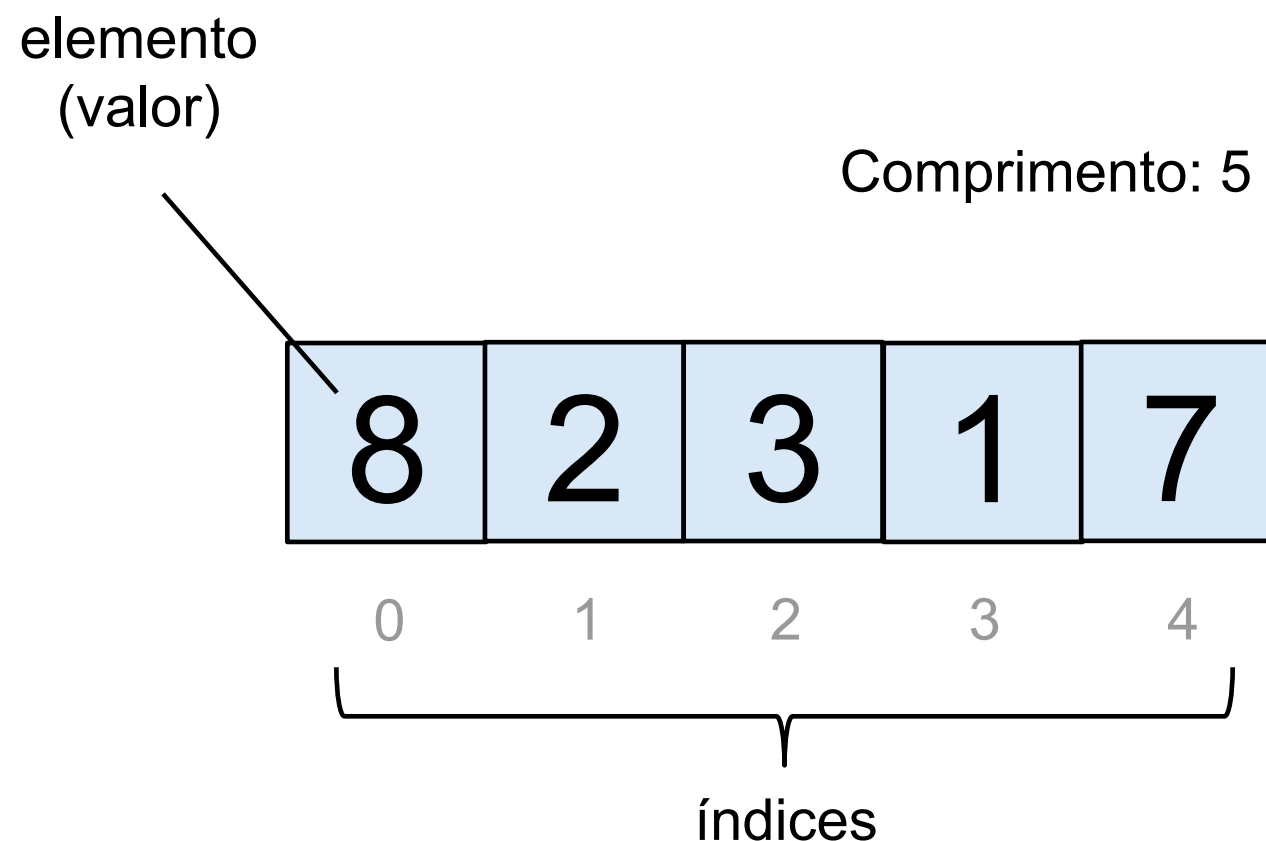


VETORES

VETORES

- Um vector é uma **estrutura de dados elementar**, capaz de armazenar uma **coleção de elementos ordenada**.
- Os acessos ao **elementos do vetor** são efectuados mediante um **índice**.



