02

Declare, instancie, inicialize e use um array uni-dimensional

As linguagens de programação, normalmente, fornecem algum recurso para o armazenamento de variáveis em memória sequencial. No Java, os arrays permitem esse tipo de armazenamento.

Um array é um objeto que armazena sequencialmente "uma porção" de variáveis de um determinado tipo. É importante reforçar que os arrays são objetos. Uma referência para um objeto array deve ser armazenada em uma variável do tipo array.

A prova de certificação verifica se o candidato está apto a manipular tanto arrays de tipos primitivos quanto de tipos não primitivos.

Os quatro pontos importantes sobre arrays são:

- Declarar
- Inicializar
- Acessar
- Percorrer

Arrays de tipos primitivos

Declaração:

Para declarar um array, é utilizado `[]` logo após ao tipo das variáveis que desejamos armazenar ou logo após ao nome da variável.

```
// Declaração de um array para guardar variáveis do tipo int.
int[] idades;

// Declaração de um array para guardar variáveis do tipo
double.
double pesos[];

// Declaração de um array para guardar variáveis do tipo long.
long []pesos;

// Declaração de um array para guardar variáveis do tipo long.
long[]tamanhos;

// Perceba as formas de declarar um array.
```

COPIAR CÓDIGO

Inicialização:

Como um array é um objeto, a inicialização envolve a criação de um objeto. O **new**, operador que cria objetos, é utilizado para construir um array. Se você não executa o **new**, qual o valor padrão? Para atributos, é **null**, e para variáveis locais, não há valor, como qualquer outra variável de referência:

```
public class Clientes {

    int[] idades;

    public static void main(String[] args) {
        Clientes c = new Clientes();
        System.out.println(c.idades); // imprime null
    }
```

```
}
public class Produtos {

    public static void main(String[] args) {
        int[] precos;
        System.out.println(precos); // nao compila, não foi
                                    // inicializada
    }
}
```

COPIAR CÓDIGO

E como instancio um array?

```
// Inicialização do array idades.
idades = new int[10];

// Inicialização do array pesos.
pesos = new double[50];
```

COPIAR CÓDIGO

Na inicialização, é definida a capacidade do array, ou seja, a quantidade de variáveis que ele terá. Quando falarmos em tamanho de um array, estaremos nos referindo à sua capacidade.

Cada variável guardada em um array é iniciada implicitamente no momento em que o array é criado. Os valores atribuídos às variáveis são os valores `default` .

```
// Imprime 0 pois esse é o valor default para int.
System.out.println(idades[0]);
```

COPIAR CÓDIGO

E temos alguns casos extremos:

```
//compila e roda
int[] numeros = new int[0];

//compila, mas joga NegativeArraySizeException
numeros = new int[-1];
```

COPIAR CÓDIGO

Durante a declaração de uma referência para um array, temos a oportunidade de criá-lo de uma maneira mais fácil se já sabemos o que queremos colocar dentro:

```
int[] numeros;
numeros = new int[]{1,2,5,7,5};

Carro[] carros = new Carro[]{new Carro(), null, new Carro()};
```

COPIAR CÓDIGO

Não passamos o tamanho e fazemos a declaração dos elementos entre chaves e separados por vírgula. Os arrays terão tamanho 5 e 3, respectivamente.

E se a declaração e a inicialização estiverem na **mesma linha** podemos simplificar ainda mais:

```
int[] numeros = {1,2,5,7,5};
```

COPIAR CÓDIGO

Mas temos que tomar um pouco de cuidado com esse modo mais simples de declarar o array. Só podemos fazer como no exemplo anterior quando **declaramos e inicializamos** o array na mesma linha. Se fizermos a declaração e a inicialização em linhas separadas, o código não compila:

```
int[] numeros = {1,2,5,7,5}; // compila
int[] numeros2;
numeros2 = {1,2,5,7,5}; //Não compila
```

COPIAR CÓDIGO

Se desejamos inicializar posteriormente, devemos adicionar o operador `new` para poder iniciar o array em outra linha:

```
int[] numeros2;
numeros2 = new int[]{1,2,5,7,5}; //compila
```

COPIAR CÓDIGO

Acesso

As posições de um array são indexadas (numeradas) de `0` até a capacidade do array menos um. Para acessar uma das variáveis do array, é necessário informar sua posição.

```
// Coloca o valor 10 na primeira variável do array idades.
int idades[] = new int[10];
idades[0] = 10;

// Coloca o valor 73.14 na última variável do array pesos.
```

```
double pesos[] = new double[50];
pesos[49] = 73.14;
```

COPIAR CÓDIGO

O que acontece se alguém tentar acessar uma posição que não existe?

```
// Erro de execução ao tentar acessar um posição que não existe.
// ArrayIndexOutOfBoundsException
pesos[50] = 88.4;
```

COPIAR CÓDIGO

Será gerado um erro de execução (não de compilação). A ::exception:: lançada pelo Java é `ArrayIndexOutOfBoundsException` .

Percorrendo

Supondo que a capacidade de um array qualquer seja `100` , os índices desse array variam de `0` até `99` , ou seja, de `0` até a capacidade menos um.

O tamanho de um array é definido na inicialização e fica guardado no próprio array, podendo ser recuperado posteriormente.

Para recuperar o tamanho ou a capacidade de um array, é utilizado um atributo chamado `length` presente em todos os arrays.

```
for (int i = 0; i < idades.length; i++) {
    idades[i] = i;
}
```

COPIAR CÓDIGO

No `for` tradicional, as posições de um array são acessadas através dos índices. Dessa forma, é possível, inclusive, modificar os valores que estão armazenados no array.

Porém, em determinadas situações, é necessário apenas ler os valores de um array sem precisar modificá-los. Nesse caso, pode ser utilizado o `for` introduzido na versão 5 do Java.

```
for(int idade : idades){
    System.out.println(idade);
}
```

COPIAR CÓDIGO

Não há índices no `for` do Java 5. Ele simplesmente percorre os valores. Assim ele não permite modificar o array facilmente.