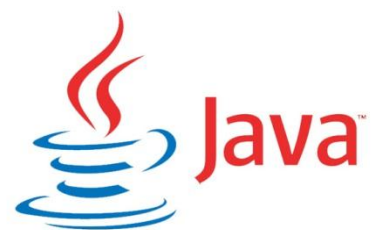


Lógica de Programação Visual

Unidade 18 – Arrays: coleções estáticas de dados



QI ESCOLAS E FACULDADES
Curso Técnico em Informática – EAD
Aline Maciel Zenker
Eligiane Ceron

SUMÁRIO

INTRODUÇÃO.....	3
O QUE É UM ARRAY?	3
CRIANDO E MANIPULANDO UM ARRAY	3
CRIANDO UM ARRAY.....	4
ARMAZENANDO VALORES NO ARRAY	5
<i>Preenchendo um array com valores informados pelo usuário</i>	<i>6</i>
EXIBINDO VALORES ARMAZENADOS NO ARRAY.....	7
<i>Utilizando um loop para exibir o conteúdo de um array</i>	<i>7</i>
<i>Utilizando a classe Arrays para exibir o conteúdo de um array</i>	<i>8</i>
O ATRIBUTO LENGTH.....	8
UM ARRAY COMO ATRIBUTO DE UMA CLASSE.....	9
REFERÊNCIAS	11

INTRODUÇÃO

Nesta unidade estudaremos as estruturas de armazenamento coletivo, também conhecidas como *arrays* (arranjos).

O QUE É UM ARRAY?

Um *array* é um **objeto** capaz de armazenar um número **fixo** de valores de **um único tipo**. A quantidade de elementos que armazena é estabelecida quando ele é criado e após a criação seu comprimento de mantém fixo (ORACLE, 2012).

O uso de *arrays* é indicado quando temos uma certa quantidade de elementos que serão manipulados/armazenados. Por exemplo: as notas de um aluno, as questões de uma prova, as alternativas das questões, um conjunto de números, e assim por diante.

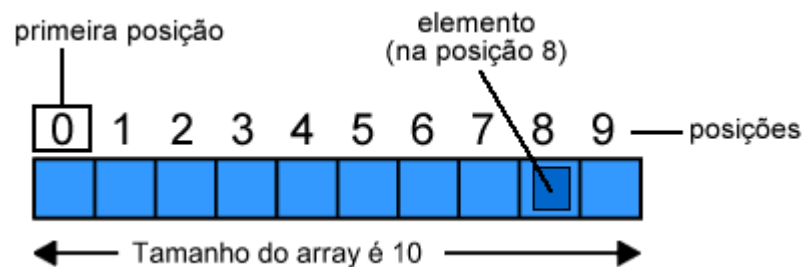


Figura 1 – Array de 10 elementos¹

“Cada item em um array é chamado de elemento, e cada elemento é acessado pela posição numérica. Como na ilustração acima as posições são numeradas a partir do 0. O 9º elemento, por exemplo, é acessado na posição 8” (LIMA, 2006).

CRIANDO E MANIPULANDO UM ARRAY

Para criar um *array* é necessário definir:

- Seu nome;
- Seu tipo (int, boolean, String, etc);
- A quantidade de elementos (um número inteiro maior que 0).

¹ Fonte: <http://www.plugmasters.com.br/sys/materias/535/1/Arrays-em-Java>

Criando um array

A declaração de um *array* utiliza um conjunto de colchetes vazio ao lado do tipo. O tipo pode ser tanto primitivo quanto de referência.

Tipos primitivos: são os tipos de dados básicos da linguagem, com eles podemos criar outros tipos. Exemplos: *int*, *char*, *byte*, *boolean*, etc.

Tipos de referência: são as classes do sistema, tanto as do Java quanto as criadas pelo programador. Exemplos: *String*, *Pessoa*, etc.

1ª forma: Declaração e instância em linhas separadas

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

O exemplo acima primeiramente declara o *array*, utilizando o tipo desejado seguido de colchetes vazios e o nome que se deseja atribuir ao *array*. Em outra linha, o comando *new* é responsável pela criação do objeto na memória. No exemplo, seria criado um *array* com capacidade para 5 números inteiros, organizados em índices de 0 até 4. Cria-se um *array* vazio, pronto para receber elementos.

2ª forma: Declaração e instância na mesma linha

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

O exemplo acima é uma forma abreviada de declarar e instanciar o *array* também vazio.

3ª forma: Declaração com atribuição de valores

```
int[] pares = {0,2,4,6,8};  
String[] cores = {"Azul", "Vermelho", "Amarelo", "Verde"};
```

O exemplo acima demonstra como criar um *array* já com elementos armazenados. O primeiro exemplo criaria um *array* com 5 elementos, conforme a figura 2:

0	1	2	3	4
0	2	4	6	8

Figura 2 – array “pares”

O segundo exemplo criaria um *array* chamado “cores” conforme mostra a figura 3:

0	1	2	3
Azul	Vermelho	Amarelo	Verde

Figura 3 - array "cores"

Observação: quando se utiliza a terceira forma, ele não traz posições vazias, mas podemos substituir o conteúdo de alguma posição por outro valor.