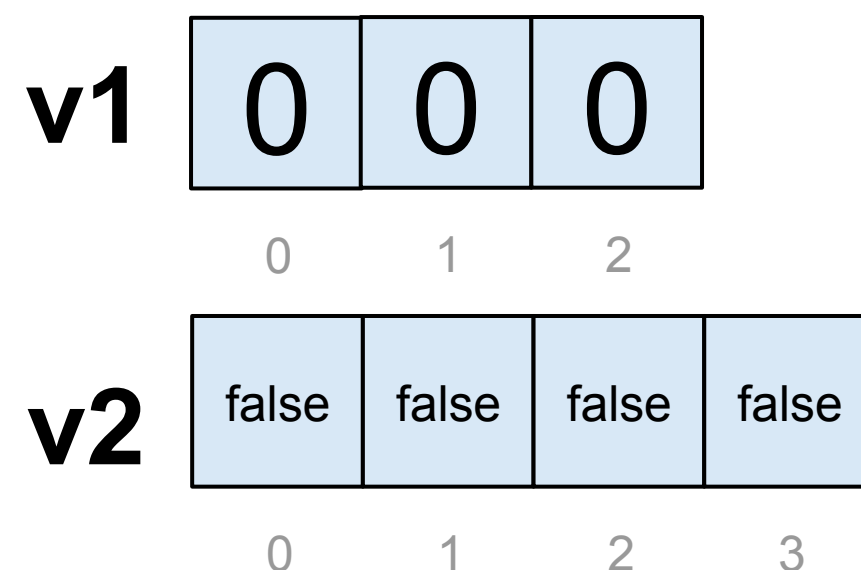
CRIAÇÃO DE VETORES

- Em Java, os vetores têm **comprimento fixo**. Desta forma, ao ser criado um vetor é necessário indicar um comprimento, o qual **não pode ser alterado após a criação**.

```
int[] v1 = new int[3];
```

```
boolean[] v2 = new boolean[4];
```

Após a criação os elementos do vetor tomam um **valor por omissão**. Para os tipos numéricos, o valor por omissão é **zero**. Para o tipo boolean, o valor por omissão é **false**.



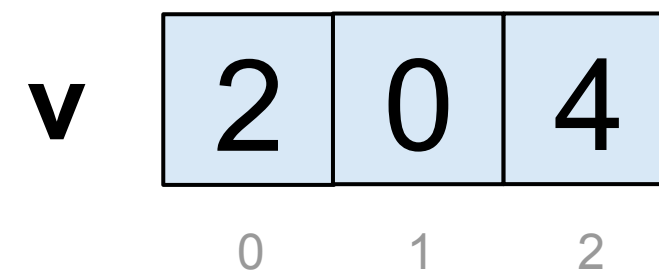
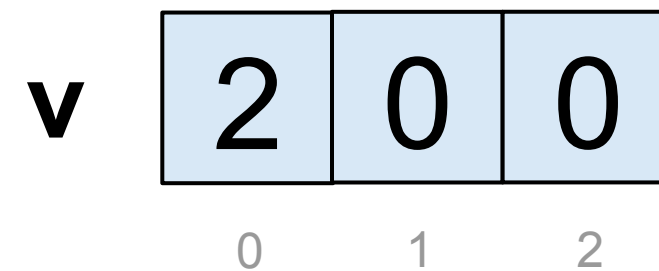
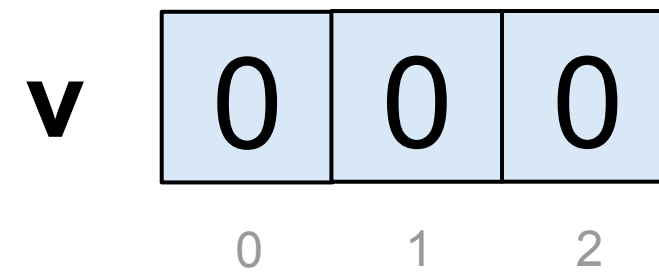
MANIPULAÇÃO DE VETORES: MODIFICAÇÃO

- A **modificação** dos elementos do vetor é feita mediante um **índice**.

```
int[] v = new int[3];
```

```
v[0] = 2;
```

```
v[2] = 4;
```



