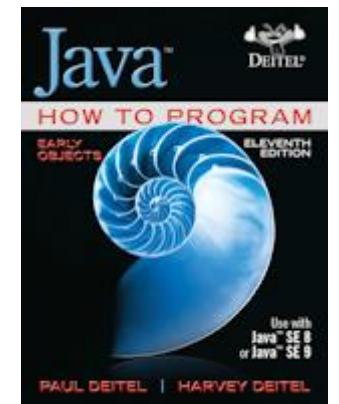


Java – Arrays

Notas de Aula 4
Prof. André Bernardi
andrebernardi@unifei.edu.br



Array

Nome do array (c) →

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	1543
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	6453
c[11]	78

Índice (ou subscrito) do elemento no array c →



Arrays uni-dimensionais

- Declarando e criando:

- `int c[];` `// declare the array variable`
- `c = new int[12];` `// create the array;`
- `int c[] = new int[12];`
- `double[] array1, array2;`

*Em uma declaração de array, especificar o número de elementos entre os colchetes da declaração (por exemplo, `int[12] c;`) **é um erro de sintaxe**.*

Arrays uni-dimensionais

Index	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

```
1 // Figura 7.2: InitArray.java
2 // Inicializando os elementos de um array como valores padrão de zero.
3
4 public class InitArray
5 {
6     public static void main(String[] args)
7     {
8         // declara array variável e o inicializa com um objeto array
9         int[] array = new int[10];           // cria o objeto array
10
11         System.out.printf("%s%8s%n", "Index", "Value"); // títulos de coluna
12
13         // gera saída do valor de cada elemento do array
14         for (int counter = 0; counter < array.length; counter++)
15             System.out.printf("%5d%8d%n", counter, array[counter]);
16     }
17 } // fim da classe InitArray
```



Arrays Uni-dimensionais

Inicializando:

- Através de um inicializador de array

```
int n[] = { 10, 20, 30, 40, 50 };
```

- Usando um loop

```
for ( int i = 0; i < n.length; i++ )
```

```
    n[ i ] = 10 + 10*i;
```

Arrays uni-dimensionais (exemplo)

Index	Value
0	32
1	27
2	64
3	18
4	95
5	14
6	90
7	70
8	60
9	37

```
1 // Figura 7.3: InitArray.java
2 // Inicializando os elementos de um array com um inicializador de array.
3
4 public class InitArray
5 {
6     public static void main(String[] args)
7     {
8         // A lista de inicializador especifica o valor inicial de cada elemento
9         int[] array = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };
10
11         System.out.printf("%s%8s%n", "Index", "Value"); // títulos de coluna
12
13         // gera saída do valor de cada elemento do array
14         for (int counter = 0; counter < array.length; counter++)
15             System.out.printf("%5d%8d%n", counter, array[counter]);
16     }
17 } // fim da classe InitArray
```

```

1 // Figura 7.6: BarChart.java
2 // programa de impressão de gráfico de barras.
3
4 public class BarChart
5 {
6     public static void main(String[] args)
7     {
8         int[] array = { 0, 0, 0, 0, 0, 0, 1, 2, 4, 2, 1 };
9
10        System.out.println("Grade distribution:");
11
12        // para cada elemento de array, gera saída de uma barra do gráfico
13        for (int counter = 0; counter < array.length; counter++)
14        {
15            // gera saída do rótulo de barra ( "00-09: ", ..., "90-99: ", "100: ")
16            if (counter == 10)
17                System.out.printf("%5d: ", 100);
18            else
19                System.out.printf("%02d-%02d: ",
20                                counter * 10, counter * 10 + 9);
21
22            // imprime a barra de asteriscos
23            for (int stars = 0; stars < array[counter]; stars++)
24                System.out.print("*");
25
26            System.out.println();
27        }
28    }
29 } // fim da classe BarChart

