ARRAY

Vetor e Matriz

Array (unidimensionais e bidimensionais)

São as estruturas de dados mais simples e mais utilizadas dentre todas.

Principais características:

- Acesso aos elementos através de índices
- Possuem tamanho finito de elementos
- Carregam dados de tipos específicos
- Indexação com início em 0 (zero)
- Unidimensional: apenas uma linha
- Bidimensional: linhas e colunas(matriz)

Array

Um *array* ou *vetor* é uma estrutura de dados utilizada para armazenar uma coleção de itens. Cada item é identificado através de seu *índice*. Podemos imaginar um array como sendo um armário com um determinado número de gavetas e cada gaveta possui um rótulo com um número de identificação.

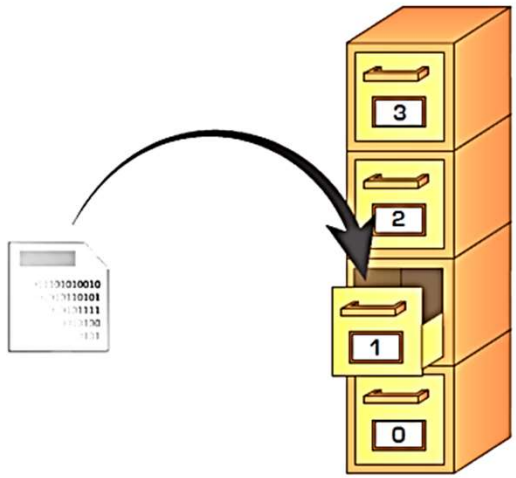
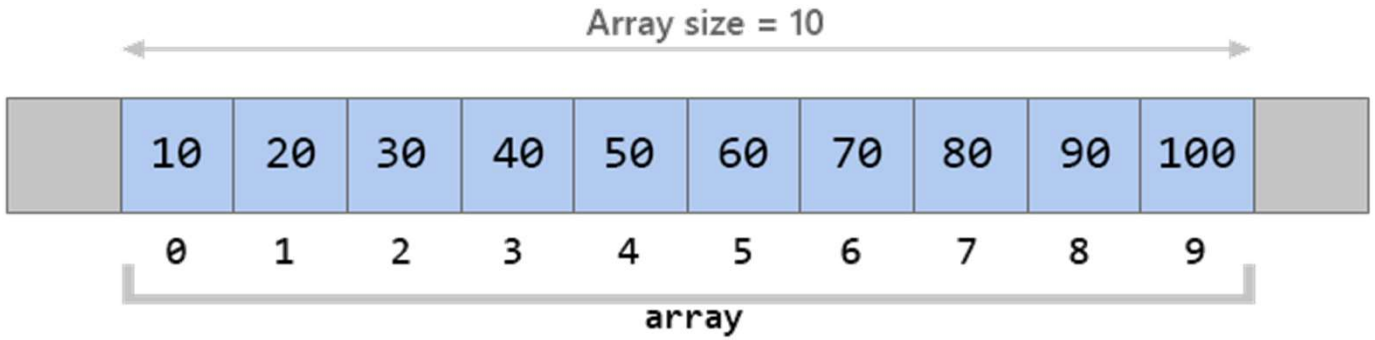


Figura 6.1: Abstração de array como um armário com gavetas.

Array

Um array é uma porção de memória fixa e sequencial dividida em pedaços idênticos indexados a partir do 0. Em cada posição do array, podemos guardar um valor.



Array - Java

Os arrays Java foram embutidos na linguagem para tratar dados que seja do mesmo tipo. Eles permitem que o desenvolvedor use uma única variável com um ou mais índices para acessar vários dados independentes.

Os arrays podem ser usados para armazenar tanto primitivos quanto objetos.

