

Capítulo - 13

13.1 - Utilização de Arrays (arranjo)

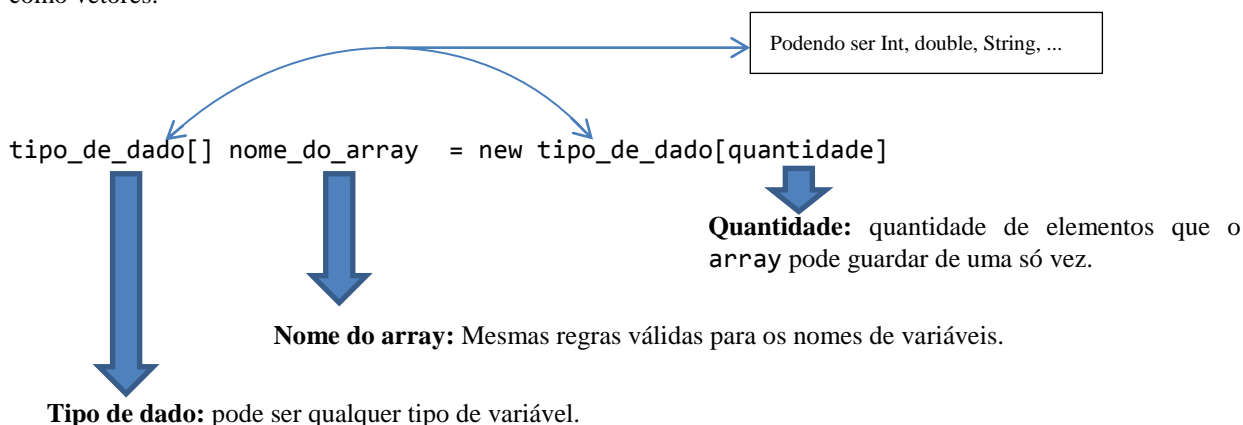
Suponha que seja necessário armazenar e manipular dezenas de nomes de pessoas num programa de computador. De acordo com o que estudamos até aqui seriam necessárias dezenas de variáveis, cada uma armazenando um nome diferente, como por exemplo, nome1= "Lucas", nome2= "Matheus" e assim por diante.

Em vez disso é possível a declaração de apenas uma variável indexada, array. Em outras palavras, podemos definir uma variável cujos elementos são referenciados por um índice no seguinte formato:

```
nome[0] = "Lucas"
nome[1] = "Matheus"
nome[2] = "..."
nome[..] = "..."
nome[10] = "Maria"
```

O número entre colchetes diferencia os valores dentro da variável. Talvez você não tenha percebido mas estamos utilizando arrays desde o início do aprendizado em Java. Pois `String[] args` nada mais é que um array chamado `args` que é um array de elementos do tipo `String`.

Os arrays podem ser unidimensionais (com única dimensão apenas) os arrays de uma dimensão são conhecidos como vetores.



Exemplo da sintaxe:

```
Int[] salaAula = new int[10];
```

A linha acima cria um array de 10 posições sendo a primeira 0 (zero) e a última 9 (nove) sendo que esta poderá armazenar números. Veja o modelo deste array no quadro abaixo:

Valor	1997	1998	1669	2015	4568	4556	12	2016	123	1
Índice	0	1	2	3	4	5	6	7	8	9

```
String[] alunosEscola = new String[5];
```

A linha acima cria um array de 5 posições sendo a primeira 0 (zero) e a última 4 (quatro) este vetor poderá armazenar valores do tipo `String`, ou seja, caracteres.

Valor	Maria	Shampoo	Casa	Provas	Curso
Índice	0	1	2	3	4

13.2 Tipos de Array

Podemos ter dois tipos de array, um do tipo primitivo e outro como tipo objeto. Não se preocupe com o termo **OBJETO** agora! Basta saber que podemos inicializar ou declarar um array sendo ele unidimensional ou bidimensional das seguinte formas:

- Primitivo não utiliza a palavra **new**, portanto não dispõe de certas funções e facilidades.
- Tipo Objeto utiliza a palavra reservada **new**

13.3 Array como tipo primitivo

Como sabemos os tipos primitivos são: **int**, **float**, **char**, **double**, **boolean**, **byte**, **short** e **long**. Portanto um array sendo de um destes tipos chamaremos de array do tipo primitivo.

