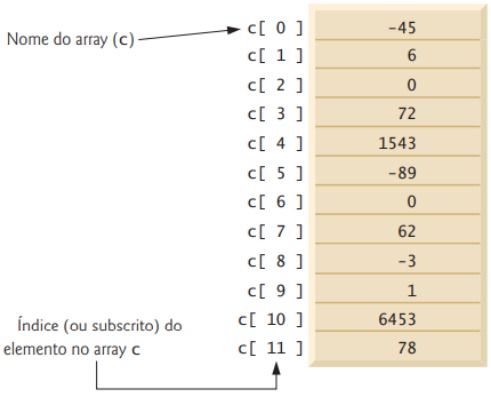
7.2 Arrays



7.4 Exemplos que utilizam arrays

```
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
```

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

```
1  // Figura 7.3: 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
```

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

```java
// Figura 7.4: InitArray.java
// Calculando os valores a serem colocados nos elementos de um array.

public class InitArray
{
   public static void main(String[] args)
   {
      final int ARRAY_LENGTH = 10; // declara constante
      int[] array = new int[ARRAY_LENGTH]; // cria o array

      // calcula valor de cada elemento do array
      for (int counter = 0; counter < array.length; counter++)
         array[counter] = 2 + 2 * counter;

      System.out.printf("%s%8s%n", "Index", "Value"); // títulos de coluna

      // gera saída do valor de cada elemento do array
      for (int counter = 0; counter < array.length; counter++)
         System.out.printf("%5d%8d%n", counter, array[counter]);
   }
} // 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
```

```java
// Figura 7.5: SumArray.java
// Calculando a soma dos elementos de um array.

public class SumArray
{
   public static void main(String[] args)
   {
      int[] array = { 87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
      int total = 0;

      // adiciona o valor de cada elemento ao total
      for (int counter = 0; counter < array.length; counter++)
         total += array[counter];

      System.out.printf("Total of array elements: %d%n", total);
   }
} // fim da classe SumArray
```