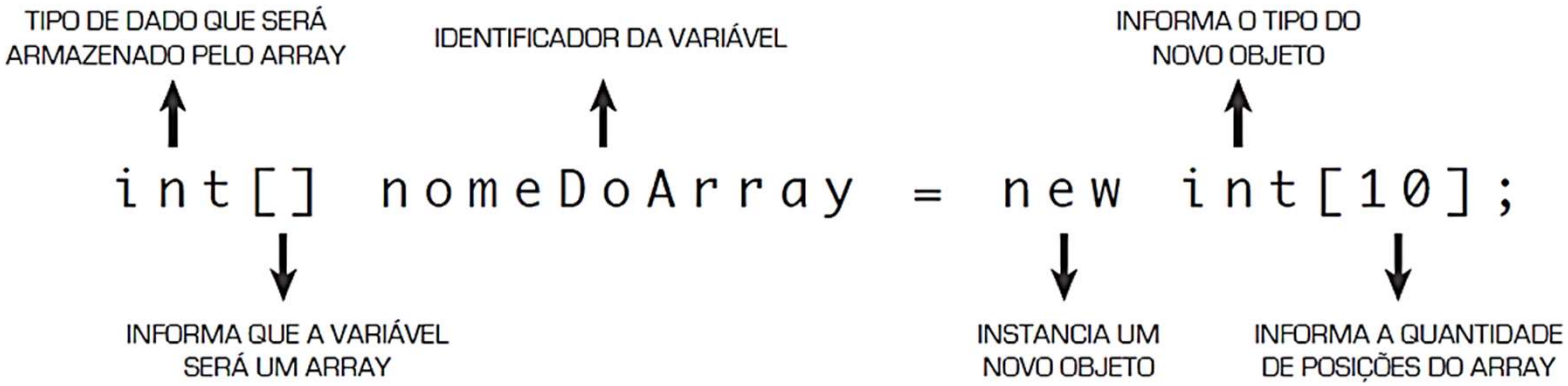
Array - Declaração

Em Java, podemos declarar um vetor de duas formas distintas:

```
int[] vetor;
```

```
int vetor[];
```

Array - Declaração e Iniciação



A capacidade de um array é fixa e deve ser informada no momento da criação do array.

Array - Declaração e Iniciação - Exemplo

```
int[] vetor = new int[6];
```

A capacidade de um array é fixa e deve ser informada no momento da criação do array.

Array - Declaração e Iniciação - Exemplo

Arrays não podem mudar de tamanho

A partir do momento que uma array foi criada, ela **não pode** mudar de tamanho.

Se você precisar de mais espaço, será necessário criar uma nova array e, antes de se referir ela, copie os elementos da array velha.

A capacidade de um array é fixa e deve ser informada no momento da criação do array.

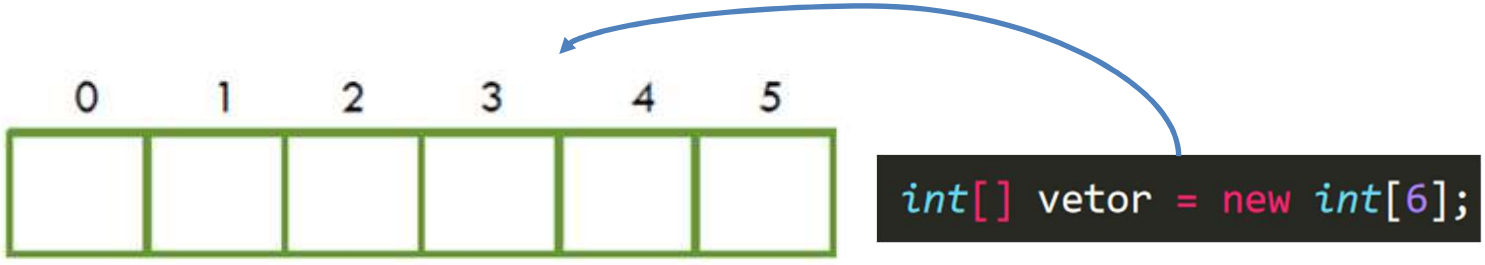
Array - Utilização

Cada item de um **array** é chamado de elemento, e cada elemento é acessado por um número, o índice.

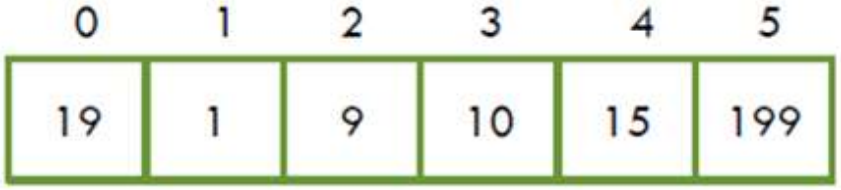
```
1
2   int[] vetor = new int[6];
3
4   vetor[0] = 19;
5   vetor[1] = 1;
6   vetor[2] = 9;
7   vetor[3] = 10;
8   vetor[4] = 15;
9   vetor[5] = 199;
10
```

