Introdução

- **Arrays:**
  - **Estruturas de dados que consistem em itens de dados do mesmo tipo relacionados;**
  - **Permanecem com o mesmo tamanho depois de criados.**
    - **Entradas de largura fixa.**

Arrays

- **Um Array:**
  - **É um grupo de variáveis (*elementos* ou *componentes*) que contém valores que são todos do mesmo tipo;**
  - **É um objeto, daí ser considerado *tipo por referência*.**

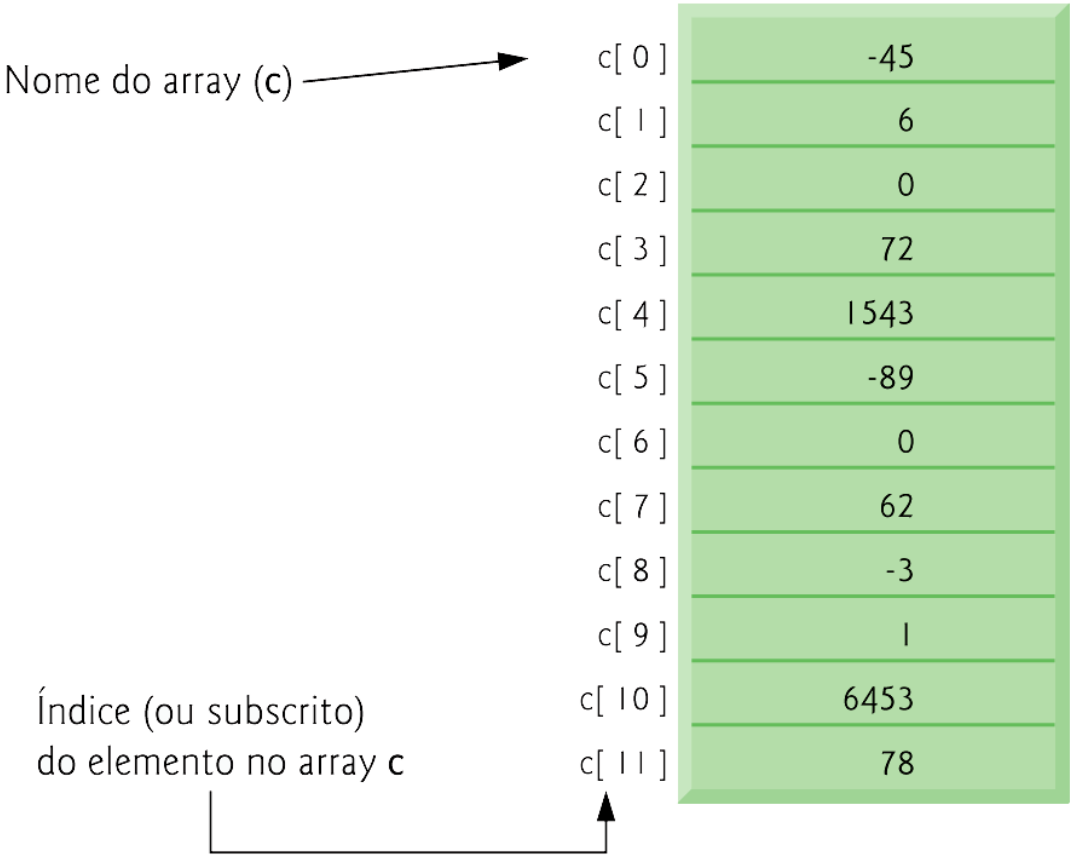

Figura 7.1 | Um array de 12 elementos.

Arrays (*Continuação*)

- **Índice:**
  - **Também chamado subscrito.**
  - **Número da posição entre colchetes.**
  - **Deve ser inteiro positivo ou expressão de inteiro.**
  - **Primeiro elemento tem o índice zero:**

```
a = 5;
b = 6;
c[ a + b ] += 2;
```

- **Adiciona 2 a `c[ 11 ]`.**

Erro comum de programação

Utilizar um valor de tipo `long` como um índice de array resulta em um erro de compilação.

Um índice deve ser um valor `int` ou um valor de um tipo que possa ser promovido para `int` — a saber, `byte`, `short` ou `char`, mas não `long`.

