



Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

Linguagem de Programação JAVA

Monael Pinheiro Ribeiro, D.Sc.

Primeiro Programa em JAVA

Primeiro Programa em JAVA

```
public class Minimo
{
    public static void main(String[] args)
    {
    }
}
```

Primeiro Programa em JAVA

```
public class Minimo
{
    public static void main(String[] args)
    {
    }
}
```

- Todo programa em JAVA é uma **Classe**.
- Neste caso a **classe** se chama Minimo.
- O mesmo nome da **classe** deve **OBRIGATORIAMENTE** ser o nome do arquivo fonte (.java), neste caso o arquivo fonte será o Minimo.java

Primeiro Programa em JAVA

```
public class Minimo
{
    public static void main(String[] args)
    {
    }
}
```

Todo programa em JAVA deve obrigatoriamente ter uma, e somente uma, “função” principal pública (main).

Primeiro Programa em JAVA

```
public class Minimo
{
    public static void main(String[] args)
    {
    }
}
```

Todo programa em JAVA deve **obrigatoriamente** ter uma, e somente uma, “função” **método** principal **público** (main).

Primeiro Programa em JAVA

```
public class Minimo
{
    public static void main(String[] args)
    {
    }
}
```

Para “compilar” o programa JAVA deve-se usar o seguinte comando, no seu console ou terminal:

```
javac <nome_do_arquivo_fonte>.java
```

Para o programa do exemplo:

```
javac Minimo.java
```

Primeiro Programa em JAVA

```
public class Minimo
{
    public static void main(String[] args)
    {
    }
}
```

Então será gerado um arquivo “executável” pela JVM. Esse arquivo tem extensão .class e é chamado de ByteCode.

Ele é “executado” pela JVM através do seguinte comando no console o terminal:

```
java <nome_do_arquivo_bytecode>
```

Para o programa do exemplo:

```
java Minimo
```


“Comando” de Saída

O “comando” para exibir uma mensagem no dispositivo de saída padrão (monitor):

~~“Comando”~~ de Saída

O ~~“comando”~~ para exibir uma mensagem no dispositivo de saída padrão (monitor):

Método de Saída

O **método de classe** para exibir uma mensagem no dispositivo de saída padrão (monitor):

Método de Saída

O **método de classe** para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

Exemplos:

```
System.out.print("UFABC");
```



UFABC

Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

Exemplos:

```
System.out.print('M');
```



M

Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

Exemplos:

```
System.out.print(97);
```



97

Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

Exemplos:

```
System.out.print(3.1415);
```



3.1415

Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

Exemplos:

```
System.out.print("UFABC");  
System.out.print('M');  
System.out.print(97);  
System.out.print(3.1415);
```



UFABCM973.1415


Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor) saltando uma linha:

```
System.out.println(<literal ou variável>);
```

Exemplos:

```
System.out.println("UFABC");  
System.out.println('M');  
System.out.println(97);  
System.out.println(3.1415);
```



```
UFABC  
M  
97  
3.1415
```

Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

Exemplos:

```
int valor = 3;  
System.out.print(valor);
```



3

Segundo Programa em JAVA

```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Ola Mundo!");
    }
}
```

Comando para “compilar”:

Segundo Programa em JAVA

```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Ola Mundo!");
    }
}
```

Comando para “compilar”:

```
javac Hello.java
```

Segundo Programa em JAVA

```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Ola Mundo!");
    }
}
```

Comando para “compilar”:

```
javac Hello.java
```

Comando para “executar”:

Segundo Programa em JAVA

```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Ola Mundo!");
    }
}
```

Comando para “compilar”:

```
javac Hello.java
```

Comando para “executar”:

```
java Hello
```

Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

O operador + concatena (junta, une) valores para serem exibidos.

Exemplo:

```
int tempo = 8;  
String universidade = "UFABC";  
System.out.println("Eu estou na " + universidade + " ha " + tempo + " anos.");
```


Método de Saída

O método de classe para exibir uma mensagem no dispositivo de saída padrão (monitor):

```
System.out.print(<literal ou variável>);
```

O operador + concatena (junta, une) valores para serem exibidos.

Exemplo:

```
int tempo = 8;  
String universidade = "UFABC";  
System.out.println("Eu estou na " + universidade + " ha " + tempo + " anos.");
```



Eu estou na UFABC ha 8 anos

“Comando” de Entrada

Para entrar dados via teclado na Linguagem JAVA necessita-se de um objeto **Scanner**.

Primeiramente, deve-se importá-lo da biblioteca da linguagem através do comando:

```
import java.util.Scanner;
```

Depois disso, no corpo do programa em JAVA deve-se criar uma variável do tipo Scanner e iniciá-la com o dispositivo de entrada padrão do sistema, através do seguinte comando:

```
Scanner <identificador> = new Scanner(System.in);
```

~~“Comando”~~ de Entrada

Para entrar dados via teclado na Linguagem JAVA necessita-se de um objeto **Scanner**.

Primeiramente, deve-se importá-lo da ~~biblioteca da linguagem~~ através do comando:

```
import java.util.Scanner;
```

Depois disso, no corpo do programa em JAVA deve-se ~~criar uma variável~~ do tipo Scanner e iniciá-la com o dispositivo de entrada padrão do sistema, através do seguinte comando:

```
Scanner <identificador> = new Scanner(System.in);
```

Método de Entrada

Para entrar dados via teclado na Linguagem JAVA necessita-se de um objeto Scanner.

Primeiramente, deve-se importá-lo do pacote de utilidades da linguagem através do comando:

```
import java.util.Scanner;
```

Depois disso, no corpo do programa em JAVA deve-se instanciar um objeto do tipo Scanner e iniciá-la com o dispositivo de entrada padrão do sistema, através do seguinte comando:

```
Scanner <identificador> = new Scanner(System.in);
```

Método de Entrada

Para entrar dados via teclado na Linguagem JAVA necessita-se de um **objeto Scanner**.

Primeiramente, deve-se importá-lo do **pacote de utilidades** da linguagem através do comando:

```
import java.util.Scanner;
```

Exemplo:

```
Scanner teclado = new Scanner(System.in);
```

Método de Entrada

```
Scanner <identificador> = new Scanner(System.in);
```

Após criar um objeto associado ao dispositivo de entrada padrão, há “~~comandos~~” para ler cada tipo de dado, tais como:

Tipo de Dado	“Comando”
boolean	nextBoolean()
byte	nextByte()
short	nextShort()
int	nextInt()
long	nextLong()
float	nextFloat()
double	nextDouble()
String	next()

Método de Entrada

```
Scanner <identificador> = new Scanner(System.in);
```

Após criar um objeto associado ao dispositivo de entrada padrão, há métodos para ler cada tipo de dado, tais como:

Tipo de Dado	“Comando”
boolean	nextBoolean()
byte	nextByte()
short	nextShort()
int	nextInt()
long	nextLong()
float	nextFloat()
double	nextDouble()
String	next()

Método de Entrada

Exemplo para ler um inteiro:

```
int valor;  
Scanner scan = new Scanner(System.in);  
valor = scan.nextInt();
```


Terceiro Programa em JAVA

```
import java.util.Scanner;
public class OhVida
{
    public static void main(String[] args)
    {
        int tempo;
        String universidade;
        Scanner scan = new Scanner(System.in);
        System.out.print("Onde voce estuda: ");
        universidade = scan.next();
        System.out.print("\nQuanto tempo: ");
        tempo = scan.nextInt();
        System.out.println("\nVoce estuda na " + universidade + " ha "
                           + tempo + " anos.");
    }
}
```

Comando para “compilar”: `javac OhVida.java`

Comando para “executar”: `java OhVida`

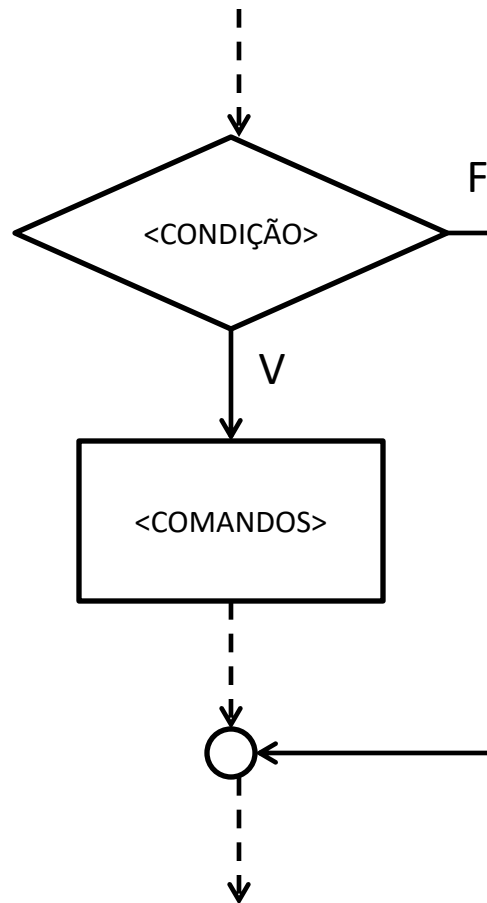
Operadores Aritméticos

- A tabela a seguir mostra os operadores aritméticos e suas funções.

