



Java

Constantes e Arrays

METODOLOGIA

- › Interprete o documento calmamente e com atenção.
- › Acompanhe a execução do exercício no seu computador.
- › Não hesite em consultar o formador para o esclarecimento de qualquer questão.
- › Não prossiga para o ponto seguinte sem ter compreendido totalmente o ponto anterior.
- › Caso seja necessário, execute várias vezes o exercício até ter compreendido totalmente o processo.

CONTEÚDO PROGRAMÁTICO

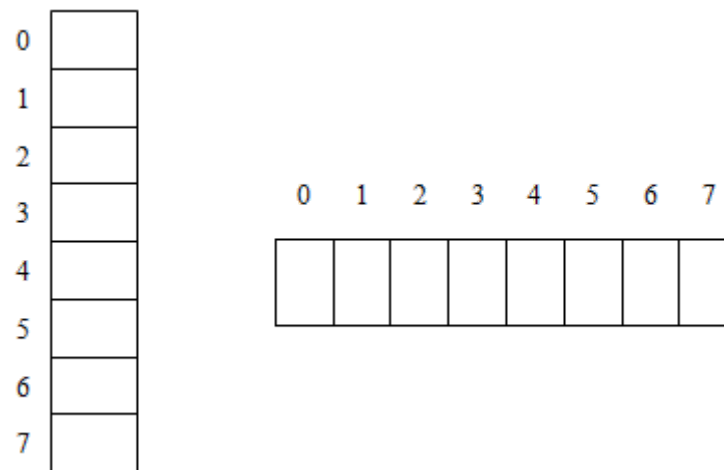
1. [Arrays](#)
2. [Constantes](#)

1. Arrays

Vamos agora trabalhar com um tipo de dados mais estruturado: os vetores, mais conhecidos por arrays. Em analogia, podemos considerar um armário com prateleiras, sendo cada prateleira uma posição do array. Tal como os armários, os arrays têm um número limitado de prateleiras, sendo esse número a dimensão do array.

Em cada posição do array (ou no caso do armário, em cada prateleira) poderemos colocar informação.

Os arrays unidimensionais também podem ser vistos como arrays linha ou arrays coluna, porque os seus elementos encontram-se sempre contíguos, ou seja, em posições de memória seguidas.



Em termos de tipos de dados Java, um array não é mais que uma lista de valores, de um determinado tipo de dados base do Java, daqueles que já conhecemos. Isto implica que todas as posições do array são obrigatoriamente do mesmo tipo.

Em Java, a sintaxe de declaração de arrays é a seguinte:

```
<TIPO DE DADOS>[] <NOME DO ARRAY> = new <TIPO DE DADOS>[n];
```

📄 Sendo **n** o número de posições ("prateleiras") do array.

Para colocarmos um valor numa determinada posição do array, utilizamos a seguinte sintaxe:

```
<NOME DO ARRAY>[n] = <VALOR>;
```

📄 Sendo **n** a posição do vetor que desejamos preencher.

Para efetuarmos a leitura de um valor de uma determinada posição do array, fazemos o seguinte:

```
<variável destino> = <NOME DO ARRAY>[n];
```

📄 Sendo **n** a posição do vector que pretendemos ler.

Vamos por este conceito em prática:


› **Crie** um novo ficheiro de nome `Arrays.java` e **introduza** o seguinte código:

```
class Arrays {  
  
    public static void main (String[] args) {  
  
        String[] listaDeStrings = new String[3];  
  
        listaDeStrings[0] = "Primeira posicao do array";  
        listaDeStrings[1] = "Segunda posicao do array";  
        listaDeStrings[2] = "Terceira posicao do array";  
  
        System.out.println(listaDeStrings[0]);  
        System.out.println(listaDeStrings[1]);  
        System.out.println(listaDeStrings[2]);  
  
    } // main  
} // Arrays
```

› **Compile** a classe, **execute** e **analise** o resultado

A primeira instrução é a declaração de um array com 3 posições:

```
String[] listaDeStrings = new String[3];
```

 Em cada posição do array pode armazenar uma String (cadeia de caracteres).

› De seguida **preenchemos** o conteúdo do array


 Como pode observar, o índice dos arrays, ou referência, começa em zero.