Arrays (*Continuação*)

- **Examine o array `c`:**
  - **`c` é o *nome* do array.**
  - **`c.length` acessa o comprimento do array `c`.**
  - **`c` tem 12 elementos (`c[0]`, `c[1]`, … `c[11]`)**
    - **O *valor* de `c[0]` é `-45`.**

Declarando e criando arrays

- **Declarando e criando arrays:**
  - **Arrays são objetos que ocupam memória.**
  - **São criados dinamicamente com a palavra-chave new:**

```
int c[] = new int[ 12 ];
```

Equivalente a:

```
int c[]; // declara a variável array
c = new int[ 12 ]; // cria o array
```

  - **Também podemos criar arrays de objetos:**

```
String b[] = new String[ 100 ];
```

Erro comum de programação

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

Erro comum de programação

Declarar múltiplas variáveis de array em uma única declaração pode levar a erros sutis. Considere a declaração `int[] a, b, c;` Se `a`, `b` e `c` devem ser declarados como variáveis array, então essa declaração é correta — colocar os colchetes logo depois do tipo indica que todos os identificadores na declaração são variáveis de array. Entretanto, se apenas `a` destina-se a ser uma variável array, e `b` e `c` variáveis `int` individuais, então essa declaração é incorreta — a declaração `int a[], b, c;` alcançaria o resultado desejado.

Resumo

```
1  // Fig. 7.2: InitArray.java
2  // Criando um array.
3
4  public class InitArray
5  {
6     public static void main( Stri
7     {
8        int array[]; // declara o array identificado
9
10       array = new int[ 10 ]; // cria o espaço
11
12       System.out.printf( "%s%8s\n", "Index", "Value" ); // títulos de coluna
13
14       // gera saída do valor de cada elemento do array
15       for ( int counter = 0; counter < array.length; counter++ )
16          System.out.printf( "%5d%8d\n", counter, array[ counter ] );
17    } // fim de main
18 } // fim da classe InitArray
```

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

Declara `array` como um array de `int`s

Cria 10 `int`s para um array; cada `int` é inicializada como 0 por padrão

`array.length` retorna o comprimento do `array`

Cada `int` é inicializado como 0 por padrão

`array[counter]` retorna o `int` associado com o índice em um array

InitArray.java

Linha 8
Declara **array** como um array de **int**s

Linha 10
Cria 10 **int**s para um array; cada **int** é inicializado como **0** por padrão

Linha 15
array.length retorna o comprimento do array

Linha 16
array[counter] retorna o **int** associado com o índice em um array

Saída do programa

Exemplos que utilizam arrays (*Continuação*)

- **Utilizando um inicializador de array:**
  - **Utiliza a *lista inicializadora*:**
    - **Itens entre chaves ({ }).**
    - **Itens em uma lista separada por vírgulas.**

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

    - **Cria um array de cinco elementos.**
    - **Indexa valores de 0, 1, 2, 3, 4.**
  - **Não precisa da palavra-chave new.**

```
1  // Fig. 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        // lista de inicializadores especifica o valor de
9        int array[] = { 32, 27, 64, 18, 95, 14, 90, 70, 60
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    } // fim de main
17 } // fim da classe InitArray
```

Declara `array` como um array de `int`s

Compilador utiliza a lista inicializadora para alocar um array

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

Resumo

itArray.java

nha 9
Declara array como um array de ints

Linha 9
Compilador utiliza a lista inicializadora para alocar um array

Saída do programa

Exemplos que utilizam arrays (*Continuação*)

- **Calculando um valor para armazenar em cada elemento de array:**
  - **Inicializa elementos do array de 10 elementos como inteiros pares.**

Resumo

```
1  // Fig. 7.4: InitArray.java
2  // Calculando valores a serem colocados em elementos de um array.
3
4  public class InitArray
5  {
6     public static void main( String args[] )
7     {
8        final int ARRAY_LENGTH = 10; // declara a constante
9        int array[] = new int[ ARRAY_LENGTH ]; // cria o ar
10
11       // calcula o valor para cada elemento do array
12       for ( int counter = 0; counter < array.length; counter++ )
13          array[ counter ] = 2 + 2 * counter;
14
15       System.out.printf( "%s%8s\n", "Index", "Value" ); // títulos de coluna
16
17       // gera saída do valor de cada elemento do array
18       for ( int counter = 0; counter < array.length; counter++ )
19          System.out.printf( "%5d%8d\n", counter
20    } // fim de main
21 } // fim da classe InitArray
```

```
Index   Value
    0       2
    1       4
    2       6
    3       8
    4      10
    5      12
    6      14
    7      16
    8      18
    9      20
```

