do vetor



# Vetores : atributo length

- Sendo objetos, os vetores possui um atributo chamado **length**, que fornece o comprimento do vetor.

```
System.out.println(a.length);
```

```
System.out.println(precos.length);
```

```
System.out.println(placas.length);
```

# Vetores : atribuição

- Atribui-se um valor a um vetor utilizando colchetes para indicar a posição.
- Ex:

```
a[0] = 10;
```

```
a[3] = 20;
```

# Vetores : atribuição

- Recupera-se um valor de um vetor utilizando colchetes para indicar a posição também.
- Ex:

```
int n = a[3];
```

# Vetores e ponteiros

- O nome de um vetor é uma referência para o mesmo, um ponteiro.

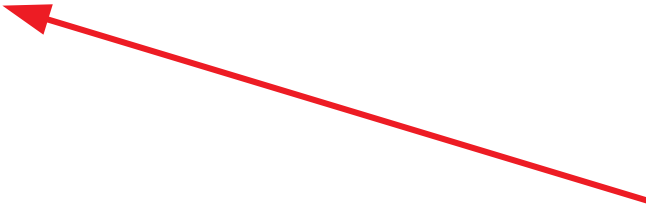
```
int[] alfa = new int[30];
```

```
int[] beta = alfa;
```



```
int[] alfa = new int[30];
```

```
int[] beta = alfa;
```



Fazer isso **não**  
**copia** o vetor



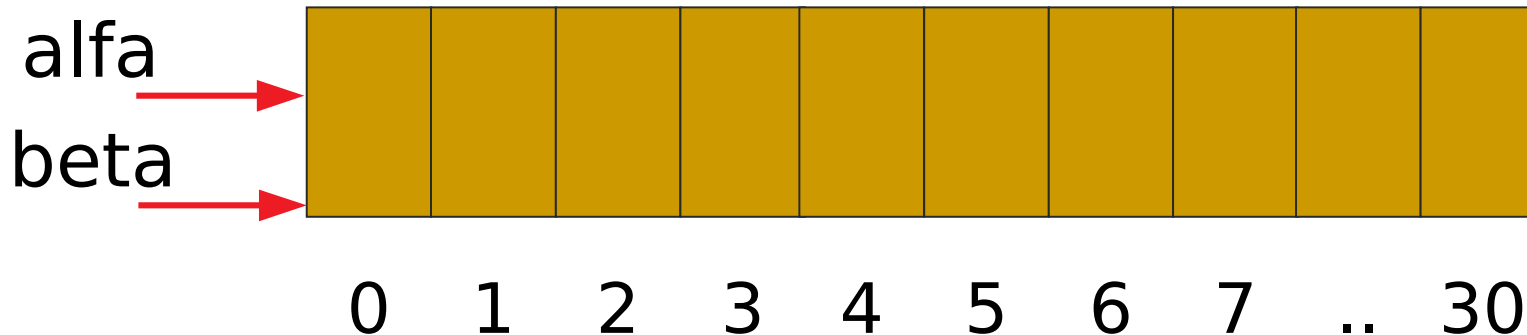


```
int[] alfa = new int[30];
```

```
int[] beta = alfa;
```

Mas faz com que  
as duas variáveis  
apontem para  
o mesmo vetor

Fazer isso **não  
copia** o vetor



# Vetores de objetos

- Vetores podem conter objetos

```
Contato[] contatos = new Contato[10];
```

Contato
- nome : String - telefone : String - e-mail : String
+ getNome() : String + setNome(nome : String) : void + getTelefone() : String + setTelefone(telefone : String) : void + getEmail() : String + setEmail(email : String) : void

# Vetores de objetos

- Acessa-se os objetos pelos índices

```
Contato[] contatos = new Contato[10];
```

```
contatos[0] = new Contato();
```

```
contatos[0].setNome("João");
```

```
contatos[0].setTelefone("98899888");
```

```
contatos[0].setEmail("joao@ucs.br");
```

```
Contato co = contatos[0];
```

Pode-se também  
Obter o objeto que  
está em uma  
determinada  
posição

# Posições não ocupadas

- Posições vazias nos vetores de objetos contém nulos

```
Contato[] contatos = new Contato[10];
```

