```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//          PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo01 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo01
{
// ---------------------------------------------  definicao de metodos

/**
  * ex0101 - comandos basicos
  */
   public static void ex0101 ( )
   {
     IO.clrscr  ( );  // limpar a tela
                      // (dependente do sistema operacional)
     IO.println ( "EXEMPLO0101 - Programa em Java" );
     IO.pause   ( );  // pausa
   } // fim ex0101 ( )

/**
  * ex0102 - comandos basicos
  *               com melhoria na interacao
  */
   public static void ex0102 ( )
   {
     IO.println ( "EXEMPLO0102 - Programa em Java" );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0102 ( )


/**
  * ex0103 - comandos basicos
  *               com identificacao
  */
   public static void ex0103 ( )
   {
     IO.println ( "EXEMPLO0103 - Programa em Java" );
     IO.println ( );  // para mudar de linha
     IO.println ( "MATRICULA: ______ ALUNO : ____________________" );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0103 ( )
```

```
/**
  * ex0104 - comandos basicos
  *               com leitura de caractere
  */
   public static void ex0104 ( )
   {
     char x;          // definir dado

     IO.println ( "EXEMPLO0104 - Programa em Java" );
     IO.println ( );  // para mudar de linha
     IO.println ( "MATRICULA: ______ ALUNO : ____________________" );
     IO.println ( );
     x = IO.readchar ( "Digitar um simbolo qualquer: " );
     IO.println ( "Valor lido = " + x );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0104 ( )

/**
  * ex0105 - comandos basicos
  *               com leitura de inteiro
  */
   public static void ex0105 ( )
   {
     int x;           // definir dado

     IO.println ( "EXEMPLO0105 - Programa em Java" );
     IO.println ( );  // para mudar de linha
     IO.println ( "MATRICULA: ______ ALUNO : ____________________" );
     IO.println ( );
     x = IO.readint ( "Digitar um valor inteiro qualquer: " );
     IO.println ( "Valor lido = " + x );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0105 ( )

/**
  * ex0106 - comandos basicos
  *               com leitura de real
  */
   public static void ex0106 ( )
   {
     double x;        // definir dado

     IO.println ( "EXEMPLO0106 - Programa em Java" );
     IO.println ( );  // para mudar de linha
     IO.println ( "MATRICULA: ______ ALUNO : ____________________" );
     IO.println ( );
     x = IO.readdouble ( "Digitar um valor real qualquer: " );
     IO.println ( "Valor lido = " + x );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0106 ( )
```

```
/**
  * ex0107 - comandos basicos
  *               com leitura de booleano
  */
   public static void ex0107 ( )
   {
     boolean x;       // definir dado

     IO.println ( "EXEMPLO0107 - Programa em Java" );
     IO.println ( );  // para mudar de linha
     IO.println ( "MATRICULA: ______ ALUNO : ____________________" );
     IO.println ( );
     x = IO.readboolean ( "Digitar um valor logico (true | false): " );
     IO.println ( "Valor lido = " + x );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0107 ( )

/**
  * ex0108 - comandos basicos
  *          com leitura de caracteres
  */
   public static void ex0108 ( )
   {
     String x;        // definir dado

     IO.println ( "EXEMPLO0108 - Programa em Java" );
     IO.println ( );  // para mudar de linha
     IO.println ( "MATRICULA: ______ ALUNO : ____________________" );
     IO.println ( );
     x = IO.readString ( "Digitar uma cadeida de caracteres qualquer: " );
     IO.println ( "Valor lido = " + x );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0108 ( )

/**
  * ex0109 - comandos basicos
  *               combinando variaveis
  */
   public static void ex0109 ( )
   {
                    // definir dados
     final  double pi = 3.1415;
     String frase     = "PI = ";

     IO.println ( "EXEMPLO0109 - Programa em Java" );
     IO.println ( );  // para mudar de linha
     IO.println ( "MATRICULA: ______ ALUNO : ____________________" );
     IO.println ( );
     frase = frase + pi;
     IO.println ( frase );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0109 ( )
```

```
/**
  * ex0110 - comandos basicos
  *               combinando variaveis de varios tipos
  */
   public static void ex0110 ( )
   {
                    // definir dados
     final   double pi = 3.1415;
     String frase      = "PI = ";
     char   simbolo  = '\n';
     int      numero   = 10;

     IO.print   ( "EXEMPLO01" + numero + " - Programa em Java" + simbolo );
     IO.print   ( simbolo );  // para mudar de linha
     IO.print   ( "MATRICULA: _____ ALUNO : _____" + simbolo );
     frase = frase + pi;
     IO.print   ( frase + simbolo );
     IO.pause   ( "APERTAR ENTER PARA TERMINAR." );
   } // fim ex0110 ( )

// ------------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // chamadas de metodos
//     ex0101 ( );
//     ex0102 ( );
//     ex0103 ( );
//     ex0104 ( );
//     ex0105 ( );
//     ex0106 ( );
//     ex0107 ( );
//     ex0108 ( );
//     ex0109 ( );
//     ex0110 ( );
   } // fim main ( )

// ------------------------------------------------

} // end class Exemplo01
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo02 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo02
{
// ---------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                  ( String titulo, String versao, String data,
                    String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                  ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )
```

```
// ---------------------------------------------  definicao de metodos

/**
  * ex0201 - comandos basicos
  *               com valor inteiro
  */
   public static void ex0201 ( )
   {
   // definir dado
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0201 - Programa em Java" );
      IO.println ( "Operar um valor inteiro" );
      IO.println ( );
   // ler dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // mostrar valor operado
      IO.println ( "Dobro do valor lido = " + (x+x) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0201 ( )

/**
  * ex0202 - comandos basicos
  *               com valores inteiros
  */
   public static void ex0202 ( )
   {
   // definir dados
      int x, y;
   // identificar o metodo
      IO.println ( "EXEMPLO0202 - Programa em Java" );
      IO.println ( "Operar valores inteiros" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // ler segundo  dado
      y = IO.readint ( "Fornecer outro valor inteiro: " );
   // mostrar valores operados
      IO.println ( "Valores operados (x+y) = " + (x+y) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0202 ( )
```

```
/**
  * ex0203 - comandos basicos
  *               com valores inteiros
  */
   public static void ex0203 ( )
   {
   // definir dados
      int x, y, z;
   // identificar o metodo
      IO.println ( "EXEMPLO0203 - Programa em Java" );
      IO.println ( "Operar valores inteiros" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // ler segundo  dado
      y = IO.readint ( "Fornecer outro valor inteiro: " );
   // mostrar valores operados
      z = x + y;
      IO.println ( "Valores operados (x+y) = " + z );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0203 ( )

/**
  * ex0204 - comandos basicos
  *               com valores inteiros e
  *               com valor   real
  */
   public static void ex0204 ( )
   {
   // definir dados
      int x, y, z;
      double  a;
   // identificar o metodo
      IO.println ( "EXEMPLO0204 - Programa em Java" );
      IO.println ( "Operar valores inteiros e valor real" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // ler segundo  dado
      y = IO.readint ( "Fornecer outro valor inteiro: " );
   // mostrar valores operados
      z = x * y;
      a = x * y;
      IO.println ( "Valores operados (x*y) = " + z + " = " + a );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0204 ( )
```

```
/**
  * ex0205 - comandos basicos
  *              com valores inteiros e
  *              com valor   real
  */
   public static void ex0205 ( )
   {
   // definir dados
      int    x, y, z;
      double a, b;
   // identificar o metodo
      IO.println ( "EXEMPLO0205 - Programa em Java" );
      IO.println ( "Operar valores inteiros e valores reais" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // ler segundo  dado
      y = IO.readint ( "Fornecer outro valor inteiro: " );
   // mostrar valores operados
      z = x / y;
      a = x / y;
      b = 1.0 * x / y;
      IO.println ( "Valores operados (x/y) = " + z + " = " + a + " = " + b );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0205 ( )

/**
  * ex0206 - comandos basicos
  *              com valores reais
  */
   public static void ex0206 ( )
   {
   // definir dados
      double x, y, z;
   // identificar o metodo
      IO.println ( "EXEMPLO0206 - Programa em Java" );
      IO.println ( "Operar valores reais" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readdouble ( "Fornecer um valor real: " );
   // ler segundo  dado
      y = IO.readdouble ( "Fornecer outro valor real: " );
   // mostrar valores operados
      z = x / y;
      IO.println ( "Valores operados (x/y) = " + z );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0206 ( )
```

```
/**
  * ex0207 - comandos basicos
  *               com valores logicos
  */
   public static void ex0207 ( )
   {
   // definir dados
      boolean x, y, z;
   // identificar o metodo
      IO.println ( "EXEMPLO0207 - Programa em Java" );
      IO.println ( "Operar valores logicos" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readboolean ( "Fornecer um valor logico (true|false): " );
   // ler segundo  dado
      y = IO.readboolean ( "Fornecer outro valor logico (true|false): " );
   // mostrar valores operados
      z = x || y;
      IO.println ( "Valores operados (x||y) = " + z );
      z = x && y;
      IO.println ( "Valores operados (x&&y) = " + z );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0207 ( )

/**
  * ex0208 - comandos basicos
  *               com caracteres
  */
   public static void ex0208 ( )
   {
   // definir dados
      char x, y;
      String  z;
   // identificar o metodo
      IO.println ( "EXEMPLO0208 - Programa em Java" );
      IO.println ( "Operar caracteres" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readchar ( "Fornecer um caractere: " );
   // ler segundo  dado
      y = IO.readchar ( "Fornecer outro caractere: " );
   // mostrar valores operados
      z = (""+x) + (""+y);    // converter e concatenar
      IO.println ( "Valores concatenados = " + z );
      z = "" + y + x;
      IO.println ( "Valores concatenados em outra ordem = " + z );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0208 ( )
```

```
/**
  * ex0209 - comandos basicos
  *               com caracteres
  */
   public static void ex0209 ( )
   {
   // definir dados
      String  x, y, z;
   // identificar o metodo
      IO.println ( "EXEMPLO0209 - Programa em Java" );
      IO.println ( "Operar caracteres" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readString ( "Fornecer uma palavra: " );
   // ler segundo  dado
      y = IO.readString ( "Fornecer outra palavra: " );
   // mostrar valores operados
      z = x + y;       // concatenar
      IO.println ( "Valores concatenados = " + z );
      z = y + x;
      IO.println ( "Valores concatenados em outra ordem = " + z );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0209 ( )

/**
  * ex0210 - comandos basicos
  *               com caracteres
  */
   public static void ex0210 ( )
   {
   // definir dados
      String  x, y;
      boolean z1;
      int     z2;
   // identificar o metodo
      IO.println ( "EXEMPLO0210 - Programa em Java" );
      IO.println ( "Operar caracteres" );
      IO.println ( );
   // ler primeiro dado
      x = IO.readString ( "Fornecer uma palavra: " );
   // ler segundo  dado
      y = IO.readString ( "Fornecer outra palavra: " );
   // mostrar valores operados
      z1= x.equals (y);
      IO.println ( "Valores comparados = " + z1 );
      z2= x.compareTo (y);
   // se x = y entao z2 sera' igual a   zero
   // se x > y entao z2 sera' maior que zero
   // se x < y entao z2 sera' menor que zero
      IO.println ( "Valores comparados = " + z2 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0210 ( )
```

```
// ------------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO02 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
      ex0201 ( );
      ex0202 ( );
      ex0203 ( );
      ex0204 ( );
      ex0205 ( );
      ex0206 ( );
      ex0207 ( );
      ex0208 ( );
      ex0209 ( );
      ex0210 ( );
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ------------------------------------------------

} // end class Exemplo02
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo03 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo03
{
// ---------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                  ( String titulo, String versao, String data,
                    String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                  ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )
```

```
// ---------------------------------------------  definicao de metodos

/**
  * ex0301 - teste de valor inteiro
  */
   public static void ex0301 ( )
   {
   // definir dado
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0301 - Programa em Java" );
      IO.println ( "Testar um valor inteiro" );
      IO.println ( );
   // ler dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // testar valor
      if ( x == 0 )
      {
            IO.println ( "Valor igual a zero." );
      }
      if ( x != 0 )
      {
            IO.println ( "Valor diferente de zero." );
      }
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0301 ( )

/**
  * ex0302 - teste de valor inteiro
  */
   public static void ex0302 ( )
   {
   // definir dado
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0302 - Programa em Java" );
      IO.println ( "Testar um valor inteiro" );
      IO.println ( );
   // ler dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // testar valor
      if ( x == 0 )
      {
            IO.println ( "Valor igual a zero." );
      }
      else
      {
            IO.println ( "Valor diferente de zero." );
      } // fim se
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0302 ( )
```

```
/**
  * ex0303 - teste de valor inteiro
  */
   public static void ex0303 ( )
   {
   // definir dado
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0303 - Programa em Java" );
      IO.println ( "Testar um valor inteiro" );
      IO.println ( );
   // ler dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // testar valor
      if ( x != 0 )
      {
            IO.println ( "Valor diferente de zero." );
      }
      else
      {
            IO.println ( "Valor igual a zero." );
      } // fim se
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0303 ( )

/**
  * ex0304 - teste de valores inteiros
  */
   public static void ex0304 ( )
   {
   // definir dados
      int x, y;
   // identificar o metodo
      IO.println ( "EXEMPLO0304 - Programa em Java" );
      IO.println ( "Testar valores inteiros" );
      IO.println ( );
   // ler dados
      x = IO.readint ( "Fornecer um valor inteiro: " );
      y = IO.readint ( "Fornecer outro valor inteiro: " );
   // testar valor
      if ( x > y )
      {
            IO.println ( "Primeiro valor maior que o segundo." );
      }
      else
      {
            IO.println ( "Primeiro valor nao e\' maior que o segundo." );
      } // fim se
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0304 ( )
```

```
/**
  * ex0305 - teste de valores inteiros
  */
   public static void ex0305 ( )
   {
   // definir dados
      int x, y;
   // identificar o metodo
      IO.println ( "EXEMPLO0305 - Programa em Java" );
      IO.println ( "Testar valores inteiros" );   IO.println ( );
   // ler dados
      x = IO.readint ( "Fornecer um valor inteiro: " );
      y = IO.readint ( "Fornecer outro valor inteiro: " );
   // testar valor
      if ( x > y )
      {
            IO.println ( "Primeiro valor maior que o segundo." );
      }
      else
      {
        if ( x < y )
        {
          IO.println ( "Primeiro valor menor que o segundo." );
        }
        else
        {
              IO.println ( "Primeiro valor igual ao segundo." );
        } // fim se
      } // fim se
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0305 ( )

/**
  * ex0306 - teste de caractere
  */
   public static void ex0306 ( )
   {
   // definir dado
      char x = ' ';
   // identificar o metodo
      IO.println ( "EXEMPLO0306 - Programa em Java" );
      IO.println ( "Testar caractere" );         IO.println ( );
   // ler dado
      x = IO.readchar ( "Fornecer um algarismo: " );
   // testar valor
      if ( '0' <= x && x <= '9' )
      {
            IO.println ( "Foi digitado um algarismo." );
      }
      else
      {
            IO.println ( "Nao foi digitado um algarismo." );
      } // fim se
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0306 ( )
```

```
/**
  * ex0307 - teste de caractere
  */
   public static void ex0307 ( )
   {
   // definir dado
      char x = ' ';
   // identificar o metodo
      IO.println ( "EXEMPLO0307 - Programa em Java" );
      IO.println ( "Testar caractere" );
      IO.println ( );
   // ler dado
      x = IO.readchar ( "Fornecer uma letra: " );
   // testar valor
      if ( 'A' <= x && x <= 'Z' )