Declara a variável constante ARRAY_LENGTH utilizando o modificador `final`

Declara e cria um `array` que contém 10 `int`s

Utiliza o índice `array` para atribuir um valor de array

InitArray.java

Linha 8
Declara a variável constante

Linha 9
Declara e cria um array que contém 10 `int`s

Linha 13
Utiliza o índice do array para atribuir o array

Saída do programa

Exemplos que utilizam arrays (*Continuação*)

- **Somando elementos de um array:**
  - **Elementos do array podem representar uma série de valores.**
    - **Podemos somar esses valores.**

```
1  // Fig. 7.5: SumArray.java
2  // Calculando a soma dos elementos de um
3
4  public class SumArray
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 counter = 0; counter < array.length; counter++ )
13          total += array[ counter ];
14
15       System.out.printf( "Total of array element
16    } // fim de main
17 } // fim da classe SumArray
```

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

Declara `array` com uma lista inicializadora

Soma todos os valores do `array`

Resumo

SumArray.java

Linha 8
Declara array com uma lista inicializadora

Linhas 12-13
Soma todos os valores de array

Saída do programa

Exemplos que utilizam arrays (*Continuação*)

- **Utilizando gráficos de barras para exibir dados de array graficamente:**
  - **Apresenta os dados graficamente.**
    - **Por exemplo: gráfico de barras.**
  - **Examina a distribuição das notas.**

```
1  // Fig. 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          // output bar label ( "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(); // inicia uma nova linha de
27       } // fim do for externo
28    } // fim de main
29 } // fim da classe BarChart
```

Declara `array` com uma lista inicializadora

Utiliza o flag 0 para exibir nota de um algarismo com um 0 inicial

Para cada elemento do `array`, imprime o número associado de asteriscos

Resumo

BarChart.java

(1 de 2)

Linha 8
Declara array com uma lista inicializadora

Linha 19
Utiliza o flag 0 para exibir nota de um

Para cada

do

prime

de

asteriscos

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

Resumo

BarChart.java

(2 de 2)

Saída do programa

Exemplos que utilizam arrays (*Continuação*)

- **Utilizando os elementos de um array como contadores:**
  - **Utiliza uma série de variáveis contadoras para resumir os dados.**

Resumo

```
1  // Fig. 7.7: RollDie.java
2  // Rola um dado de seis lados 6000 vezes.
3  import java.util.Random;
4
5  public class RollDie
6  {
7     public static void main( String args[] )
8     {
9        Random randomNumbers = new Random(); // gerador de número aleatório
10       int frequency[] = new int[ 7 ]; // array de contadores de freqüência
11
12       // lança o dados 6000 vezes; usa o valor do dado como
13       for ( int roll = 1; roll <= 6000; roll++ )
14          ++frequency[ 1 + randomNumbers.nextInt( 6 ) ];
15
16       System.out.printf( "%s%10s\n"
17
18       // gera saída do valor de cad
19       for ( int face = 1; face < fr
20          System.out.printf( "%4d%10d\n", face, frequency[ face ] );
21    } // fim de main
22 } // fim da classe RollDie
```

```
Face Frequency
   1       988
   2       963
   3      1018
   4      1041
   5       978
   6      1012
```

Declara `frequency` como um array de 7 `int`s

Gera 6000 inteiros aleatórios no intervalo 1-6

Incrementa os valores de `frequency` no índice associado com um número aleatório

...Die.java

Linha 10
Declara **frequency** como um array de 7

Linhas 13-14
Geram 6000 inteiros aleatórios no intervalo 1-6

Linha 14
Incrementa valores de frequency no índice associado com o número aleatório

Saída do programa

Exemplos que utilizam arrays (*Continuação*)

