PUC-Minas - Ciência da Computação
AED1 – Estudo Dirigido 09

Tema: Introdução à programação IV
Atividade: Grupos de dados homogêneos

01.) Editar e salvar um esboço de programa em Java:

```
/**
  * Exemplo0161
  *
  * @author
  * @version 01
  */

// ---------------------------------------------  dependencias

import IO.*;

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

public class Exemplo0161
{
// ---------------------------------------------  definicao de metodo auxiliar

 /**
  * ler valores inteiros de arquivo e guardar em uma matriz.
  * @return tabela com os valores lidos de arquivo
  * @param nome do arquivo com os dados
  */
   public static int [ ][ ] lerDoArquivo ( String nome )
  {
   // definir dados
     FILE arquivo = new FILE ( FILE.INPUT, nome );
     int  linhas, colunas;
     int  x, y;
      int [ ][ ] tabela = null;
     String linha;

   // identificar
     IO.println ( "Montar matriz com valores lidos de arquivo" );
   // tentar ler uma linha do arquivo
     linha   = arquivo.readln ( );
```

```
  // testar a disponibilidade de dados
     if ( linha == null )
     {
       IO.println ( "ERRO: Nao ha' dados no arquivo." );
     }
     else
     {
      // tentar obter a quantidade de linhas
         linhas  = IO.getint ( linha );
      // tentar obter a quantidade de colunas
         linha     = arquivo.readln ( );
         colunas = IO.getint ( linha );
      // testar se quantidade valida
          if ( linhas <= 0 || colunas <= 0 )
         {
           IO.println ( "ERRO: Tamanho invalido." );
         }
          else
          {
         // reservar espaco para os dados
             tabela = new int [ linhas ] [ colunas ];
         // repetir para cada dado no arquivo
             for ( x = 0; x < linhas; x = x + 1 )
             {
               for ( y = 0; y < colunas; y = y + 1 )
               {
              // ler linha do arquivo
                 linha = arquivo.readln ( );
              // armazenar em um posicao da matriz
              // valor convertido para inteiro
                   tabela [ x ][ y ] = IO.getint ( linha );
               } // fim for
             } // fim for
           } // fim se
      } // fim se
    // fechar arquivo
      arquivo.close ( );
    // retornar matriz montada
       return ( tabela );
    } // fim lerDoArquivo ( )

/**
  * recuperar dados de arquivo.
  */
public static void teste01 ( )
{
 // 1. definir dados
    int [ ] [ ] tabela = null;

 // 2. identificar
    IO.println ( );
    IO.println ( "Recuperar dados de arquivo" );
    IO.println ( );

 // 3. montar dados
    tabela = lerDoArquivo ( "DADOS.TXT" );
```

```
  // 4. testar a existencia de dados
     if ( tabela == null )
     {
       IO.println ( "ERRO: Matriz vazia." );
     }
     else
     {
       if ( tabela.length == 0 )
       {
         IO.println ( "ERRO: Matriz vazia." );
       }
       else
       {
          IO.println ( "Matriz montada com " +
                       tabela.length + "x" +
                       tabela[ 0 ].length + " dados." );
       } // fim se
     } // fim se

  // 5. encerrar
     IO.println ( );
     IO.pause ( "Apertar ENTER para continuar." );
  } // fim teste01 ( )


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

    /**
     * main() – metodo principal
     */
    public static void main ( String [ ] args )
    {
      // identificar
         IO.println ( "EXEMPLO0141 - Programa em Java" );
         IO.println ( "Autor: _____" );
      // executar o metodo auxiliar
         teste01 ( );
      // encerrar
         IO.pause   ( “Apertar ENTER para terminar.” );
    } // fim main( )
} // fim class Exemplo0141
```

OBS.:
O tamanho em linhas é dado como quantidade de grupos,
enquanto em colunas é dado como quantidade de elementos em um grupo.
O arquivo de dados deve ser preparado de acordo com o seguinte formato
(quantidade de linhas, de colunas e dados):

2
2
1
2
3
4

02.) Compilar o programa.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.
Em caso de dúvidas, consultar a apostila, recorrer aos monitores ou apresentá-las ao professor.

03.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.
Em caso de erro (ou dúvida), usar comentários para registrar a ocorrência e,
posteriormente, tentar resolvê-lo (ou esclarecer a dúvida).
04.) Copiar a versão atual do programa para outra nova – Exemplo0162.java.