Operador	Função	Tipo	Resultado
+	Manutenção de sinal	Unário	
-	Inversão de sinal	Unário	
+	Adição	Binário	Inteiro ou Real
-	Subtração	Binário	Inteiro ou Real
*	Multiplicação	Binário	Inteiro ou Real
/	Divisão	Binário	Inteiro ou Real
%	Resto da Divisão	Binário	Inteiro

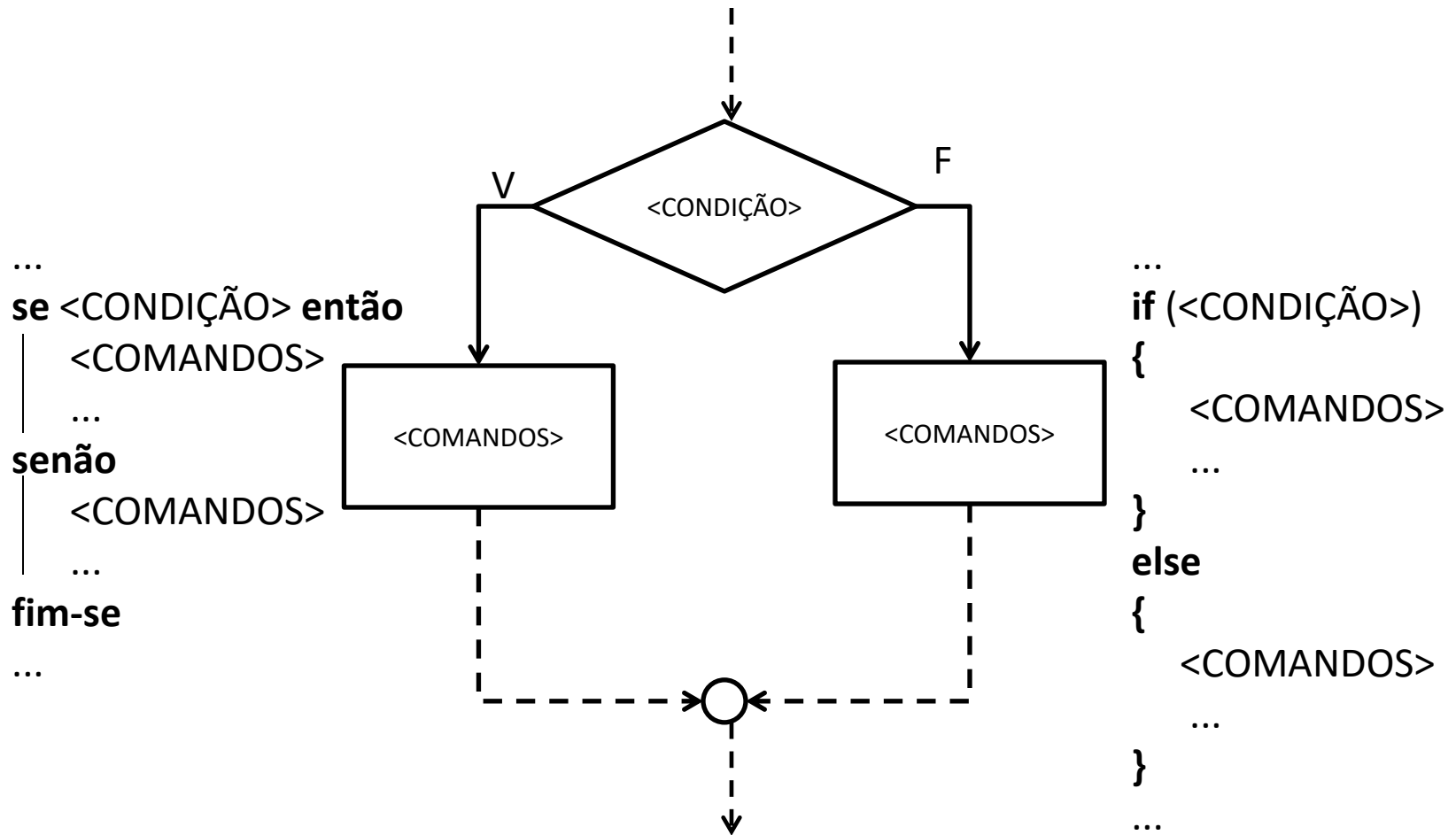
Estrutura Condicional Simples

```
...  
se <CONDIÇÃO> então  
|   <COMANDOS>  
|   ...  
fim-se  
...
```



```
...  
if (<CONDIÇÃO>)  
{  
    <COMANDOS>  
    ...  
}  
...
```

Estrutura Condicional Composto



Operadores Relacionais

- São operadores que operam sobre variáveis numéricas ou expressões aritméticas e retornam valores lógicos.

Operador	Operação	Descrição
>	Maior que	Verifica se um valor é maior a outro.
<	Menor que	Verifica se um valor é menor a outro.
>=	Maior ou igual a	Verifica se um valor é maior ou igual a outro.
<=	Menor ou igual a	Verifica se um valor é menor ou igual a outro.
==	Igualdade	Verifica se dois valores são iguais.
!=	Diferença	Verifica se dois valores são diferentes