- **Utilizando arrays para analisar resultados de pesquisas:**
  - **40 alunos avaliam a qualidade da comida:**
    - **Escala de avaliação de `1-10` — `1` significa horrível, `10` significa excelente.**
  - **Coloque 40 respostas no array de inteiros.**
  - **Resuma os resultados da enquete.**

Resumo

```
1  // Fig. 7.8: StudentPoll.java
2  // Programa de análise de enquete.
3
4  public class StudentPoll
5  {
6     public static void main( String args[] )
7     {
8        // array de respostas da pesquisa
9        int responses[] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
10          10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5,
11          4, 8, 6, 8, 10 };
12       int frequency[] = new int[ 11 ]; // array de contad
13
14       // para cada resposta, seleciona elemento de respostas e usa esse valor
15       // como índice de freqüência para determinar elemento a incrementar
16       for ( int answer = 0; answer < responses.length; answer++ )
17          ++frequency[ responses[ answer ] ];
18
19       System.out.printf( "%s%10s", "Rating", "Frequency" );
20
21       // gera saída do valor de cada elemento do array
22       for ( int rating = 1; rating < frequency.length; rating++ )
23          System.out.printf( "%d%10d", rating, frequency[ rating ] );
24    } // fim de main
25 } // fim da classe StudentPoll
```

Declare `responses` as array to store 40 responses

Declara `frequency` como um array de 11 `int`s e ignora o primeiro elemento

Para cada resposta, incrementa os valores de `frequency` no índice associados com essa resposta

StudentPoll.java

(1 de 2)

onses
y para
armazenar 40
respostas
Linha 12
Declara frequency
como array de 11
gnora o
elemento
5-17
a resposta,
tam os
de
frequency no índice
associado com essa
resposta

```
Rating Frequency
     1         2
     2         2
     3         2
     4         2
     5         5
     6        11
     7         5
     8         7
     9         1
    10         3
```

Resumo

StudentPoll.java

(2 de 2)

Saída do programa

Estudo de caso: Simulação de embaralhamento e distribuição de cartas

- **O programa simula o ato de embaralhar e distribuir cartas:**
  - **Utiliza geração de números aleatórios.**
  - **Utiliza um array de elementos de tipo por referência para representar as cartas.**
  - **Três classes:**
    - `Card`:
      - **Representa uma carta do baralho.**
    - `DeckOfCards`:
      - **Representa um baralho com 52 cartas.**
    - `DeckOfCardsTest`:
      - **Demonstra o embaralhamento e a distribuição de cartas.**

Resumo

Card.java

Linhas 17-20

```
1  // Fig. 7.9: Card.java
2  // Classe Card representa uma carta de baralho.
3
4  public class Card
5  {
6     private String face; // face da carta ("Ace", "Deuce", ...)
7     private String suit; // naipe da carta ("Hearts", "Diamonds", ...)
8
9     // construtor de dois argumentos inicializa face e naipe da carta
10    public Card( String cardFace, String cardSuit )
11    {
12       face = cardFace; // inicializa face da carta
13       suit = cardSuit; // inicializa naipe da carta
14    } // fim do construtor Card de dois argume
15
16    // retorna representação String de Card
17    public String toString()
18    {
19       return face + " of " + suit;
20    } // fim do método toString
21 } // fim da classe Card
```

Retorna a representação de string de uma carta

Resumo

```
1  // Fig. 7.10: DeckOfCards.java
2  // Classe DeckOfCards representa um baralho.
3  import java.util.Random;
4
5  public class DeckOfCards
6  {
7     private Card deck[]; // array de objetos Card
8     private int currentCard; // índice do próximo Card a ser
9     private final int NUMBER_OF_CARDS = 52; // número constante de Cards
10    private Random randomNumbers; // gerador de número aleatório
11
12    // construtor preenche baralho de c
13    public DeckOfCards()
14    {
15       String faces[] = { "Ace", "Deuce
16          "Seven", "Eight", "Nine", "Te
17       String suits[] = { "Hearts", "Di
18
19       deck = new Card[ NUMBER_OF_CARDS ]; // cria array de objetos Card
20       currentCard = 0; // configura currentCard então o primeiro Card distribuído é deck[ 0 ]
21       randomNumbers = new Random(); // cria gerador de r
22
23       // preenche baralho com objetos Card
24       for ( int count = 0; count < deck.length; count++ )
25          deck[ count ] =
26             new Card( faces[ count % 13 ], suits[ count / 13 ] );
27    } // fim do construtor DeckOfCards
```