Array - Utilização

Para inserir um valor em um array, devemos informar a posição/índice desejada(o).




```
vetor[0] = 19;
vetor[1] = 1;
vetor[2] = 9;
vetor[3] = 10;
vetor[4] = 15;
vetor[5] = 199;
```



Array - Utilização

Outra forma de inicializar um vetor:

```
int vetor2[] = {10,20,40};
```

```
String[] nomes = {"carlos", "larissa", "manoel"};
```

Array - Utilização

```
int[] intervalos = {2, 5, 6};
// x recebe 2
int x = intervalos[0];
// ERRO!
// Vetor de 3 elementos tem índice de 0 a 2
int y = intervalos[3];
```

Múltiplas Variáveis vs Vetor

Ex: Armazenamento de Temperaturas

```
double tempDia001 = 31.3;
double tempDia002 = 32;
double tempDia003 = 33.7;
double tempDia004 = 34;
double tempDia005 = 33.1;
```

Array - Visualizando

```
//DECLARA UM ARRAY DE INTEIROS
int[] meuArray;

//ALOCA MEMÓRIA PARA 10 INTEIROS
meuArray = new int[10];

//INICIALIZA O PRIMEIRO ELEMENTO
meuArray [0] = 100;
meuArray [1] = 85;
meuArray [2] = 88;
meuArray [3] = 93;
meuArray [4] = 123;
meuArray [5] = 952;
meuArray [6] = 344;
meuArray [7] = 233;
meuArray [8] = 622;
meuArray [9] = 8522;
//meuArray [10] = 564; //ESTOURA O ARRAY POIS NÃO EXISTE O ÍNDICE 10

System.out.println(meuArray[9]);
System.out.println(meuArray[2]);
```

DECLARAÇÃO

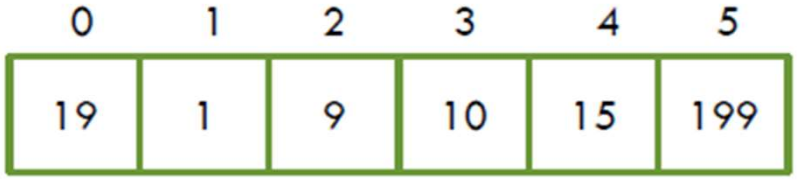
INSTANCIAÇÃO

ATRIBUIÇÃO

EXIBIÇÃO

Array - Visualizando

Percorrendo todos os elementos do vetor:




```
int[] vetor = new int[6];
```

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

```
for (int w : vetor){
    System.out.println("Elemento: " + w);
}
```

Array - Resumo

- Os arrays Java padrão podem armazenar tanto primitivos quanto objetos.
- Os arrays Java padrão podem ter uma ou muitas dimensões.
- Os arrays Java padrão são objetos e devem ser inicializados após serem declarados.
- Os arrays Java padrão podem ser declarados com colchetes após o tipo ou após o nome.
- Os arrays Java padrão podem ser inicializados com o operador new e seu tamanho em colchetes após o tipo.
- Os arrays Java padrão podem ser inicializados pela inserção dos valores que preencherão em chaves.

Exercícios

1. Desenvolva um programa que leia 10 nomes, armazene em um vetor (String) e depois percorra esse vetor exibindo os nomes presentes nele.

2. Crie um programa que leia 10 números e armazene-os em um vetor (int). Após o armazenamento, mostre a média dos números armazenados.

3. Desenvolva um programa que leia 10 salários, armazene em um vetor (double) e depois percorra esse vetor identificando qual o índice do maior salário.

Array - Importante

Um vetor, ao ser declarado, apenas reserva uma referência que pode apontar para um “objeto” vetor.

Ex:

```
byte[] mascara;
```

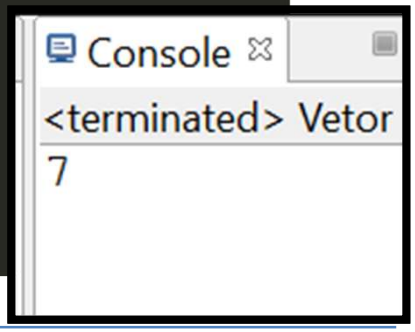

```
mascara = new byte[5];
```

Um vetor de tipos primitivos, ao ser instanciado, tem o seu espaço alocado em memória, e todos valores alocados são inicializados com 0.