      {
            IO.println ( "Foi digitada uma letra maiúscula." );
      }
      else
      {
        if ( 'a' <= x && x <= 'z' )
        {
          IO.println ( "Foi digitada uma letra minuscula." );
        }
        else
        {
          IO.println ( "Nao foi digitada uma letra." );
        } // fim se
      } // fim se
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0307 ( )

/**
  * ex0308 - teste de caractere
  */
   public static void ex0308 ( )
   {
   // definir dado
      char x = ' ';
   // identificar o metodo
      IO.println ( "EXEMPLO0308 - Programa em Java" );
      IO.println ( "Testar caractere" );
      IO.println ( );
   // ler dado
      x = IO.readchar ( "Fornecer uma letra: " );
```

```
  // testar valor
     if ( 'A' <= x && x <= 'Z' )
     {
           IO.println ( "Foi digitada uma letra maiúscula." );
     }
     else
     {
       if ( 'a' <= x && x <= 'z' )
       {
         IO.println ( "Foi digitada uma letra minuscula." );
       }
       else
       {
         IO.println ( "Nao foi digitada uma letra." );
       } // fim se
     } // fim se
  // pausa
     IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
  } // fim ex0308 ( )

/**
  * ex0309 - teste de caractere
  */
   public static void ex0309 ( )
   {
  // definir dado
     final char maior = '>',
                menor = '<',
                igual = '=';
     char x = ' ';
  // identificar o metodo
     IO.println ( "EXEMPLO0309 - Programa em Java" );
     IO.println ( "Testar caractere" );
     IO.println ( );
  // ler dado
     x = IO.readchar ( "Fornecer um caractere [>,<,=]: " );
  // testar valor
     switch ( x )
     {
       case maior :
           IO.println ( "Foi digitado o sinal de maior." );
        break;
       case menor :
           IO.println ( "Foi digitado o sinal de menor." );
        break;
       case igual :
           IO.println ( "Foi digitado o sinal de igual." );
        break;
       default:
           IO.println ( "Foi digitado um outro caractere qualquer." );
     } // fim escolher
  // pausa
     IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
  } // fim ex0309 ( )
```

```
/**
  * ex0310 - teste de caractere
  */
   public static void ex0310 ( )
   {
   // definir dado
      char x = ' ';
   // identificar o metodo
      IO.println ( "EXEMPLO0310 - Programa em Java" );
      IO.println ( "Testar caractere" );            IO.println ( );
   // ler dado
      x = IO.readchar ( "Fornecer uma letra ou um algarismo: " );
   // testar valor
      switch ( x )
      {
        case 'A' : case 'a' : case 'E' : case 'e' : case  'I'  : case 'i' :
        case 'O' : case 'o' : case 'U' : case 'u' :
            IO.println ( "Foi digitada uma vogal." );
         break;
        case '0' : case '1' : case '2' : case '3' : case '4' :
        case '5' : case '6' : case '7' : case '8' : case '9' :
            IO.println ( "Foi digitado um algarismo." );
         break;
        default:
            IO.println ( "Foi digitado um outro caractere qualquer." );
      } // fim escolher
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0310 ( )

// ----------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO03 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
      ex0301 ( );
      ex0302 ( );
      ex0303 ( );
      ex0304 ( );
      ex0305 ( );
      ex0306 ( );
      ex0307 ( );
      ex0308 ( );
      ex0309 ( );
      ex0310 ( );
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ----------------------------------------------

} // end class Exemplo03
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo04 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo04
{
// ---------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                  ( String titulo, String versao, String data,
                    String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                  ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )
```

```
// ---------------------------------------------  definicao de metodos

/**
  * ex0401 - repetir 3 vezes
  */
   public static void ex0401 ( )
   {
   // definir dado
      int x, contador;
   // identificar o metodo
      IO.println ( "EXEMPLO0401 - Programa em Java" );
      IO.println ( "Repetir 3 vezes" );
      IO.println ( );
   // repetir 3 vezes
      contador = 1;
      while ( contador <= 3 )
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // proximo
         contador = contador + 1;
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0401 ( )

/**
  * ex0402 - repetir N vezes
  */
   public static void ex0402 ( )
   {
   // definir dado
      int x, n, contador;
   // identificar o metodo
      IO.println ( "EXEMPLO0402 - Programa em Java" );
      IO.println ( "Repetir N vezes" );
      IO.println ( );

   // ler dado
      n = IO.readint ( "Repetir quantas vezes ? " );
   // repetir n vezes
      contador = 1;
      while ( contador <= n )
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // proximo
         contador = contador + 1;
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0402 ( )
```

```
/**
  * ex0403 - repetir N vezes
  */
   public static void ex0403 ( )
   {
   // definir dado
      int x, n;
   // identificar o metodo
      IO.println ( "EXEMPLO0403 - Programa em Java" );
      IO.println ( "Repetir N vezes" );
      IO.println ( );
   // ler dado
      n = IO.readint ( "Repetir quantas vezes ? " );
   // repetir n vezes
      while ( n > 0 )
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // proximo
         n = n - 1;
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0403 ( )

/**
  * ex0404 - repetir N vezes
  */
   public static void ex0404 ( )
   {
   // definir dado
      int x, n, contador;
   // identificar o metodo
      IO.println ( "EXEMPLO0404 - Programa em Java" );
      IO.println ( "Repetir N vezes" );
      IO.println ( );
   // ler dado
      n = IO.readint ( "Repetir quantas vezes ? " );
   // repetir n vezes
      for ( contador = 1; contador <= n; contador = contador + 1 )
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0404 ( )
```

```
/**
  * ex0405 - repetir N vezes
  */
   public static void ex0405 ( )
   {
   // definir dado
      int x, n, contador;
   // identificar o metodo
      IO.println ( "EXEMPLO0405 - Programa em Java" );
      IO.println ( "Repetir N vezes" );
      IO.println ( );
   // ler dado
      n = IO.readint ( "Repetir quantas vezes ? " );
   // repetir n vezes
      for ( contador = n; contador > 0; contador = contador - 1 )
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0405 ( )

/**
  * ex0406 - repetir N vezes
  */
   public static void ex0406 ( )
   {
   // definir dado
      int x, n;
   // identificar o metodo
      IO.println ( "EXEMPLO0406 - Programa em Java" );
      IO.println ( "Repetir N vezes" );
      IO.println ( );
   // ler dado
      n = IO.readint ( "Repetir quantas vezes ? " );
   // repetir n vezes
      do
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // proximo
         n = n - 1;
      } // fim repetir
      while ( n > 0 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0406 ( )
```

```
/**
  * ex0407 - repetir N vezes
  */
   public static void ex0407 ( )
   {
   // definir dado
      int x, n, contador;
   // identificar o metodo
      IO.println ( "EXEMPLO0407 - Programa em Java" );
      IO.println ( "Repetir N vezes" );
      IO.println ( );
   // ler dado
      n = IO.readint ( "Repetir quantas vezes ? " );
   // repetir n vezes
      contador = 1;
      do
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // proximo
         contador = contador + 1;
      } // fim repetir
      while ( contador <= n );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0407 ( )

/**
  * ex0408 - repetir ate' encontrar zero
  */
   public static void ex0408 ( )
   {
   // definir dado
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0408 - Programa em Java" );
      IO.println ( "Repetir ate' encontrar zero" );
      IO.println ( );
   // ler dado
      x = IO.readint ( "Fornecer um valor inteiro: " );
   // repetir ate' encontrar zero
      while ( x != 0 )
      {
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // ler outro dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0408 ( )
```

```
/**
  * ex0409 - repetir ate' encontrar zero
  */
   public static void ex0409 ( )
   {
   // definir dado
      int x, n;
   // identificar o metodo
      IO.println ( "EXEMPLO0409 - Programa em Java" );
      IO.println ( "Repetir ate' encontrar zero" );
      IO.println ( );
   // repetir ate' encontrar zero
      n = 1;
      while ( n != 0 )
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // ler controle
         n = IO.readint ( "Repetir [sim=1,nao=0] ? " );
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0409 ( )

/**
  * ex0410 - repetir ate' parar
  */
   public static void ex0410 ( )
   {
   // definir dado
      int    x;
      char n;
   // identificar o metodo
      IO.println ( "EXEMPLO0410 - Programa em Java" );
      IO.println ( "Repetir ate' parar" );
      IO.println ( );
   // ler controle
      n = IO.readchar ( "Repetir [sim=S,nao=N] ? " );
      while ( n != 'N' && n != 'n' )
      {
      // ler dado
         x = IO.readint ( "Fornecer um valor inteiro: " );
      // mostrar valor
         IO.println ( "Valor lido = " + x );
      // ler controle
         n = IO.readchar ( "Repetir [sim=S,nao=N] ? " );
      } // fim repetir
   // pausa
      IO.pause   ( "APERTAR ENTER PARA CONTINUAR." );
   } // fim ex0410 ( )
```

```
// ------------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO04 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
      ex0401 ( );
      ex0402 ( );
      ex0403 ( );
      ex0404 ( );
      ex0405 ( );
      ex0406 ( );
      ex0407 ( );
      ex0408 ( );
      ex0409 ( );
      ex0410 ( );
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ------------------------------------------------

} // end class Exemplo04
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo05 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo05
{
// ---------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                  ( String titulo, String versao, String data,
                    String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                  ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )

// ---------------------------------------------  definicao de metodos

//  definicao de dado global
    static int globalX = 0;
```

```
// ---------------------------------------------  exemplo 01

/**
  * p01 - exemplo de procedimento
  */
   public static void p01 ( )
   {
   // identificar o metodo
      IO.println ( "CHAMADO O PROCEDIMENTO P01" );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P01." );
   } // fim p01 ( )

/**
  * ex0501 - comandos basicos com procedimento
  */
   public static void ex0501 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0501 - Programa em Java" );
      IO.println ( "Chamar procedimento" );   IO.println ( );
   // chamar procedimento
      p01 ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0501." );
   } // fim ex0501 ( )

// ---------------------------------------------  exemplo 02

/**
  * p02 - exemplo de procedimento com dado global
  */
   public static void p02 ( )
   {
   // identificar o metodo
      IO.println ( );
      IO.println ( "CHAMADO O PROCEDIMENTO P02 " + globalX + " VEZES." );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P02." );
   } // fim p02 ( )

/**
  * ex0502 - comandos basicos com procedimento
  */
   public static void ex0502 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0502 - Programa em Java" );
      IO.println ( "Chamar procedimento" );   IO.println ( );
   // chamar procedimento
      for ( globalX = 1; globalX <= 3; globalX = globalX + 1 )
      {
         p02 ( );
      } // fim for
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0502." );
   } // fim ex0502 ( )

// ---------------------------------------------  exemplo 03
```

```
/**
  * p03 - exemplo de procedimento
  *       com dado global
  */
   public static void p03 ( )
   {
   // identificar o metodo
      IO.println ( );
      IO.println ( "CHAMADO O PROCEDIMENTO P03 " + globalX + " VEZES." );
   // modificar o dado global
      globalX = globalX + 1;
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P03." );
   } // fim p03 ( )

/**
  * ex0503 - comandos basicos com procedimento
  */
   public static void ex0503 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0503 - Programa em Java" );
      IO.println ( "Chamar procedimento" );   IO.println ( );
   // chamar procedimento
      globalX = 1;
      while ( globalX <= 3 )
      {
         p03 ( );
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0503." );
   } // fim ex0503 ( )

// ----------------------------------------------  exemplo 04

/**
  * p04 - exemplo de procedimento
  *       com dado global
  */
   public static void p04 ( )
   {
   // executar
      while ( globalX <= 3 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO P04 " + globalX + " VEZES." );
      // modificar o dado global
         globalX = globalX + 1;
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P04." );
   } // fim p04 ( )
```

```
/**
  * ex0504 - comandos basicos
  *               com procedimento
  */
   public static void ex0504 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0504 - Programa em Java" );
      IO.println ( "Chamar procedimento" );
      IO.println ( );
   // chamar procedimento
      globalX = 1;
      p04 ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0504." );
   } // fim ex0504 ( )

// ------------------------------------------------  exemplo 05

/**
  * p05 - exemplo de procedimento
  *         com parametro
  */
   public static void p05 ( int valor )
   {
   // executar
      globalX = valor;
      while ( globalX <= 3 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO P05 " + globalX + " VEZES." );
      // modificar o dado global
         globalX = globalX + 1;
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P05." );
   } // fim p05 ( )

/**
  * ex0505 - comandos basicos
  *               com procedimento
  */
   public static void ex0505 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0505 - Programa em Java" );
      IO.println ( "Chamar procedimento" );
      IO.println ( );
   // chamar procedimento com parametro
      p05 ( 1 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0505." );
   } // fim ex0505 ( )
```

```
// ---------------------------------------------  exemplo 06

/**
  * p06 - exemplo de procedimento com parametro e dado local
  */
   public static void p06 ( int valor )
   {
   // definir dado local
      int localX = valor;
   // executar
      while ( localX <= 3 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO P06 " + localX + " VEZES." );
      // modificar o dado local
         localX = localX + 1;
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P06." );
   } // fim p06 ( )

/**
  * ex0506 - comandos basicos com procedimento
  */
   public static void ex0506 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0506 - Programa em Java" );
      IO.println ( "Chamar procedimento" );   IO.println ( );
   // chamar procedimento com parametro
      p06 ( 1 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0506." );
   } // fim ex0506 ( )

// ---------------------------------------------  exemplo 07

/**
  * p07 - exemplo de procedimento
  *         com parametro e dado local
  */
   public static void p07 ( int vezes )
   {
   // definir dado local
      int localX = 1;
   // executar
      while ( localX <= vezes )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO P07 " + localX + " VEZES." );
      // modificar o dado local
         localX = localX + 1;
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P07." );
   } // fim p07 ( )
```

```
/**
  * ex0507 - comandos basicos
  *               com procedimento
  */
   public static void ex0507 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0507 - Programa em Java" );
      IO.println ( "Chamar procedimento" );
      IO.println ( );
   // chamar procedimento com parametro
      p07 ( 3 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0507." );
   } // fim ex0507 ( )

// ----------------------------------------------  exemplo 08

/**
  * p08 - exemplo de procedimento
  *          com parametro e dado local
  */
   public static void p08 ( int vezes )
   {
   // definir dado local
      int localX = vezes;
   // executar
      while ( localX > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO P08 " + (vezes-localX+1) + " VEZES." );
      // modificar o dado local
         localX = localX - 1;
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P08." );
   } // fim p08 ( )

/**
  * ex0508 - comandos basicos
  *               com procedimento
  */
   public static void ex0508 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0508 - Programa em Java" );
      IO.println ( "Chamar procedimento" );
      IO.println ( );
   // chamar procedimento com parametro
      p08 ( 3 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0508." );
   } // fim ex0508 ( )
```

```
// ---------------------------------------------  exemplo 09

/**
  * p09 - exemplo de procedimento com chamada recursiva
  */

   public static void p09 ( int vezes )
   {
   // definir dado local
      int localX = vezes;
   // executar
      if ( localX > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO P09 COM PARAMETRO = " + vezes + "." );
      // modificar o dado local