Declara `deck` como um array para armazenar objetos `Card`

A constante `NUMBER_OF_CARDS` indica o número de `Cards` no baralho

(1 de 2)

Declara e inicializa `faces` com `String`s que representam a face da carta

Linha 9

Declara e inicializa `suits` com `String`s que representam o naipe da carta

Linhas 15-16

Linha 17

Linhas 24-26

Preenche o array `deck` com `Cards`

Resumo

DeckOfCards.java

(2 de 2)

```
28
29    // embaralha as cartas com um algoritmo de uma passagem
30    public void shuffle()
31    {
32       // depois de embaralhar, a distribuição deve iniciar em deck[ 0 ] novamente
33       currentCard = 0; // reinicializa currentCard
34
35       // para cada Card, seleciona outro Card aleatório e os compara
36       for ( int first = 0; first < deck.length; first++ )
37       {
38          // seleciona um número aleatório entre 0 e 51
39          int second =  randomNumbers.nextInt( NUMBER_OF_CARDS );
40
41          // compara Card atual com Card aleatoriamente selecio
42          Card temp = deck[ first ];
43          deck[ first ] = deck[ second ];
44          deck[ second ] = temp;
45       } // fim de for
46    } // fim do método shuffle
47
48    // distribui um Card
49    public Card dealCard()
50    {
51       // determina se ainda há Cards a ser distribuídos
52       if ( currentCard < deck.length )
53          return deck[ currentCard++ ]; // retorna Card atual no array
54       else
55          return null; // retorna nulo p/ indicar que todos os Cards foram distribuídos
56    } // fim do método dealCard
57 } // fim da classe DeckOfCards
```

Troca a `Card` atual por uma `Card` aleatoriamente selecionada

Linha 52

Determine se `deck` está vazio

Resumo

DeckOfCards Test

.java

(1 de 2)

```
1  // Fig. 7.11: DeckOfCardsTest.java
2  // Aplicativo de embaralhar e distribuir cartas.
3
4  public class DeckOfCardsTest
5  {
6     // executa o aplicativo
7     public static void main( String args[] )
8     {
9        DeckOfCards myDeckOfCards = new DeckOfCards();
10       myDeckOfCards.shuffle(); // coloca Cards em ordem aleatória
11
12       // imprime todas as 52 cartas na ordem em que elas são distribuídas
13       for ( int i = 0; i < 13; i++ )
14       {
15          // distribui e imprime 4 Cards
16          System.out.printf( "%-20s%-20s%-20s%-20s\n",
17             myDeckOfCards.dealCard(), myDeckOfCards.dealCard(),
18             myDeckOfCards.dealCard(), myDeckOfCards.dealCard() );
19       } // fim de for
20    } // fim de main
21 } // fim da classe DeckOfCardsTest
```

```
Six of Spades      Eight of Spades    Six of Clubs       Nine of Hearts
Queen of Hearts    Seven of Clubs     Nine of Spades     King of Hearts
Three of Diamonds  Deuce of Clubs     Ace of Hearts      Ten of Spades
Four of Spades     Ace of Clubs       Seven of Diamonds  Four of Hearts
Three of Clubs     Deuce of Hearts    Five of Spades     Jack of Diamonds
King of Clubs      Ten of Hearts      Three of Hearts    Six of Diamonds
Queen of Clubs     Eight of Diamonds  Deuce of Diamonds  Ten of Diamonds
Three of Spades    King of Diamonds   Nine of Clubs      Six of Hearts
Ace of Spades      Four of Diamonds   Seven of Hearts    Eight of Clubs
Deuce of Spades    Eight of Hearts    Five of Hearts     Queen of Spades
Jack of Hearts     Seven of Spades    Four of Clubs      Nine of Diamonds
Ace of Diamonds    Queen of Diamonds  Five of Clubs      King of Spades
Five of Diamonds   Ten of Clubs       Jack of Spades     Jack of Clubs
```

Resumo

DeckOfCardsTest.java

(2 de 2)