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

```java
// Figura 7.6: BarChart.java
// programa de impressão de gráfico de barras.

public class BarChart
{
   public static void main(String[] args)
   {
      int[] array = { 0, 0, 0, 0, 0, 0, 1, 2, 4, 2, 1 };

      System.out.println("Grade distribution:");

      // para cada elemento de array, gera saída de uma barra do gráfico
      for (int counter = 0; counter < array.length; counter++)
      {
         // gera saída do rótulo de barra ( "00-09: ", ..., "90-99: ", "100: ")
         if (counter == 10)
            System.out.printf("%5d: ", 100);
         else
            System.out.printf("%02d-%02d: ",
               counter * 10, counter * 10 + 9);

         // imprime a barra de asteriscos
         for (int stars = 0; stars < array[counter]; stars++)
            System.out.print("*");

         System.out.println();
      }
   }
} // 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: *
```

```java
// Figura 7.7: RollDie.java
// Programa de jogo de dados utilizando arrays em vez de switch.
import java.security.SecureRandom;

public class RollDie
{
   public static void main(String[] args)
   {
      SecureRandom randomNumbers = new SecureRandom();
      int[] frequency = new int[7]; // array de contadores de frequência

      // lança o dado 6.000.000 vezes; usa o valor do dado como índice de frequência
      for (int roll = 1; roll <= 6000000; roll++)
         ++frequency[1 + randomNumbers.nextInt(6)];

      System.out.printf("%s%10s%n", "Face", "Frequency");

      // gera saída do valor de cada elemento do array
      for (int face = 1; face < frequency.length; face++)
         System.out.printf("%4d%10d%n", face, frequency[face]);
   }
} // fim da classe RollDie
```

```
Face Frequency
   1    999690
   2    999512
   3   1000575
   4    999815
   5    999781
   6   1000627
```

```java
// Figura 7.8: StudentPoll.java
// Programa de análise de enquete.

public class StudentPoll
{
   public static void main(String[] args)
   {
      // array das respostas dos alunos (mais tipicamente, inserido em tempo de execução)
      int[] responses = { 1, 2, 5, 4, 3, 5, 2, 1, 3, 3, 1, 4, 3, 3, 3,
         2, 3, 3, 2, 14 };
      int[] frequency = new int[6]; // array de contadores de frequência

      // para cada resposta, seleciona elemento de respostas e utiliza esse valor
      // como índice de frequência para determinar elemento a incrementar
      for (int answer = 0; answer < responses.length; answer++)
      {
         try
         {
            ++frequency[responses[answer]];
         }
         catch (ArrayIndexOutOfBoundsException e)
         {
            System.out.println(e); // invoca o método toString
            System.out.printf("   responses[%d] = %d%n%n",
               answer, responses[answer]);
         }
      }

      System.out.printf("%s%10s%n", "Rating", "Frequency");

      // gera saída do valor de cada elemento do array
      for (int rating = 1; rating < frequency.length; rating++)
         System.out.printf("%6d%10d%n", rating, frequency[rating]);
   }
} // fim da classe StudentPoll
```

```
java.lang.ArrayIndexOutOfBoundsException: 14
   responses[19] = 14

Rating Frequency
     1         3
     2         4
     3         8
     4         2
     5         2
```

```java
// Figura 7.9: Card.java
// Classe Card representa uma carta de baralho.

public class Card
{
   private final String face; // face da carta ("Ace", "Deuce", ...)
   private final String suit; // naipe da carta ("Hearts", "Diamonds", ...)

   // construtor de dois argumentos inicializa face e naipe da carta
   public Card(String cardFace, String cardSuit)
   {
      this.face = cardFace; // inicializa face da carta
      this.suit = cardSuit; // inicializa naipe da carta
   }

   // retorna representação String de Card
   public String toString()
   {
      return face + " of " + suit;
   }
} // fim da classe Card
```

```java
// Figura 7.10: DeckOfCards.java
// classe DeckOfCards representa um baralho.
import java.security.SecureRandom;

public class DeckOfCards
{
   private Card[] deck; // array de objetos Card
   private int currentCard; // índice da próxima Card a ser distribuída (0-51)
   private static final int NUMBER_OF_CARDS = 52; // número constante de Cards
   // gerador de número aleatório
   private static final SecureRandom randomNumbers = new SecureRandom();

   // construtor preenche baralho de cartas
   public DeckOfCards()
   {
      String[] faces = { "Ace", "Deuce", "Three", "Four", "Five", "Six",
         "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King" };
      String[] suits = { "Hearts", "Diamonds", "Clubs", "Spades" };

      deck = new Card[NUMBER_OF_CARDS]; // cria array de objetos Card
      currentCard = 0; // a primeira Card distribuída será o deck[0]

      // preenche baralho com objetos Card
      for (int count = 0; count < deck.length; count++)
         deck[count] =
            new Card(faces[count % 13], suits[count / 13]);
   }

   // embaralha as cartas com um algoritmo de uma passagem
   public void shuffle()
   {
      // a próxima chamada para o método dealCard deve começar no deck[0] novamente
      currentCard = 0;

      // para cada Card, seleciona outra Card aleatória (0-51) e as compara
      for (int first = 0; first < deck.length; first++)
      {
         // seleciona um número aleatório entre 0 e 51
         int second = randomNumbers.nextInt(NUMBER_OF_CARDS);

         // compara Card atual com Card aleatoriamente selecionada
         Card temp = deck[first];
         deck[first] = deck[second];
         deck[second] = temp;
      }
   }

   // distribui uma Card
   public Card dealCard()
   {
      // determina se ainda há Cards a serem distribuídas
      if (currentCard < deck.length)
         return deck[currentCard++]; // retorna Card atual no array
      else
         return null; // retorna nulo para indicar que todos as Cards foram distribuídas
   }
} // fim da classe DeckOfCards
```

```
1  // Figura 7.11: DeckOfCardsTest.java
2  // Embaralhando e distribuindo 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 = 1; i <= 52; i++)
14       {
15          // distribui e exibe uma Card
16          System.out.printf("%-19s", myDeckOfCards.dealCard());
17
18          if (i % 4 == 0) // gera uma nova linha após cada quarta carta
19             System.out.println();
20       }
21    }
22 } // 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
```