Exemplo da forma de uso:

```
int[ ] idade = {45, 48, 21, 35, 12};
```

Está subentendido que este array é unidimensional. Veja a tabela abaixo com a representação do array criado:

Índices:	0	1	2	3	4
Valores:	45	48	21	35	12

13.4 Array como tipo objeto

Podemos também declarar um array como tipo objeto. Para isto devemos proceder como se segue:

```
int[ ] idade = new int[5];
```

Neste caso utilizamos a palavra chave **new** e indicamos o tamanho do array. Agora podemos atribuir os valores da seguinte forma:

```
idade[0] = 45;  
idade[1] = 48;  
idade[2] = 21;  
idade[3] = 35;  
idade[4] = 12;
```

13.5 Acessando os valores de um array

Para buscar valores em um array podemos fazer de forma direta como visto abaixo:

```
1 public class ArraySimples1 {
2
3     public static void main(String[] args) {
4
5         String[] alunos = new String[5];
6
7         alunos[0] = "Amanda";
8         alunos[1] = "Diego";
9         alunos[2] = "Solange";
10        alunos[3] = "Ivan";
11        alunos[4] = "Otávio";
12
13        System.out.println("Nome do aluno nº1 "+alunos[0]);
14        System.out.println("Nome do aluno nº2 "+alunos[1]);
15        System.out.println("Nome do aluno nº3 "+alunos[2]);
16        System.out.println("Nome do aluno nº4 "+alunos[3]);
17        System.out.println("Nome do aluno nº5 "+alunos[4]);
18    }
19 }
```

Console

<terminated> ArraySimples1 [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (16/1)

Nome do aluno nº1 Amanda
Nome do aluno nº2 Diego
Nome do aluno nº3 Solange
Nome do aluno nº4 Ivan
Nome do aluno nº5 Otávio

Porém vemos que será necessário muitas linhas de código para atribuir e extrair valores do array. Para automatizar e reduzir o código nos utilizaremos da estrutura de laço de repetição **for** para percorrer todos os índices do vetor aproveitando o índice **i** para indicar o índice do vetor **alunos** a serem mostrados.

```
1 package Array_Apostila;
2
3 public class ArraySimples {
4
5     public static void main(String[] args) {
6
7         String[] alunos = new String[5];
8
9         alunos[0] = "Amanda";
10        alunos[1] = "Diego";
11        alunos[2] = "Solange";
12        alunos[3] = "Ivan";
13        alunos[4] = "Otávio";
14
15        for (int i = 0; i < 5; i++)
16            System.out.println("Nome do aluno nº"+i+" "+alunos[i]);
17    }
18 }
19 }
```

Console

<terminated> ArraySimples [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (16/1)

Nome do aluno nº0 Amanda
Nome do aluno nº1 Diego
Nome do aluno nº2 Solange
Nome do aluno nº3 Ivan
Nome do aluno nº4 Otávio

Como vimos no início do tópico um array pode ser inicializado de outra maneira. Em vez de usar o operador new para criar um array, podemos determinar os elementos que ficarão no array entre {} chaves e separados por vírgula. Esses elementos devem ser do mesmo tipo, isto é, se for um array do tipo int deveremos apenas preencher com valores do tipo int.

```

3 public class ArrayComChaves {
4
5     public static void main(String[] args) {
6         String[] semana = {"Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado"};
7
8         for(int i = 0; i < semana.length; i++)
9             System.out.println(semana[i]);
10    }
11 }

```

Console

```

<terminated> ArrayComChaves [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (28/05/2015 11:25:37)
Domingo
Segunda
Terça
Quarta
Quinta
Sexta
Sábado

```

13.6 - Array Multidimensional

Os arrays podem também ser multidimensionais (com duas ou mais dimensões). Array com mais de uma dimensão é conhecido como matriz. A linguagem Java não permite array bidimensional como em outras linguagens (no formato linha-coluna), porém é possível obter o mesmo efeito criando um array de arrays. Os mais comuns são os que envolvem dois arrays, mas é possível criar arrays com quantas dimensões forem necessárias. No exemplo dos nomes acima podemos agora atribuir uma nota para cada aluno por exemplo.