A estrutura for aprimorada

- **Instrução `for` aprimorada:**
  - **Novo recurso do J2SE 5.0.**
  - **Permite iterar por elementos de um array ou por uma coleção sem utilizar um contador.**
  - **Sintaxe:**

```
for ( parâmetro : nomeDoArray )
   instrução
```

Resumo

EnhancedForTest.java

```
1  // Fig. 7.12: EnhancedForTest.java
2  // Utilizando 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    } // fim de main
17 } // fim da classe EnhancedForTest
```

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

Para cada iteração, atribui o próximo elemento do `array` à variável `int number`, e então o adiciona a `total`

A estrutura de repetição for (*Continuação*)

- **As linhas 12-13 são equivalentes a:**

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

- **Uso:**
  - **Pode acessar elementos do array.**
  - **Não pode modificar elementos do array.**
  - **Não pode acessar o contador que indica o índice.**

Passando arrays para métodos

- **Para passar um argumento de array para um método:**
  - **Especifique o nome de array sem colchetes:**
    - **Array `hourlyTemperatures` é declarado como**

```
int hourlyTemperatures = new int[
24 ];
```

    - **A chamada de método:**

```
modifyArray( hourlyTemperatures );
```

    - **Passa o array `hourlyTemperatures` para o método `modifyArray`.**

Resumo

PassArray.java

(de 2)

Linha 9

Linha 19

```
1  // Fig. 7.13: PassArray.java
2  // Passando arrays e elementos do arrays individuais aos métodos.
3
4  public class PassArray
5  {
6     // main cria array e chama modifyArr
7     public static void main( String args[] )
8     {
9        int array[] = { 1, 2, 3, 4, 5 };
10
11       System.out.println(
12          "Effects of passing reference to entire array:\n" +
13          "The values of the original array are:" );
14
15       // gera saída de elementos do array original
16       for ( int value : array )
17          System.out.printf( "   %d", value );
18
19       modifyArray( array ); // passa referência de array
20       System.out.println( "\n\nThe values of the modified array are:" );
21
22       // gera saída de elementos do array modificado
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 ] );
```

Declara `array` de 5 `int`s com uma lista inicializadora

Passa o array inteiro para o método `modifyArray`

```
29
30       modifyElement( array[ 3 ] ); // tenta modificar o array[ 3 ]
31       System.out.printf(
32          "array[3] after modifyElement
33    } // fim de main
34
35    // multiplica cada elemento de um array por 2
36    public static void modifyArray( int array2[] )
37    {
38       for ( int counter = 0; counter < array2.length; counter++ )
39          array2[ counter ] *= 2;
40    } // fim do método modifyArray
41
42    // multiplica o argumento por 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    } // fim do método modifyElement
49 } // fim da classe PassArray
```

Passa o elemento de array `array[3]` para o método `modifyElement`

O método `modifyArray` manipula o array diretamente

O método `modifyElement` manipula a cópia de um primitivo

```
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
```

Saída do programa

Resumo

y.j

(2 de 2)

Linha 30

Linhas 36-40

Linhas 43-48

Passando arrays para métodos (*Continuação*)

- **Notas sobre a passagem de argumentos para métodos:**
  - **Duas maneiras de passar argumentos para métodos:**
    - **Passagem *por valor*:**
      - **Cópia do valor do argumento é passada para o método chamado.**
      - **No Java, todo primitivo é passado por valor.**
    - **Passagem *por referência*:**
      - **O chamador fornece ao método chamado acesso direto aos dados do chamador.**
      - **Método chamado pode manipular esses dados.**
      - **Desempenho aprimorado em relação à passagem por valor.**
      - **No Java, todo objeto é passado por referência.**
        - **No Java, arrays são objetos.**
        - **Portanto, arrays são passados para os métodos por referência.**

Estudo de caso: Classe GradeBook utilizando um array para armazenar notas

- **Desenvolve ainda mais a classe GradeBook.**
- **Classe GradeBook:**
  - **Representa um boletim de notas que armazena e analisa notas.**
  - **Não mantém valores de notas individuais.**
  - **A repetição dos cálculos exige a reinserção das mesmas notas.**
    - **Isso pode ser resolvido armazenando notas em um array.**

Resumo

GradeBook.java

(de 5)

