

Material de Apoio Aula 12

Arrays

Nos exercícios propostos até o momento, nos restringimos a utilizar representações *escalares* de variáveis e referências a objetos. Ou seja, cada nome simbólico de uma variável ou de uma referência, diz respeito a apenas uma instância de um determinado tipo de dado ou de uma determinada classe. No entanto, outros tipos de organização são possíveis. Notadamente uma organização que permite referenciar a várias instâncias de um mesmo tipo ou de uma classe. Esta estrutura é denominada *array* (ou vetores, ou ainda matrizes unidimensionais).

Sintaxe para definição de um array em Java

```
<tipo> <identificador> [];
```

Note que um array em Java também é um objeto. Portanto, deve ser explicitamente instanciado. No momento em que esta instanciação for realizada é necessário informar o número de elementos que o array deverá conter. Este tamanho é fixo e não poderá ser alterado.

Criação de um array em Java

```
<identificador> = new <tipo> [<tamanho>];
```

Exemplos:

Um array de inteiros

```
int vet []; // vet é uma referência para array de inteiros  
vet = new int[10]; // o array criado possui 10 posições
```

De forma abreviada, um array pode ser definido e criado na mesma linha de código

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

Um array de Strings para nomes de alunos

```
String meusalunos [] = { "Andrei", "Andrey", "Carla", "Michel" };
```

O acesso a uma determinada posição do array se dá através do operador [] e de um índice. Lembre que um array definido com tamanho *n* (onde *n* é um número positivo maior que 0) possui *n* posições: **numeradas de 0 a n-1**. No exemplo acima, o array *vet* possui 10 posições: 0, 1, 2, ..., 8 e 9. Já o array *meusalunos* possui tamanho 4, onde a posição 0 é Andrei, a posição 1 é Andrey, a posição 3 é Carla e a posição 3 (que é a última) Michel. É possível obter acesso ao tamanho de um vetor através de um campo público no objeto array em questão: *length*.

Exemplo de acesso a elementos de um array

```
// inicializando um vetor de inteiros  
int i, vet [] = new int[10];  
for( i = 0 ; i < vet.length ; i++ )  
    vet[i] = i;  
  
// imprimindo (na tela) o nome de um dos alunos  
String meusalunos [] = { "Andrei", "Andrey", "Carla", "Michel" };  
System.out.print("O segundo nome: " + meusalunos[1] );
```

Atenção: é comum lembrar de inicializar a variável que percorre um array em um comando de iteração (tal no exemplo com *for* dado acima) com o valor 0 (zero), considerando que a primeira posição do array possui este índice. Mas, também é um erro comum esquecer que o índice da última posição é tamanho - 1.

Observe também que é possível manter um array de referências para objetos.

Array de contas em banco

```
ContaEmBanco contas[] = new ContaEmBanco[5]; // array para 5 contas em banco  
  
contas[0] = new ContaEmBanco("Fulano", 617 );  
contas[1] = new ContaEmBanco("Fulano", 761);
```

Observe que cada posição do array necessita ter seu objeto criado explicitamente.

Exercícios

1. Exercício exemplo: realize a implementação final do código abaixo e comente a solução proposta neste exercício exemplo. Implemente o método `char[] concatena(char s1[], char s2[])` que recebe dois arrays de caracteres (`char`) `s1` e `s2` contendo listas de caracteres válidos. Este método deve retornar um novo array contendo a concatenação de `s1` e `s2`.

<pre>public char[] concatena(char s1[], char s2[]) { int i, tam = s1.length + s2.length; char s3[] = new char[tam]; for(i = 0 ; i < s1.length ; i++) s3[i] = s1[i]; for(i = 0 ; i < s2.length ; i++) s3[s1.length+i] = s2[i]; return s3; }</pre>	<pre>static public void main(String args[]) { char toto[] = new char[] { 'a', 'b', 'c', 'd' }; char titi[] = new char[] { 'f', 'g', 'h', 'i', 'j' }; char tutu[] = null; tutu = concatena(toto,titi); System.out.print(tutu); }</pre>
--	--

2. Implemente em uma classe `manipulaArray`, que não possui nenhum atributo, os seguintes métodos:
 - a. `void inicializaArray(int vet [] , int val)`. Este método inicializa o array `vet` atribuindo a cada posição de `vet` o valor `val`.
 - b. `void inicializaArray(int vet [])`. Este método inicializa o array `vet` atribuindo a cada posição de `vet` o valor correspondente a sua posição no array (0, 1, 2, ..., `length`).
 - c. `void inicializaArrayInvertido(int vet [])`. Este método inicializa o array `vet` atribuindo a cada posição de `vet` o valor correspondente a sua posição no array invertido (`length-1`, `length-2` ..., 2, 1, 0).
 - d. `void inicializaArrayAleatório(int vet [])`. Este método inicializa o array `vet` atribuindo a cada posição de `vet` um valor definido aleatoriamente.
 - e. `int maior(int vet [])`. Este método deve retornar o maior valor em `vet`.
 - f. `int soma(int vet [])`. Este método deve retornar a soma de todos valores em `vet`.
 - g. `double media(int vet [])`. Este método deve retornar a média dos valores em `vet`.
3. Implementar uma classe de testes que avalie os métodos do Exercício 2.

4. Implementar na classe `meuArray` o seguinte método:

```
int [] extremos( int int vet [] ).
```

Este método deve receber um array `vet` e retornar um novo array, de tamanho 2, cujo primeiro elemento represente o menor valor encontrado no array `vet` e o segundo o maior valor encontrado.

5. Implemente um array de objetos da classe `Fatorial` (Apoio 9). Em cada posição deste array coloque um novo objeto `Fatorial`, inicializado com a própria posição no vetor (Fatorial de 1, de 2, de 3, ...). Invoque o método de cálculo de Fatorial em cada um destes objetos e apresente a saída da seguinte forma:

```
Fatorial(0) = 1
Fatorial(1) = 1
Fatorial(2) = 2
Fatorial(3) = 6
Fatorial(4) = 24
```