         localX = localX - 1;
      // chamada recursiva
         p09 ( localX );
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P09 COM PARAMETRO = " + vezes + "." );
   } // fim p09 ( )

/**
  * ex0509 - comandos basicos com procedimento
  */
   public static void ex0509 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0509 - Programa em Java" );
      IO.println ( "Chamar procedimento" );   IO.println ( );
   // chamar procedimento com parametro
      p09 ( 3 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0509." );
   } // fim ex0509 ( )

// ---------------------------------------------  exemplo 10

/**
  * p10 - exemplo de procedimento com chamada recursiva
  */
   public static void p10 ( int vezes )
   {
   // executar
      if ( vezes > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO P09 COM PARAMETRO = " + vezes + "." );
      // chamada recursiva
         p09 ( vezes - 1 );
      } // fim while
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR P09 COM PARAMETRO = " + vezes + "." );
   } // fim p10 ( )
```

```
/**
  * ex0510 - comandos basicos
  *               com procedimento
  */
   public static void ex0510 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0510 - Programa em Java" );
      IO.println ( "Chamar procedimento" );
      IO.println ( );
   // chamar procedimento com parametro
      p10 ( 3 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0510." );
   } // fim ex0510 ( )

// ------------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO05 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
/*
      ex0501 ( );
      ex0502 ( );
      ex0503 ( );
      ex0504 ( );
      ex0505 ( );
      ex0506 ( );
      ex0507 ( );
      ex0508 ( );
      ex0509 ( );
      ex0510 ( );
*/
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ------------------------------------------------

} // end class Exemplo05
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo06 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo06
{
// ---------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                  ( String titulo, String versao, String data,
                    String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                  ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )
```

```
// ---------------------------------------------  exemplo 01

/**
  * contar01 - procedimento para contar N vezes
  */
   public static void contar01 ( int n )
   {
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO O PROCEDIMENTO COM PARAMETRO = " + n + "." );
      // chamada recursiva
         contar01 ( n - 1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR COM PARAMETRO = " + n + "." );
   } // fim contar01 ( )

/**
  * ex0601 - comandos basicos
  *               com procedimento
  */
   public static void ex0601 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0601 - Programa em Java" );
      IO.println ( "Chamar procedimento" );
      IO.println ( );
   // chamar procedimento
      contar01 ( 5 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0601." );
   } // fim ex0601 ( )

// ---------------------------------------------  exemplo 02

/**
  * contar02 - funcao para contar N vezes
  */
   public static int contar02 ( int n )
   {
   // definir dado local
      int x = 0;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO = " + n + "." );
      // chamada recursiva
         x = contar02 ( n - 1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + x + "." );
      return ( x );
   } // fim contar02 ( )
```

```
/**
  * ex0602 - comandos basicos
  *               com funcao
  */
   public static void ex0602 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0602 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + contar02 ( 5 ) );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0602." );
   } // fim ex0602 ( )

// ----------------------------------------------  exemplo 03

/**
  * somar03 - funcao para somar os N primeiros naturais
  */
   public static int somar03 ( int n )
   {
   // definir dado local
      int x = 0;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO = " + n + "." );
      // chamada recursiva
         x = n + somar03 ( n - 1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + x + "." );
      return ( x );
   } // fim somar03 ( )

/**
  * ex0603 - comandos basicos
  *               com funcao
  */
   public static void ex0603 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0603 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + somar03 ( 5 ) );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0603." );
   } // fim ex0603 ( )
```

```
// ---------------------------------------------  exemplo 04

/**
  * mostrar04 - funcao para mostrar os N primeiros pares
  */
   public static int mostrar04 ( int n )
   {
   // definir dado local
      int x = 0;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO = " + n + "." );
      // chamada recursiva
         IO.println ( "PAR = " + ( 2 * n ) );
         x = mostrar04 ( n - 1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + n + "." );
      return ( x );
   } // fim mostrar04 ( )

/**
  * ex0604 - comandos basicos com funcao
  */
   public static void ex0604 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0604 - Programa em Java" );            IO.println ( );
      IO.println ( "Retorno final da funcao = " + mostrar04 ( 5 ) );   IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0604." );
   } // fim ex0604 ( )

// ---------------------------------------------  exemplo 05

/**
  * somar05 - funcao para somar os N primeiros pares
  */
   public static int somar05 ( int n )
   {
   // definir dado local
      int x = 0;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO = " + n + "." );
      // chamada recursiva
         x = 2 * n + somar05 ( n - 1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + x + "." );
      return ( x );
   } // fim somar05 ( )
```

```
/**
  * ex0605 - comandos basicos
  *               com funcao
  */
   public static void ex0605 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0605 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + somar05 ( 5 ) );
      IO.println ( );
   // pausa
      IO.pause  ( "APERTAR ENTER PARA TERMINAR EX0605." );
   } // fim ex0605 ( )

// ----------------------------------------------  exemplo 06

/**
  * mostrar06 - funcao para mostrar os N primeiros pares
  */
   public static int mostrar06 ( int n )
   {
   // definir dado local
      int x = 0;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO = " + n + "." );
      // chamada recursiva
         IO.println ( "IMPAR = " + ( 2 * n - 1 ) );
         x = mostrar06 ( n - 1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + n + "." );
      return ( x );
   } // fim mostrar06 ( )

/**
  * ex0606 - comandos basicos
  *               com funcao
  */
   public static void ex0606 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0606 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + mostrar06 ( 5 ) );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0606." );
   } // fim ex0606 ( )
```

```
// ---------------------------------------------  exemplo 07

/**
  * somar07 - funcao para somar os N primeiros impares
  */
   public static int somar07 ( int n )
   {
   // definir dado local
      int x = 0;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO = " + n + "." );
      // chamada recursiva
         x = ( 2 * n - 1 ) + somar07 ( n - 1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + x + "." );
      return ( x );
   } // fim somar07 ( )

/**
  * ex0607 - comandos basicos com funcao
  */
   public static void ex0607 ( )
   {
   // identificar o metodo
      IO.println ( "EXEMPLO0607 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + somar07 ( 5 ) );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0607." );
   } // fim ex0607 ( )

// ---------------------------------------------  exemplo 08

/**
  * mostrar08 - funcao para mostrar as N primeiras letras de uma palavra
  */
   public static int mostrar08 ( String palavra, int n )
   {
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO n = " + n + "." );
         IO.println ( "LETRA = " + palavra.charAt ( n-1 ) );
      // chamada recursiva
         n = mostrar08 ( palavra, n-1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + n + "." );
      return ( n );
   } // fim mostrar08 ( )
```

```
/**
  * ex0608 - comandos basicos com funcao
  */
   public static void ex0608 ( )
   {
   // definir dado
      String p = new String ( "abcde" );
   // identificar o metodo
      IO.println ( "EXEMPLO0608 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + mostrar08 ( p, 5 ) );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0608." );
   } // fim ex0608 ( )

// ----------------------------------------------  exemplo 09

/**
  * somar09 - funcao para mostrar e somar as N primeiras letras de uma palavra
  */
   public static int somar09 ( String palavra, int n )
   {
   // definir dado local
      int x = 0;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO n = " + n + "." );
         x = (int) palavra.charAt ( n - 1 );
         IO.println ( "LETRA = " + ( char ) x );
      // chamada recursiva
         x = x + somar09 ( palavra, n-1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + x + "." );
      return ( x );
   } // fim somar09 ( )

/**
  * ex0609 - comandos basicos com funcao
  */
   public static void ex0609 ( )
   {
   // definir dado
      String p = new String ( "abcde" );
   // identificar o metodo
      IO.println ( "EXEMPLO0609 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + somar09 ( p, 5 ) );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0609." );
   } // fim ex0609 ( )


// ----------------------------------------------  exemplo 10
```

```
/**
  * procurar10 - funcao para procurar uma letra entre
  *                  as N primeiras letras de uma palavra
  */
   public static boolean procurar10 ( char letra, String palavra, int n )
   {
   // definir dado local
      boolean resposta = false;
   // executar
      if ( n > 0 )
      {
      // identificar o metodo
         IO.println ( );
         IO.println ( "CHAMADO A FUNCAO COM PARAMETRO n = " + n + "." );
         IO.println ( "LETRA A SER TESTADA = " + palavra.charAt ( n-1 ) );
      // chamada recursiva
         if ( letra == palavra.charAt ( n-1 ) )
           resposta = true;
         else
           resposta = procurar10 ( letra, palavra, n-1 );
      } // fim if
   // pausa
      IO.pause   ( "APERTAR ENTER PARA RETORNAR COM PARAMETRO = " + n + "." );
      return ( resposta );
   } // fim procurar10 ( )

/**
  * ex0610 - comandos basicos
  *              com funcao
  */
   public static void ex0610 ( )
   {

   // definir dado
      String p = new String ( "abcde" );
      char   c = 'c';
   // identificar o metodo
      IO.println ( "EXEMPLO0610 - Programa em Java" );
      IO.println ( );
      IO.println ( "Retorno final da funcao = " + procurar10 ( c, p, p.length( ) ) );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0610." );
   } // fim ex0610 ( )
```

```
// ------------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO06 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
/*
      ex0601 ( );
      ex0602 ( );
      ex0603 ( );
      ex0604 ( );
      ex0605 ( );
      ex0606 ( );
      ex0607 ( );
      ex0608 ( );
      ex0609 ( );
      ex0610 ( );
*/
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ------------------------------------------------

} // end class Exemplo06
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo07 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao da classe principal

public class Exemplo07
{
// ---------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                  ( String titulo, String versao, String data,
                    String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                  ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )
```

```
// ---------------------------------------------  exemplo 01

/**
  * ex0701 - comandos basicos
  *               para ler e mostrar tabela
  */
   public static void ex0701 ( )
   {
   // definir dados
      int [ ] tabela = new int [ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0701 - Programa em Java" );
      IO.println ( "Ler e mostrar tabela" );
      IO.println ( );
   // ler dados
      IO.println ( "Fornecer valor para a posicao" );
      IO.println ( );
      for ( x = 0; x < 5; x = x + 1 )
      {
            tabela [ x ] = IO.readint ( "" + x + " : " );
      } // fim for
   // mostrar dados
      IO.println ( );
      IO.println ( "Valores lidos: " );
      for ( x = 0; x < 5; x = x + 1 )
      {
            IO.print ( " " + tabela [ x ] );
      } // fim for
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0701." );
   } // fim ex0701 ( )

// ---------------------------------------------  exemplo 02

/**
  * ler - procedimento para ler tabela
  * <br>

  * @param v - tabela
  * @param n - numero de elementos
  */
   public static void ler ( int [] v, int n )
   {
   // definir dado local
      int x = 0;
   // ler dados
      IO.println ( "Fornecer valor para a posicao" );
      IO.println ( );
      for ( x = 0; x < n; x = x + 1 )
      {
         v [ x ] = IO.readint ( "" + x + " : " );
      } // fim for
   } // fim ler ( )
```

```
/**
  * mostrar - procedimento para mostrar tabela
  * @param v - tabela
  * @param n - numero de elementos
  */
   public static void mostrar ( int [] v, int n )
   {
   // definir dado local
      int x = 0;
   // mostrar dados
      IO.println ( );
      IO.println ( "Valores lidos: " );
      for ( x = 0; x < n; x = x + 1 )
      {
            IO.print ( " " + v [ x ] );
      } // fim for
      IO.println ( );
   } // fim mostrar ( )

/**
  * ex0702 - comandos basicos
  *              para ler e mostrar tabela
  *              com procedimentos
  */
   public static void ex0702 ( )
   {
   // definir dados
      int [ ] tabela = new int [ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0702 - Programa em Java" );
      IO.println ( "Ler e mostrar tabela" );
      IO.println ( );
   // ler dados
      ler ( tabela, 5 );
   // mostrar dados
      mostrar ( tabela, 5 );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0702." );
   } // fim ex0702 ( )
```

```
// ---------------------------------------------  exemplo 03

/**
  * somar - funcao para somar valores
  *             em uma tabela
  * <br>
  * @return  - soma dos valores na tabela
  * @param v - tabela
  * @param n - numero de elementos
  */
   public static int somar ( int [] v, int n )
   {
   // definir dados locais
      int x    = 0,
          soma = 0;
   // somar valores
      for ( x = 0; x < n; x = x + 1 )
      {
            soma = soma + v [ x ];
      } // fim for
      return ( soma );
   } // fim somar ( )

/**
  * ex0703 - comandos basicos
  *               para calcular a soma dos valores
  *               em uma tabela com metodos
  */
   public static void ex0703 ( )
   {
   // definir dados
      int [ ] tabela = new int [ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0703 - Programa em Java" );
      IO.println ( "Ler e somar valores de uma tabela" );
      IO.println ( );
   // ler dados
      ler ( tabela, 5 );
   // mostrar dados
      IO.println ( "Soma dos valores = " + somar ( tabela, 5 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0703." );
   } // fim ex0703 ( )
```

```
// ---------------------------------------------  exemplo 04

/**
  * media - funcao para calcular a media
  *             dos valores em uma tabela
  * <br>
  * @return  - media dos valores na tabela
  * @param v - tabela
  * @param n - numero de elementos
  */
   public static double media ( int [] v, int n )
   {
   // definir dado local
      double m = 0.0;
   // calcular a media
      if ( n > 0 )
      {
            m = somar ( v, n ) / n;
      } // fim if
      return ( m );
   } // fim media ( )

/**
  * ex0704 - comandos basicos
  *               para calcular a media dos valores
  *               em uma tabela com metodos
  */
   public static void ex0704 ( )
   {
   // definir dados
      int [ ] tabela = new int [ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0704 - Programa em Java" );
      IO.println ( "Ler e achar a media de uma tabela" );
      IO.println ( );
   // ler dados
      ler ( tabela, 5 );
   // mostrar dados
      IO.println ( "Media dos valores = " + media ( tabela, 5 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0704." );
   } // fim ex0704 ( )
```

```
// ---------------------------------------------  exemplo 05

/**
  * maior - funcao para achar o maior
  *             dos valores em uma tabela
  * <br>
  * @return  - maior valor na tabela
  * @param v - tabela
  * @param n - numero de elementos
  */
   public static int maior ( int [] v, int n )
   {
   // definir dados locais
      int x = 0,
          m = v [ 0 ];
   // achar o maior valor
      for ( x = 1; x < n; x = x + 1 )
      {
        if ( v [ x ] > m )
        {
          m = v [ x ];
        } // fim if
      } // fim for
      return ( m );
   } // fim maior ( )

/**
  * ex0705 - comandos basicos
  *                para calcular o maior dentre os valores
  *                em uma tabela com metodos
  */
   public static void ex0705 ( )
   {
   // definir dados
      int [ ] tabela = new int [ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0705 - Programa em Java" );
      IO.println ( "Achar o maior valor em uma tabela" );
      IO.println ( );