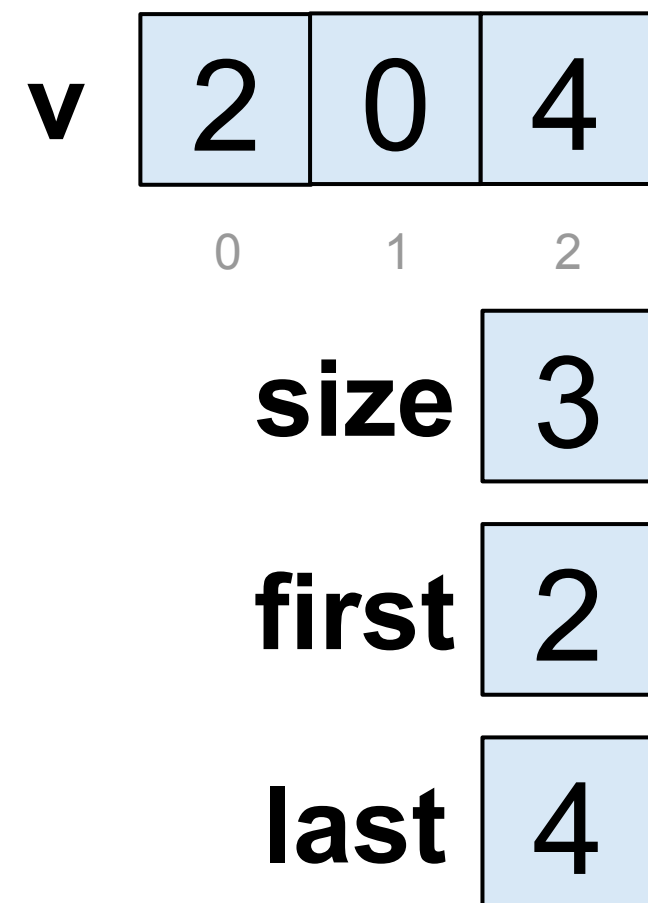
MANIPULAÇÃO DE VETORES: ACESSO

- O **comprimento** de um vector pode ser obtido através do atributo `length`.
- O último elemento tem índice igual ao comprimento menos um.

```
int size = v.length;
```

```
int first = v[0];
```

```
int last = v[v.length - 1];
```



OPERAÇÕES COM VETORES: ITERAÇÃO

- O processo de **percorrer os elementos de um vetor** pode ser caracterizado como uma **iteração**. Tipicamente, a iteração faz uso de uma variável que toma sucessivamente os valores dos índices do vetor cujos elementos se quer aceder (variável **i**).

```
static int sum(int[] v) {  
    int sum = 0;  
    int i = 0;  
    while(i != v.length) {  
        sum = sum + v[i];  
        i = i + 1;  
    }  
    return sum;  
}
```

OPERAÇÕES COM VETORES: ACUMULAÇÃO

- O seguinte exemplo combina uma iteração sobre o vetor e uma **acumulação** para calcular o somatório de todos os elementos (variável **sum**).

```
static int sum(int[] v) {  
    int sum = 0;  
    int i = 0;  
    while(i != v.length) {  
        sum = sum + v[i];  
        i = i + 1;  
    }  
    return sum;  
}
```

OPERAÇÕES COM VETORES:

CONTAGEM

- O seguinte exemplo combina uma iteração sobre o vetor e uma **contagem** para devolver o número de ocorrências de determinado número no vetor (variável count).

```
static int numberOfOccurrences(int a, int[] v) {  
    int count = 0;  
    int i = 0;  
    while(i != v.length) {  
        if(v[i] == a) {  
            count = count + 1;  
        }  
        i = i + 1;  
    }  
    return count;  
}
```


OPERAÇÕES COM VETORES: PESQUISA

- O seguinte exemplo efectua uma iteração sobre o vetor para realizar uma **pesquisa** de determinado número no vetor.

```
static boolean exists(int a, int[] v) {  
    int i = 0;  
    while(i != v.length) {  
        if(v[i] == a) {  
            return true;  
        }  
        i = i + 1;  
    }  
    return false;  
}
```

OPERAÇÕES COM VETORES: MÁXIMO

- O seguinte exemplo efetua uma iteração sobre os elementos do vetor para encontrar o valor **máximo** (variável **max**).

```
static int max(int[] v) {  
    int max = v[0];  
    int i = 1;  
    while(i != v.length) {  
        if(v[i] > max) {  
            max = v[i];  
        }  
        i = i + 1;  
    }  
    return max;  
}
```

FUNÇÕES NUMÉRICAS EM JAVA

- O Java oferece a classe estática **Math** (`java.lang.Math`) que contém diversas funções úteis para cálculo numérico
 - Módulo (valor absoluto):
`Math.abs(int/double)`
 - Raiz quadrada:
`Math.sqrt(double)`
 - Potência:
`Math.pow(double, double)`
 - Seno:
`Math.sin(double)`
 - Coseno:
`Math.cos(double)`
 - Tangente:
`Math.tan(double)`
 - Logaritmo:
`Math.log(double)`
 - Máximo:
`Math.max(int/double, int/double)`
 - Mínimo:
`Math.min(int/double, int/double)`
 - Arredondamento:
`Math.round(double)`
 - Números aleatórios:
`Math.random()`

A RETER

- Vetores
 - Criação de vetores
 - Manipulação de vetores
 - Modificação
 - Acesso
 - Comprimento de um vetor
- Operações com vetores
 - Iteração
 - Acumulação
 - Contagem
 - Pesquisa
 - Máximo

