Aula 06 – Vetores e matrizes

Baseado nos slides oficiais do livro Java – Como programar – Deitel e Deitel – 10ª edição

Arrays ou vetores

- Conjunto de elementos de um mesmo tipo
- Um array é considerado um objeto em Java, e deve ser instanciado através do operador new
- Os vetores podem ser de tipos primitivos ou de objetos
- Tem um tamanho pré-determinado no momento da instanciação, armazenado no atributo length
- Elementos indexados da posição 0 até tamanho-1

Declaração

- Declara-se da mesma forma que as demais variáveis, mas colocando colchetes vazios antes ou depois do nome da variável
  - int faltas [ ];
  - String [ ] nomes;
- Não é possível criar o array no estilo C/C++
  - double notas[10]; // Erro de compilação

Instanciação

- Usamos o operador new, junto ou não à declaração, o tipo (compatível) e o tamanho entre colchetes
  - String [ ] nomes = new String[10];
  - int faltas [ ];
  - faltas = new int[20];
- Cada elemento do array é inicalizado no momento da sua criação com os valores 0, false ou null

```
1  // Fig. 7.2: InitArray.java
2  // Initializing the elements of an array to default values of zero.
3
4  public class InitArray
5  {
6     public static void main(String[] args)
7     {
8        // declare variable array and initialize it with an array object
9        int[] array = new int[10]; // create the array object
10
11       System.out.printf("%s%8s%n", "Index", "Value"); // column headings
12
13       // output each array element's value
14       for (int counter = 0; counter < array.length; counter++)
15          System.out.printf("%5d%8d%n", counter, array[counter]);
16    }
17 } // end class InitArray
```

Fig. 7.2 | Initializing the elements of an array to default values of zero. (Part 1 of 2.)

Inicialização com lista de valores

- Ao invés de inicializar com new, é possível inicializar um vetor com uma lista de valores entre chaves
  - char opções [ ] = {‘a’, ‘i’, ‘e’};
- O tamanho do vetor é calculado de acordo com o número de elementos especificados

```
1  // Fig. 7.3: InitArray.java
2  // Initializing the elements of an array with an array initializer.
3
4  public class InitArray
5  {
6     public static void main(String[] args)
7     {
8        // initializer list specifies the initial value for each element
9        int[] array = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10
11       System.out.printf("%s%8s%n", "Index", "Value"); // column headings
12
13       // output each array element's value
14       for (int counter = 0; counter < array.length; counter++)
15          System.out.printf("%5d%8d%n", counter, array[counter]);
16
17 } // end class InitArray
```

Fig. 7.3 | Initializing the elements of an array with an array initializer. (Part 1 of 2.)

Laço for (each) para vetores

- O java tem um laço for específico para percorrer vetores e coleções de estruturas de dados
  - Sintaxe: for (variável: vetor)
- A cada iteração a variável assume o valor de um elemento do vetor em ordem. É equivalente a se fazer:
  - for (int i = 0 ; i < vetor.length; i++)

    variavel = vetor [i];
- Não é possível modificar os elementos, só acessá-los

```
1  // Fig. 7.12: EnhancedForTest.java
2  // Using the enhanced for statement to total integers in an array.
3
4  public class EnhancedForTest
5  {
6     public static void main(String[] args)
7     {
8        int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
9        int total = 0;
10
11       // add each element's value to total
12       for (int number : array)
13          total += number;
14
15       System.out.printf("Total of array elements: %d%n", total);
16    }
17 } // end class EnhancedForTest
```

```
Total of array elements: 849
```

Fig. 7.12 | Using the enhanced `for` statement to total integers in an array.

Exemplo

- Faça uma classe GeradorLogin que tem dois atributos: o nome da pessoa e o ano de nascimento. Crie um construtor que recebe dois argumentos, o nome e ano. Crie métodos getters e setters. Crie também os seguintes métodos que dão sugestões de nome de login:
  - Contendo as iniciais de todos os nomes
  - Contendo o primeiro e último nome
  - Contendo as iniciais dos nomes até o penúltimo, o último nome e o ano de nascimento
- Usar o método split da classe String

Matrizes

- Matrizes são estruturas com um mesmo tipo onde o elemento é identificado por mais de um índice
- Em java matrizes são vetores de vetores
- Elas seguem o mesmo padrão dos vetores só que replicado para cada dimensão

Declaração

- Declara-se da mesma forma que os vetores, mas colocando um par de colchets para cada dimentsão
  - int faltas [ ] [ ];
  - String [ ] [ ] nomesAluno;

Instanciação

- Usamos o operador new, de duas formas
  - Para alocar todas as dimensões de uma vez

```
nomesAlunos = new String[10] [30];
```

  - Para alocar cada dimensão com um tamanho

```
int trianguloPascal [ ] [ ] = new int [10];
for (int i = 0 ; i < 10 ; i++)
    trianguloPascal [i] = new int [i+1];
```

Inicialização com lista de valores

- Também é possível inicializar uma matriz com uma lista de valores entre chaves. Cada elemento deve ser também um vetor
  - double matriz [ ] [ ] = { {1.2, 9.2}, {2.3, 5.3}, {1.2, 2.5}, {10.2, -1.7} };
- O atributo length varia de acordo com a dimensão da matriz (matriz.length=4, matriz[0].length=2)