   // ler dados
      ler ( tabela, 5 );
   // mostrar dados
      IO.println ( "Maior valor = " + maior ( tabela, 5 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0705." );
   } // fim ex0705 ( )
```

```
// ---------------------------------------------  exemplo 06

/**
  * avaliar - funcao para avaliar o polinomio
  *              com os coeficientes em uma tabela
  * <br>
  * @return     - valor do polinomio
  * @param v  - tabela com coeficientes
  * @param n  - numero de elementos
  * @param px - ponto de avaliacao
  */
   public static double avaliar ( int [] v, int n, double px )
   {
   // definir dados locais
      int       x = 0;
      double p = 0.0;
   // avaliar o polilnomio
      for ( x = 0; x < n; x = x + 1 )
      {
        p = p * px + v [ x ];
      } // fim for
      return ( p );
   } // fim avaliar ( )

/**
  * ex0706 - comandos basicos
  *               para avaliar o polinomio
  *               com os coeficientes
  *               em uma tabela com metodos
  */
   public static void ex0706 ( )
   {
   // definir dados
      int [ ] tabela = new int [ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0706 - Programa em Java" );
      IO.println ( "Avaliar o polinomio" );
      IO.println ( );
   // ler dados
      ler ( tabela, 3 );
   // mostrar dados
      IO.println ( "Polinomio no ponto ( 2.0 ) = " + avaliar ( tabela, 3, 2.0 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0706." );
   } // fim ex0706 ( )
```

```
// ---------------------------------------------  exemplo 07

/**
  * comprimento - funcao para avaliar o comprimento
  *                       de um vetor
  * <br>
  * @return     - comprimento do vetor
  * @param v  - tabela com elementos
  * @param n  - numero de elementos
  */
   public static double comprimento ( int [] v, int n )
   {
   // definir dados locais
      int       x = 0;
      double p = 0.0;
   // avaliar o polilnomio
      for ( x = 0; x < n; x = x + 1 )
      {
        p = p + v [ x ] * v [ x ];
      } // fim for
      return ( Math.sqrt ( p ) );
   } // fim comprimento ( )

/**
  * ex0707 - comandos basicos
  *               para calcular o comprimento
  *               de um vetor com metodos
  */
   public static void ex0707 ( )
   {
   // definir dados
      int [ ] tabela = new int [ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0707 - Programa em Java" );
      IO.println ( "Calcular o comprimento de um vetor" );
      IO.println ( );
   // ler dados
      ler ( tabela, 3 );
   // mostrar dados
      IO.println ( "Comprimento do vetor = " + comprimento ( tabela, 3 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0707." );
   } // fim ex0707 ( )
```

```
// ---------------------------------------------  exemplo 08

/**

  * lerMatriz - procedimento para ler matriz
  * <br>
  * @param a - matriz
  * @param m - numero de linhas
  * @param n  - numero de colunas
  */
   public static void lerMatriz ( int [][] a, int m, int n )
   {
   // definir dado local
      int x = 0,
          y = 0;
   // ler dados
      IO.println ( "Fornecer valor para a posicao" );
      IO.println ( );
      for ( x = 0; x < m; x = x + 1 )
      {
        for ( y = 0; y < n; y = y + 1 )
        {
          a [ x ][ y ] = IO.readint ( "[" + x + "," + y + "] : " );
        } // fim for ( y )
      } // fim for ( x )
   } // fim lerMatriz ( )

/**
  * mostrarMatriz - procedimento para mostrar matriz
  * @param a - matriz
  * @param m - numero de linhas
  * @param n - numero de colunas
  */
   public static void mostrarMatriz ( int [][] a, int m, int n )
   {
   // definir dado local
      int x = 0,
          y = 0;
   // mostrar dados
      for ( x = 0; x < m; x = x + 1 )
      {
        IO.println ( );
        for ( y = 0; y < n; y = y + 1 )
        {
          IO.print ( " " + a [ x ][ y ] );
        } // fim for ( y )
      } // fim for ( x )
   } // fim mostrarMatriz ( )
```

```
/**
  * ex0708 - comandos basicos
  *               para ler e mostrar matriz
  *               com procedimentos
  */
   public static void ex0708 ( )
   {
   // definir dados
      int [ ][ ] a = new int [ 10 ][ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0708 - Programa em Java" );
      IO.println ( "Ler e mostrar matriz" );
      IO.println ( );
   // ler dados
      lerMatriz ( a, 2, 2 );
   // mostrar dados
      IO.println ( );
      IO.println ( "Matriz lida:" );
      mostrarMatriz ( a, 2, 2 );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0708." );
   } // fim ex0708 ( )

// ----------------------------------------------  exemplo 09

/**
  * transpor - procedimento para transpor matriz
  * @param b - matriz transposta
  * @param a - matriz
  * @param m - numero de linhas
  * @param n  - numero de colunas
  */
   public static void transpor ( int [][] b, int [][] a, int m, int n )
   {
   // definir dado local
      int x = 0,
          y = 0;
   // transpor matriz
      for ( x = 0; x < n; x = x + 1 )
      {
        for ( y = 0; y < n; y = y + 1 )
        {
            b [ y ][ x ] = a [ x ][ y ];
        } // fim for ( y )
      } // fim for ( x )
      IO.println ( );
   } // fim transpor ( )
```

```
/**
  * ex0709 - comandos basicos
  *               para transpor matriz
  *               com procedimentos
  */
   public static void ex0709 ( )
   {
   // definir dados
      int [ ][ ] a = new int [ 10 ][ 10 ];
      int [ ][ ] b = new int [ 10 ][ 10 ];
      int x;
   // identificar o metodo
      IO.println ( "EXEMPLO0709 - Programa em Java" );
      IO.println ( "Transpor matriz" );
      IO.println ( );
   // ler dados
      lerMatriz ( a, 2, 3 );
   // mostrar dados
      IO.println ( "Matriz A:" );
      mostrarMatriz ( a, 2, 3 );
      IO.println ( );
   // transpor matriz
      transpor ( b, a, 2, 3 );
   // mostrar dados
      IO.println ( "Matriz B:" );
      mostrarMatriz ( b, 3, 2 );
      IO.println ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0709." );
   } // fim ex0709 ( )

// ----------------------------------------------  exemplo 10

/**
  * somarDiagonal - procedimento para somar
  *                         valores na diagonal da matriz
  * @param a - matriz
  * @param n - tamanho da matriz
  */
   public static int somarDiagonal ( int [][] a, int n )
   {
   // definir dado local
      int x    = 0,
          y    = 0;
      int soma = 0;
   // somar valores da diagonal
      for ( x = 0; x < n; x = x + 1 )
      {
         soma = soma + a [ x ][ y ];
      } // fim for ( x )
      return ( soma );
   } // fim somarDiagonal ( )
```

```
/**
  * ex0710 - comandos basicos
  *              para somar valores na diagonal da matriz
  *              com procedimentos
  */
   public static void ex0710 ( )
   {
   // definir dados
      int [ ][ ] a = new int [ 10 ][ 10 ];
   // identificar o metodo
      IO.println ( "EXEMPLO0710 - Programa em Java" );
      IO.println ( "Somar valores na diagonal da matriz" );
      IO.println ( );
   // ler dados
      lerMatriz ( a, 2, 2 );
   // mostrar dados
      IO.println ( );
      IO.println ( "Matriz A:" );
      mostrarMatriz ( a, 2, 2 );
      IO.println ( );
   // somar valores na diagonal da matriz
      IO.println ( );
      IO.println ( "Soma da diagonal = " + somarDiagonal ( a, 2 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0710." );
   } // fim ex0710 ( )

// ------------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO07 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
/*
      ex0701 ( );
      ex0702 ( );
      ex0703 ( );
      ex0704 ( );
      ex0705 ( );
      ex0706 ( );
      ex0707 ( );
      ex0708 ( );
      ex0709 ( );
      ex0710 ( );
*/
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ------------------------------------------------

} // end class Exemplo07
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo08 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao de classes

class Ponto01
{
// armazenadores
   public double X, Y, Z;
} // fim class Ponto01

class Ponto02
{
// armazenadores
   public double X, Y, Z;

// metodos
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     X = IO.readdouble ( "X = " );
     Y = IO.readdouble ( "Y = " );
     Z = IO.readdouble ( "Z = " );
   } // fim Ponto02.ler ( )
} // fim class Ponto02
```

```
class Ponto03
{
// armazenadores
   public double X, Y, Z;

// metodos
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     X = IO.readdouble ( "X = " );
     Y = IO.readdouble ( "Y = " );
     Z = IO.readdouble ( "Z = " );
   } // fim Ponto03.ler ( )

   public double distanciaAte ( Ponto03 B )
   {
   // definir dado local
      double distancia = 0.0;
   // calcular distancia entre pontos
      distancia = Math.sqrt ( Math.pow ( this.X - B.X, 2 ) +
                              Math.pow ( this.Y - B.Y, 2 ) +
                              Math.pow ( this.Z - B.Z, 2 ) );
   // retornar o resultado
      return ( distancia );
   } // fim Ponto03.distanciaAte ( )
} // fim class Ponto03


class Ponto04
{
// armazenadores
   public double X, Y, Z;

// metodos
   public Ponto04 subtrair ( Ponto04 B )
   {
   // definir dado local
      Ponto04 diferenca = new Ponto04 ( );
   // calcular a diferenca entre pontos
      diferenca.X = X - B.X;
      diferenca.Y = Y - B.Y;
      diferenca.Z = Z - B.Z;
   // retornar o resultado
      return ( diferenca );
   } // fim Ponto04.diferenca ( )

   public double comprimento ( )
   {
   // definir dado local
      double distancia = 0.0;
   // calcular distancia entre pontos
      distancia = Math.sqrt ( X * X + Y * Y + Z * Z );
   // retornar o resultado
      return ( distancia );
   } // fim Ponto04.comprimento ( )
```

```
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     X = IO.readdouble ( "X = " );
     Y = IO.readdouble ( "Y = " );
     Z = IO.readdouble ( "Z = " );
   } // fim Ponto04.ler ( )

   public double distanciaAte ( Ponto04 B )
   {
   // definir dado local
      double  distancia = 0.0;
      Ponto04 temp;
   // calcular distancia entre pontos
      temp      = this.subtrair ( B );
      distancia = temp.comprimento ( );
   // retornar o resultado
      return ( distancia );
   } // fim Ponto04.distanciaAte ( )
} // fim class Ponto04

class Ponto05
{
// armazenadores privados
   private double X, Y, Z;

// metodos privados
   private Ponto05 subtrair ( Ponto05 B )
   {
   // definir dado local
      Ponto05 diferenca = new Ponto05 ( );
   // calcular a diferenca entre pontos
      diferenca.X = X - B.X;
      diferenca.Y = Y - B.Y;
      diferenca.Z = Z - B.Z;
   // retornar o resultado
      return ( diferenca );
   } // fim Ponto05.diferenca ( )

   private double comprimento ( )
   {
   // definir dado local
      double distancia = 0.0;
   // calcular distancia entre pontos
      distancia = Math.sqrt ( X * X + Y * Y + Z * Z );
   // retornar o resultado
      return ( distancia );
   } // fim Ponto05.comprimento ( )

// metodos privados publicos
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     X = IO.readdouble ( "X = " );
     Y = IO.readdouble ( "Y = " );
     Z = IO.readdouble ( "Z = " );
   } // fim Ponto04.ler ( )
```

```
   public double distanciaAte ( Ponto05 B )
   {
   // definir dado local
      double  distancia = 0.0;
      Ponto05 temp;
   // calcular distancia entre pontos
      temp      = this.subtrair ( B );
      distancia = temp.comprimento ( );
   // retornar o resultado
      return ( distancia );
   } // fim Ponto05.distanciaAte ( )
} // fim class Ponto05

class Ponto06
{
// armazenadores
   private double [ ] vetor = new double [ 3 ];
   private final static int X = 0, Y = 1 , Z = 2;

// metodos
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     vetor [ X ] = IO.readdouble ( "X = " );
     vetor [ Y ] = IO.readdouble ( "Y = " );
     vetor [ Z ] = IO.readdouble ( "Z = " );
   } // fim Ponto06.ler ( )

   public double distanciaAte ( Ponto06 B )
   {
   // definir dado local
      double distancia = 0.0;
   // calcular distancia entre pontos
      distancia = Math.sqrt ( Math.pow ( vetor [ X ] - B.vetor [ X ], 2 ) +
                  Math.pow ( vetor [ Y ] - B.vetor [ Y ], 2 ) +
                  Math.pow ( vetor [ Z ] - B.vetor [ Z ], 2 ) );
   // retornar o resultado
      return ( distancia );
   } // fim Ponto06.distanciaAte ( )
} // fim class Ponto06

class Ponto07
{
// armazenadores
   private double [ ] vetor = new double [ 3 ];
   private final static int X = 0, Y = 1 , Z = 2;

// metodos
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     vetor [ X ] = IO.readdouble ( "X = " );
     vetor [ Y ] = IO.readdouble ( "Y = " );
     vetor [ Z ] = IO.readdouble ( "Z = " );
   } // fim Ponto07.ler ( )
```

```
   public double distanciaAte ( Ponto07 B )
   {
   // definir dado local
      double distancia = 0.0;
      int    n;
   // calcular distancia entre pontos
      for ( n = X; n <= Z; n = n + 1 )
        distancia = distancia + Math.pow ( vetor [ n ] - B.vetor [ n ], 2 );
   // retornar o resultado
      return ( Math.sqrt ( distancia ) );
   } // fim Ponto07.distanciaAte ( )
} // fim class Ponto07

class Pontos08
{
// armazenadores
   private double [ ] vetor1 = new double [ 3 ];
   private double [ ] vetor2 = new double [ 3 ];
   private final static int X = 0, Y = 1 , Z = 2;

// metodos
   private void ler ( int ponto )
   {
    if ( ponto == 1 )
    {
      vetor1 [ X ] = IO.readdouble ( "X = " );
      vetor1 [ Y ] = IO.readdouble ( "Y = " );
      vetor1 [ Z ] = IO.readdouble ( "Z = " );
    }
    else
    {
      vetor2 [ X ] = IO.readdouble ( "X = " );
      vetor2 [ Y ] = IO.readdouble ( "Y = " );
      vetor2 [ Z ] = IO.readdouble ( "Z = " );
    }
   } // fim Pontos08.ler ( )

   public void ler ( String mensagem )
   {
    IO.println ( mensagem );
    IO.println ( "Primeiro ponto:" );
    ler ( 1 );

    IO.println ( "Segundo  ponto:" );
    ler ( 2 );
   } // fim Pontos08.ler ( )

   public double distancia ( )
   {
   // definir dado local
      double d = 0.0;
      int    n;
   // calcular distancia entre pontos
      for ( n = X; n <= Z; n = n + 1 )
        d = d + Math.pow ( vetor1 [ n ] - vetor2 [ n ], 2 );
   // retornar o resultado
      return ( Math.sqrt ( d ) );
   } // fim Pontos08.distancia ( )
} // fim class Pontos08
```

```
class Pontos09
{
// armazenadores
   private double [ ][ ] vetor = new double [ 2 ][ 3 ];
   private final static int X = 0, Y = 1 , Z = 2;

// metodos
   private void ler ( int ponto )
   {
     vetor [ ponto ][ X ] = IO.readdouble ( "X = " );
     vetor [ ponto ][ Y ] = IO.readdouble ( "Y = " );
     vetor [ ponto ][ Z ] = IO.readdouble ( "Z = " );
   } // fim Pontos09.ler ( )

   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     IO.println ( "Primeiro ponto:" );
     ler ( 0 );
     IO.println ( "Segundo  ponto:" );
     ler ( 1 );
   } // fim Pontos09.ler ( )

   public double distancia ( )
   {
   // definir dado local
      double d = 0.0;
      int    n;
   // calcular distancia entre pontos
      for ( n = X; n <= Z; n = n + 1 )
        d = d + Math.pow ( vetor [ 0 ][ n ] - vetor [ 1 ][ n ], 2 );
   // retornar o resultado
      return ( Math.sqrt ( d ) );
   } // fim Pontos08.distancia ( )
} // fim class Pontos09

class Ponto10
{
// armazenadores
   protected double X, Y, Z;

// metodos
   protected Ponto10 ( String mensagem )
   {
   // construtor
      IO.println ( mensagem );
      X = IO.readdouble ( "X = " );
      Y = IO.readdouble ( "Y = " );
      Z = IO.readdouble ( "Z = " );
   } // fim Ponto10.ler ( )
} // fim class Ponto10
```

```
class Pontos10
{
// armazenadores
   private Ponto10 P1 = new Ponto10 ( "Fornecer coordenadas do primeiro ponto:" ),
                   P2 = new Ponto10 ( "Fornecer coordenadas do segundo  ponto:" );

// metodos
   public double distancia ( )
   {
   // definir dado local
      double d = 0.0;
   // calcular distancia entre pontos
      d = Math.sqrt ( Math.pow ( P1.X - P2.X, 2 ) +
            Math.pow ( P1.Y - P2.Y, 2 ) +
            Math.pow ( P1.Z - P2.Z, 2 ) );
   // retornar o resultado
      return ( d );
   } // fim Pontos10.distancia ( )
} // fim class Pontos10

// ----------------------------------------------  definicao da classe principal

public class Exemplo08
{
// ----------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                  ( String titulo, String versao, String data,
                    String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                  ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )
```

```
// ---------------------------------------------  exemplo 01

/**
  * ex0801 - calcular a distancia entre dois pontos
  */
   public static void ex0801 ( )
   {
   // definir dados
      Ponto01 P1 = new Ponto01 ( ),
               P2 = new Ponto01 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0801 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      IO.println ( "Fornecer coordenadas do primeiro ponto:" );
      P1.X = IO.readdouble ( "X = " );
      P1.Y = IO.readdouble ( "Y = " );
      P1.Z = IO.readdouble ( "Z = " );
      IO.println ( "Fornecer coordenadas do segundo  ponto:" );
      P2.X = IO.readdouble ( "X = " );
      P2.Y = IO.readdouble ( "Y = " );
      P2.Z = IO.readdouble ( "Z = " );
   // calcular distancia
      distancia = Math.sqrt ( Math.pow ( P1.X - P2.X, 2 ) +
                      Math.pow ( P1.Y - P2.Y, 2 ) +
                      Math.pow ( P1.Z - P2.Z, 2 ) );
   // mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + distancia );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0801." );
   } // fim ex0801 ( )
```

```
// ---------------------------------------------  exemplo 02