› No final, **imprimimos** o conteúdo de cada posição do array, com base no respetivo índice

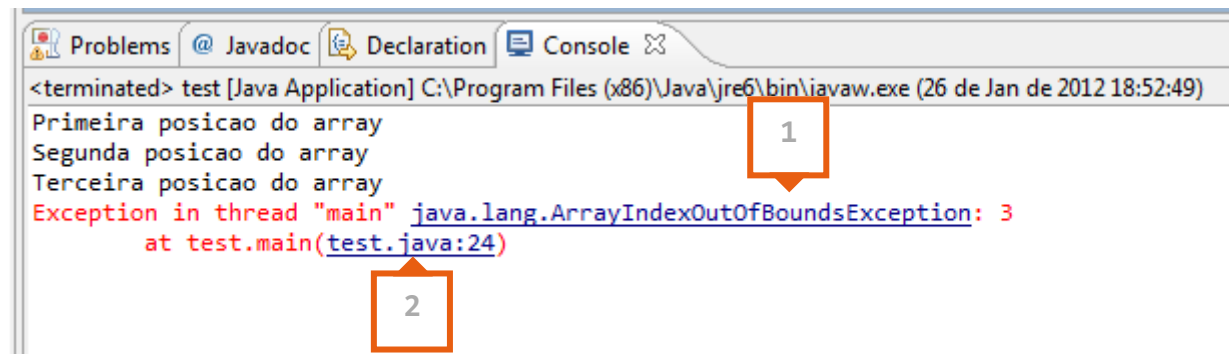
› **Adicione** agora a **instrução** seguinte na secção de preenchimento do **array**:

```
listaDeStrings[3] = "Quarta posição do array";
```

› **Compile** a classe, **execute** e **analise** o resultado

 O que aconteceu? Um erro? Repare que o array foi declarado apenas com 3 posições e a última instrução está a utilizar uma posição que não existe. Quando ocorre um erro na altura em que executamos o programa diz-se que houve uma exceção. Isto significa que o

programa teve de terminar de forma inesperada devido ao erro. Esta informação é visível na indicação do erro da **exceção 1**, assim como a respetiva **linha de código onde este erro ocorreu 2**.



› **Remova** a linha que resultou no erro e **introduza** agora o seguinte código no final da classe:

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

saldos[0] = 500;
saldos[1] = 350;
saldos[2] = 430;

System.out.println("posicao 0 mais a posicao 1 - " + (saldos[0] + saldos[1]));
System.out.println("posicao 0 menos a posicao 2 - " + (saldos[0] - saldos[2]));
System.out.println("tamanho do array saldos - " + saldos.length);
```

› **Compile** a classe, **execute** e **analise** o resultado

📄 Este exemplo declara um array também com 3 posições, mas agora em cada posição podemos armazenar um número inteiro. Podemos fazer operações diretamente com os valores dos arrays.

📄 Tal como qualquer outro objeto, os arrays disponibilizam métodos, algo que vamos observar em detalhe mais à frente. A última linha de código permite obter o tamanho, ou número de posições, do array.

A sua real utilidade só será perceptível quando abordarmos as instruções de controlo de fluxo.

2. Constantes

Um dos conceitos mais importantes nas boas práticas de programação é a utilização generalizada de constantes.

A utilização de constantes em programação tem duas grandes vantagens:

- 1 - A possibilidade de utilizar em todo o código um valor constante que não pode ser alterado por instruções ou operações, que se traduz num risco menor de erros.
- 2 - A estruturação do código, que permite num só local definir um valor utilizado em vários locais.

📄 Um caso típico da utilização de constantes é a taxa de IVA. Se o governo mudar a taxa, só será necessário alterar o valor num só local.

Vamos exemplificar:

- › **Crie** uma nova classe de nome `CalcularIva.java` e **adicione** o seguinte código:

```
import java.util.Scanner;

class CalcularIva {

    public static final double TAXA_IVA = 0.23;

    public static void main (String[] args) {

        Scanner teclado = new Scanner(System.in);

        System.out.println("Introduza o primeiro numero");
        int numero1 = teclado.nextInt();
        System.out.println("Introduza o segundo numero");
        int numero2 = teclado.nextInt();

        double soma_com_iva = (numero1 + numero2)*(1 + TAXA_IVA);
        System.out.println("A soma dos dois numeros que introduziu acrescido de iva é "
            + soma_com_iva);

    }
}
```

› **Compile e execute** a classe que acabou de criar

📄 Os termos `static` e `final` na declaração da variável `TAXA_IVA` indicam ao Java que se trata de uma variável especial (uma constante) que não permite alterações no seu valor.

📄 Note também que o valor do resultado teve de ser guardado num `double`, isto porque um dos valores da operação era `double`.