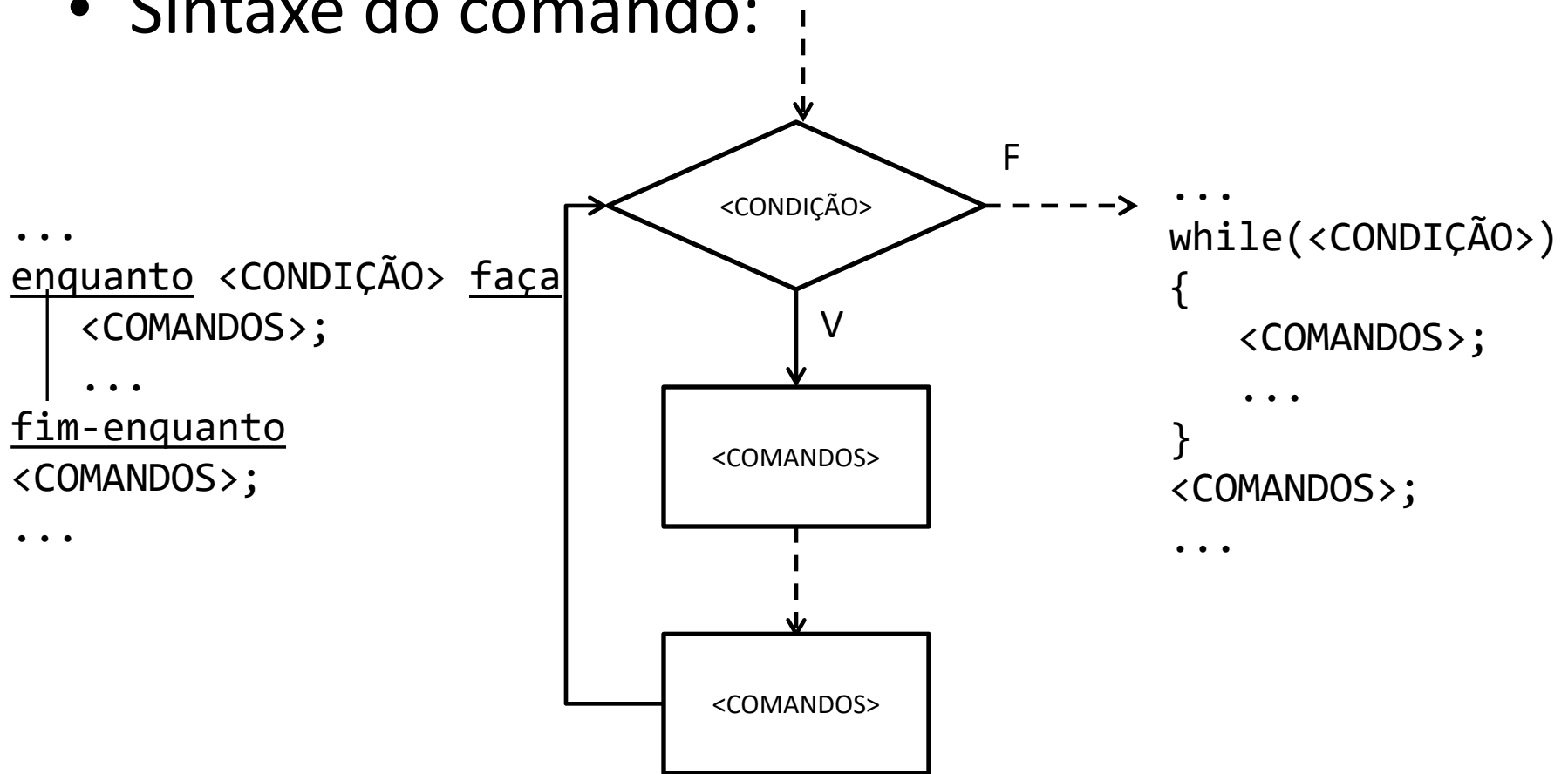
Operadores Lógicos

- Quando você precisa criar uma condição **com mais de uma** expressão relacional, então usa-se os operadores lógicos.

Operador	Operação	Descrição
&&	E lógico	Retorna verdade apenas se os dois componentes forem verdadeiros.
	Ou Lógico	Retorna falso apenas se os dois componentes forem falsos.
!	Negação	Contradiz o argumento.

Estrutura de Repetição

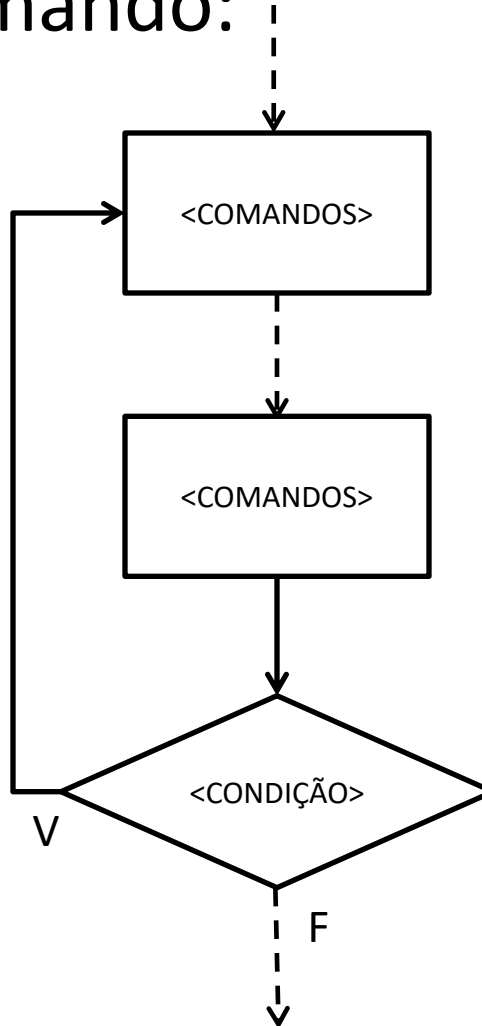
- Sintaxe do comando:



Estrutura de Repetição

- Sintaxe do comando:

...
faça <COMANDOS>
 ...
até que <CONDIÇÃO>
<COMANDOS>
...

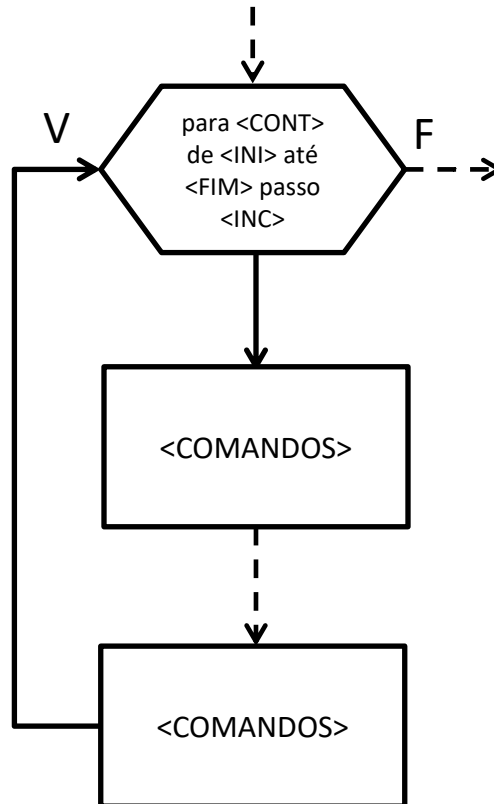


...
do
{
 <COMANDOS>;
 ...
}while(<CONDIÇÃO>);
<COMANDOS>;
...

Estrutura de Repetição

- Sintaxe do comando:

...
para <VAR> de <INI> até <FIM> passo <INC> faça
|
<COMANDOS>
|
...
fim-para
<COMANDOS>
...



...
for(<INI>;<COND>;<INC>)
{
 <COMANDOS>;
 ...
}
<COMANDOS>;
...

Variáveis Acumuladoras e Contadores

- São variáveis que acumulam valores
- Acumula seu valor anterior, mais outro valor
- No caso dos contadores o valor adicionado é sempre uma constante.
- Costumam ser usadas em ciclos para acumularem valores
- As variáveis acumuladoras devem SEMPRE ser inicializadas com um valor NULO.

Operadores de Incremento e Decremento

Operador	Tipo	Descrição
++	Unário	Incremento Unitário
--	Unário	Decremento Unitário
+=	Binário	Atribuição por Adição
-=	Binário	Atribuição por Subtração
*=	Binário	Atribuição por Multiplicação
/=	Binário	Atribuição por Divisão
%=	Binário	Atribuição por Módulo

Vetores

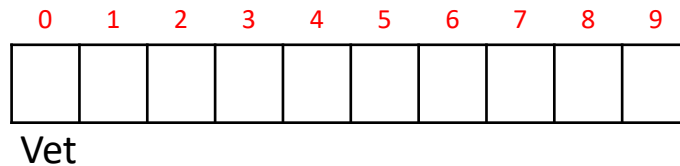
- Declaração

- Sintaxe:

<tipo> nome_da_variavel [] = new **<tipo>**[tamanho];

- Exemplo: declaração de um vetor de inteiros chamado Vet com 10 elementos.

int vet[] = new **int**[10];



- Exemplo: declaração de um vetor de flutuantes chamado pesos com 100 elementos.

float pesos[] = new **float**[100];

Vetores

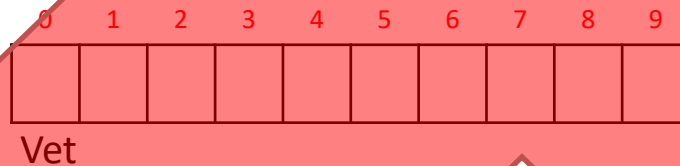
- Declaração

- Sintaxe:

<tipo> nome_da_variavel [] = new **<tipo>**[tamanho];

- Exemplo: declaração de um vetor de inteiros chamado Vet com 10 elementos.

int vet[] = new **int**[10];



- Exemplo: declaração de um vetor de flutuantes chamado pesos com 100 elementos.

float pesos[] = new **float**[100];

Vetores

- **Declaração:**

- Sintaxe: **<tipo>** nome_da_variavel [];

- **Instanciação:**

- Sintaxe: nome_da_variavel [] = new **<tipo>**[tamanho];

- Exemplo: declaração e **instanciação** de um vetor de inteiros chamado Vet com 10 elementos.

int vet[] = new **int**[10];



Vet

- Exemplo: declaração e **instanciação** de um vetor de flutuantes chamado pesos com 100 elementos.