/**
  * ex0802 - calcular a distancia entre dois pontos
  */
   public static void ex0802 ( )
   {
   // definir dados
      Ponto02 P1 = new Ponto02 ( ),
               P2 = new Ponto02 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0802 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P1.ler ( "Fornecer coordenadas do primeiro ponto:" );
      P2.ler ( "Fornecer coordenadas do segundo  ponto:" );
   // calcular distancia
      distancia = Math.sqrt ( Math.pow ( P1.X - P2.X, 2 ) +
                  Math.pow ( P1.Y - P2.Y, 2 ) +
                  Math.pow ( P1.Z - P2.Z, 2 ) );
   // mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + distancia );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0802." );
   } // fim ex0802 ( )

// ---------------------------------------------  exemplo 03

/**
  * ex0803 - calcular a distancia entre dois pontos
  */
   public static void ex0803 ( )
   {
   // definir dados
      Ponto03 P1 = new Ponto03 ( ),
               P2 = new Ponto03 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0803 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P1.ler ( "Fornecer coordenadas do primeiro ponto:" );
      P2.ler ( "Fornecer coordenadas do segundo  ponto:" );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P1.distanciaAte ( P2 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0803." );
   } // fim ex0803 ( )
```

```
// ---------------------------------------------  exemplo 04

/**
  * ex0804 - calcular a distancia entre dois pontos
  */
   public static void ex0804 ( )
   {
   // definir dados
      Ponto04 P1 = new Ponto04 ( ),
               P2 = new Ponto04 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0804 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P1.ler ( "Fornecer coordenadas do primeiro ponto:" );
      P2.ler ( "Fornecer coordenadas do segundo  ponto:" );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P1.distanciaAte ( P2 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0804." );
   } // fim ex0804 ( )

// ---------------------------------------------  exemplo 05

/**
  * ex0805 - calcular a distancia entre dois pontos
  */
   public static void ex0805 ( )
   {
   // definir dados
      Ponto05 P1 = new Ponto05 ( ),
               P2 = new Ponto05 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0805 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P1.ler ( "Fornecer coordenadas do primeiro ponto:" );
      P2.ler ( "Fornecer coordenadas do segundo  ponto:" );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P1.distanciaAte ( P2 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0805." );
   } // fim ex0805 ( )
```

```
// ---------------------------------------------  exemplo 06

/**
  * ex0806 - calcular a distancia entre dois pontos
  */
   public static void ex0806 ( )
   {
   // definir dados
      Ponto06 P1 = new Ponto06 ( ),
               P2 = new Ponto06 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0806 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P1.ler ( "Fornecer coordenadas do primeiro ponto:" );
      P2.ler ( "Fornecer coordenadas do segundo  ponto:" );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P1.distanciaAte ( P2 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0806." );
   } // fim ex0806 ( )

// ---------------------------------------------  exemplo 07

/**
  * ex0807 - calcular a distancia entre dois pontos
  */
   public static void ex0807 ( )
   {
   // definir dados
      Ponto07 P1 = new Ponto07 ( ),
               P2 = new Ponto07 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0807 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P1.ler ( "Fornecer coordenadas do primeiro ponto:" );
      P2.ler ( "Fornecer coordenadas do segundo  ponto:" );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P1.distanciaAte ( P2 ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0807." );
   } // fim ex0807 ( )
```

```
// ---------------------------------------------  exemplo 08

/**
  * ex0808 - calcular a distancia entre dois pontos
  */
   public static void ex0808 ( )
   {
   // definir dados
      Pontos08 P = new Pontos08 ( );
      double distancia;
   // identificar o metodo
      IO.println ( "EXEMPLO0808 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P.ler ( "Fornecer coordenadas:" );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P.distancia ( ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0808." );
   } // fim ex0808 ( )

// ---------------------------------------------  exemplo 09

/**
  * ex0809 - calcular a distancia entre dois pontos
  */
   public static void ex0809 ( )
   {
   // definir dados
      Pontos09 P = new Pontos09 ( );
   // identificar o metodo
      IO.println ( "EXEMPLO0809 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P.ler ( "Fornecer coordenadas:" );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P.distancia ( ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0809." );
   } // fim ex0809 ( )
```

```
// ---------------------------------------------  exemplo 10

/**
  * ex0810 - calcular a distancia entre dois pontos
  */
   public static void ex0810 ( )
   {
   // definir dados
      Pontos10 P;
   // identificar o metodo
      IO.println ( "EXEMPLO0810 - Programa em Java" );
      IO.println ( "Calcular a distancia entre dois pontos" );
      IO.println ( );
   // ler dados
      P = new Pontos10 ( );
   // calcular distancia e mostrar resultado
      IO.println ( );
      IO.println ( "Distancia = " + P.distancia ( ) );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0810." );
   } // fim ex0810 ( )

// ---------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO08 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
/*
      ex0801 ( );
      ex0802 ( );
      ex0803 ( );

      ex0804 ( );
      ex0805 ( );
      ex0806 ( );
      ex0807 ( );
      ex0808 ( );
      ex0809 ( );
      ex0810 ( );
*/
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ---------------------------------------------

} // end class Exemplo08
```

```
//
// OBS.: RETIRAR OS COMENTARIOS /* */
//           PARA TESTAR CADA EXEMPLO INDIVIDUALMENTE.
//
/**
  *      Exemplo09 class
  *
  *      Copyright (c) 2007 by PUC-Minas / DCC
  *      All rights reserved.
  *
  * @author  PUC-Minas/DCC
  * @version 0.99
  */

// ---------------------------------------------  classes nativas para entrada e saida

import IO.*;
import java.util.*;

// ---------------------------------------------  definicao de classes

class Ponto01
{
// armazenadores
   public double X = 0, Y = 0, Z = 0;

// metodos
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     X = IO.readdouble ( "X = " );
     Y = IO.readdouble ( "Y = " );
     Z = IO.readdouble ( "Z = " );
   } // fim Ponto01.ler ( )

} // fim class Ponto01

class Ponto02
{
// armazenadores
   public double X = 0, Y = 0, Z = 0;

// metodos
   public void ler ( String mensagem )
   {
     IO.println ( mensagem );
     X = IO.readdouble ( "X = " );
     Y = IO.readdouble ( "Y = " );
     Z = IO.readdouble ( "Z = " );
   } // fim Ponto02.ler ( )
```

```
   public void lerDe ( FILE arquivo )
   {
     String linha;

     if ( arquivo.isOpen ( ) )
     {
       linha = arquivo.read ( );
       linha = linha.trim ( );
       if ( linha.length ( ) > 0 )
       {
         X     = Double.parseDouble ( linha );
         linha = arquivo.read ( );
         linha = linha.trim ( );
         if ( linha.length ( ) > 0 )
         {
           Y     = Double.parseDouble ( linha );
           linha = arquivo.read ( );
           linha = linha.trim ( );
           if ( linha.length ( ) > 0 )
           {
             Z   = Double.parseDouble ( linha );
           }
         }
       }
     } // fim se ( arquivo aberto )
   } // fim Ponto02.ler ( )

} // fim class Ponto02

// ---------------------------------------------  definicao da classe principal

public class Exemplo09
{
// ---------------------------------------------  definicao de metodos gerais

/**
  * identificar - comandos para fornecer dados sobre o programa
  * <br>
  */
   public static void identificar
                ( String titulo, String versao, String data,
                  String matricula, String nome )
   {
     IO.println ( titulo + "v." + versao + " - " + data );
     IO.println ( "MATRICULA: " + matricula + " ALUNO: " + nome );
     IO.println (  );
   } // fim identificar ( )

/**
  * pausa - comandos para controlar o programa
  * <br>
  */
   public static void pausa
                ( String mensagem )
   {
     IO.println (  );
     IO.pause   ( mensagem );
   } // fim pausa ( )
```

```
// ---------------------------------------------  exemplo 01

/**
  * ex0901 - gravar coordenadas de um ponto
  */
   public static void ex0901 ( )
   {
   // definir dados
      Ponto01 P1 = new Ponto01 ( );
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.OUTPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0901 - Programa em Java" );
      IO.println ( "Gravar arquivo" );
      IO.println ( );
   // ler dados
      P1.ler ( "Fornecer coordenadas do ponto:" );
   // gravar dados
      arquivo.println ( P1.X + " " + P1.Y + " " + P1.Z );
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0901." );
   } // fim ex0901 ( )

// ---------------------------------------------  exemplo 02

/**
  * ex0902 - ler coordenadas de um ponto do arquivo
  */
   public static void ex0902 ( )
   {
   // definir dados
      Ponto01 P1 = new Ponto01 ( );
      String  linha;
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.INPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0902 - Programa em Java" );
      IO.println ( "Ler arquivo" );
      IO.println ( );
   // ler dados do arquivo
      linha = arquivo.readln ( );
   // mostrar dados
      IO.println ( "Lido: " + linha );
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0902." );
   } // fim ex0902 ( )
```

```
// ---------------------------------------------  exemplo 03

/**
  * ex0903 - ler coordenadas de um ponto do arquivo
  */
   public static void ex0903 ( )
   {
   // definir dados
      Ponto01 P1 = new Ponto01 ( );
      String  linha;
      double  X, Y, Z;
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.INPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0903 - Programa em Java" );
      IO.println ( "Ler arquivo" );
      IO.println ( );
   // ler dados
      linha = arquivo.read ( );
      P1.X  = Double.parseDouble ( linha );
      linha = arquivo.read ( );
      P1.Y  = Double.parseDouble ( linha );
      linha = arquivo.read ( );
      P1.Z  = Double.parseDouble ( linha );
   // mostrar dados
      IO.println ( "Lido: " + P1.X + " " + P1.Y + " " + P1.Z );
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0903." );
   } // fim ex0903 ( )

// ---------------------------------------------  exemplo 04

/**
  * ex0904 - ler coordenadas de um ponto do arquivo
  */
   public static void ex0904 ( )
   {
   // definir dados
      Ponto02 P1 = new Ponto02 ( );
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.INPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0904 - Programa em Java" );
      IO.println ( "Ler arquivo" );
      IO.println ( );
   // ler dados
      P1.lerDe ( arquivo );
   // mostrar dados
      IO.println ( "Lido: " + P1.X + " " + P1.Y + " " + P1.Z );
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0904." );
   } // fim ex0904 ( )
```

```
// ---------------------------------------------  exemplo 05
/**
  * ex0905 - gravar coordenadas de outro ponto
  */
   public static void ex0905 ( )
   {
   // definir dados
      Ponto02 P2 = new Ponto02 ( );
   // abrir arquivo para acrescentar
      FILE arquivo = new FILE ( FILE.APPEND, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0905 - Programa em Java" );
      IO.println ( "Regravar arquivo" );
      IO.println ( );
   // ler dados
      P2.ler ( "Fornecer coordenadas do ponto:" );
   // gravar dados
      arquivo.println ( P2.X + " " + P2.Y + " " + P2.Z );
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0905." );
   } // fim ex0905 ( )

// ---------------------------------------------  exemplo 06

/**
  * ex0906 - ler coordenadas de pontos do arquivo
  */
   public static void ex0906 ( )
   {
   // definir dados
      Ponto01 P1 = new Ponto01 ( );
      String  linha;
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.INPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0906 - Programa em Java" );
      IO.println ( "Ler arquivo" );
      IO.println ( );
   // ler e mostrar dados do arquivo
      linha = arquivo.readln ( );
      IO.println ( "Lido: " + linha );
      linha = arquivo.readln ( );
      IO.println ( "Lido: " + linha );
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0906." );
   } // fim ex0906 ( )
```

```
// ---------------------------------------------  exemplo 07

/**
  * ex0907 - ler coordenadas de pontos do arquivo
  */
   public static void ex0907 ( )
   {
   // definir dados
      Ponto01 P1 = new Ponto01 ( );
      String  linha;
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.INPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0907 - Programa em Java" );
      IO.println ( "Ler arquivo" );      IO.println ( );
   // ler e mostrar dados do arquivo
      linha = arquivo.readln ( );
      while ( ! arquivo.eof ( ) )
      {
        IO.println ( "Lido: " + linha );
        linha = arquivo.readln ( );
      } // fim repetir enquanto houver dados
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0907." );
   } // fim ex0907 ( )

// ---------------------------------------------  exemplo 08

/**
  * ex0908 - ler coordenadas de pontos do arquivo
  */
   public static void ex0908 ( )
   {
   // definir dados
      Ponto02 P1 = new Ponto02 ( );
      String  linha;
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.INPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0908 - Programa em Java" );
      IO.println ( "Ler arquivo" );      IO.println ( );
   // ler e mostrar dados do arquivo
      P1.lerDe ( arquivo );
      while ( ! arquivo.eof ( ) )
      {
      // mostrar dados
         IO.println ( "Lido: " + P1.X + " " + P1.Y + " " + P1.Z );
      // ler de novo
         P1.lerDe ( arquivo );
      } // fim repetir enquanto houver dados
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0908." );
   } // fim ex0908 ( )
```

```
// ---------------------------------------------  exemplo 09

/**
  * ex0909 - alterar coordenadas de pontos do arquivo
  */
   public static void ex0909 ( )
   {
   // definir dados
      Ponto02 P1 = new Ponto02 ( );
      String  linha;
      char    resposta;
      int     posicao;
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.ALTER, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0909 - Programa em Java" );
      IO.println ( "Ler arquivo" );
      IO.println ( );
   // ler, mostrar e alterar dados do arquivo
      posicao = arquivo.tell ( );
      P1.lerDe ( arquivo );
      while ( ! arquivo.eof ( ) )
      {
      // mostrar dados
         IO.println ( "Lido: " + P1.X + " " + P1.Y + " " + P1.Z );
      // alterar dados
         resposta = IO.readchar ( "Quer alterar [S|N] ? " );
         if ( resposta == 'S' || resposta == 's' )
         {
         // ler novas coordenadas
            P1.ler ( "Fornecer novas coordenadas do ponto: " );
         // reposicionar
            arquivo.seek ( posicao );
         // regravar
            arquivo.println ( P1.X + " " + P1.Y + " " + P1.Z );
         }
      // ler de novo
         posicao = arquivo.tell ( );
         P1.lerDe ( arquivo );
      } // fim repetir enquanto houver dados
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0909." );
   } // fim ex0909 ( )
```

```
// ---------------------------------------------  exemplo 10

/**
  * ex0910 - ler coordenadas de pontos do arquivo
  *               com acesso direto
  */
   public static void ex0910 ( )
   {
   // definir dados
      Ponto02 P1 = new Ponto02 ( );
      int     posicao;
   // abrir arquivo
      FILE arquivo = new FILE ( FILE.INPUT, "DADOS.TXT" );
   // identificar o metodo
      IO.println ( "EXEMPLO0910 - Programa em Java" );
      IO.println ( "Ler arquivo" );
      IO.println ( );
   // ler e mostrar dados do arquivo
      posicao = arquivo.length ( );
      while ( posicao > 0 )
      {
      // posicionar
         arquivo.seek ( posicao-13 );
      // ler dados
         P1.lerDe ( arquivo );
      // mostrar dados
         IO.println ( "Lido: " + P1.X + " " + P1.Y + " " + P1.Z );
      // novas coordenadas
         posicao = posicao - 13;
      } // fim repetir enquanto houver dados
   // fechar arquivo
      arquivo.close ( );
   // pausa
      IO.pause   ( "APERTAR ENTER PARA TERMINAR EX0910." );
   } // fim ex0910 ( )
```

```
// ------------------------------------------------  definicao do metodo principal

   public static void main ( String [ ] args )
   {
   // identificar
      identificar ( "EXEMPLO09 - Programas em Java", "v.1", "01/08/2004",
                    "999999", "xxx yyy zzz" );
   // chamadas de metodos
/*
      ex0901 ( );
      ex0902 ( );
      ex0903 ( );
      ex0904 ( );
      ex0905 ( );
      ex0906 ( );
      ex0907 ( );
      ex0908 ( );
      ex0909 ( );
      ex0910 ( );
*/
   // pausa
      pausa ( "APERTAR ENTER PARA TERMINAR." );
   } // fim main ( )

// ------------------------------------------------

} // end class Exemplo09
```

```
// OBS.: COLOCAR O CÓDIGO ABAIXO NO ARQUIVO
//           Testar_Positivo1.java

/*
 * Positivo1 - classe para lidar com valores positivos.