```

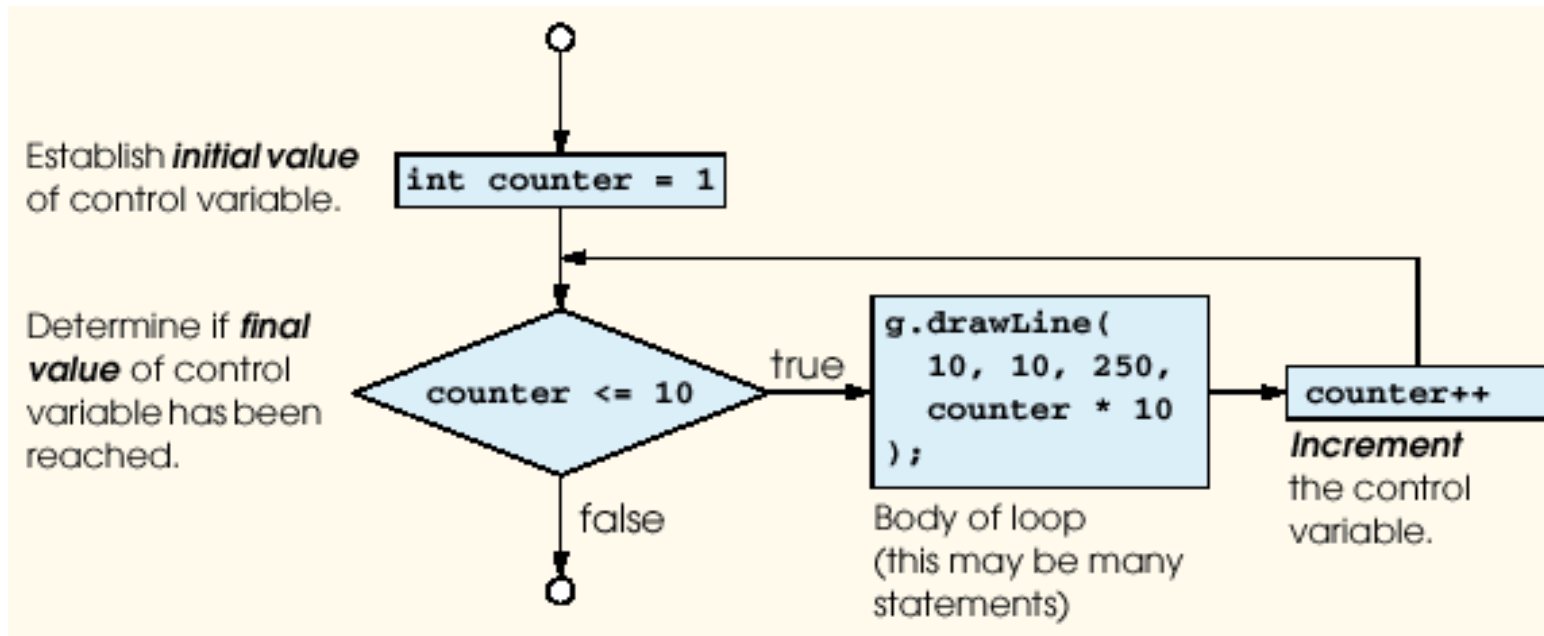
```

Grade distribution:
00-09:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
100: *

```

Estrutura for

```
for ( int counter = 1; counter <= 10; counter++ )  
    g.drawLine( 10, 10, 250, counter * 10 );
```





Enhanced for

```
for ( parametro : arrayName )  
    instruções
```

- Onde o **parametro** deve ser uma variável do tipo do **arrayName**
- Utilizado apenas para recuperação de valores do array.

Não pode ser usado para alteração de dados no array.