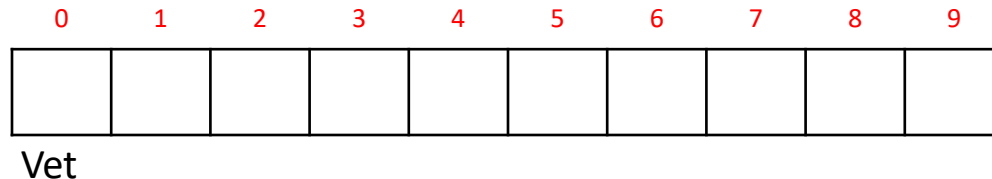
float pesos[] = new **float**[100];

Vetores

- Acessando os índices de um vetor

– Sintaxe:

nome_da_variavel [**<indice>**] = valor;



Vetores

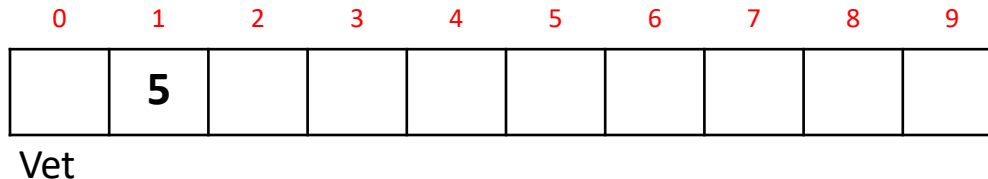
- Acessando os índices de um vetor

- Syntax:

nome_da_variavel [<indice>**] = valor;**

- Exemplo: atribuir 5 ao vetor Vet no índice 1:

Vet[1] = 5;



Vet

Vetores

- Acessando os índices de um vetor

- Syntax:

nome_da_variavel [**<indice>**] = valor;

- Exemplo: atribuir 5 ao vetor Vet no índice 1:

Vet[1] = 5;

- **Exemplo:** atribuir -7 ao vetor Vet no índice 0:

Vet[0] = -7;

0	1	2	3	4	5	6	7	8	9
-7	5								

Vet

Vetores

- Lendo e imprimindo valores a partir de um vetor:
 - Igual se faz com uma variável não vetor, porém jamais se esqueça de informar o índice.
 - Leitura:
 - Ler um flutuante via teclado e armazenar no índice 5 do vetor pesos.

```
pesos[5] = scan.nextFloat();
```

- Impressão:
 - Imprimir na tela o valor armazenado no índice 3 do vetor pesos.

```
System.out.println(pesos[3]);
```

Vetores

- Como manipular eficientemente vetores?

Vetores

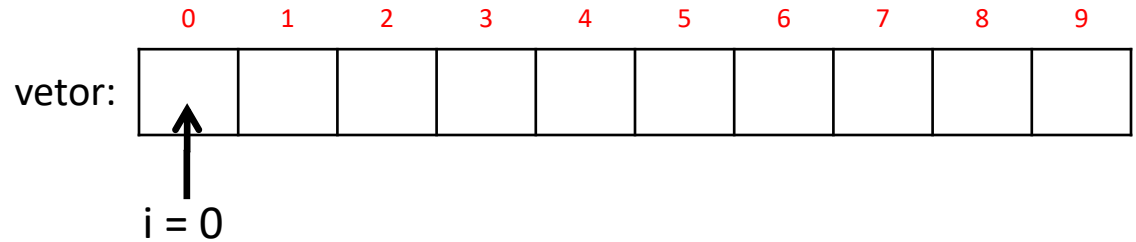
- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```

Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

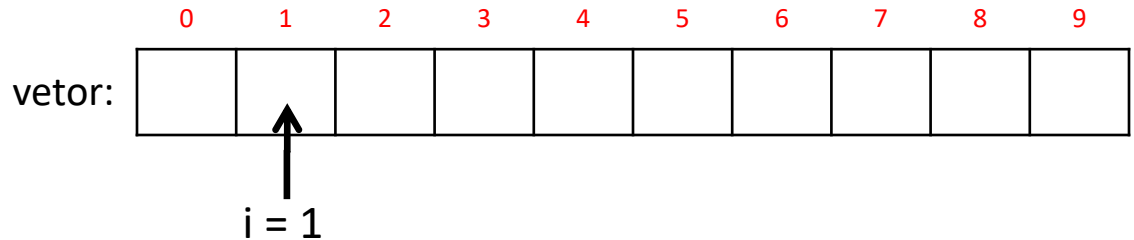
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

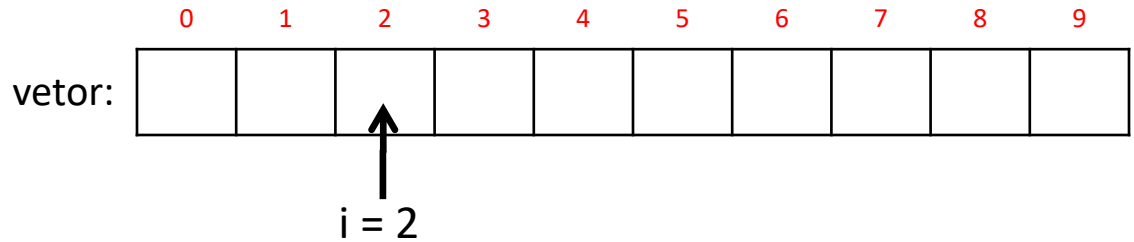
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

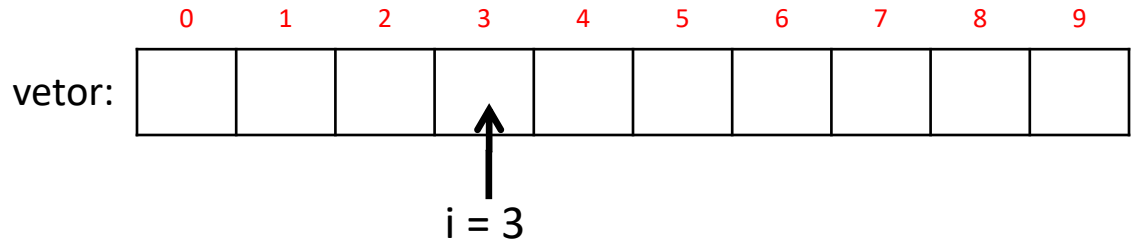
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

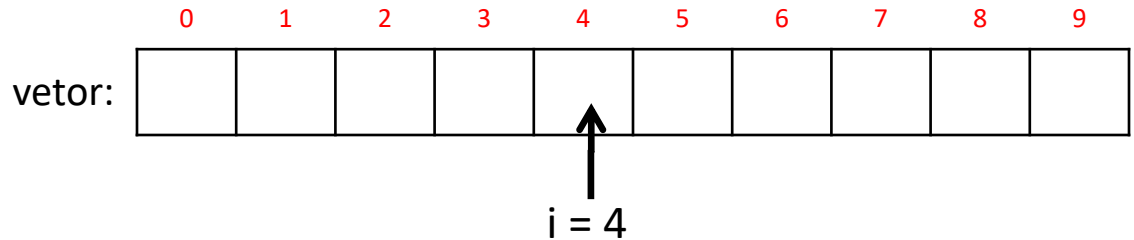
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

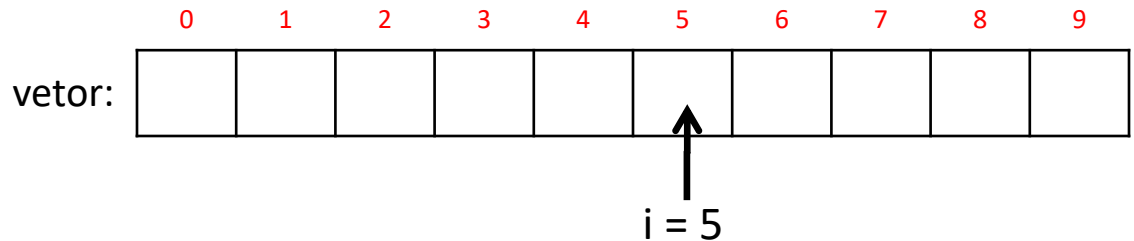
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