Armazenando valores no array

Após a declaração e instância do *array*, ele está pronto para armazenar valores. Para isso, é necessário indicar em qual posição queremos armazenar o valor. Esta posição pode ser indicada por um número inteiro definido ou por uma variável do tipo inteiro que contenha um valor de índice válido.

Vamos utilizar como exemplo o seguinte *array*:

```
int[] numeros = new int[6];
```

0	1	2	3	4	5

Figura 4 - array "numeros"

Vamos armazenar um número qualquer na primeira posição do *array*:

```
numeros[0] = 15;
```

0	1	2	3	4	5
15					

Ocupando a posição 0, as demais ficam à disposição para armazenar outros valores. Caso você utilize novamente a posição 0 para armazenar um valor, ele substituirá o valor antigo pelo novo. Observe o exemplo:

```
numeros[0] = 20;
```

0	1	2	3	4	5
20					

Preenchendo outras posições:

```
numeros[1] = 8;  
numeros[4] = 16;
```

0	1	2	3	4	5
20	8			16	

A posição pode ser indicada por uma variável inteira. Observe o exemplo (considere que o *array* está vazio):

```
int i = 0; //declara a variável e inicializa com 0
valores[i]=15; //armazena o valor 15 na posição i (0)
i++; //incrementa o valor de i
valores[i]=21; //armazena o valor 21 na posição i (1)
i=i+2; //aumenta 2 no valor de i
valores[i]=33; //armazena o valor 33 na posição i (3)
```

0	1	2	3	4	5
15	21		33		

Se tentarmos acessar uma posição que não existe, ocorrerá um erro em tempo de execução. Observe o exemplo:

```
int x = 10;
numeros[x] = 15; //ERRO! A POSIÇÃO 10 NÃO EXISTE!
```

Preenchendo um array com valores informados pelo usuário

Para preencher um *array* com valores informados pelo usuário podemos utilizar um laço de repetição. Observe o exemplo, que preencherá um *array* criado na classe *Main* com valores digitados no terminal:

```
1 import java.util.Scanner;
2 public class Main{
3     public static void main(String args[]){
4         Scanner ler = new Scanner(System.in);
5         int[] numeros = new int[6];
6
7         for(int i = 0; i < 6; i++){
8             System.out.print("Digite um número: ");
9             numeros[i] = ler.nextInt();
10        }
11    }
12 }
```

Figura 5 – Exemplo de código

Observe que na linha 9 foi utilizada a variável “i” para indicar a posição que receberá o elemento digitado pelo usuário. Assim, a cada volta do *loop*, o número digitado pelo usuário é armazenado numa posição diferente, partindo da primeira (0). Também poderíamos preencher o *array* de trás para frente, invertendo o *for*:

```

1 import java.util.Scanner;
2 public class Main{
3     public static void main(String args[]){
4         Scanner ler = new Scanner(System.in);
5         int[] numeros = new int[6];
6
7         for(int i = 5; i>=0; i--){
8             System.out.print("Digite um número: ");
9             numeros[i] = ler.nextInt();
10        }
11    }
12 }

```

Figura 6 – Exemplo de código

Exibindo valores armazenados no array

Para exibir os valores de um *array* podemos utilizar um laço de repetição, imprimindo uma posição de cada vez no terminal, ou utilizar o método *toString* da classe *Arrays*.

Utilizando um loop para exibir o conteúdo de um array

```

1 import java.util.Scanner;
2 public class Main{
3     public static void main(String args[]){
4         Scanner ler = new Scanner(System.in);
5         int[] pares = {0,2,4,6,8};
6
7         System.out.println("Números pares:");
8         for(int i=0;i<5; i++){
9             System.out.println(pares[i]);
10        }
11    }
12 }

```

Figura 7

Resultado no terminal:

```

Números pares:
0
2
4
6
8

```

Figura 8

Utilizando a classe Arrays para exibir o conteúdo de um array

A classe *Arrays* deve ser importada no início da classe. O método *toString* já possui um formato padrão de exibição, que apresenta os elementos entre colchetes separados por ponto e vírgula. Observe o exemplo:

```

1 import java.util.Scanner;
2 import java.util.Arrays;
3 public class Main{
4     public static void main(String args[]){
5         Scanner ler = new Scanner(System.in);
6         int[] pares = {0,2,4,6,8};
7
8         System.out.println("Números pares:");
9         System.out.println(Arrays.toString(pares));
10    }
11 }

```

Figura 9

Resultado no terminal:

```

Números pares:
[0, 2, 4, 6, 8]

```

Figura 10

O atributo *length*

O atributo *length* armazena a quantidade de elementos que um *array* pode armazenar (considera inclusive as posições vazias). Observe seu uso na linha 10 do exemplo da figura 11:

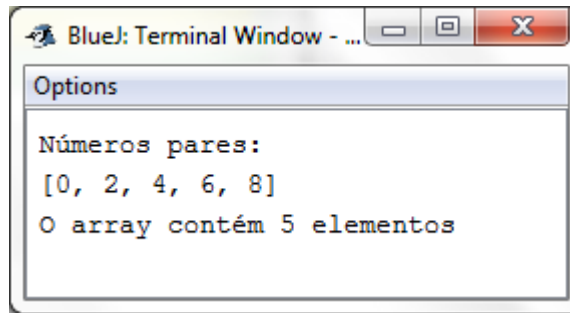
```

1 import java.util.Scanner;
2 import java.util.Arrays;
3 public class Main{
4     public static void main(String args[]){
5         Scanner ler = new Scanner(System.in);
6         int[] pares = {0,2,4,6,8};
7
8         System.out.println("Números pares:");
9         System.out.println(Arrays.toString(pares));
10        System.out.println("O array contém " + pares.length + " elementos");
11    }
12 }

```

Figura 11

Resultado no terminal:



UM ARRAY COMO ATRIBUTO DE UMA CLASSE

Considere que um treinador de um nadador precisa guardar os tempos que seu atleta leva para realizar o nado de 50 metros. Durante o treino, ele repete o percurso 10 vezes, e cada vez que ele faz o percurso o treinador anota o tempo, para depois averiguar a média de tempo, a maior marca e a menor marca. Considerando isso, entendemos que o treino de um nadador possui um conjunto de 10 tempos. Vamos observar o diagrama de classe:

Treino
-nomeAtleta: String -marcas:double[10]
+Treino() +gets/sets +armazenarMarca(numero:int, tempo:double):void +calcularMediaDeMarcas():double +obterMaiorMarca():double +toString():String

Observe que o atributo “marcas” é um conjunto de valores *double* com capacidade para 10 elementos. Vamos analisar o código da classe.

```

1 import java.util.Arrays;
2 public class Treino{
3     private String nome;
4     private double[] marcas;
5
6     public Treino(){
7         this.marcas = new double[10];
8     }
9
10    public String getNome(){return this.nome;}
11    public double[] getMarcas(){return this.marcas;}
12    public void setNome(String nome){this.nome=nome;}
13    public void setMarcas(double[] marcas){this.marcas=marcas;}
14
15    public void armazenarMarca(int numero, double marca){
16        if(numero>=0 && numero<this.marcas.length){
17            this.marcas[numero] = marca;
18        }
19    }

```

Declaramos o atributo `marcas` na lista de atributos da classe, e depois instanciamos o objeto no construtor, determinando que possui a quantidade de 10 elementos. O método “armazenarMarca” recebe o número da marca por argumento e também a marca que será armazenada. Se o número for um maior ou igual a 0 e menor que a quantidade de elementos, ele armazena a marca no número especificado. Vamos observar o restante do código.

```
public double calcularMediaDeMarcas(){
    double soma = 0;
    for(int i=0; i<this.marcas.length; i++){
        soma = soma + this.marcas[i];
    }
    return soma/this.marcas.length;
}

public double obterMaiorMarca(){
    double maior = this.marcas[0];
    for (int i=1; i<this.marcas.length; i++){
        if(this.marcas[i] > maior){
            maior = this.marcas[i];
        }
    }
    return maior;
}

public String toString(){
    return this.nome + "\nMarcas: " + Arrays.toString(this.marcas);
}
}
```

O método “calcularMedia” soma todas as marcas e retorna a divisão da soma pela quantidade de elementos. Já o método `obterMaiorMarca`, armazena a marca de número 0 como sendo a maior. Então, parte da posição seguinte (1) comparando todas com a “maior”. Se localizar alguma maior, substitui o valor da variável pelo novo valor.

Vamos montar a classe *Main*, de forma bem simples, de modo que o treinador consiga informar o nome do atleta, suas 10 marcas, e depois visualize as marcas, a média de marcas e a maior marca.

```
import java.util.Scanner;
public class Main{
    public static void main(String args[]){
        Scanner ler = new Scanner(System.in);
        Treino t1 = new Treino();

        System.out.print("Informe o nome do atleta: ");
        t1.setNome(ler.next());

        for(int i=0; i<10; i++){
            System.out.print("Informe a marca " + i + ": ");
            t1.armazenarMarca(i, ler.nextDouble());
        }

        System.out.println(t1);
        double media = t1.calcularMediaDeMarcas();
        double maior = t1.obterMaiorMarca();

        System.out.println("Média de marcas: " + media);
        System.out.println("Maior marca: " + maior);
    }
}
```

REFERÊNCIAS

ORACLE. **The Java Tutorials: Arrays**. Disponível em <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>. Acesso em 03 de novembro de 2012.

LIMA, Dayvid. **Arrays em Java**. Plugmasters, 2006. Disponível em: <http://www.plugmasters.com.br/sys/materias/535/1/Arrays-em-Java>. Acesso em 25 de setembro de 2012.