Exemplo - Enhanced for

```
1 // Figura 7.12: EnhancedForTest.java
2 // Utilizando a instrução for aprimorada para somar inteiros em um 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         // adiciona o valor de cada elemento ao total
12         for (int number : array)
13             total += number;
14
15         System.out.printf("Total of array elements: %d\n", total);
16     }
17 } // fim da classe EnhancedForTest
```



Arrays Uni-dimensionais

- Acessando elementos com **enhanced for**

```
for ( parameter : arrayName )  
    statement
```

- Exemplo:

```
for ( int i = 0; i < array.length; i++ )  
    total += array[ i ];
```

- Será substituído por:

```
for ( int number : array )  
    total += number;
```



Arrays de Objetos

- Quando criamos um ***array***, do tipo de uma classe na verdade estamos criando um ***array*** de **referências para Objetos**.
- Essas referências devem ser alocadas antes de se utilizar:

```
String name[] = { "B1", "B2", "B3", "B4" };
```

```
// declarando e criando o array
```

```
JButton array[] = new JButton[name.length];
```

```
// inicializando as referencias
```

```
for(int i = 0; i < array.length; i++)  
    array[i] = new JButton(name[i]);
```

Array Multi-Dimensional

	Coluna 0	Coluna 1	Coluna 2	Coluna 3
Linha 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Linha 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Linha 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Índice de coluna
Índice de linha
Nome do array



Array Multi-Dimensional

- `int b[][] = { { 1, 2 }, { 3, 4 } };`
- `int b[][] = { { 1, 2 }, { 3, 4, 5 } };`
- `int b[][];`
`b = new int [3][4]; // retangular`
- `int b[][];`
`b = new int[2][]; // cria 2 linhas`
`b[0] = new int[5]; // cria 5 colunas`
`b[1] = new int[3]; // cria 3 colunas`



Array Multi-Dimensional

- Usando com duas estruturas for aninhadas

```
int total = 0;
for ( int linha = 0; linha < a.length; linha++ )
{
    for ( int col = 0; col < a[ linha ].length; col++ )
        total += a[ linha ][ col ];
}
```



Exercícios

- 1) Crie um programa em Java que declare e inicialize dois arrays unidimensionais do mesmo tamanho, e calcule o produto escalar destes dois arrays. Por exemplo, se os dois arrays forem $\{ 9; 2; 6; 7; 0 \}$ e $\{ 1; 4; 5; 9; 2 \}$ o produto escalar será $9 \times 1 + 2 \times 4 + 6 \times 5 + 7 \times 9 + 0 \times 2 = 110$. Monte uma interface gráfica com JTextFields e eventos de ação para pegar os valores dos vetores e retornar o produto escalar em um JLabel.
- 2) Escreva uma classe que represente um numero inteiro longo com 50 dígitos. Acrescente funções que permitam que estes números possam ser somados e subtraídos. Utilize-a em um programa principal simples. *Sugestão:* utilize um vetor para armazenar os dígitos do numero.



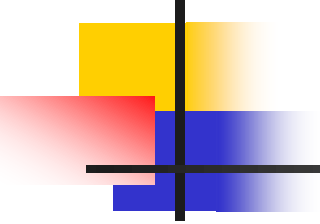
Listas de argumentos de comprimento variável

- O tipo de um argumento, seguido de uma **reticências (...)** na lista de parâmetros de um método indica que ele pode receber um número variável de parâmetros daquele tipo em particular.
- A reticências pode ocorrer apenas **uma vez** na lista de parâmetros formais.
- Deve ser usado sempre no **final** da lista de parâmetros do método.
- É tratado no corpo do método como um **array**.