Array bidimensional

		Índice Nomes				
		0	1	2	N	10
Índice Notas	0	Lucas	Matheus	Lucas	...	Maria
	1	2,0	5,0	1,8	2,9	7,5
	2	3,8	6,0	6,5	7,0	7,9
	3	4,0	4,5	6,3	7,8	9,0
	10	10,0	10,0	9,5	8,4	10,0

Analizando este array temos:

Linha coluna
 Array[0] [0] teremos apenas "Lucas" Array[0] [1] teremos apenas "Matheus"
 Array[1] [0] teremos 2,0 e Lucas Array[1] [1] teremos 5,0 e Matheus
 Array[2] [0] teremos 3,8 e Lucas Array[2] [1] teremos 6,0 e Matheus
 Array[3] [0] teremos 4,0 e Lucas Array[3] [1] teremos 4,5 e Matheus
 Array[10][0] teremos 10,0 e Lucas Array[10][1] teremos 10,0 e Matheus

Dica! Em Java o primeiro valor de um array sempre inicia por

A forma de uso do array multidimensional é:

Tipo_de_dado nome-do-array[][] = new tipo_de_dado [<indice1>][<indice2>]

O exemplo abaixo mostra o uso de um array bidimensional para armazenar seis números.

```
3 import javax.swing.JOptionPane;
4
5 public class MatrizBidimensional {
6
7     public static void main(String[] args) {
8         int matriz[][] = new int[2][3];
9
10        matriz[0][0] = 1;
11        matriz[1][0] = 4;
12
13        matriz[0][1] = 2;
14        matriz[1][1] = 5;
15
16        matriz[0][2] = 3;
17        matriz[1][2] = 6;
18
19
20        int soma = 0;
21
22        for (int linha = 0; linha < 2; linha++)
23            for (int coluna = 0; coluna < 3; coluna++){
24                System.out.println(matriz[linha][coluna]);
25                soma += matriz [linha][coluna];
26            }
27        JOptionPane.showMessageDialog(null, "Soma dos elementos da Matriz: "+soma);
28    }
29 }
30
```

Tabela 3x2

Mensagem

Soma dos elementos da Matriz: 21

OK

	0	1	2
0	1	2	3
1	4	5	6

Console

MatrizBidimensional [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (28/05/2015 11:53:01)

```
1
2
3
4
5
6
```

13.7 Busca em Arrays

Por vezes desejamos realizar uma busca de um dos elementos do array, neste caso teremos que recorrer ao já estudado método equals("valor-procurado") da classe String.

```
3 public class BuscaArray {
4
5     public static void main(String[] args) {
6
7         String nomes[] = new String[5];
8
9         nomes[0] = "Maria";
10        nomes[1] = "José";
11        nomes[2] = "Diogo";
12        nomes[3] = "Estevao";
13        nomes[4] = "Diana";
14
15        String procurado = "Diana";
16        boolean status = false;
17        for (int i = 0; i <= nomes.length-1; i++){
18            if (nomes[i].equals(procurado)){
19                System.out.println(procurado +" encontrado na posição: " + (i+1)+" !");
20                status = true;
21            }
22        }
23
24        if (status == false)
25            System.out.println("Valor procurado não encontrado!");
26    }
27 }
28
```

Valor procurado!

Console

<terminated> BuscaArray [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (28/05/2015 18:18:57)

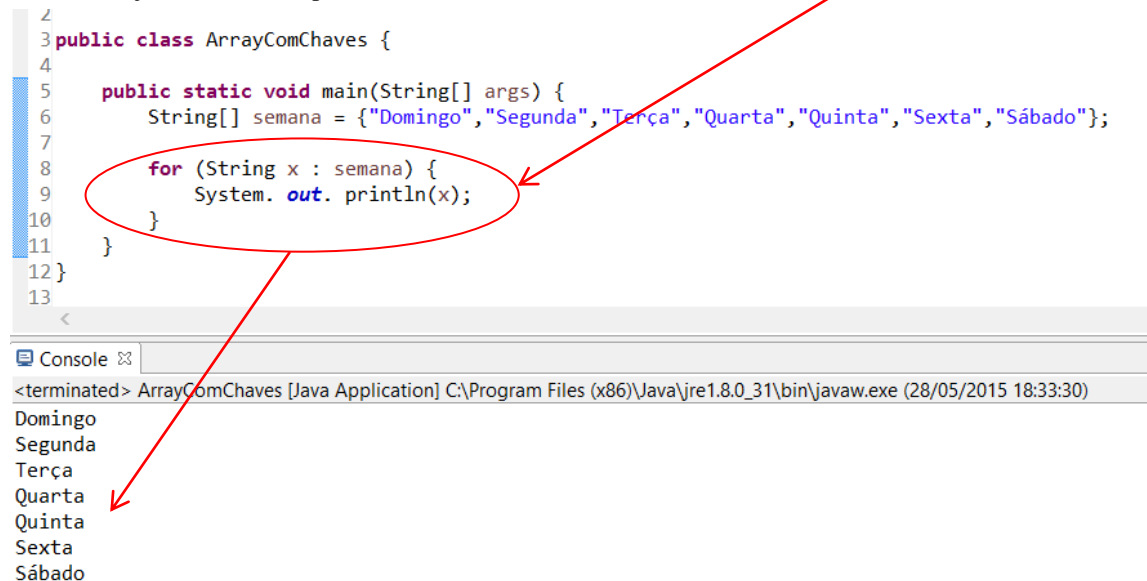
Diana encontrado na posição: 5 !