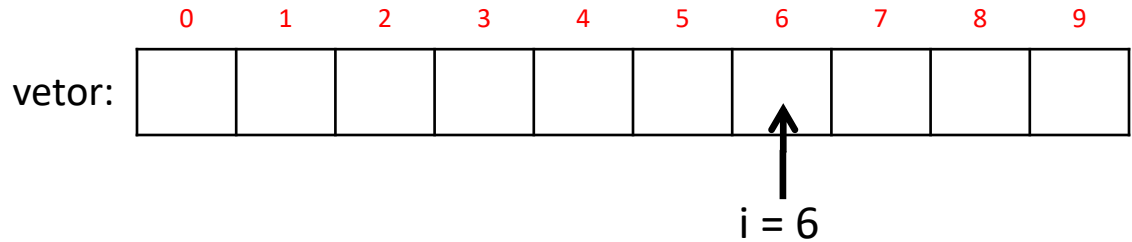
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

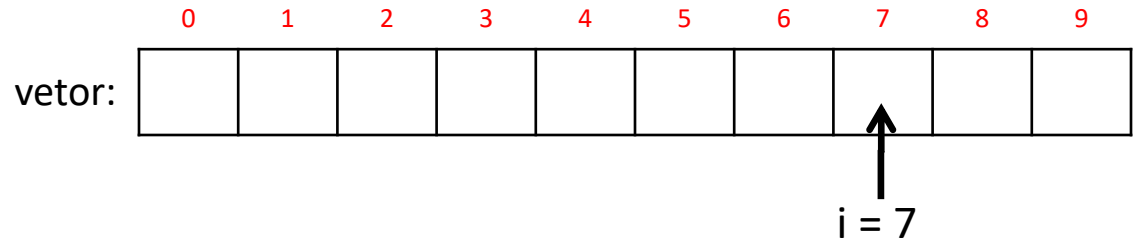
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

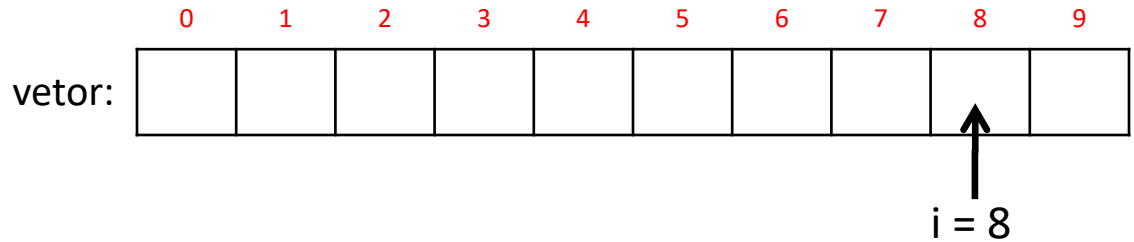
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

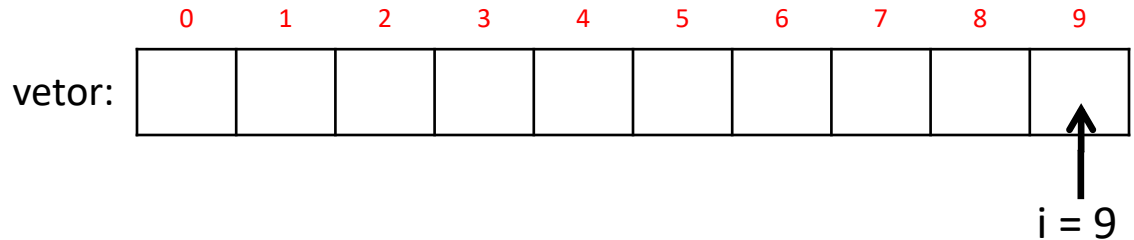
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

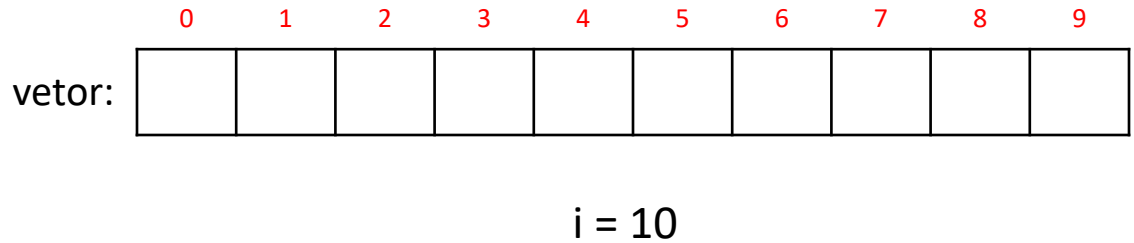
```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - O que faz o programa a seguir ?

```
public static void main(String[] args)
{
    int i, vetor[] = new int[10];
    for(i=0; i<10; i++)
    {
        vetor[i];
    }
}
```



Vetores

- Como manipular eficientemente vetores?
 - Lendo valores via teclado...

Vetores

- Como manipular eficientemente vetores?
 - Lendo valores via teclado...

```
import java.util.Scanner;
public class Exemplo
{
    public static void main(String[] args)
    {
        int i, vetor[] = new int[10];
        Scanner scan = new Scanner(System.in);
        for(i=0; i<10; i++)
        {
            vetor[i] = scan.nextInt();
        }
    }
}
```

Vetores

- Como manipular eficientemente vetores?
 - Imprimindo valores na tela ...

Vetores

- Como manipular eficientemente vetores?
 - Imprimindo valores na tela ...

```
import java.util.Scanner;
public class Exemplo
{
    public static void main(String[] args)
    {
        int i, vetor[] = new int[10];
        Scanner scan = new Scanner(System.in);
        for(i=0; i<10; i++)
        {
            System.out.print(vetor[i] + " ");
        }
    }
}
```

Vetores

- Como manipular eficientemente vetores?
 - Somando os valores de cada índice ...

Vetores

- Como manipular eficientemente vetores?
 - Somando os valores de cada índice ...

```
import java.util.Scanner;
public class Exemplo
{
    public static void main(String[] args)
    {
        int i, soma=0, vetor[] = new int[10];
        Scanner scan = new Scanner(System.in);
        for(i=0; i<10; i++)
        {
            soma = soma + vetor[i];
        }
    }
}
```

Vetores

- Cuidados
 - **Jamais** ultrapasse os limites do vetor.
 - Linguagem JAVA trata a extrapolação dos limites do vetor. Deste modo, caso você ultrapasse os limites estabelecidos ascenderá uma excessão: `ArrayOutOfBoundsException`.

Vetores

- Passagem de Parâmetros
 - Por padrão na Linguagem JAVA, todo vetor é automaticamente passado por referência.
 - Atenção:
 - É necessário explicitar o recebimento de um vetor no “cabeçalho da função”.

Vetores

- Passagem de Parâmetros
 - Por padrão na Linguagem JAVA, todo vetor é automaticamente passado por referência, pois ...
 - Atenção:
 - É necessário explicitar o recebimento de um vetor no ~~“cabeçalho da função”~~.

Vetores

- Passagem de Parâmetros
 - Por padrão na Linguagem JAVA, todo vetor é automaticamente passado por referência, **pois todo vetor em JAVA é um objeto.**
 - Atenção:
 - É necessário explicitar o recebimento de um vetor na **assinatura do método.**

Vetores

- Passagem de Parâmetros (exemplo)
 - Suponha a função `exVet()` que recebe um vetor de inteiros e um inteiro.

- Invocando `exVet()`

`exVet (vetor, n);`

Não é necessário explicitar a passagem de um vetor.

- Cabeçalho da função `exVet()`

`void exVet (int[] v, int n);`

Entretanto é necessário explicitar o recebimento de um vetor.

Matrizes

- Declaração:

- Sintaxe:

<tipo> nome_da_variavel[][] = new **<tipo>** [<tamanho>] [<tamanho>];

- Exemplo: declaração de uma matriz de inteiros chamado Mat com 12 elementos.

int Mat[][] = new **int**[3][4];

	0	1	2	3
0				
1				
2				

- Exemplo: declaração de uma matriz de flutuantes chamado dados com 1050 elementos, em 30 linhas e 35 colunas.

float dados[][] = new **float**[30][35];

Matrizes

- Declaração:

- Sintaxe:

<tipo> nome_da_variavel[][] = new **<tipo>** [**<tamanho>**][**<tamanho>**];

- Exemplo: declaração de uma matriz de inteiros chamado Mat com 12 elementos.

int Mat[][] = new **int**[3][4];

	0	1	2	3
0				
1				
2				

- Exemplo: declaração de uma matriz de flutuantes chamado dados com 1050 elementos, em 30 linhas e 35 colunas.

float dados[][] = new **float**[30][35];

Matrizes

- **Declaração:**

- Sintaxe: **<tipo>** nome_da_variavel[][];

- **Instanciação:**

- Sintaxe: nome_da_variavel[][] = new **<tipo>** [<tamanho>] [<tamanho>];

- Exemplo: declaração e **instanciação** de uma matriz de inteiros chamado Mat com 12 elementos.