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

null  
null  
null  
null  
null  
null  
null  
null  
null  
null

# Posições não ocupadas

- Posições vazias nos vetores de tipos primitivos numéricos contém zeros

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

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

0  
0  
0  
0  
0



# Exemplo : Preenchendo um vetor de objetos

```
import java.util.Scanner;

public class ExemploVetorObjetos {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        String resposta = "n";
        int contador = 0;
        Contato[] contatos = new Contato[30];

        do {
            Contato c = new Contato();
            System.out.println("Informe um nome");
            c.setNome(in.nextLine());
            System.out.println("Informe o número do telefone");
            c.setTelefone(in.nextLine());

            contatos[contador++] = c;

            System.out.println("Deseja continuar (\"S\"-Sim/\"N\"-Não)?");
            resposta = in.nextLine();
        } while ("s".equalsIgnoreCase(resposta) && contador < 30);

        for(int i = 0; i < contador; i++) {
            System.out.println(contatos[i].getNome() + " " + contatos[i].getTelefone());
        }

        in.close();

    }
}
```

Contato
- nome : String - telefone : String - e-mail : String
+ getNome() : String + setNome(nome : String) : void + getTelefone() : String + setTelefone(telefone : String) : void + getEmail() : String + setEmail(email : String) : void

# Exemplo: Procurando um nome em um vetor de objetos

```
public Contato procura(Contato[] contatos, String nome) {  
  
    Contato contato = null;  
  
    for(int i = 0; i < contatos.length; i++) {  
        if(contatos[i] != null && nome.equals(contatos[i].getNome())) {  
            return contatos[i];  
        }  
    }  
    return null;  
}
```

# Exemplo: Procurando um nome em um vetor de objetos

```
public Contato procura(Contato[] contatos, String nome) {  
  
    Contato contato = null;  
  
    for(int i = 0; i < contatos.length; i++) {  
        if(contatos[i] != null && nome.equals(contatos[i].getNome())) {  
            return contatos[i];  
        }  
    }  
    return null;  
}
```

O método retorna um contato Contido no vetor com nome igual ao informado no parâmetro.

Se não encontrar, retorna nulo

# For enhanced

- Java possui um tipo de comando ***for*** específico para vetores e coleções:

```
for(Contato c : contatos) {  
    System.out.println(c.getNome());  
}
```

Lê-se:  
Para cada Contato c  
no vetor contatos,  
mostre o nome do contato

# For enhanced

```
for(Contato c : contatos) {  
    System.out.println(c.getNome());  
}
```

É equivalente à

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



# Vetores : Tamanho fixo

- O tamanho de um vetor não pode ser modificado após sua instanciação:

```
valores = new double[50];
```

```
valores = new double[90];
```

Isso não altera o tamanho do vetor, mas sim cria um novo.

Valores existentes no vetor são perdidos nessa redefinição.

# Vetores : Instanciação automática

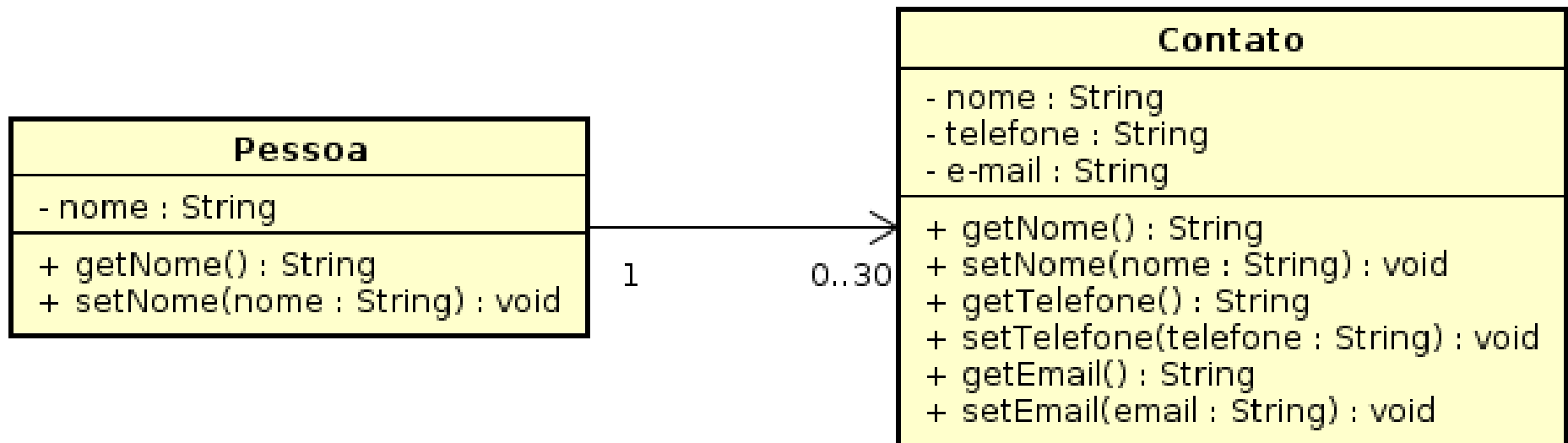
- Vetores podem ser instanciados e preenchidos automaticamente na sua declaração:

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

```
String[] nomes = {"Huguinho", "Zézinho", "Luizinho"};
```

Nesse caso, o **tamanho do vetor é igual ao número de elementos fornecidos** no momento da inicialização do mesmo.