7.7 A instrução for aprimorada

```
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
```

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

7.8 Passando arrays para métodos

```
1  // Figura 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 modifyArray e 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          // gera saída de elementos do array original
16          for (int value : array)
17             System.out.printf("   %d", value);
18
19          modifyArray(array); // passa a referência do array
20          System.out.printf("%n%nThe values of the modified array are:%n");
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]);
29
30          modifyElement(array[3]); // tenta modificar o array[3]
31          System.out.printf(
32             "array[3] after modifyElement: %d%n", array[3]);
33       }
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       }
41
42       // multiplica 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       }
49    } // fim da classe 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
```

7.10 Estudo de caso: classe GradeBook utilizando um array para armazenar notas

```
1  // Figura 7.14: GradeBook.java
2  // classe GradeBook 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
10    public GradeBook(String courseName, int[] grades)
11    {
12       this.courseName = courseName;
13       this.grades = grades;
14    }
15
16    // método para configurar o nome do curso
17    public void setCourseName(String courseName)
18    {
19       this.courseName = courseName;
20    }
21
```

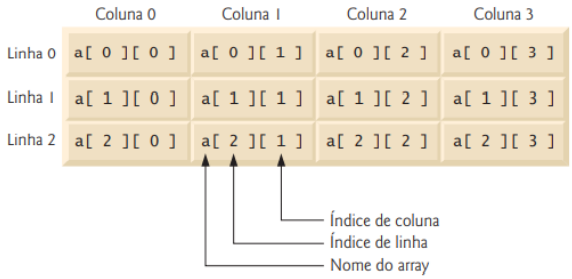
```
22      // método para recuperar o nome do curso
23      public String getCourseName()
24      {
25         return courseName;
26      }
27
28      // realiza várias operações nos dados
29      public void processGrades()
30      {
31         // gera saída de array de notas
32         outputGrades();
33
34         // chama método getAverage para calcular a nota média
35         System.out.printf("%nClass average is %.2f%n", getAverage());
36
37         // chama métodos getMinimum e getMaximum
38         System.out.printf("Lowest grade is %d%nHighest grade is %d%n%n",
39            getMinimum(),getMaximum());
40
41         // chama outputBarChart para imprimir gráfico de distribuição de nota
42         outputBarChart();
43      }
44
45      // localiza nota mínima
46      public int getMinimum()
47      {
48         int lowGrade = grades[0]; // supõe que grades[0] é a menor nota
49
50         // faz um loop pelo array de notas
51         for (int grade : grades)
52         {
53            // se nota for mais baixa que lowGrade, atribui essa nota a lowGrade
54            if (grade < lowGrade)
55               lowGrade = grade; // nova nota mais baixa
56         }
57
58         return lowGrade;
59      }
60
61      // localiza nota máxima
62      public int getMaximum()
63      {
64         int highGrade = grades[0]; // supõe que grades[0] é a maior nota
65
66         // faz um loop pelo array de notas
67         for (int grade : grades)
68         {
69            // se a nota for maior que highGrade, atribui essa nota a highGrade
70            if (grade > highGrade)
71               highGrade = grade; // nova nota mais alta
72         }
73
74         return highGrade;
75      }
76
77      // determina média para o teste
78      public double getAverage()
79      {
80         int total = 0;
81
82         // soma notas de um aluno
83         for (int grade : grades)
84            total += grade;
85
86         // retorna média de notas
87         return (double) total / grades.length;
88      }
89
90      // gera a saída do gráfico de barras exibindo distribuição de notas
91      public void outputBarChart()
92      {
93         System.out.println("Grade distribution:");
94
```

```
95          // armazena frequência de notas em cada intervalo de 10 notas
96          int[] frequency = new int[11];
97
98          // para cada nota, incrementa a frequência apropriada
99          for (int grade : grades)
100            ++frequency[grade / 10];
101
102         // para cada frequência de nota, imprime barra no gráfico
103         for (int count = 0; count < frequency.length; count++)
104         {
105            // gera saída do rótulo de barra ( "00-09: ", ..., "90-99: ", "100: ")
106            if (count == 10)
107               System.out.printf("%5d: ", 100);
108            else
109               System.out.printf("%02d-%02d: ",
110                  count * 10, count * 10 + 9);
111
112            // imprime a barra de asteriscos
113            for (int stars = 0; stars < frequency[count]; stars++)
114               System.out.print("*");
115
116            System.out.println();
117         }
118      }
119
120      // gera a saída do conteúdo do array de notas
121      public void outputGrades()
122      {
123         System.out.printf("The grades are:%n%n");
124
125         // gera a saída da nota de cada aluno
126         for (int student = 0; student < grades.length; student++)
127            System.out.printf("Student %2d: %3d%n",
128               student + 1, grades[student]);
129      }
130   } // fim da classe GradeBook
```

```
1    // Figura 7.15: GradeBookTest.java
2    // GradeBookTest cria um objeto GradeBook utilizando um array de notas,
3    // e, então, invoca o método processGrades para analisá-las.
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         System.out.printf("Welcome to the grade book for%n%s%n%n",
15            myGradeBook.getCourseName());
16         myGradeBook.processGrades();
17      }
18   } // fim da classe GradeBookTest
```

```
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: *
```