Exemplo

```
1 // Figura 7.20: VarargsTest.java
2 // Utilizando listas de argumentos de comprimento variável.
3
4 public class VarargsTest
5 {
6     // calcula a média
7     public static double average( double... numbers )
8     {
9         double total = 0.0;
10
11         // calcula total utilizando a instrução for aprimorada
12         for (double d : numbers)
13             total += d;
14
15         return total / numbers.length ;
16     }
17
```



```
d1 = 10.0  
d2 = 20.0  
d3 = 30.0  
d4 = 40.0
```

```
Average of d1 and d2 is 15.0  
Average of d1, d2 and d3 is 20.0  
Average of d1, d2, d3 and d4 is 25.0
```

```
18 public static void main(String[] args)  
19 {  
20     double d1 = 10.0;  
21     double d2 = 20.0;  
22     double d3 = 30.0;  
23     double d4 = 40.0;  
24  
25     System.out.printf("d1 = %.1f%d2 = %.1f%d3 = %.1f%d4 = %.1f%n%n",  
26                       d1, d2, d3, d4);  
27  
28     System.out.printf("Average of d1 and d2 is %.1f%n",  
29                       average(d1, d2) );  
30     System.out.printf("Average of d1, d2 and d3 is %.1f%n",  
31                       average(d1, d2, d3) );  
32     System.out.printf("Average of d1, d2, d3 and d4 is %.1f%n",  
33                       average(d1, d2, d3, d4) );  
34 }  
35 } // fim da classe VarargsTest
```

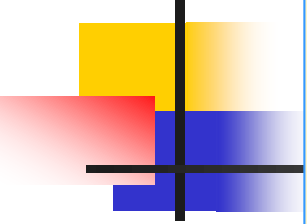


Classe Arrays

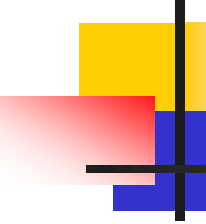
- Fornece métodos estáticos comuns na manipulação de arrays como exemplo:
 - **sort** – para ordenar elementos
 - **binarySearch** – para buscar em um array ordenado
 - **equals** – para comparar arrays
 - **fill** – para preencher os itens de um array.
 - **arraycopy** – copiar elementos de um array para outro



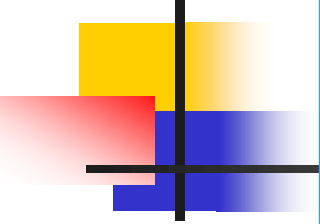
Exemplo



```
1 // Figura 7.22: ArrayManipulations.java
2 // Métodos da classe Arrays e System.arraycopy.
3 import java.util.Arrays;
4
5 public class ArrayManipulations
6 {
7     public static void main(String[] args)
8     {
9         // classifica doubleArray em ordem crescente
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         // preenche o array de 10 elementos com 7s
18         int[] filledIntArray = new int[10];
19         Arrays.fill(filledIntArray, 7);
20         displayArray(filledIntArray, "filledIntArray");
```



```
21
22     // copia array intArray em array intArrayCopy
23     int[] intArray = { 1, 2, 3, 4, 5, 6 };
24     int[] intArrayCopy = new int[intArray.length];
25     Arrays.arraycopy(intArray, 0, intArrayCopy, 0, intArray.length);
26     displayArray(intArray, "intArray");
27     displayArray(intArrayCopy, "intArrayCopy");
28
29     // verifica a igualdade de intArray e intArrayCopy
30     boolean b = Arrays.equals(intArray, intArrayCopy);
31     System.out.printf("%n%nintArray %s intArrayCopy%n",
32         (b ? "==" : "!="));
33
34     // verifica a igualdade de intArray e filledIntArray
35     b = Arrays.equals(intArray, filledIntArray);
36     System.out.printf("intArray %s filledIntArray%n",
37         (b ? "==" : "!="));
38
```



```
39      // pesquisa o valor 5 em intArray
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");
47
48      // pesquisa o valor 8763 em intArray
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
```


Exemplo

```
58 // gera saída de valores em cada 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 } // fim da classe ArrayManipulations
```

```
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
```



Coleções : Classe *ArrayList*

- A API Java fornece várias estruturas de dados predefinidas, chamadas **coleções**, usadas para armazenar grupos de objetos relacionados na memória.
- Um **ArrayList<T>** é semelhante a um array, mas pode ser redimensionado dinamicamente.