 */
class Positivo1
{
// definir dados globais publicos
   public int erro;      // indicador de erro
   public int valor;    // armazenador de valor

   /*
    * toString ( ) - converter conteudo
                      para caracteres.
    * @return valor convertido
                 para caracteres
    */
   public String toString ( )
   {
     return ( ""+valor );
   } // fim toString ( )

   /*
    * Positivo1 ( ) - construtor padrao.
    */
   public Positivo1 ( )
   {
     erro  = 0;         // nao ha' erro
     valor = 0          // valor inicial zero
   } // fim construtor padrao ( )

   /*
    * Positivo1 ( ) - construtor alternativo.
    * @param inicial - valor inicial
                  a ser atribuido ao objeto
    */
   public Positivo1 ( int inicial )
   {
     erro  = 0;         // supoe nao haver erro
     atribuir ( inicial );
   } // fim construtor padrao ( )

   /*
    * Positivo1 ( ) - construtor alternativo.
    * @param inicial - valor inicial
                  a ser atribuido ao objeto
    */
   public Positivo1 ( Positivo1 inicial )
   {
     this  ( inicial.valor );
   } // fim construtor padrao ( )
```

```
/*
 * parsePositivo1 ( ) - converter caracteres
                        para inteiro positivo.
 * @return    valor inteiro convertido
 * @param s - caracteres para converter
 */
public int parsePositivo1 ( String s )
{
// definir dados locais
   int  x;
   char c;

// converter String para int
   erro  = 0;
   valor = 0;
   x     = 0;
   while ( x < s.length( ) && erro == 0 )
   {
         c = s.charAt ( x );
         if ( c < '0' || c > '9' )
           erro = 1;     // erro na conversao
         else
           valor = valor * 10 + ( c - '0' );
         x = x + 1;
   } // fim while
// testar se nao ha' erro
   if ( erro != 0 )
     valor = 0;      // corrigir o valor invalido
// retornar resultado
   return ( valor );
} // fim parsePositivo ( )

/*
 * atribuir ( ) - dar valor a um objeto
                  a partir de outro.
 * @param x - valor a ser atribuido
 */
public void atribuir ( Positivo1 x )
{
// atribuir
   if ( x.valor < 0 )
     erro = 2;       // atribuir valor invalido
   else
     valor = x.valor;
} // fim atribuir ( )
```

```
   /*
    * atribuir ( ) - atribuir valor inteiro
    * @param x - valor inteiro a ser atribuido
    */
   public void atribuir ( int x )
   {
   // testar e atribuir
      if ( x < 0 )
        erro = 2;       // atribuir valor invalido
      else
        valor = x;
   } // fim atribuir ( )

   /*
    * atribuir ( ) - dar valor a um objeto a partir
                     de cadeia de caracteres.
    * @param x - caracteres de onde extrair
                     o valor a ser atribuido
    */
   public void atribuir ( String s )
   {
   // converter e atribuir
      valor = parsePositivo1 ( s );
   } // fim atribuir ( )

} // fim class Positivo1
```

```
import IO.*;

/*
 * Testar_Positivo1 - classe para lidar
                          com valores positivos.
 */
class Testar_Positivo1
{
  public static void main ( String [ ] args )
  {
  // definir dados locais
     Positivo1 x1 = new Positivo1 (    );
     Positivo1 x2 = new Positivo1 (  3 );
     Positivo1 x3 = new Positivo1 ( x2 );
  // identificar
     IO.println ( "Programa para tratar
                     positivos - v.1" );
     IO.println ( );
  // mostrar dados
     IO.println ( "Definicao de dados" );
     IO.println ( "x1.erro= "+x1.erro+
                  " x1.valor= "+x1.valor );
     IO.println ( "x2.erro= "+x2.erro+
                  " x2.valor= "+x2.valor );
     IO.println ( "x3.erro= "+x3.erro+
                  " x3.valor= "+x3.valor );
  // atribuir valor a ser convertido
     x1.atribuir( "123" );
     x2.atribuir( "1a2" );
  // mostrar dados apos atribuicao
     IO.println ( );
     IO.println ( "Atribuicao de valores" );
     IO.println ( "x1=\"123\" x1.erro= "+
         x1.erro+" x1.valor= "+x1.valor );
     IO.println ( "x2=\"1a2\" x2.erro="+
         x2.erro+" x2.valor= "+x2.valor );
  // atribuir valor sem conversao
     IO.println ( );
     IO.println ( "Atribuicao de valores
                     positivos" );
     x3.atribuir( x1 );
     IO.println ( "x3=x1    x3.erro= "+x3.erro+
                  " x3.valor= "+x3.valor );
     x3.atribuir( x2 );
     IO.println ( "x3=x2    x3.erro= "+x3.erro+
                  " x3.valor= "+x3.valor );
     x3.atribuir( "456" );
     IO.println ( "x3=\"456\" x3.erro= "+x3.erro
                 +" x3.valor= "+x3.valor );
  // encerrar
     IO.println ( );
     IO.println ( "Apertar ENTER
                     para terminar." );
     IO.pause   ( );
  } // fim main ( )
} // fim class
```

```
// OBS.: COLOCAR O CÓDIGO ABAIXO NO ARQUIVO
//           Testar_Positivo2.java

/*
 * Positivo2 - classe para lidar
                com valores positivos.
 */
class Positivo2
{
// definir dados globais publicos
   private int erro;     // indicador de erro
   private int valor;    // armazenador de valor

   /*
    * toString ( ) - converter conteudo
                     para caracteres.
    * @return valor convertido
                     para caracteres
    */
   public String toString ( )
   {
     return ( ""+valor );
   } // fim toString ( )

   /*
    * Positivo2 ( ) - construtor padrao.
    */
   public Positivo2 ( )
   {
     erro  = 0;        // nao ha' erro
     valor = 0;        // valor inicial zero
   } // fim construtor padrao ( )

   /*
    * Positivo2 ( ) - construtor alternativo.
    * @param inicial - valor inicial
                a ser atribuido ao objeto
    */
   public Positivo2 ( int inicial )
   {
     erro  = 0;        // supoe nao haver erro
     atribuir ( inicial );
   } // fim construtor padrao ( )

   /*
    * Positivo1 ( ) - construtor alternativo.
    * @param inicial - valor inicial
                a ser atribuido ao objeto
    */
   public Positivo2 ( Positivo2 inicial )
   {
     erro  = 0;        // supoe nao haver erro
     atribuir ( inicial.valor );
   } // fim construtor padrao ( )
```

```
/*
 * extrairErro ( ) - obter codigo de erro.
 * @return codigo do erro
 *         0 - nao ha' erro
 *         1 - erro na conversao para positivo
 *         2 - valor invalido
 */
public int extrairErro ( )
{
// extrair erro do objeto
   return ( erro );
} // fim extrairErro ( )

/*
 * extrairValor ( ) - obter valor do objeto.
 * @return valor armazenado no objeto
 */
public int extrairValor ( )
{
// extrair valor do objeto
   return ( valor );
} // fim extrairValor ( )

/*
 * parsePositivo2 ( ) - converter caracteres
                        para inteiro positivo.
 * @return    valor inteiro convertido
 * @param s - caracteres a converter
 */
public int parsePositivo2 ( String s )
{
// definir dados locais
   int  x;
   char c;

// converter String para int
   erro  = 0;
   valor = 0;
   x     = 0;
   while ( x < s.length( ) && erro == 0 )
   {
         c = s.charAt ( x );
         if ( c < '0' || c > '9' )
           erro = 1;     // erro na conversao
         else
           valor = valor * 10 + ( c - '0' );
         x = x + 1;
   } // fim while
// testar se nao ha' erro
   if ( erro != 0 )
     valor = 0;      // corrigir o valor invalido
// retornar resultado
   return ( valor );
} // fim parsePositivo ( )
```

```
/*
 * atribuir ( ) - atribuir valor ao objeto
                  a partir de outro.
 * @param x - valor a ser atribuido
 */
public void atribuir ( Positivo2 x )
{
// atribuir
   if ( x.erro != 0 | x.valor < 0 )
   {
     erro  = 2;      // atribuir valor invalido
     valor = 0;
   }
   else
   {
     erro  = x.extrairErro ( );
     valor = x.extrairValor( );
   } // fim se
} // fim atribuir ( )

/*
 * atribuir ( ) - atribuir valor inteiro
 * @param x - valor inteiro a ser atribuido
 */
public void atribuir ( int x )
{
// testar e atribuir
   if ( x < 0 )
   {
     erro  = 2;      // atribuir valor invalido
     valor = 0;
   }
   else
   {
         erro  = 0;
     valor = x;
   } // fim se
} // fim atribuir ( )

/*
 * atribuir ( ) - dar valor a um objeto
    a partir de cadeia de caracteres.
 * @param x - caracteres de onde extrair
                  o valor a ser atribuido
 */
public void atribuir ( String s )
{
// converter e atribuir
   valor = parsePositivo2 ( s );
} // fim atribuir ( )
```

```
   /*
    * lerPositivo2 ( ) - ler do teclado valor
                         para o objeto.
    * @param msg - caracteres com
                  mensagem a ser exibida
    */
   public void lerPositivo2 ( String msg )
   {
   // definir dado local
      String dado;
   // ler dado
      dado = IO.readString ( msg );
   // converter e atribuir
      parsePositivo2 ( dado );
   } // fim lerPositivo ( )

   /*
    * trocar ( ) - trocar valor com outro objeto.
    * @param y - valor do outro objeto
                 a ser trocado
    */
   public void trocar ( Positivo2 y )
   {
   // definir dado local
      Positivo2 z = new Positivo2 ( );
   // salvar o conteudo de y
      z.atribuir( y );
      y.atribuir( this );
        atribuir( z );
   } // fim trocar ( )

} // fim class Positivo2
```

```
import IO.*;

/*
 * Testar_Positivo2 - classe para lidar com
                          valores positivos.
 */
class Testar_Positivo2
{
  public static void main ( String [ ] args )
  {
  // definir dados locais
     Positivo2 x1 = new Positivo2 (    );
     Positivo2 x2 = new Positivo2 (  3 );
     Positivo2 x3 = new Positivo2 ( x2 );
  // identificar
     IO.println ( "Programa para tratar
                     positivos - v.2" );
     IO.println ( );
  // mostrar dados
     IO.println ( "Definicao de dados" );
     IO.println ( "x1.erro= "+x1.extrairErro( )+
                  " x1.valor= "+x1.extrairValor( ) );
     IO.println ( "x2.erro= "+x2.extrairErro( )+
                  " x2.valor= "+x2.extrairValor( ) );
     IO.println ( "x3.erro= "+x3.extrairErro( )+
                  " x3.valor= "+x3.extrairValor( ) );
  // atribuir valor a ser convertido
     x1.atribuir( "123" );
     x2.atribuir( "1a2" );
  // mostrar dados apos atribuicao
     IO.println ( );
     IO.println ( "Atribuicao de valores" );
     IO.println ( "x1=\"123\" x1.erro= "+
         x1.extrairErro( )+" x1.valor= "+x1 );
     IO.println ( "x2=\"1a2\" x2.erro= "+
         x2.extrairErro( )+" x2.valor= "+x2 );
  // atribuir valor sem conversao
     IO.println ( );
     IO.println ( "Atribuicao de valores
                     positivos" );
     x3.atribuir( x1 );
     IO.println ( "x3=x1    x3.erro= "+
         x3.extrairErro( )+" x3.valor= "+x3 );
     x3.atribuir( x2 );
     IO.println ( "x3=x2    x3.erro= "+
         x3.extrairErro( )+" x3.valor= "+x3 );
     x3.atribuir( "456" );
     IO.println ( "x3=\"456\" x3.erro= "+
         x3.extrairErro( )+" x3.valor= "+x3 );
```

```
  // ler valor do teclado
     IO.println ( );
     IO.println ( "Leitura de valor" );
     x1.lerPositivo2 ( "Fornecer um valor
                        positivo: " );
     IO.println ( "x1.erro= "+x1.extrairErro( )+
                  " x1.valor= "+x1 );
     x2.lerPositivo2 ( "Fornecer outro valor
                        positivo: " );
     IO.println ( "x2.erro= "+x2.extrairErro( )+
                  " x2.valor= "+x2 );
  // trocar valores
     IO.println ( );
     IO.println ( "Troca de valores" );
     x1.trocar  ( x2 );
     IO.println ( "x1.erro= "+x1.extrairErro( )+
                  " x1.valor= "+x1 );
     IO.println ( "x2.erro= "+x2.extrairErro( )+
                  " x2.valor= "+x2 );
  // encerrar
     IO.println ( );
     IO.println ( "Apertar ENTER
                    para terminar." );
     IO.pause   ( );
   } // fim main ( )
} // fim class
```

```
// OBS.: COLOCAR O CÓDIGO ABAIXO NO ARQUIVO
//           Testar_Tabela.java

/**
  * Tabela - modelo de classe para lidar com
                tabelas.