7.11 Arrays multidimensionais




```
1  // Figura 7.17: InitArray.java
2  // Inicializando arrays bidimensionais.
3
4  public class InitArray
5  {
6     // cria e gera saída de arrays bidimensionais
7     public static void main(String[] args)
8     {
9        int[][] array1 = {{1, 2, 3}, {4, 5, 6}};
10       int[][] array2 = {{1, 2}, {3}, {4, 5, 6}};
11
12       System.out.println("Values in array1 by row are");
13       outputArray(array1); // exibe array1 por linha
14
15       System.out.printf("%nValues in array2 by row are%n");
16       outputArray(array2); // exibe array2 por linha
17    }
18
```

```
19      // gera saída de linhas e colunas de um array bidimensional
20      public static void outputArray(int[][] array)
21      {
22        // faz um loop pelas linhas do array
23        for (int row = 0; row < array.length; row++)
24        {
25            // faz um loop pelas colunas da linha atual
26            for (int column = 0; column < array[row].length; column++)
27               System.out.printf("%d  ", array[row][column]);
28
29            System.out.println();
30        }
31      }
32   } // fim da classe InitArray
```

```
Values in array1 by row are
1  2  3
4  5  6

Values in array2 by row are
1  2
3
4  5  6
```

7.12 Estudo de caso: classe GradeBook utilizando um array bidimensional

```
1   // Figura 7.18: GradeBook.java
2   // classe GradeBook utilizando um array bidimensional para armazenar notas.
3
4   public class GradeBook
5   {
6      private String courseName; // nome de curso que este livro de nota representa
7      private int[][] grades; // array bidimensional de notas de aluno
8
9      // construtor de dois argumentos inicializa courseName e array de notas
10     public GradeBook(String courseName, int[][] grades)
11     {
12        this.courseName = courseName;
13        this.grades = grades;
14     }
15
16     // método para configurar o nome do curso
17     public void setCourseName(String courseName)
18     {
19        this.courseName = courseName;
20     }
21
22     // método para recuperar o nome do curso
23     public String getCourseName()
24     {
25        return courseName;
26     }
27
28     // realiza várias operações nos dados
29     public void processGrades()
30     {
31        // gera saída de array de notas
32        outputGrades();
33
34        // chama métodos getMinimum e getMaximum
35        System.out.printf("%n%s %d%n%s %d%n%n",
36           "Lowest grade in the grade book is", getMinimum(),
37           "Highest grade in the grade book is", getMaximum());
38
39        // gera a saída de gráfico de distribuição de notas de todas as notas em todos os testes
40        outputBarChart();
41     }
42
```

```
43      // localiza nota mínima
44      public int getMinimum()
45      {
46         // supõe que o primeiro elemento de array de notas é o menor
47         int lowGrade = grades[0][0];
48
49         // faz um loop pelas linhas do array de notas
50         for (int[] studentGrades : grades)
51         {
52            // faz um loop pelas colunas da linha atual
53            for (int grade : studentGrades)
54            {
55               // se a nota for menor que lowGrade, atribui a nota a lowGrade
56               if (grade < lowGrade)
57                  lowGrade = grade;
58            }
59         }
60
61         return lowGrade;
62      }
63
64      // localiza nota máxima
65      public int getMaximum()
66      {
67         // supõe que o primeiro elemento de array de notas é o maior
68         int highGrade = grades[0][0];
69
70         // faz um loop pelas linhas do array de notas
71         for (int[] studentGrades : grades)
72         {
73            // faz um loop pelas colunas da linha atual
74            for (int grade : studentGrades)
75            {
76               // se a nota for maior que highGrade, atribui essa nota a highGrade
77               if (grade > highGrade)
78                  highGrade = grade;
79            }
80         }
81
82         return highGrade;