```
int Mat[][] = new int[3][4];
```

	0	1	2	3
0				
1				
2				

- Exemplo: declaração e **instanciação** de uma matriz de flutuantes chamado dados com 1050 elementos, em 30 linhas e 35 colunas.

```
float dados[][] = new float[30][35];
```

Matrizes

- Acessando os valores de uma matriz

Mat:

	0	1	2	3
0				
1				
2				

Matrizes

- Acessando os valores de uma matriz
 - Sintaxe:
nome_da_variavel [**<linha>**][**<coluna>**] = valor;

Mat:

	0	1	2	3
0				
1				
2				

Matrizes

- Acessando os valores de uma matriz

– Sintaxe:

nome_da_variavel [**<linha>**][**<coluna>**] = valor;

– Exemplo: atribuir 5 a matriz Mat na linha 1 coluna 2.

Mat[**1**][**2**] = 5;

Mat:

	0	1	2	3
0				
1			5	
2				

Matrizes

- Acessando os valores de uma matriz

- Sintaxe:

- `nome_da_variavel [<linha>][<coluna>] = valor;`

- Exemplo: atribuir 5 a matriz Mat na linha 1 coluna 2.

- `Mat[1][2] = 5;`

- Exemplo: atribuir -7 a matriz Mat na linha 0 coluna 1:

- `Mat[0][1] = -7;`

Mat:

	0	1	2	3
0		-7		
1			5	
2				

Matrizes

- Lendo e imprimindo valores a partir de uma matriz:
 - Igual se faz com um vetor, porém jamais se esqueça de informar os índices linha e coluna.
 - Leitura:
 - Ler um flutuante via teclado e armazenar na linha 3, coluna 4 da matriz valores.

```
valores[3][4] = scan.nextFloat();
```

- Impressão:
 - Imprimir na tela o valor armazenado na linha 5 coluna 7 da matriz val.

```
System.out.println (val[5][7]);
```

Matrizes

- Como manipular eficientemente matrizes?

Matrizes

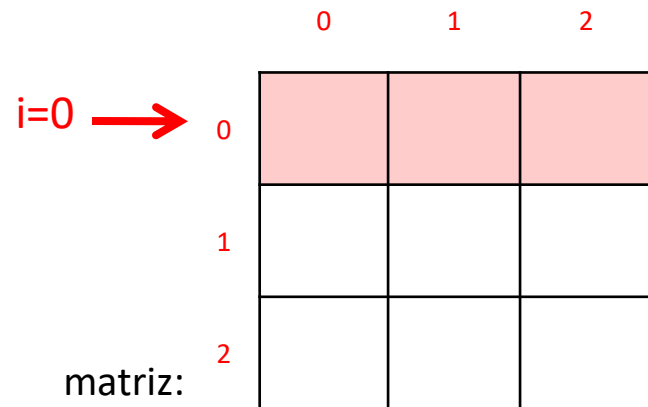
- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```

Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

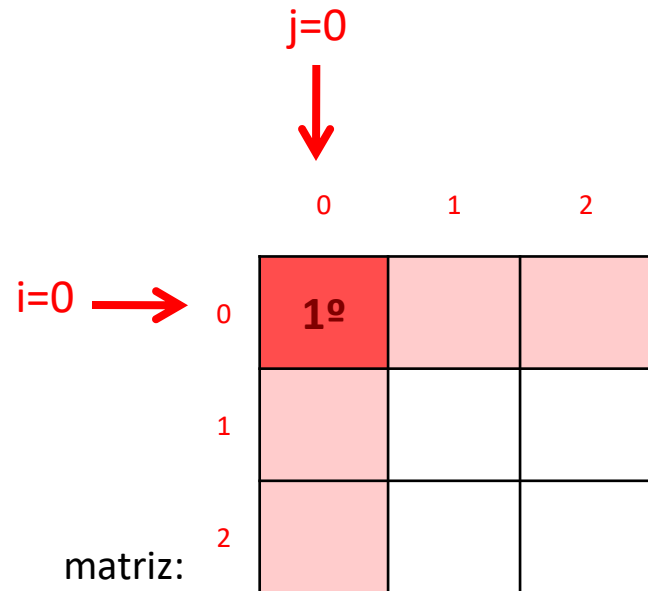
```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```



Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

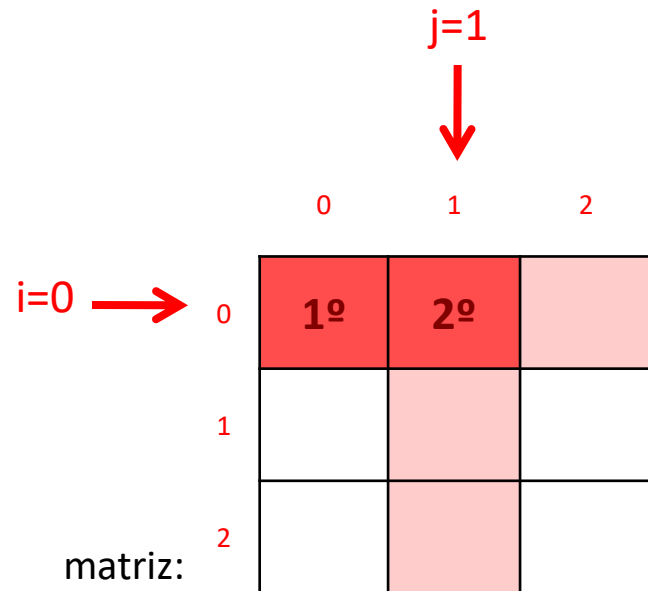
```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```



Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

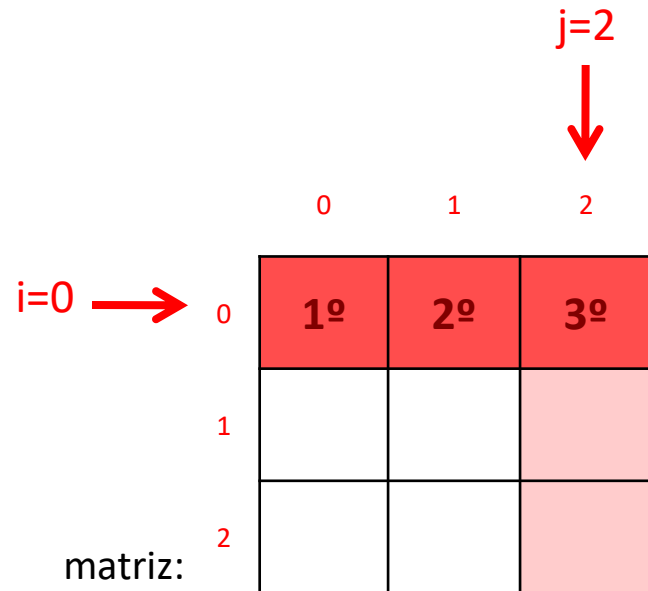
```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```



Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

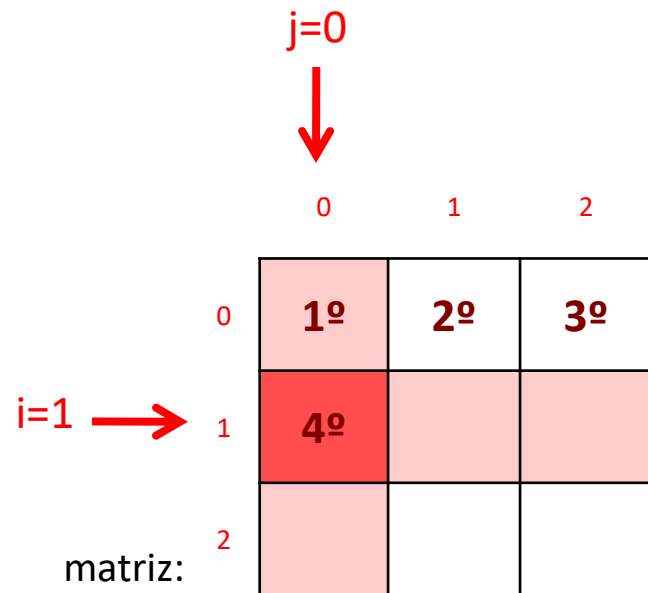
```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```



Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

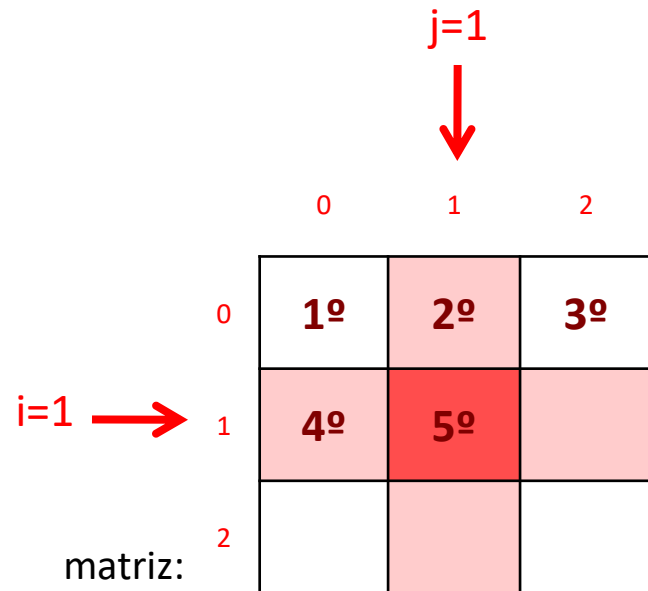
```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```



Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```



Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```

Diagram illustrating a 3x3 matrix with indices *i* and *j*.

Indices: *i*=1 (row 1), *j*=2 (column 2).

Matrix content (row-major order):

	0	1	2
0	1º	2º	3º
1	4º	5º	6º
2			

matriz:

Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```

j=0

↓

012

012

1º	2º	3º
4º	5º	6º
7º		

i=2→

2matriz:

Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```

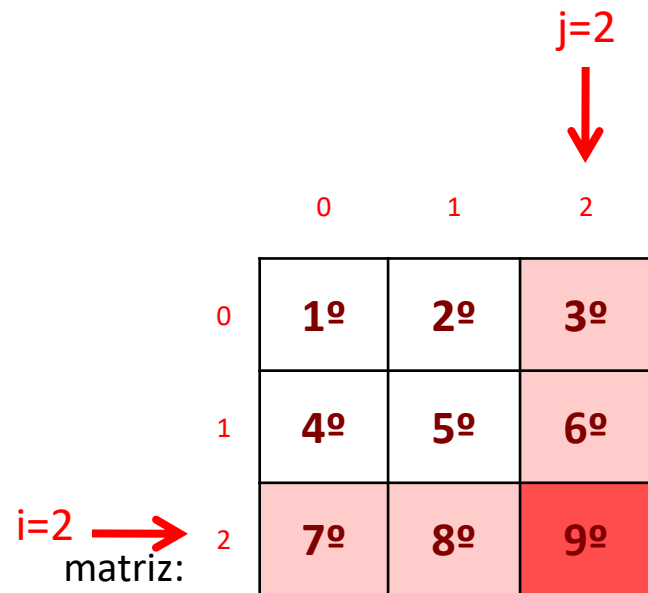
	0	1	2
0	1º	2º	3º
1	4º	5º	6º
2	7º	8º	

i=2 → matriz:

Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```



	0	1	2
0	1º	2º	3º
1	4º	5º	6º
2	7º	8º	9º

Matrizes

- Como manipular eficientemente matrizes?
 - O que faz o programa a seguir ?

```
1. public static void main(String[] args)
2. {
3.     int matriz[][] = new int[3][3], i, j;
4.     for(i=0; i<3; i++)
5.     {
6.         for(j=0; j<3; j++)
7.         {
8.             matriz[i][j];
9.         }
10.    }
11. }
```

matriz:

	0	1	2
0	1º	2º	3º
1	4º	5º	6º
2	7º	8º	9º

i=3

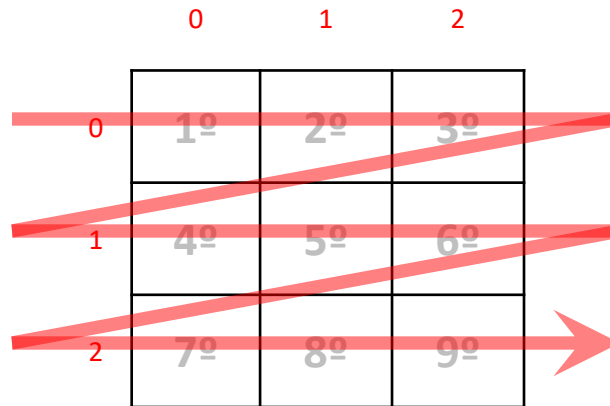
Matrizes

- Ordem de visita aos elementos
 - No código anterior a ordem de visitação foi essa.

	0	1	2
0	1º	2º	3º
1	4º	5º	6º
2	7º	8º	9º

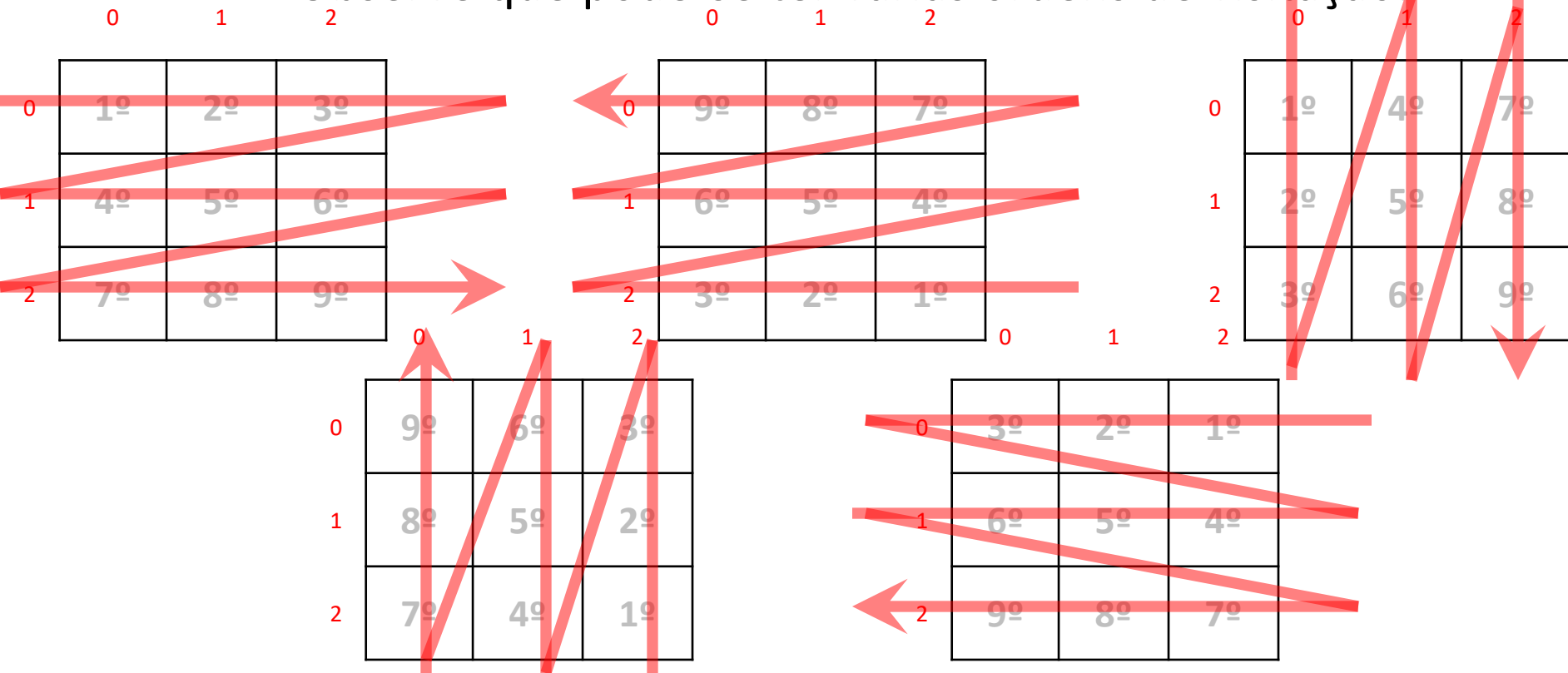
Matrizes

- Ordem de visita aos elementos
 - No código anterior a ordem de visitação foi essa.