Linha 7

Linha 13

```
1  // Fig. 7.14: GradeBook.java
2  // Grade book utilizando um array para armazenar notas de teste.
3
4  public class GradeBook
5  {
6     private String courseName; // nome do curso que essa GradeBook representa
7     private int grades[]; // array de notas de aluno
8
9     // construtor de dois argumentos inicializa courseName e array de notas
10    public GradeBook( String name, int gradesArray[] )
11    {
12       courseName = name; // inicializa courseName
13       grades = gradesArray; // armazena notas
14    } // construtor de dois argumentos inicializa courseName e array de notas
15
16    // método para configura o nome do curso
17    public void setCourseName( String name )
18    {
19       courseName = name; // armazena o nome do curso
20    } // fim do método setCourseName
21
22    // método para recuperar o nome do curso
23    public String getCourseName()
24    {
25       return courseName;
26    } // fim do método getCourseName
27
```

Declara o array `grades` para armazenar notas individuais

Atribui a referência do array à variável de instância `grades`

Resumo

GradeBook.java

(2 de 5)

```
28    // exibe uma mensagem de boas-vindas para o usuário GradeBook
29    public void displayMessage()
30    {
31       // getCourseName obtém o nome do curso
32       System.out.printf( "Welcome to the grade book for\n%s!\n\n",
33          getCourseName() );
34    } // fim do método displayMessage
35
36    // realiza várias operações nos dados
37    public void processGrades()
38    {
39       // gera saída de array de notas
40       outputGrades();
41
42       // chama método getAverage para calcular a média
43       System.out.printf( "\nClass average is %.2f\n", getAverage() );
44
45       // chama métodos getMinimum e getMaximum
46       System.out.printf( "Lowest grade is %d\nHighest grade is %d\n\n",
47          getMinimum(), getMaximum() );
48
49       // chama outputBarChart para imprimir gráfico de distribuição de nota
50       outputBarChart();
51    } // fim do método processGrades
52
53    // encontra nota mínima
54    public int getMinimum()
55    {
56       int lowGrade = grades[ 0 ]; // assume que grades[ 0 ] é a menor nota
57
```

Resumo

GradeBook.java

(3 de 5)

Linhas 59-64

Linhas 75-80

```
58       // faz um loop pelo array de notas
59       for ( int grade : grades )
60       {
61          // se nota for mais baixa que lowGrade, atribui-a a lowGrade
62          if ( grade < lowGrade )
63             lowGrade = grade; // nova nota mais baixa
64       } // for final
65
66       return lowGrade; // retorna a menor nota
67    } // fim do método getMinimum
68
69    // localiza nota máxima
70    public int getMaximum()
71    {
72       int highGrade = grades[ 0 ]; // assume que grades[ 0 ] é a maior nota
73
74       // faz um loop pelo array de notas
75       for ( int grade : grades )
76       {
77          // se a nota for maior que highGrade, atribui essa nota a highGrade
78          if ( grade > highGrade )
79             highGrade = grade; // nova nota mais alta
80       } // for final
81
82       return highGrade; // retorna a nota mais alta
83    } // fim do método getMaximum
84
```

Faz um loop pelas `grades` a fim de encontrar a nota mais baixa

Faz um loop pelas `grades` para encontrar a nota mais alta

Resumo

GradeBook.java

(4 de 5)

Linhas 91-92

Linhas 107-108

```
85   // determina média para o teste
86   public double getAverage()
87   {
88      int total = 0; // inicializa o total
89
90      // soma notas de um aluno
91      for ( int grade : grades )
92         total += grade;
93
94      // retorna a média de notas
95      return (double) total / grades.length;
96   } // fim do método getAverage
97
98   // gera a saída do gráfico de barras exibindo distribuição de notas
99   public void outputBarChart()
100    {
101       System.out.println( "Grade distribution:" );
102
103       // armazena freqüência de notas em cada intervalo de 10 notas
104       int frequency[] = new int[ 11 ];
105
106       // para cada nota, incrementa a freqüência apropriada
107       for ( int grade : grades )
108          ++frequency[ grade / 10 ];
109
```

Faz um loop pelas `grades` para somar as notas de um aluno

Faz um loop pelas `grades` para calcular a freqüência

Resumo

GradeBook.java

(5 de 5)