83      }
84
85      // determina a média do conjunto particular de notas
86      public double getAverage(int[] setOfGrades)
87      {
88         int total = 0;
89
90         // soma notas de um aluno
91         for (int grade : setOfGrades)
92            total += grade;
93
94         // retorna média de notas
```

```
95          return (double) total / setOfGrades.length;
96       }
97
98       // gera a saída do gráfico de barras para exibir distribuição total de notas
99       public void outputBarChart()
100      {
101         System.out.println("Overall grade distribution:");
102
103         // armazena frequência de notas em cada intervalo de 10 notas
104         int[] frequency = new int[11];
105
106         // para cada nota em GradeBook, incrementa a frequência apropriada
107         for (int[] studentGrades : grades)
108         {
109            for (int grade : studentGrades)
110               ++frequency[grade / 10];
111         }
112
113         // para cada frequência de nota, imprime barra no gráfico
114         for (int count = 0; count < frequency.length; count++)
115         {
116            // gera saída do rótulo de barra ("00-09: ", ..., "90-99: ", "100: ")
117            if (count == 10)
118               System.out.printf("%5d: ", 100);
119            else
120               System.out.printf("%02d-%02d: ",
121                  count * 10, count * 10 + 9);
122
123            // imprime a barra de asteriscos
124            for (int stars = 0; stars < frequency[count]; stars++)
125               System.out.print("*");
126
127            System.out.println();
128         }
129      }
130
131      // gera a saída do conteúdo do array de notas
132      public void outputGrades()
133      {
134         System.out.printf("The grades are:%n%n");
135         System.out.print("            "); // alinha títulos de coluna
136
137         // cria um título de coluna para cada um dos testes
138         for (int test = 0; test < grades[0].length; test++)
139            System.out.printf("Test %d  ", test + 1);
140
141         System.out.println("Average"); // título da coluna de média do aluno
142
143         // cria linhas/colunas de texto que representam notas de array
144         for (int student = 0; student < grades.length; student++)
145         {
146            System.out.printf("Student %2d", student + 1);
147
148            for (int test : grades[student]) // gera saída de notas do aluno
149               System.out.printf("%8d", test);
150
151            // chama método getAverage para calcular a média do aluno;
152            // passa linha de notas como o argumento para getAverage
153            double average = getAverage(grades[student]);
154            System.out.printf("%9.2f%n", average);
155         }
156      }
157   } // fim da classe GradeBook
```

```
1  // Figura 7.19: GradeBookTest.java
2  // GradeBookTest cria o objeto GradeBook utilizando um array bidimensional
3  // das notas e, então, invoca o método processGrades para analisá-las.
4  public class GradeBookTest
5  {
6     // método main inicia a execução de programa
7     public static void main(String[] args)
8     {
9        // array bidimensional de notas de aluno
10       int[][] gradesArray = {{87, 96, 70},
11                              {68, 87, 90},
```

```
                                {94, 100, 90},
                                {100, 81, 82},
                                {83, 65, 85},
                                {78, 87, 65},
                                {85, 75, 83},
                                {91, 94, 100},
                                {76, 72, 84},
                                {87, 93, 73}};

            GradeBook myGradeBook = new GradeBook(
               "CS101 Introduction to Java Programming", gradesArray);
            System.out.printf("Welcome to the grade book for%n%s%n%n",
               myGradeBook.getCourseName());
            myGradeBook.processGrades();
         }
      } // fim da classe GradeBookTest
```