Array - Importante

Qual seria a saída desse programa ?

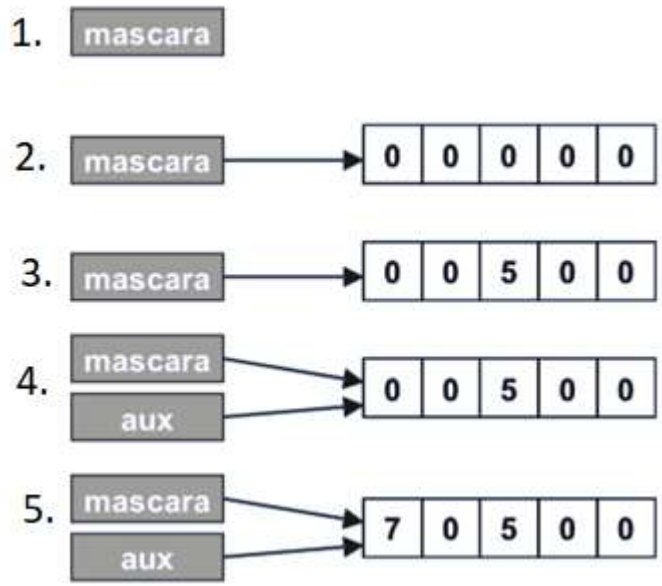
```
1
2  public class Vetor {
3
4      public static void main(String[] args) {
5
6          byte[] mascara = new byte[5];
7          mascara[2] = 5;
8
9          byte[] aux = mascara;
10         aux[0] = 7;
11
12         System.out.print(mascara[0]);
13
14     }
15 }
```



Array - Importante

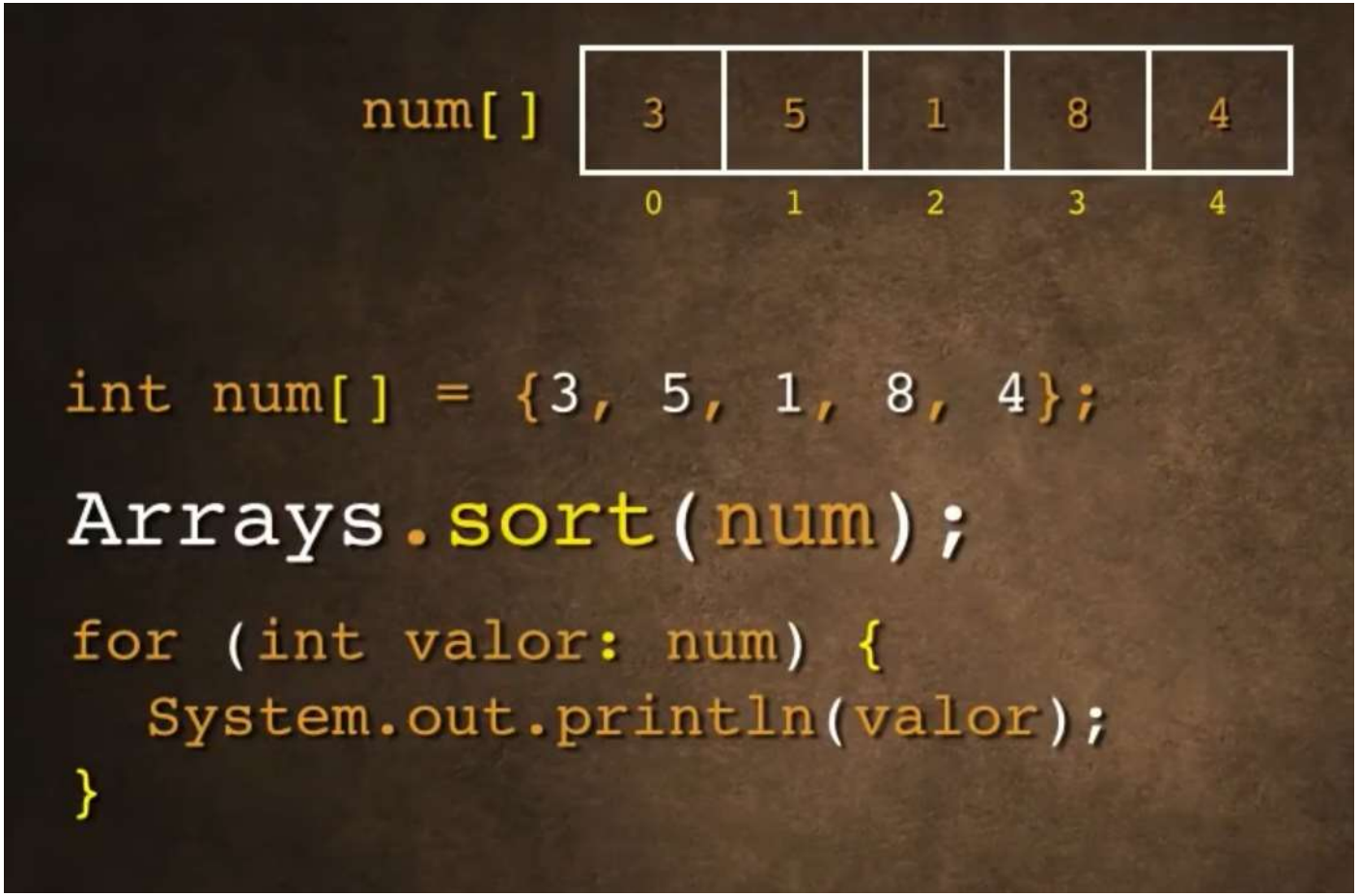
Mais de uma referência podem apontar para o mesmo vetor ao mesmo tempo.

```
1  byte[] mascara;
2  mascara = new byte[5];
3  mascara[2] = 5;
4  byte[] aux = mascara;
5  aux[0] = 7;
```