  */

public class Tabela
{
// definir armazenadores
   private int erro;        // indicador de erro

   private int tamanho;// quantidade de itens
   private int livre;       // proxima posicao livre
   private Object [ ] valor;   // armazenador

/**
  * Tabela ( ) - construtor padrao
                   da classe Tabela.
  * @param quantidade – quantidade
                   maxima de elementos
  */
  public Tabela ( int quantidade )
  {
  // atribuir valores iniciais
     tamanho = 0;
     livre   = 0;

     if ( quantidade <= 0 )
     {
       setError ( 4 );  // valor invalido
     }
     else
     {
       valor   = new Object [ tamanho ];
       if ( valor == null )
       {
         setError ( 1 ); // nao ha' mais espaco
       }
       else
       {
         tamanho = quantidade;
         resetError ( ); // nao ha' erro
       } // fim se
     } // fim se
  } // fim construtor padrao
```

```
/**
  * toString ( ) - converter todo o conteudo
                   para String.
  * <br>
  * @return conteudo convertido para
               caracteres
  */
   public String toString  ( )
   {
   // definir dado local
      String saida = "";
      int x;
   // coletar dados
      for ( x = 0; x < livre ; x = x + 1 )
      {
        saida = saida + valor [ x ] + "\n" ;
      } // fim for
     return ( saida );
   } // fim toString ( )

/**
  * empty ( ) - informar se a tabela
                esta' vazia.
  * <br>
  * @return resposta - se estiver vazia (true),
                       ou nao (false)
  */
   public boolean empty ( )
   {
      return ( livre == 0 );
   } // fim empty ( )

/**
  * push ( ) - adicionar um valor 'a tabela.
  * <br>
  * @param resultado - se houver inclusao
                       (true), ou nao (false)
  * @param x - objeto a ser adicionado
  */
   public void push ( Object x )
   {
   // adicionar um valor 'a tabela
      if ( livre >= tamanho ) // sem espaco
      {
            setError ( 1 ); // nao ha' mais espaco
      }
      else                    // se houver espaco
      {
        valor [ livre ] = x;//  guardar valor
        livre = livre + 1;  //  proxima posicao
        resetError ( );     //  nao ha' erro
      } // fim se
   } // fim push ( )
```

```
/**
  * pop ( ) - retirar um valor da tabela.
  * <br>
  * @return resultado - objeto obtido, ou null
  */
   public Object pop ( )
   {
   // definir dado local
      Object x = null;
      int    y;
   // remover um valor da tabela
      if ( empty ( ) )        // tabela vazia
      {
            setError ( 2 ); // nao ha' elementos
      }
      else                    // ha' elementos
      {
        x = valor [ 0 ];     // separar o primeiro
                             // deslocar os outros
        for ( y = 0; y < livre-1; y = y + 1 )
          valor [ y ] = valor [ y + 1 ];
        livre = livre - 1;   // um espaco a menos
        valor [ livre ] = null; // desconectar a
                                //  referencia
        resetError ( );     // nao ha' erro
      }
      return ( x );          //  retornar o elemento
   } // fim pop ( )
```

```
/**
  * peek ( ) - retornar o valor de uma posicao
                da tabela.
  * <br>
  * @return resultado - objeto obtido, ou null
  * @param  position  - posicao do objeto
                        na tabela
  */
   public Object peek ( int position )
   {
   // definir dado local
      Object x = null;
      int    y;
   // obter um valor da tabela
      if ( empty ( ) )       // tabela vazia
      {
            setError ( 2 ); //  nao ha' elementos
      }
      else                   // ha' elementos
      {
        if ( position < 0 || position >= livre )
        {
          setError ( 3 );   // posicao invalida
        }
        else
        {
          x = valor [ position ]; //  copiar o valor
           da posicao
        } // fim se ( posicao invalida )
      } // fim se ( tabela vazia )
      return ( x );   //  retornar o elemento
   } // fim peek ( )

/**
  * maxSize ( ) - informar a quantidade
                  maxima de valores na tabela.
  * <br>
  * @return - numero maximo de valores
                  na tabela
  */
   public int maxSize ( )
   {
      return ( tamanho );
   } // fim maxSize ( )

/**
  * size ( ) - informar o numero atual
                de valores na tabela.
  * <br>
  * @return - numero atual de valores
                  na tabela
  */
   public int size ( )
   {
      return ( livre );
   } // fim size ( )

/**
  * getError ( ) - informar o codigo do erro.
  * <br>
```

```
  * @return - codigo do erro
  *          0 - nao ha' erro
  *          1 - nao ha' mais espaco
  *          2 - nao ha' elementos
  *          3 - posicao invalida
  *          4 - valor invalido
  */
   public int getError ( )
   {
      return ( erro );
   } // fim getError ( )

/**
  * setError ( ) - estabelecer um codigo de
                   erro.
  * <br>
  * @param code - codigo do erro
  */
   public void setError ( int codigo )
   {
      erro = codigo;
   } // fim setError ( )

/**
  * resetError ( ) - voltar erro 'a condicao
                     normal.
  * <br>
  */
   public void resetError ( )
   {
      erro = 0;
   } // fim resetError ( )
```

```
/**
  * errorMsg ( ) - obter mensagem de erro.
  * <br>
  * @return - mensagem de erro
  *          0 - nao ha' erro
  *          1 - nao ha' mais espaco
  *          2 - nao ha' elementos
  *          3 - posicao invalida
  *          4 - valor invalido
  */
   public String errorMsg ( )
   {
   // definir dado local
      String msg = new String ( "ERRO: " );
   // selecionar mensagem
      switch ( erro )
      {
        case 0:
          msg = msg + "Nao ha' erro.";
          break;
        case 1:
          msg = msg + "Nao ha' mais espaco.";
          break;
        case 2:
          msg = msg + "Nao ha' elemento.";
          break;
        case 3:
          msg = msg + "Posicao invalida.";
          break;
        case 4:
          msg = msg + "Valor invalido.";
          break;
        default:
          msg = msg + "Indefinido.";
      } // fim selecionar
   // retornar mensagem
      return ( msg );
   } // fim errorMsg ( )
```

```
/**
  * className ( ) - informar o nome da classe de um objeto.
  * <br>
  * @return - nome da classe
                de um objeto na tabela
  */
   public String className ( )
   {
      String resposta = "";
      Object x = null;;

      if ( empty ( ) )
      {
        setError ( 2 ); //  nao ha' elementos
      }
      else
      {
         x = valor [ 0 ]; //  extrair a informacao
         resposta = x.getClass( ).getName ( );
      }
      return ( resposta );
   } // fim className ( )

} // fim class Tabela
```

```
import IO.*;

/**
  * Testar_Tabela - modelo de classe para
                          testar tabelas
  */

public class Testar_Tabela
{
/**
  * teste1 - testar tabela com caracteres
  */
  public static void teste1 ( )
  {
  // definir dados
     Tabela  a = new Tabela ( 10 );
     Object  x;

  // identificar
     IO.println ( "Teste 1 - Tabela de
                      caracteres" );
     IO.println ( );

  // adicionar dados
     IO.println ( "Tabela com capacidade para
               " + a.maxSize( ) + " elementos:" );
     a.push ( "123" );
     if ( a.getError ( ) != 0 )
     {
       IO.println ( a.errorMsg ( ) );
     }
     else
     {
       a.push ( "abc" );
       if ( a.getError ( ) != 0 )
       {
         IO.println ( a.errorMsg ( ) );
       }
       else
       {
       // mostrar dados
          IO.println ( "Tabela com " + a.size( )
                      + " elemento(s):" );
          while ( ! a.empty ( ) )
          {
            IO.println ( a );
            IO.println ( );
            x = (String) a.pop ( );
            IO.println ( "Retirado da tabela
                           o elemento " + x );
            IO.println ( );
            IO.println ( "Tabela com " + a.size( )
                        + " elemento(s):" );
          }
       } // fim se
     } // fim se

  // mostrar dados
     IO.println ( );
     IO.println ( "Tabela atualmente com " +
```

```
                    a.size( ) + " elemento(s):" );
    IO.println ( a );
  } // fim teste1

/**
  * teste2 - testar tabela com inteiros
  */
  public static void teste2 ( )
  {
  // definir dados
    Tabela  a = new Tabela ( 10 );
    Object  x;
    int     y, z;

  // identificar
    IO.println ( "Teste 2-Tabela de inteiros" );
    IO.println ( );

  // adicionar dados
    y = 0;
    while ( y < 3 && a.getError ( ) == 0 )
    {
          a.push ( new Integer ( y ) );
          y = y + 1;
    } // fim while
    if ( a.getError ( ) != 0 )
    {
      IO.println ( a.errorMsg ( ) );
    }

  // mostrar dados
    IO.println ( "Tabela com " + a.size ( ) +
                 " elemento(s):" );
    IO.println ( a );
    IO.println ( );

    z = a.size ( );
    for ( y = 0; y < z; y = y + 1 )
    {
      if ( a.className( ).equals
           ( "java.lang.Integer" ) )
         x = (Integer) a.pop ( );
      else
        x = null;
      IO.println ( "Retirado da tabela
                    o elemento " + x );
      IO.println ( "Tabela com " + a.size ( ) +
                   " elemento(s):" );
      IO.println ( a );
      IO.println ( );
    } // fim repetir
  } // fim teste2

/**
  * main - testar tabelas com objetos
  */
  public static void main ( String [ ] args )
  {
  // definir dados
    int opcao = 1;
```

```
  // repetir ate' parar
     do
     {
     // oferecer opcoes
        IO.println ( "Teste de tabelas" );
        IO.println ( );
        IO.println ( "Opcoes:" );
        IO.println ( );
        IO.println ( "0. Terminar" );
        IO.println ( "1. Caracteres" );
        IO.println ( "2. Inteiros" );
        IO.println ( );

        opcao = IO.readint ( "Escolher
                                  sua opcao : " );

     // escolher opcao
        switch ( opcao )
        {
          case 0:
            IO.println ( "Encerrar testes." );
            break;
          case 1:
            teste1 ( );
            break;
          case 2:
            teste2 ( );
            break;
          default:
            IO.println ("ERRO: Opcao invalida.");
        } // fim escolher
     // encerrar
        IO.println ( );
        IO.println ( "Apertar ENTER." );
        IO.pause ( );
     }
     while ( opcao != 0 );
   } // fim main
} // fim class Testar_Tabela
```

```
// OBS.: COLOCAR O CÓDIGO ABAIXO NO ARQUIVO
//          Vetor.java

/**
  * Erro - modelo de classe para lidar com
           erros.
  */

class Erro
{
// definir tratamento de erro
   private int code;        // indicador de erro

/**
  * Erro ( ) - construtor da classe Erro.
  * <br>
  */
   public Erro  ( )
   {
     code = 0;
   } // fim construtor padrao

/**
  * toString ( ) - converter conteudo para
                   String.
  * <br>
  * @return indicar o codigo de erro atual
  */
   public String toString  ( )
   {
     return ( "ERRO = " + code );
   } // fim toString ( )

/**
  * get ( ) - informar o codigo do erro.
  * <br>
  * @return - codigo do erro
  *          0 - nao ha' erro
  *          1 - nao ha' mais espaco
  *          2 - nao ha' elementos
  *          3 - posicao invalida
  *          4 - valor invalido
  */
   public int get ( )
   {
      return ( code );
   } // fim get ( )

/**
  * set ( ) - estabelecer um codigo do erro.
  * <br>
  * @param code - codigo do erro
  */
   public void set ( int codigo )
   {
      code = codigo;
   } // fim set ( )
```

```
/**
  * reset ( ) - voltar erro 'a condicao normal.
  * <br>
  */
   public void reset ( )
   {
      code = 0;
   } // fim resetError ( )

/**
  * msg ( ) - obter mensagem de erro.
  * <br>
  * @return - mensagem de erro
  *          0 - nao ha' erro
  *          1 - nao ha' mais espaco
  *          2 - nao ha' elementos
  *          3 - posicao invalida
  *          4 - valor invalido
  */
   public String msg ( )
   {
   // definir dado local
      String msg = new String ( "ERRO: " );
   // selecionar mensagem
      switch ( code )
      {
        case 0:
          msg = msg + "Nao ha' erro.";
          break;
        case 1:
          msg = msg + "Nao ha' mais espaco.";
          break;
        case 2:
          msg = msg + "Nao ha' elemento.";
          break;
        case 3:
          msg = msg + "Posicao invalida.";
          break;
        case 4:
          msg = msg + "Valor invalido.";
          break;
        default:
          msg = msg + "Indefinido.";
      } // fim selecionar
   // retornar mensagem
      return ( msg );
   } // fim msg ( )

} // fim class Erro
```

```
/**
  * Vetor - modelo de classe para lidar com
              tabelas.
  */

public class Vetor
{
// definir armazenadores
                          // tratamento de erro
   public  Erro erro = new Erro ( ) ;

   private int ultimo ; // posicao do ultimo
   private int tamanho; // quantidade de itens
   private Object [ ] valor; // armazenador

/**
  * Vetor ( ) - construtor padrao
                da classe Vetor.
  * @param  quantidade – quantidade
                maxima de elementos
  */
   public Vetor ( int quantidade )
   {
  // atribuir valores iniciais
     ultimo  = 0;
     tamanho = 0;
     if ( quantidade <= 0 )
     {
       erro.set ( 4 );      // valor invalido
     }
     else
     {
       valor   = new Object [ quantidade ];
       if ( valor == null )
       {
         erro.set ( 1 );   // nao ha' mais espaco
       }
       else
       {
         tamanho = quantidade;
         erro.reset ( );  // nao ha' erro
       } // fim se
     } // fim se
   } // fim construtor padrao
```

```
/**
  * toString ( ) - converter conteudo para
                   String.
  * <br>
  * @return conteudo convertido para
                   caracteres
  */
   public String toString  ( )
   {
   // definir dado local
      String saida = "";
      int x;
   // coletar dados
      for ( x = 0; x < size ( ) ; x = x + 1 )
      {
        saida = saida + valor [ x ] + "\n";
      } // fim for
     return ( saida );
   } // fim toString ( )

/**
  * set ( ) - modificar valor em uma posicao
              do vetor.
  * <br>
  * @param position - posicao onde alterar
  * @param x        - objeto alterado
  */
   public void set ( int position, Object x )
   {
   // definir dado local
      boolean resposta = false;
   // adicionar um valor ao vetor
      if ( position < 0 || position >= tamanho )
      {
            erro.set ( 1 );     // posicao invalida
      }
      else
      {
        valor [ position ] = x;
        if ( position > ultimo )
          ultimo = position;
        erro.reset ( );         // nao ha' erro
      } // fim se
   } // fim set ( )
```

```
/**
  * get ( ) - obter valor de uma posicao
                do vetor.