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

The grades are:

            Test 1  Test 2  Test 3  Average
Student  1      87      96      70    84.33
Student  2      68      87      90    81.67
Student  3      94     100      90    94.67
Student  4     100      81      82    87.67
Student  5      83      65      85    77.67
Student  6      78      87      65    76.67
Student  7      85      75      83    81.00
Student  8      91      94     100    95.00
Student  9      76      72      84    77.33
Student 10      87      93      73    84.33

Lowest grade in the grade book is 65
Highest grade in the grade book is 100

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

7.13 Listas de argumentos de comprimento variável

```
// Figura 7.20: VarargsTest.java
// Utilizando listas de argumentos de comprimento variável.

public class VarargsTest
{
   // calcula a média
   public static double average(double... numbers)
   {
      double total = 0.0;

      // calcula total utilizando a instrução for aprimorada
      for (double d : numbers)
         total += d;

      return total / numbers.length;
   }

   public static void main(String[] args)
   {
      double d1 = 10.0;
      double d2 = 20.0;
      double d3 = 30.0;
      double d4 = 40.0;

      System.out.printf("d1 = %.1f%nd2 = %.1f%nd3 = %.1f%nd4 = %.1f%n%n",
         d1, d2, d3, d4);

```

```
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
```

```
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
```

7.14 Utilizando argumentos de linha de comando

```
1    // Figura 7.21: InitArray.java
2    // Inicializando um array com argumentos de linha de comando.
3
4    public class InitArray
5    {
6       public static void main(String[] args)
7       {
8          // verifica número de argumentos de linha de comando
9          if (args.length != 3)
10            System.out.printf(
11               "Error: Please re-enter the entire command, including%n" +
12               "an array size, initial value and increment.%n");
13         else
14         {
15            // obtém o tamanho do array a partir do primeiro argumento de linha de comando
16            int arrayLength = Integer.parseInt(args[0]);
17            int[] array = new int[arrayLength];
18
19            // obtém o valor inicial e o incrementa a partir dos argumentos de linha de comando
20            int initialValue = Integer.parseInt(args[1]);
21            int increment = Integer.parseInt(args[2]);
22
23            // calcula valor de cada elemento do array
24            for (int counter = 0; counter < array.length; counter++)
25               array[counter] = initialValue + increment * counter;
26
27            System.out.printf("%s%8s%n", "Index", "Value");
28
29            // exibe o valor e o índice de array
30            for (int counter = 0; counter < array.length; counter++)
31               System.out.printf("%5d%8d%n", counter, array[counter]);
32         }
33      }
34   } // fim da classe InitArray
```

```
java InitArray
Error: Please re-enter the entire command, including
an array size, initial value and increment.
```

```
java InitArray 5 0 4
Index   Value
    0       0
    1       4
    2       8
    3      12
    4      16
```

```
java InitArray 8 1 2
Index   Value
    0       1
    1       3
    2       5
    3       7
    4       9
    5      11
    6      13
    7      15
```