Matrizes

- Ordem de visita aos elementos
 - No código anterior a ordem de visitação foi essa.
 - Observe que pode-se ter várias ordens de visitação.



Matrizes

- Cuidados
 - **Jamais** ultrapasse os limites da matriz.
 - Linguagem JAVA trata a extrapolação dos limites de uma matriz ascendendo uma exceção:

`java.lang.ArrayIndexOutOfBoundsException`

Matrizes

- Passagem de Parâmetros (exemplo)
 - Suponha a função `exMat()` que recebe uma matriz de inteiros.

- Invocando `exMat()`

`exMat (matriz, n, m);`

Não é necessário explicitar a matriz

- Cabeçalho da função `exMat()`

`public static void exMat (int m[][], int n, int m);`

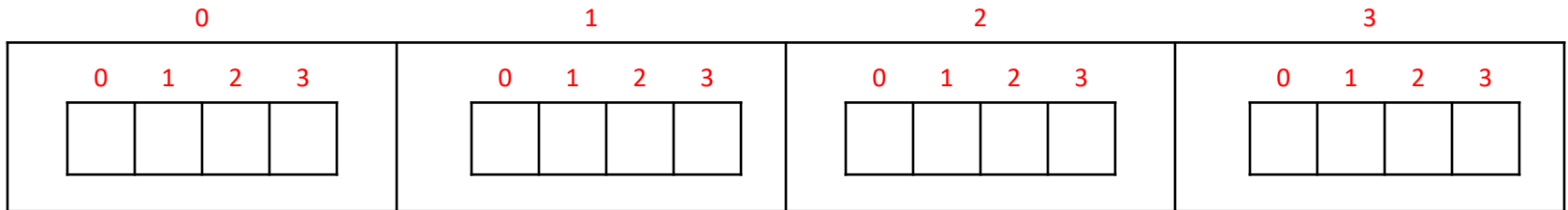
É necessário explicitar.

Matrizes

- Internamente
 - Uma matriz pode ser entendida como um vetor unidimensional, onde cada item deste vetor é outro vetor unidimensional.

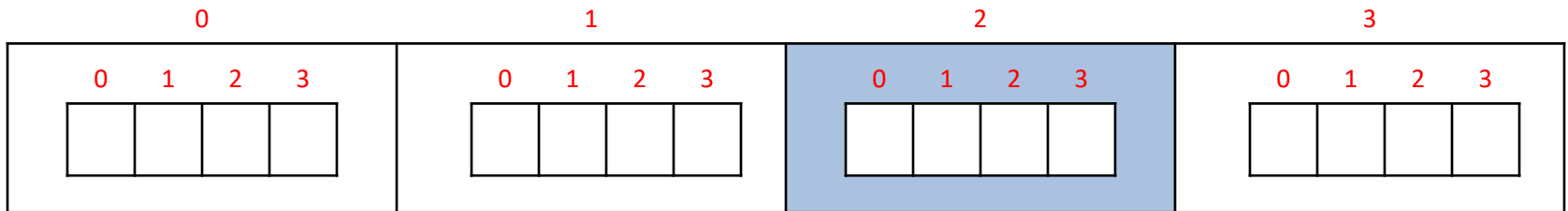
Matrizes

- Internamente
 - A matriz pode ser vista como um vetor de vetores.



Matrizes

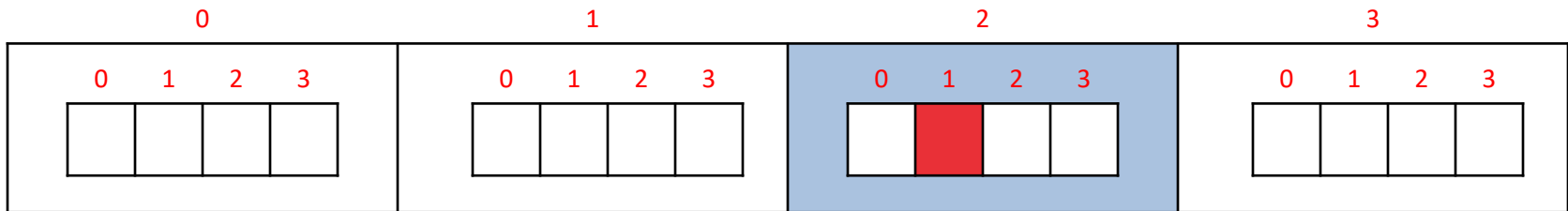
- Internamente
 - A matriz pode ser vista como um vetor de vetores.



- Ao referenciar a matriz na linha 2 coluna 1:
 - a linha resolve o índice do primeiro vetor.

Matrizes

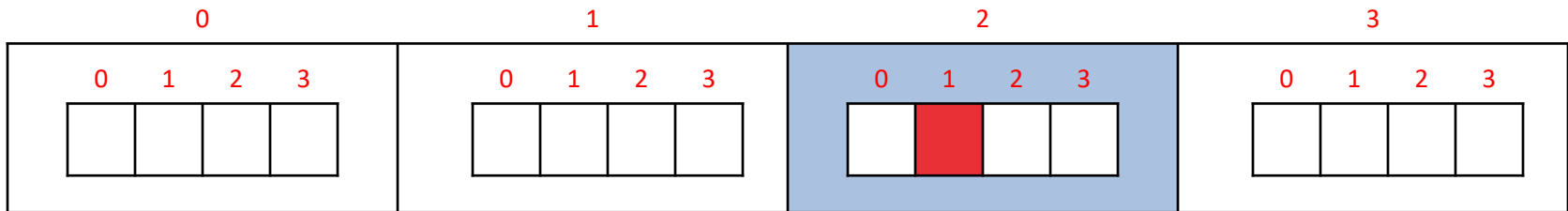
- Internamente
 - A matriz pode ser vista como um vetor de vetores.



- Ao referenciar a matriz na linha 2 coluna 1:
 - a linha resolve o índice do primeiro vetor.
 - Enquanto a coluna resolve o índice do segundo vetor.

Matrizes

- Internamente
 - A matriz pode ser vista como um vetor de vetores.



- Ao referenciar a matriz na linha 2 coluna 1:
 - a linha resolve o índice do primeiro vetor.
 - Enquanto a coluna resolve o índice do segundo vetor.
- Desta forma podemos tratar a matriz como um conjunto de vetores.

Matrizes

- Internamente
 - Desta forma podemos tratar a matriz como um conjunto de vetores.
 - Para isto, basta omitirmos a segunda dimensão da matriz e então teríamos um vetor.
 - Deste modo, ao referenciar `Mat[1]`, temos:

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							

O vetor formado pela linha 1 da matriz Mat

Matrizes

- Internamente
 - Desta forma podemos tratar a matriz como um conjunto de vetores.
 - Para isto, basta omitirmos a segunda dimensão da matriz e então teríamos um vetor.
 - Deste modo, ao referenciar `Mat[3]`, temos:

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							

O vetor formado pela linha 3 da matriz Mat

Matrizes

- Internamente
 - Desta forma podemos tratar a matriz como um conjunto de vetores.
 - Para isto, basta omitirmos a segunda dimensão da matriz e então teríamos um vetor.
 - Desta maneira, pode-se obter este vetor e tratá-lo como um vetor tradicional.
 - Vejamos um exemplo ...


Matrizes

- Usando uma matriz como um conjunto de vetores.
 - Agora, usando a função somaVet() feita anteriormente, faça um programa que exiba na tela a somatória de cada linha da matriz a seguir:
- ```
1. public static void main(String[] args) {
2. int mat[][] = new int ...
3. ...
4. for(i=0; i<n; i++) {
5. System.out.print("\nLinha " + i + ", soma" + somaVet(mat[i], n, m));
6. }
7. }
```

# Matrizes

- Usando uma matriz como um conjunto de vetores.
  - Agora, usando a função somaVet() feita anteriormente, faça um programa que exiba na tela a somatória de cada linha da matriz a seguir:

```
1. public static void main(String[] args) {
2. int mat[][] = new int ...
3. ...
4. for(i=0; i<n; i++) {
5. System.out.print("\nLinha " + i + ", soma" + somaVet(mat[i], n, m));
6. }
7. }
```



Aqui espera-se um vetor...  
E isso é um vetor...