13.8 Acessando valores de um array (*enhanced-for*)

Quando **não há** a necessidade de manter uma variável com o índice que indica a posição do elemento no vetor (que é uma grande parte dos casos), podemos usar o **enhanced-for**.

Vejamos este exemplo:

```
2
3 public class ArrayComChaves {
4
5     public static void main(String[] args) {
6         String[] semana = {"Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado"};
7
8         for (String x : semana) {
9             System.out.println(x);
10        }
11    }
12 }
13
```



Console

<terminated> ArrayComChaves [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (28/05/2015 18:33:30)

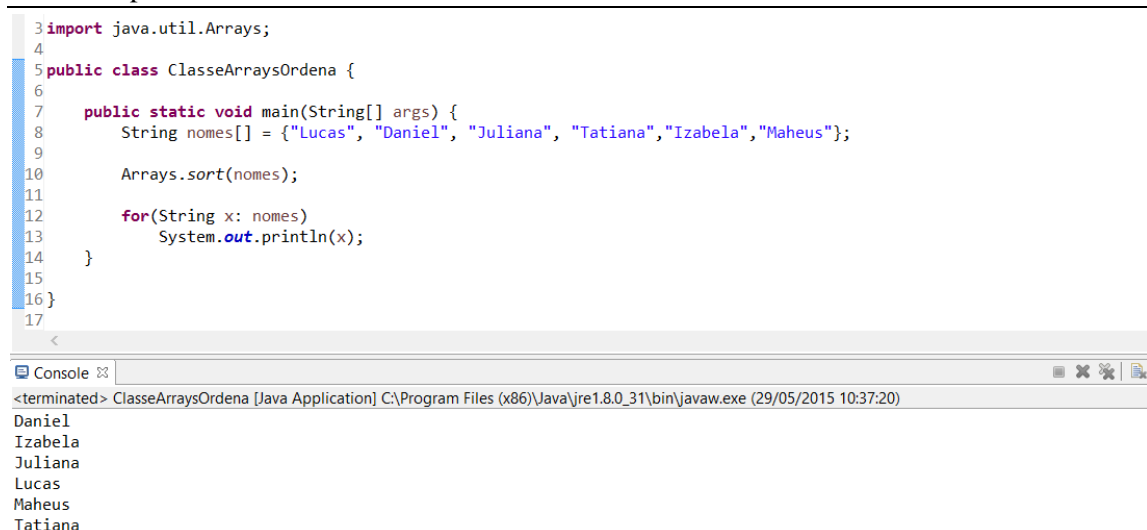
Domingo
Segunda
Terça
Quarta
Quinta
Sexta
Sábado

Agora vamos comparar com o 3º exemplo do tópico 13.5 deste capítulo! Não precisamos mais do `length` para percorrer matrizes cujo tamanho não conhecemos.

13.9 – A classe Arrays –(ordenando dados-String)

A classe Arrays permite manipular os elementos de um array (ordená-los ou realizar uma pesquisa com eles) O próximo exemplo demonstra a utilização da classe Arrays para ordenar um array de nomes de pessoas.

```
3 import java.util.Arrays;
4
5 public class ClasseArraysOrdena {
6
7     public static void main(String[] args) {
8         String nomes[] = {"Lucas", "Daniel", "Juliana", "Tatiana", "Izabela", "Maheus"};
9
10        Arrays.sort(nomes);
11
12        for(String x: nomes)
13            System.out.println(x);
14    }
15 }
16 }
17
```



Console

<terminated> ClasseArraysOrdena [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (29/05/2015 10:37:20)

Daniel
Izabela
Juliana
Lucas
Maheus
Tatiana

Linha 3: importamos a classe Arrays que está no pacote java/útil

Linha 8: Declaramos um array chamado nomes contendo os seis nomes: Lucas, Daniel, Juliana, Tatiana, Izabela, Maheus (índices de 0 a 5).