7.15 Classe Arrays

```java
// Figura 7.22: ArrayManipulations.java
// Métodos da classe Arrays e System.arraycopy.
import java.util.Arrays;

public class ArrayManipulations
{
   public static void main(String[] args)
   {
      // classifica doubleArray em ordem crescente
      double[] doubleArray = { 8.4, 9.3, 0.2, 7.9, 3.4 };
      Arrays.sort(doubleArray);
      System.out.printf("%ndoubleArray: ");

      for (double value : doubleArray)
         System.out.printf("%.1f ", value);

      // preenche o array de 10 elementos com 7s
      int[] filledIntArray = new int[10];
      Arrays.fill(filledIntArray, 7);
      displayArray(filledIntArray, "filledIntArray");

      // copia array intArray em array intArrayCopy
      int[] intArray = { 1, 2, 3, 4, 5, 6 };
      int[] intArrayCopy = new int[intArray.length];
      System.arraycopy(intArray, 0, intArrayCopy, 0, intArray.length);
      displayArray(intArray, "intArray");
      displayArray(intArrayCopy, "intArrayCopy");

      // verifica a igualdade de intArray e intArrayCopy
      boolean b = Arrays.equals(intArray, intArrayCopy);
      System.out.printf("%n%nintArray %s intArrayCopy%n",
         (b ? "==" : "!="));

      // verifica a igualdade de intArray e filledIntArray
      b = Arrays.equals(intArray, filledIntArray);
      System.out.printf("intArray %s filledIntArray%n",
         (b ? "==" : "!="));

      // pesquisa o valor 5 em intArray
      int location = Arrays.binarySearch(intArray, 5);

      if (location >= 0)
         System.out.printf(
            "Found 5 at element %d in intArray%n", location);
      else
         System.out.println("5 not found in intArray");

      // pesquisa o valor 8763 em intArray
      location = Arrays.binarySearch(intArray, 8763);

      if (location >= 0)
         System.out.printf(
            "Found 8763 at element %d in intArray%n", location);
      else
         System.out.println("8763 not found in intArray");
   }

   // gera saída de valores em cada array
   public static void displayArray(int[] array, String description)
   {
      System.out.printf("%n%s: ", description);

      for (int value : array)
         System.out.printf("%d ", value);
   }
} // 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
```

7.16 Introdução a coleções e classe `ArrayList`

Método	Descrição
`add`	Adiciona um elemento ao *final* 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.

```
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
19       // exibe as cores na lista
20       for (int i = 0; i < items.size(); i++)
21          System.out.printf(" %s", items.get(i));
22
23       // exibe 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       // exibe o número de elementos na List
42       System.out.printf("Size: %s%n", items.size());
43    }
44
45    // exibe elementos do ArrayList no console
46    public static void display(ArrayList<String> items, String header)
47    {
48       System.out.printf(header); // exibe o cabeçalho
49
50       // exibe 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
```

7.17 (Opcional) Estudo de caso de GUIs e imagens gráficas: desenhando arcos

```
1  // Figura 7.25: DrawRainbow.java
2  // Desenhando um arco-íris com arcos e um array de cores.
3  import java.awt.Color;
4  import java.awt.Graphics;
5  import javax.swing.JPanel;
6
7  public class DrawRainbow extends JPanel
8  {
9     // define as cores índigo e violeta
10    private final static Color VIOLET = new Color(128, 0, 128);
11    private final static Color INDIGO = new Color(75, 0, 130);
12
13    // Cores a utilizar no arco-íris, iniciando da parte mais interna
14    // As duas entradas em branco resultam em um arco vazio no centro
15    private Color[] colors =
16       { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
17         Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
18
19    // construtor
20    public DrawRainbow()
21    {
22       setBackground(Color.WHITE); // configura o fundo como branco
23    }
24
25    // desenha um arco-íris utilizando arcos concêntricos
26    public void paintComponent(Graphics g)
27    {
28       super.paintComponent(g);
29
30       int radius = 20; // raio de um arco
31
32       // desenha o arco-íris perto da parte central inferior
33       int centerX = getWidth() / 2;
34       int centerY = getHeight() - 10;
35
36       // desenha arcos preenchidos com o mais externo
37       for (int counter = colors.length; counter > 0; counter--)
38       {
39          // configura a cor para o arco atual
40          g.setColor(colors[counter - 1]);
41
42          // preenche o arco de 0 a 180 graus
43          g.fillArc(centerX - counter * radius,
44             centerY - counter * radius,
45             counter * radius * 2, counter * radius * 2, 0, 180);
46       }
47    }
48 } // fim da classe DrawRainbow
```

```
1  // Figura 7.26: DrawRainbowTest.java
2  // Aplicativo de teste para exibir um arco-íris.
3  import javax.swing.JFrame;
4
5  public class DrawRainbowTest
6  {
7     public static void main(String[] args)
8     {
9        DrawRainbow panel = new DrawRainbow();
10       JFrame application = new JFrame();
11
12       application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13       application.add(panel);
14       application.setSize(400, 250);
15       application.setVisible(true);
16    }
17 } // fim da classe DrawRainbowTest
```