  * <br>
  * @return objeto obtido, ou null
  * @param position - posicao onde
                              obter objeto
  */
   public Object get ( int position )
   {
   // definir dado local
      Object x = null;
   // obter um valor de uma posicao
      if ( position < 0 || position >= tamanho )
      {
            erro.set ( 1 );     // posicao invalida
      }
      else
      {
        x = valor [ position ];
        erro.reset ( );     // nao ha' erro
      } // fim se
      return ( x );
   } // fim get ( )

/**
  * maxSize ( ) - informar o tamanho maximo
                    do vetor.
  * <br>
  * @return quantidade maxima de dados
                    no vetor
  */
   public int maxSize ( )
   {
      return ( tamanho );
   } // fim maxSize ( )

/**
  * size ( ) - informar o tamanho atual
                do vetor.
  * <br>
  * @return quantidade atual de dados
                no vetor
  */
   public int size ( )
   {
      return ( ultimo+1 );
   } // fim size ( )

} // fim class Vetor
```

```
// OBS.: COLOCAR O CÓDIGO ABAIXO NO ARQUIVO
//           Testar_Vetor.java

import IO.*;

/**
  * Testar_Vetor - modelo de classe para
                        testar vetores
  */

public class Testar_Vetor
{
/**
  * teste1 - testar vetor com caracteres
  */
  public static void teste1 ( )
  {
  // definir dados
     Vetor   a = new Vetor ( 5 );
     Object  x;
     int     y;

  // identificar
     IO.println ( "Teste 1 - Vetor com
                      caracteres" );
     IO.println ( );

  // adicionar dados
     IO.println ( "Vetor com capacidade para "
                  + a.maxSize ( ) + " dados." );
     IO.println ( );
     a.set ( 0, "123" );
     if ( a.erro.get( ) == 0 )
     {
       a.set ( 1, "abc" );
       if ( a.erro.get( ) == 0 )
       {
       // mostrar dados
          for ( y = 0; y < a.size ( ); y = y + 1 )
          {
            x = a.get ( y );
            if ( x == null )
              IO.println ( y + " : vazio" );
            else
              IO.println ( y + " : " + x  );
          } // fim repetir
       } // fim se
     } // fim se

  // mostrar dados
     IO.println ( );
     IO.println ( "Vetor atualmente com " +
                     a.size ( ) + " elementos:" );
     IO.println ( a );
  } // fim teste1
```

```
/**
  * teste2 - testar vetor com inteiros
  */
  public static void teste2 ( )
  {
  // definir dados
     Vetor  a = new Vetor ( 5 );
     Object  x;
     int     y, z;

  // identificar
     IO.println ( "Teste 2-Vetor com inteiros" );
     IO.println ( );

  // adicionar dados
     y = 0;
     while ( y < 5 && a.erro.get( ) == 0 )
     {
           a.set ( y, new Integer ( y ) );
           y = y + 1;
     } // fim while
  // mostrar dados
     IO.println ( "Vetor com " + a.size ( ) +
                  " elementos:" );
     IO.println ( a );
     IO.println ( );

     z = a.size ( );
     for ( y = 0; y < z; y = y + 1 )
     {
       x = a.get ( y );
       IO.println ( "Elemento na posicao " + y
                    + " igual a " + x );
       IO.println ( );
     } // fim repetir
  } // fim teste2
```

```
/**
  * main - testar tabelas com objetos
  */
  public static void main ( String [ ] args )
  {
  // definir dados
     int opcao = 1;

  // repetir ate' parar
     do
     {
     // oferecer opcoes
        IO.println ( "Teste de vetores" );
        IO.println ( );
        IO.println ( "Opcoes:" );
        IO.println ( );
        IO.println ( "0. Terminar" );
        IO.println ( "1. Caracteres" );
        IO.println ( "2. Inteiros" );
        IO.println ( );
        opcao = IO.readint ( "Escolher sua
                                    opcao : " );

     // escolher opcao
        switch ( opcao )
        {
          case 0:
            IO.println ( "Encerrar testes." );
            break;
          case 1:
            teste1 ( );
            break;
          case 2:
            teste2 ( );
            break;
          default:
            IO.println ("ERRO: Opcao invalida.");
        } // fim escolher
     // encerrar
        IO.println ( );
        IO.println ( "Apertar ENTER." );
        IO.pause ( );
     }
     while ( opcao != 0 );
  } // fim main

} // fim class Testar_Vetor
```

```
// OBS.: COLOCAR O CÓDIGO ABAIXO NO ARQUIVO
//          Matriz.java

/**
  * Erro - modelo de classe para lidar com
           erros.
  */

class Erro
{
// definir tratamento de erro
   protected int  code;   // indicador de erro
   private String name;

/**
  * Erro ( ) - construtor da classe Erro.
  * <br>
  */
   public Erro  ( )
   {
     code = 0;
     name = "";
   } // fim construtor padrao

/**
  * Erro ( ) - construtor alternativo
                da classe Erro.
  * @param name - nome da classe
                associada
  * <br>
  */
   public Erro  ( String name1 )
   {
     reset ( );
     name = name1;
   } // fim construtor padrao

/**
  * toString ( ) - converter conteudo
                   para String.
  * <br>
  * @return indicar o codigo de erro atual
  */
   public String toString  ( )
   {
     return ( "[" + name + "] ERRO = "+code );
   } // fim toString ( )
```

```
/**
  * get ( ) - informar o codigo do erro.
  * <br>
  * @return - codigo do erro
  *          0 - nao ha' erro
  *          1 - nao ha' mais espaco
  *          2 - nao ha' elementos
  *          3 - posicao invalida
  *          4 - valor invalido
  */
   public int get ( )
   {
      return ( code );
   } // fim get ( )

/**
  * set ( ) - estabelecer um codigo do erro.
  * <br>
  * @param code - codigo do erro
  */
   public void set ( int codigo )
   {
      code = codigo;
   } // fim set ( )

/**
  * reset ( ) - voltar erro 'a condicao normal.
  * <br>
  */
   public void reset ( )
   {
      code = 0;
   } // fim reset ( )

/**
  * msg ( ) - obter mensagem de erro.
  * <br>
  * @return - indicador de erro e classe
  */
   protected String msg ( )
   {
   // definir dado local
      String m = new String ( "[" + name +
                                    "] ERRO" );
      return ( m );
   } // fim msg

} // fim class Erro
```

```
/**
  * Matriz - modelo de classe para lidar
               com matrizes
  */

public class Matriz extends Erro
{
// definir tratamento de erro
   public Erro erro = new Erro ( "Matriz" );

// definir armazenadores
   private int maxLinhas , // quantidade de
                                 linhas
             maxColunas;   // quantidade de
                                 colunas
   private int linhas ,        // linhas  usadas
             colunas;          // colunas usadas
   private Object [ ] [ ] valor; // armazenador

/**
  * Matriz ( ) - construtor do tipo Matriz.
  * @param nLinhas   - quantas linhas
  * @param nColunas - quantas colunas
  */
  public Matriz ( int nLinhas, int nColunas )
  {
  // definir dados locais
     int x, y;

  // atribuir valores iniciais
     maxLinhas  = 0;
     maxColunas = 0;
     linhas     = 0;
     colunas    = 0;

     if ( nLinhas <= 0 || nColunas <= 0 )
     {
       valor      = null;
       erro.set ( 4 );          // valor invalido
     }
     else
     {
       valor      = new Object
                         [ nLinhas ][ nColunas ];
       if ( valor == null )
       {
         erro.set ( 1 ); // nao ha' mais espaco
       }
       else
       {
         maxLinhas  = nLinhas ;
         maxColunas = nColunas;
         erro.reset ( ); // nao ha' erro
       } // fim se
     } // fim se
  } // fim construtor padrao
/**
  * toString ( ) - converter conteudo para
                   String.
  * <br>
```

```
* @return conteudo convertido para
                    caracteres
*/
public String toString  ( )
{
// definir dado local
   String saida = "";
   int x, y;
// coletar dados
   for ( x = 0; x < lines( ) ; x = x + 1 )
   {
     for ( y = 0; y < columns( ) ; y = y + 1 )
     {
       saida = saida + "\t" + valor [ x ] [ y ];
     } // fim for
     saida = saida + "\n";
   } // fim for
  return ( saida );
} // fim toString ( )
```

```
/**
  * msg ( ) - obter mensagem de erro.
  * <br>
  * @return - mensagem de erro
  *          0 - nao ha' erro
  *          1 - nao ha' mais espaco
  *          2 - nao ha' elementos
  *          3 - posicao invalida
  *          4 - valor invalido
  */
   public String msg ( )
   {
   // definir dado local
      String m = new String (erro.msg ( )+" : ");
   // selecionar mensagem
      switch ( erro.get( ) )
      {
        case 0:
          m = m + "Nao ha' erro.";
          break;
        case 1:
          m = m + "Nao ha' mais espaco.";
          break;
        case 2:
          m = m + "Nao ha' elemento.";
          break;
        case 3:
          m = m + "Posicao invalida.";
          break;
        case 4:
          m = m + "Valor invalido.";
          break;
        default:
          m = m + "Indefinido.";
      } // fim selecionar
   // retornar mensagem
      return ( m );
   } // fim msg ( )
```

```
/**
  * set ( ) - adicionar um valor 'a matriz.
  * <br>
  * @param x - linha a ser alterada
  * @param y - coluna a ser alterada
  * @param z - objeto a ser adicionado
  */
  public void set ( int x, int y, Object z  )
  {
  // adicionar um valor 'a matriz
     if ( ( x < 0 || x >= maxLinhas  ) ||
          ( y < 0 || y >= maxColunas ) )
     {
       erro.set ( 3 ); // posicao invalida
     }
     else
     {
       valor [ x ] [ y ] = z;
       if ( x > linhas  )
        linhas = x:    // ultima linha usada
       if ( y > colunas )
        colunas = y; // ultima coluna usada

       erro.reset ( ); // nao ha' erro
     }
  } // fim set ( )

/**
  * get ( )  - obter valor de uma posicao
                da matriz.
  * <br>
  * @return objeto obtido da posicao, ou null
  * @param x - linha  de onde obter objeto
  * @param y - coluna de onde obter objeto
  */
   public Object get ( int x, int y )
   {
   // definir dado local
      Object z = null;
   // obter um valor de uma posicao
      if ( ( x < 0 || x >= maxLinhas  ) ||
           ( y < 0 || y >= maxColunas ) )
      {
            erro.set ( 3 ); // posicao invalida
      }
      else
      {
        z = valor [ x ][ y ];
        erro.reset ( );     // nao ha' erro
      } // fim se
      return ( z );
   } // fim get ( )
```

```
/**
  * maxLines ( ) - informar a quantidade
                   maxima de linhas na matriz.
  * <br>
  * @return numero maximo de linhas
                na matriz
  */
  public int maxLines ( )
  {
     return ( maxLinhas );
  } // fim maxLines ( )

/**
  * maxColumns ( ) - informar a quantidade
                    maxima de colunas na matriz.
  * <br>
  * @return numero maximo de colunas
                na matriz
  */
  public int maxColumns ( )
  {
     return ( maxColunas );
  } // fim maxColumns ( )

/**
  * lines ( ) - informar a quantidade de linhas
                   usadas na matriz.
  * <br>
  * @return numero de linhas usadas
                na matriz
  */
  public int lines ( )
  {
     return ( linhas+1 );
  } // fim lines ( )

/**
  * columns ( ) - informar a quantidade de
                     colunas usadas na matriz.
  * <br>
  * @return numero de colunas usadas
                na matriz
  */
  public int columns ( )
  {
     return ( colunas+1 );
  } // fim columns ( )

} // fim class Matriz
```

```
// OBS.: COLOCAR O CÓDIGO ABAIXO NO ARQUIVO
//           Testar_Matriz.java

import IO.*;

/**
  * Testar_Matriz - modelo de classe para
                        testar matrizes
  */

public class Testar_Matriz
{
/**
  * teste1 - testar matriz com caracteres
  */
  public static void teste1 ( )
  {
  // definir dados
     Matriz  a = new Matriz ( 5, 5 );
     Object  z;
     int     x, y;

  // identificar
     IO.println ( "Teste 1 - Matriz com
                                caracteres" );
     IO.println ( );

  // testar posicao invalida
     IO.println ( "Teste de colocar valor
                    em posicao invalida" );
     IO.println ( );
     a.set ( -1, 10, new String ( "erro" ) );
     if ( a.erro.get ( ) != 0 )
       IO.println ( a.msg ( ) );

  // testar obter valor de posicao vazia
     IO.println ( );
     IO.println ( "Teste de obter valor
                    de posicao invalida" );
     IO.println ( );
     a.get ( -1, 10 );
     if ( a.erro.get ( ) != 0 )
       IO.println ( a.msg ( ) );

  // adicionar dados
     for ( x = 0; x < 3; x = x + 1 )
       for ( y = 0; y < 3; y = y + 1 )
       {
         a.set ( x, y, new String ( "" + x + y ) );
         if ( a.erro.get ( ) != 0 )
           IO.println ( a.msg ( ) );
       } // fim repetir

  // mostrar dados
     IO.println ( );
     IO.println ( "Matriz com " + a.lines ( ) +
          "x" + a.columns ( ) + " elementos:" );
     IO.println ( a );
  } // fim teste1
```

```
/**
  * teste2 - testar matriz com inteiros
  */
  public static void teste2 ( )
  {
  // definir dados
     Matriz  a = new Matriz ( 5, 5 );
     Object  z;
     int     x, y;

  // identificar
     IO.println ("Teste 2-Matriz com inteiros");
     IO.println ( );

  // adicionar dados
     IO.println ( "Matriz com capacidade
          para " + a.maxLines ( ) + "x" +
          a.maxColumns ( ) + " elementos." );
     IO.println ( );
     for ( x = 0; x < 3; x = x + 1 )
       for ( y = 0; y < 3; y = y + 1 )
       {
         a.set ( x, y, new Integer ( x*10 + y ) );
       } // fim repetir

  // mostrar dados
     IO.println ( "Matriz atualmente com " +
                  a.lines ( ) + "x" +
                  a.columns ( ) + " elementos:" );
     IO.println ( );

     for ( x = 0; x < a.lines( ); x = x + 1 )
     {
       for ( y = 0; y < a.columns( ); y = y + 1 )
       {
         z = a.get ( x, y );
         IO.println ( "Elemento na posicao (" +
                      x + "," + y + ") igual a " + z );
       } // fim repetir
     } // fim repetir
  } // fim teste2
```

```
/**
  * main - testar matrizes com objetos
  */
  public static void main ( String [ ] args )
  {
  // definir dados
     int opcao = 1;

  // repetir ate' parar
     do
     {
     // oferecer opcoes
        IO.println ( "Teste de matrizes" );
        IO.println ( );
        IO.println ( "Opcoes:" );
        IO.println ( );
        IO.println ( "0. Terminar" );
        IO.println ( "1. Testes de  posicao" );
        IO.println ( "2. Testes com alteracao e
                          consulta" );
        IO.println ( );
        opcao = IO.readint ( "Escolher sua
                                  opcao : " );

     // escolher opcao
        switch ( opcao )
        {
          case 0:
            IO.println ( "Encerrar testes." );
            break;
          case 1:
            teste1 ( );
            break;
          case 2:
            teste2 ( );
            break;
          default:
            IO.println ("ERRO: Opcao invalida.");
        } // fim escolher

     // encerrar
        IO.println ( );
        IO.println ( "Apertar ENTER." );
        IO.pause ( );
     }
     while ( opcao != 0 );
  } // fim main ( )

} // fim class Testar_Matriz
```