Linhas 134-136

```
110      // para cada freqüência de nota, imprime barra no gráfico
111      for ( int count = 0; count < frequency.length; count++ )
112      {
113         // gera saída do rótulo de barra ( "00-09: ", ..., "90-99: ", "100: " )
114         if ( count == 10 )
115            System.out.printf( "%5d: ", 100 );
116         else
117            System.out.printf( "%02d-%02d: ",
118               count * 10, count * 10 + 9  );
119
120         // imprime a barra de asteriscos
121         for ( int stars = 0; stars < frequency[ count ]; stars++ )
122            System.out.print( "*" );
123
124         System.out.println(); // inicia uma nova linha de saída
125      } // fim do for externo
126   } // fim do método outputBarChart
127
128   // gera a saída do conteúdo do array de notas
129   public void outputGrades()
130   {
131      System.out.println( "The grades are:\n" );
132
133      // gera a saída da nota de cada aluno
134      for ( int student = 0; student < grades.length; student++ )
135         System.out.printf( "Student %2d: %3d\n",
136            student + 1, grades[ student ] );
137   } // fim do método outputGrades
138 } // fim da classe GradeBook
```

Faz um loop pelas `grades` para exibir cada nota

Resumo

```
1  // Fig. 7.15: GradeBookTest.java
2  // Cria objeto GradeBook utilizando um array de notas.
3
4  public class GradeBookTest
5  {
6     // método main inicia a execução de programa
7     public static void main( String args[] )
8     {
9        // array de notas de aluno
10       int gradesArray[] = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
11
12       GradeBook myGradeBook = new GradeBook(
13          "CS101 Introduction to Java Programming", gradesArray );
14       myGradeBook.displayMessage();
15       myGradeBook.processGrades();
16    } // fim de main
17 } // fim da classe GradeBookTest
```

Declara e inicializa `gradesArray` com 10 elementos

Passa `gradesArray` para construtor de `GradeBook`

deBook
Test
.java

(1 de 2)

Linha 10

Linha 13

```
Welcome to the grade book for
CS101 Introduction to Java Programming!

The grades are:

Student  1:  87
Student  2:  68
Student  3:  94
Student  4: 100
Student  5:  83
Student  6:  78
Student  7:  85
Student  8:  91
Student  9:  76
Student 10:  87

Class average is 84.90
Lowest grade is 68
Highest grade is 100

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

Resumo

GradeBook Test .java

(2 de 2)

Saída do programa

Listas de argumentos de comprimento variável

- **Listas de argumentos de comprimento variável:**
  - **Novo recurso no J2SE 5.0.**
  - **Número não-especificado de argumentos.**
  - **Utilize reticências (…) na lista de parâmetros do método.**
    - **Pode ocorrer somente uma vez na lista de parâmetros.**
    - **Deve ser colocado no final da lista de parâmetros**
  - **O array cujos elementos são todos do mesmo tipo.**

```
1  // Fig. 7.20: VarargsTest.java
2  // Utilizando listas de argumentos de comprimento variável.
3
4  public class VarargsTest
5  {
6     // calcula média
7     public static double average( double... numbers )
8     {
9        double total = 0.0; // inicializa total
10
11       // calcula total usando a instruç
12       for ( double d : numbers )
13          total += d;
14
15       return total / numbe
16    } // fim do método averag
17
18    public static void m
19    {
20       double d1 = 10.0;
21       double d2 = 20.0;
22       double d3 = 30.0;
23       double d4 = 40.0;
24
```

O método `average` recebe uma seqüência variável de comprimento de `doubles`

Calcula o total dos `doubles` no array

Acessa `numbers.length` para obter o tamanho do array `numbers`

Resumo

`VarargsTest.java`

e 2)

Linha 7

Linhas 12-13

Linha 15

```
25       System.out.printf( "d1 = %.1f\nd2 = %.1f\nd3 = %.1f\nd4 = %.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 d
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    } // fim de main
35 } // fim da classe VarargsTest
```

```
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
```

Invoca o método `average` com dois argumentos

Invoca o método `average` com três argumentos

Invoca método `average` com quatro argumentos

Resumo

`VarargsTest.java`

(de 2)

Linha 29

Linha 31

Linha 33

Saída do programa