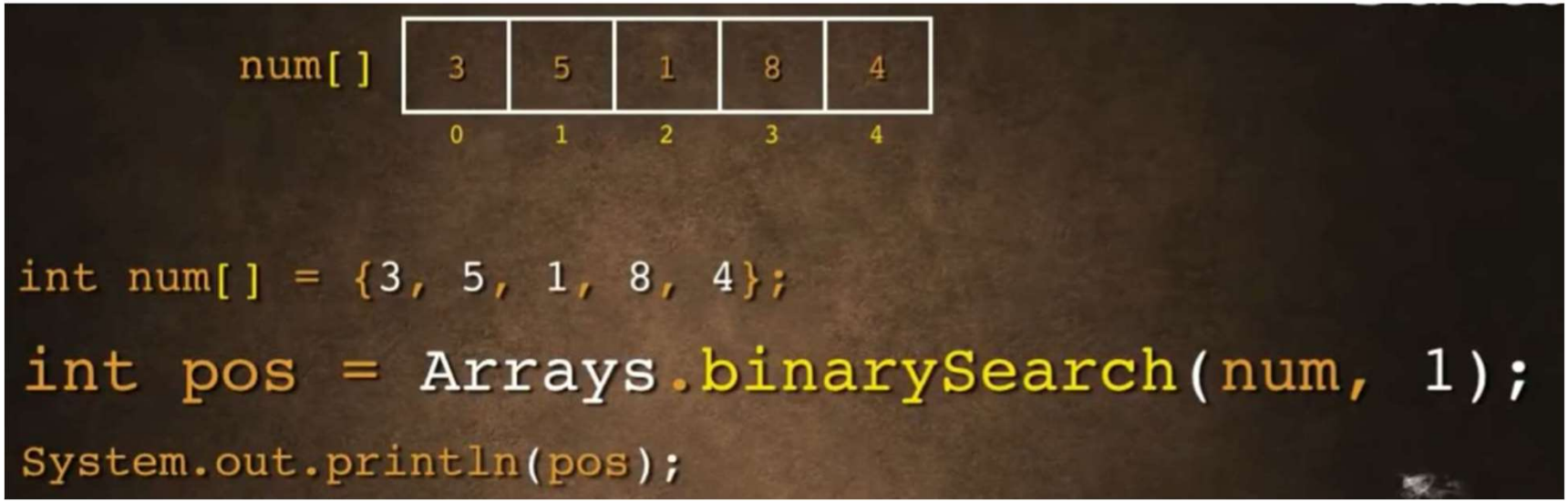


No exemplo, mascara e aux apontam para o mesmo vetor.

Ordenando um Array



Buscando um valor



IMPORTANTE: esse método só funciona se o array estiver na ordem crescente! Ou seja, só use o 'binarySearch' após usar o 'sort' !