Métodos da classe *ArrayList*

Método	Descrição
add	Adiciona um elemento ao <i>final</i> do ArrayList.
clear	Remove todos os elementos do ArrayList.
contains	Retorna true se o ArrayList contém o elemento especificado; caso contrário, retorna false.
get	Retorna o elemento no índice especificado.
indexOf	Retorna o índice da primeira ocorrência do elemento especificado no ArrayList.
remove	Sobrecarregado. Remove a primeira ocorrência do valor especificado ou o elemento no índice especificado.
Size	Retorna o número de elementos armazenados em ArrayList.
trimToSize	Corta a capacidade do ArrayList para o número atual de elementos.



Métodos da classe *ArrayList*

- O método **add** com um argumento adiciona um elemento ao final de um *ArrayList*.
- O método **add** com dois argumentos insere um novo elemento em uma posição especificada em um *ArrayList*.
- O método **size** retorna o número dos elementos atualmente em um *ArrayList*.
- O método **remove** com uma referência a um objeto como um argumento remove o primeiro elemento que corresponde ao valor do argumento.



Métodos da classe *ArrayList*

- O método **remove**, com um argumento **inteiro** remove o elemento no índice especificado, e todos os elementos acima desse índice são deslocados para baixo por um.
- O método **contains** retorna *true* se o elemento é encontrado no *ArrayList* e, do contrário, *false*.



Demonstrando um **ArrayList <String>**

```
1 // Figura 7.24: ArrayListCollection.java
2 // Demonstração da coleção ArrayList<T> genérica.
3 import java.util.ArrayList;
4
5 public class ArrayListCollection
6 {
7     public static void main(String[] args)
8     {
9         // cria um novo ArrayList de strings com uma capacidade inicial de 10
10         ArrayList<String> items = new ArrayList<String>();
11
12         items.add("red");           // anexa um item à lista
13         items.add(0, "yellow");     // insere "yellow" no índice 0
14
15         // cabeçalho
16         System.out.print(
17             "Display list contents with counter-controlled loop:");
18     }
```



Demonstrando um ArrayList <String>

```
19      // exhibe as cores na lista
20      for (int i = 0; i < items.size(); i++)
21          System.out.printf(" %s", items.get(i));
22
23      // exhibe as cores usando for aprimorada no método display
24      display(items,
25          "%nDisplay list contents with enhanced for statement:");
26
27      items.add("green"); // adiciona "green" ao fim da lista
28      items.add("yellow"); // adiciona "yellow" ao fim da lista
29      display(items, "List with two new elements:");
30
31      items.remove("yellow"); // remove o primeiro "yellow"
32      display(items, "Remove first instance of yellow:");
33
34      items.remove(1); // remove o item no índice 1
35      display(items, "Remove second list element (green):");
36
```

```

37 // verifica se um valor está na List
38 System.out.printf("\red\" is %sin the list%n",
39     items.contains("red") ? "": "not ");
40
41 // exhibe o número de elementos na List
42 System.out.printf("Size: %s%n", items.size());
43 }
44
45 // exhibe elementos do ArrayList no console
46 public static void display(ArrayList<String> items, String header)
47 {
48     System.out.printf(header); // exhibe o cabeçalho
49
50     // exhibe cada elemento em itens
51     for (String item : items)
52         System.out.printf(" %s", item);
53
54     System.out.println();
55 }
56 } // fim da classe ArrayListCollection

```

```

Display list contents with counter-controlled loop: yellow red
Display list contents with enhanced for statement: yellow red
List with two new elements: yellow red green yellow
Remove first instance of yellow: red green yellow
Remove second list element (green): red yellow
"red" is in the list
Size: 2

```




Referencias

- Java How to program 3, 4, 5, 6, 7, 8, 9, 10 ed.
Deitel e Deitel

- Sun

<http://java.sun.com>