Passando arrays como argumentos

- Podemos passar arrays como parâmetros de métodos ou tipo de retorno, usando apenas os colchetes
- Não é preciso passar o tamanho deles porque isso já está presente no atributo length
- Assim como todo objeto em Java, um vetor/matriz é passado por referência, e as modificações feitas são replicadas no chamador

```
1  // Fig. 7.13: PassArray.java
2  // Passing arrays and individual array elements to methods.
3
4  public class PassArray
5  {
6     // main creates array and calls modifyArray and modifyElement
7     public static void main(String[] args)
8     {
9        int[] array = { 1, 2, 3, 4, 5 };
10
11       System.out.printf(
12          "Effects of passing reference to entire array:%n" +
13          "The values of the original array are:%n");
14
15       // output original array elements
16       for (int value : array)
17          System.out.printf("   %d", value);
18
19       modifyArray(array); // pass array reference
20       System.out.printf("%n%nThe values of the modified array are:%n");
21
```

Fig. 7.13 | Passing arrays and individual array elements to methods. (Part 1 of 3.)

```
22          // output modified array elements
23          for (int value : array)
24             System.out.printf("   %d", value);
25
26          System.out.printf(
27             "%n%nEffects of passing array element value:%n" +
28             "array[3] before modifyElement: %d%n", array[3]);
29
30          modifyElement(array[3]); // attempt to modify array[3]
31          System.out.printf(
32             "array[3] after modifyElement: %d%n", array[3]);
33       }
34
35       // multiply each element of an array by 2
36       public static void modifyArray(int[] array2)
37       {
38          for (int counter = 0; counter < array2.length; counter++)
39             array2[counter] *= 2;
40       }
```

Fig. 7.13 | Passing arrays and individual array elements to methods. (Part 2 of 3.)

```
41
42      // multiply argument by 2
43      public static void modifyElement(int element)
44      {
45         element *= 2;
46         System.out.printf(
47            "Value of element in modifyElement: %d%n", element);
48      }
49   } // end class PassArray
```

```
Effects of passing reference to entire array:
The values of the original array are:
   1   2   3   4   5

The values of the modified array are:
   2   4   6   8   10

Effects of passing array element value:
array[3] before modifyElement: 8
Value of element in modifyElement: 16
array[3] after modifyElement: 8
```

Fig. 7.13 | Passing arrays and individual array elements to methods. (Part 3 of 3.)

Classe Arrays

- Classe com algumas funções estáticas para manipular vetores, como: sort, binarySearch, equals, fill
- A classe System tem uma função chamada arrayCopy que copia o conteúdo de um vetor para outro

```
1  // Fig. 7.22: ArrayManipulations.java
2  // Arrays class methods and System.arraycopy.
3  import java.util.Arrays;
4
5  public class ArrayManipulations
6  {
7     public static void main(String[] args)
8     {
9        // sort doubleArray into ascending order
10       double[] doubleArray = { 8.4, 9.3, 0.2, 7.9, 3.4 };
11       Arrays.sort(doubleArray);
12       System.out.printf("%ndoubleArray: ");
13
14       for (double value : doubleArray)
15          System.out.printf("%.1f ", value);
16
17       // fill 10-element array with 7s
18       int[] filledIntArray = new int[10];
19       Arrays.fill(filledIntArray, 7);
20       displayArray(filledIntArray, "filledIntArray");
21
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 1 of 4.)

```
22          // copy array intArray into array intArrayCopy
23          int[] intArray = { 1, 2, 3, 4, 5, 6 };
24          int[] intArrayCopy = new int[intArray.length];
25          System.arraycopy(intArray, 0, intArrayCopy, 0, intArray.length);
26          displayArray(intArray, "intArray");
27          displayArray(intArrayCopy, "intArrayCopy");
28
29          // compare intArray and intArrayCopy for equality
30          boolean b = Arrays.equals(intArray, intArrayCopy);
31          System.out.printf("%n%nintArray %s intArrayCopy%n",
32             (b ? "==" : "!="));
33
34          // compare intArray and filledIntArray for equality
35          b = Arrays.equals(intArray, filledIntArray);
36          System.out.printf("intArray %s filledIntArray%n",
37             (b ? "==" : "!="));
38
39          // search intArray for the value 5
40          int location = Arrays.binarySearch(intArray, 5);
41
42          if (location >= 0)
43             System.out.printf(
44                "Found 5 at element %d in intArray%n", location);
45          else
46             System.out.println("5 not found in intArray");
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 2 of 4.)

```
47
48          // search intArray for the value 8763
49          location = Arrays.binarySearch(intArray, 8763);
50
51          if (location >= 0)
52             System.out.printf(
53                "Found 8763 at element %d in intArray%n", location);
54          else
55             System.out.println("8763 not found in intArray");
56       }
57
58       // output values in each array
59       public static void displayArray(int[] array, String description)
60       {
61          System.out.printf("%n%s: ", description);
62
63          for (int value : array)
64             System.out.printf("%d ", value);
65       }
66    } // end class ArrayManipulations
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 3 of 4.)

```
doubleArray: 0.2 3.4 7.9 8.4 9.3
filledIntArray: 7 7 7 7 7 7 7 7 7 7
intArray: 1 2 3 4 5 6
intArrayCopy: 1 2 3 4 5 6

intArray == intArrayCopy
intArray != filledIntArray
Found 5 at element 4 in intArray
8763 not found in intArray
```

Fig. 7.22 | `Arrays` class methods and `System.arraycopy`. (Part 4 of 4.)