# Vetores como atributos de instâncias




Essa relação entre duas classes pode ser lida como :  
Uma pessoa possui de zero até 30 contatos

# Vetores como atributos de instâncias

```
public class Pessoa {  
  
    private String nome;  
    private Contato[] contatos;  
  
    public Pessoa() {  
        contatos = new Contato[30];  
    }  
  
    public Pessoa(String nome) {  
        this();  
        this.nome = nome;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public Contato[] getContatos() {  
        return contatos;  
    }  
  
    public void setContatos(Contato[] contatos) {  
        this.contatos = contatos;  
    }  
}
```

Note que os vetores  
são inicializados  
nos construtores.



# Vetores como atributos de instâncias

```
public class Pessoa {  
  
    private String nome;  
    private Contato[] contatos;  
  
    public Pessoa() {  
        contatos = new Contato[30];  
    }  
  
    public Pessoa(String nome) {  
        this();  
        this.nome = nome;  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public Contato[] getContatos() {  
        return contatos;  
    }  
  
    public void setContatos(Contato[] contatos) {  
        this.contatos = contatos;  
    }  
}
```

Invocar **this()** faz  
Com que seja  
Chamado o  
Construtor sem  
parâmetros



# Vetores como parâmetros de métodos

- Vetores podem ser passados por parâmetros para métodos
- Deve-se tomar cuidado, pois essa passagem é feita por referência, o que significa que o método pode alterar o vetor que foi passado.

```
public int achaMaior(int[ ] vetor) {...}
```

# Vetores como retorno de métodos

- Vetores podem ser retornados por métodos.

```
public int[] subVetor(int[ ] vetor, int  
inicio, int fim) {  
  
    ..  
  
}
```



# Roteiro

- Vetores
- Matrizes
- Exercícios

# Matrizes

- Matrizes são semelhantes aos vetores, só que possuem mais de uma dimensão
- Ex: Uma matriz 4 x 4 de números inteiros:

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

# Matrizes

## ■ Declaração e inicialização de uma matriz

- definir o tipo
- definir a variável/identificador
- definir o tamanho (linhas e colunas) ou os elementos iniciais

- Matriz declarada mas não inicializada (null)

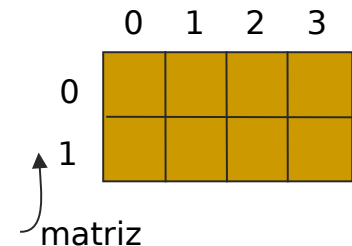
```
int[][] matriz;
```

- Matriz de 10x2 inteiros:

```
int[][] matrizInt = new int[10][2];
```

- Matriz de 2x3 inteiros (valores definidos na criação)

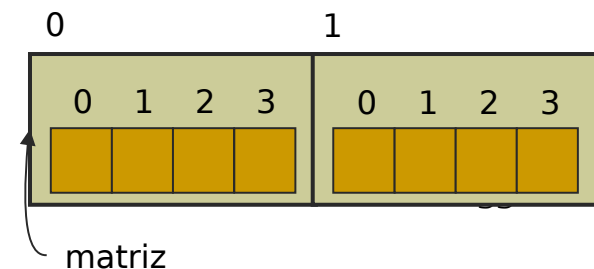
```
int[][] matrizInt = {{1, 2, 3}, {4, 5, 6}};  
int[][] matrizInt = {{1, 2, 3},  
                      {4, 5, 6}};
```



- Captura do tamanho da matriz

```
int linhas = matrizInt.length;  
int colunas = matrizInt[0].length;
```

Programação Orientada a Objetos



# Matrizes : percorrendo

- Para percorrer uma matriz, utiliza-se quantos laços **for** quantas dimensões a matriz possuir:

```
for (int i = 0; i < 4; i++) {  
    for (int j = 0; j < 4; j++) {  
        System.out.printf("%2d ", m[i][j]);  
    }  
    System.out.println();  
}
```

# Matrizes = vetor de vetores

- Matrizes são vetores de vetores e portanto, podem possuir linhas com quantidades diferentes de colunas.

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

```
m[0] = new int[2];  
m[1] = new int[3];  
m[2] = new int[4];  
m[3] = new int[5];
```

Fazer isso aumenta a complexidade do programa, pois deve-se cuidar para não “estourar” o número de colunas de cada linha.



# Dúvidas?







# Roteiro

- Vetores
- Matrizes
- Exercícios



# Atividades

- Execute as atividades presentes no documento

08.Lista.de.Exercícios.POO.pdf

# Próximos passos

- Coleções