Linha 10: ordena o array nomes por ordem alfabética pelo método sort da classe Arrays.

Linhas 12-23: exibe através do enhanced for os nomes ordenados.

13.10 – A classe Arrays –(ordenando dados-Números)

O exemplo abaixo se utiliza da classe Math para gerar números aleatórios, colocando-os em um array de 10 elementos. Em seguida novamente utilizamos a classe Arrays com o método sort para ordenar os valores.

```
3 import java.util.Arrays;
4
5 public class ClasseArraysOrdena2 {
6
7     public static void main(String[] args) {
8         int numeros[] = new int[10];
9
10        for (int i =0;i<10;i++){
11            numeros[i] = (int) (Math.random() * 100);
12        }
13        System.out.print("Sorteado: ");
14        for(int x: numeros){
15            System.out.print(x+ ", ");
16        }
17        System.out.println("\n");
18
19        System.out.print("Ordenado: ");
20        Arrays.sort(numeros);
21        for(int n: numeros){
22            System.out.print(n+ ", ");
23        }
24    }
25 }
26 }
27
28
29
```

<

Console

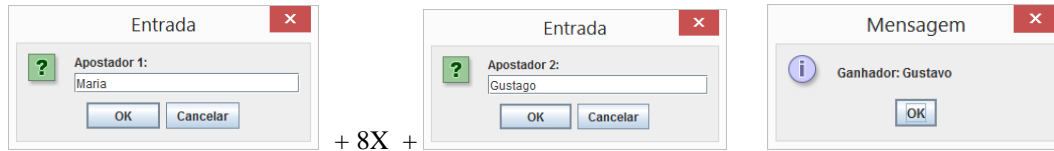
<terminated> ClasseArraysOrdena2 [Java Application] C:\Program Files (x86)\Java\jre1.8.0_31\bin\javaw.exe (29/05/2015 10:55:02)

Sorteado: 43, 47, 51, 4, 28, 51, 34, 61, 58, 76,

Ordenado: 4, 28, 34, 43, 47, 51, 51, 58, 61, 76,

ATIVIDADES – 13

1) Elabore uma classe que receba dez nomes de pessoas por meio do `showInputDialog`, armazene estes nomes em um array de dez elementos, sorteie um ganhador e imprima o seu nome em tela, conforme a figura abaixo.



2) Crie uma classe que receba dez números por meio do `showInputDialog`, armazene esses números em um array de dez elementos e apresente em tela as somas dos números, o maior e o menor.

3) Faça uma classe que gere aleatoriamente mil números inteiros quaisquer e os armazene num array bidimensional (considere 50 linhas e 20 colunas). A seguir, peça para o usuário digitar um número para ser pesquisado no array. Informe ao usuário se o número existe ou não no array.

4) Dada a sequência de números {52, 10, 85, 1324, 01, 13, 62, 30, 12, 127} desenvolva uma aplicação que exiba os mesmos números em ordem crescente e em ordem decrescente.

5) Faça uma classe contendo um array bidimensional que recebe o nome e o sexo de cinco pessoas: A seguir o usuário fornece o sexo das pessoas que devem ser apresentadas na tela.

6) Faça uma classe que simule a ocupação de quartos em um hotel. Considere que existem cinco andares e dez quartos por andar. O objetivo é saber se um quarto está ou não ocupado. A classe deve possuir os menus com as opções: 1-IMPRIMIR LISTA DE QUARTOS, 2-OCUPAR QUARTO e 3 Sair. A opção 1, lista todos os quartos, informando se estão ou não ocupados, a opção 2 permite definir o status “OCUPADO” ou “LIVRE” para qualquer um dos quartos. A opção 3 encerra o programa.

Parabéns aluno:

O próximo capítulo apresenta a criação de métodos e a troca de mensagens entre eles. De forma geral, ele fornece a base para que você mesmo possa criar seus métodos, compreendendo os conceitos de **modularidade**, **encapsulamento** e **reaproveitamento** do código.