05.) Editar mudanças no nome do programa e versão.
Acrescentar outro método para mostrar dados em matriz.
Na parte principal, editar a chamada do método para o novo.

```
// ----------------------------------------------  definicao de metodo auxiliar

/**
  * exibir dados em matriz.
  * @param tabela - matriz com dados
  */
  public static void mostrar ( int [ ][ ] tabela )
 {
  // definir dados
    int  linhas, colunas;
    int  x, y;
  // identificar
    IO.println ( );
  // testar se a matriz foi montada
    if ( tabela == null )
    {
      IO.println ( "ERRO: Tabela vazia." );
    }
    else
    {
     // verificar o tamanho da matriz
       linhas    = tabela.length;
       colunas = tabela[0].length;
       if ( linhas <= 0 || colunas <= 0 )
       {
         IO.println ( "ERRO: Tabela vazia." );
       }
       else
       {
        // mostrar matriz
          IO.println ( "Matriz montada com " +
                       linhas + "x" +
                       colunas + " dados." );
```

```
        // repetir para cada dado na matriz
            for ( x = 0; x < linhas; x = x + 1 )
            {
                for ( y = 0; y < colunas; y = y + 1 )
                {
              // mostrar dado em um posicao da matriz
              // convertido para inteiro
                    IO.print ( " " + tabela [ x ][ y ] );
                } // fim for
             // mudar de linha
                IO.println ( );
            } // fim for
        } // fim se
    } // fim se
} // fim mostrar ( )

/**
 * recuperar e mostrar dados de arquivo.
 */
public static void teste02 ( )
{
 // 1. definir dados
    int [ ][ ] tabela = null;

 // 2. identificar
    IO.println ( );
    IO.println ( "Recuperar e mostrar dados de arquivo" );
    IO.println ( );

 // 3. montar dados
    tabela = lerDoArquivo ( "DADOS.TXT" );

 // 4. testar a existencia de dados
    if ( tabela == null )
    {
      IO.println ( "ERRO: Matriz nula." );
    }
    else
    {
      if ( tabela.length == 0 )
      {
        IO.println ( "ERRO: Matriz vazia." );
      }
      else
      {
        mostrar ( tabela );
      } // fim se
    } // fim se

 // 5. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim teste02 ( )
```

OBS.:
Só poderá ser mostrada a matriz que tiver algum conteúdo
(ser diferente de ***null*** = inexistência de dados).

06.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

07.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

08.) Copiar a versão atual do programa para outra nova – Exemplo0163.java.

09.) Editar mudanças no nome do programa e versão.
Acrescentar função para copiar e retornar dados de uma matriz.
Na parte principal, editar a chamada do método para o novo.

```
// ------------------------------------------------  definicao de metodo auxiliar

 /**
  * copiar dados em matriz.
  * @return nova matriz com dados copiados
  * @param tabela - matriz com dados
  */
  public static int [ ][ ] copiar ( int [ ][ ] tabela )
 {
  // definir dados
    int linhas, colunas;
    int x, y;
     int [ ][ ] nova = null;

  // testar existencia de dados
     if ( tabela == null )
    {
      IO.println ( "ERRO: Matriz vazia." );
    }
     else
    {
     // reservar espaco na nova matriz para os dados
        linhas    = tabela.length;
        colunas = tabela[0].length;
        nova = new int [ linhas ][ colunas ];
      // testar o espaco disponivel
        if ( nova == null )
       {
         IO.println ( "ERRO: Nao ha' espaco." );
       }
        else
       {
```

```
        // repetir para cada dado no arquivo
            for ( x = 0; x < linhas; x = x + 1 )
            {
                for ( y = 0; y < colunas; y = y + 1 )
                {
                 // copiar cada posicao da matriz
                    nova [ x ][ y ] = tabela [ x ][ y ];
                } // fim for
            } // fim for
        } // fim se
    } // fim se

  // retornar nova matriz
     return ( nova );
} // fim copiar ( )

/**
  * recuperar e mostrar dados de arquivo.
  */
public static void teste03 ( )
{
 // 1. definir dados
    int [ ][ ] tabela1 = null;
    int [ ][ ] tabela2 = null;

 // 2. identificar
   IO.println ( );
   IO.println ( "Recuperar e mostrar dados de arquivo" );
   IO.println ( );

 // 3. montar dados
    tabela1 = lerDoArquivo ( "DADOS.TXT" );

 // 4. testar a existencia de dados
   if ( tabela1 == null )
   {
     IO.println ( "ERRO: Matriz nula." );
   }
   else
   {
     if ( tabela1.length == 0 )
     {
       IO.println ( "ERRO: Matriz vazia." );
     }
     else
     {
        mostrar ( tabela1 );
        tabela2 = copiar ( tabela1 );
       IO.println ( );
       IO.println ( "Apos copiar:" );
        mostrar ( tabela2 );
     } // fim se
   } // fim se

 // 5. encerrar
   IO.println ( );
   IO.pause ( "Apertar ENTER para continuar." );
} // fim teste03 ( )
```

OBS.:
Se existir dados na matriz original, espaço para igual quantidade deverá ser reservado.

10.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

11.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

12.) Copiar a versão atual do programa para outra nova – Exemplo0164.java.

13.) Editar mudanças no nome do programa e versão.
Acrescentar outra função para somar e retornar dados em matrizes.
Na parte principal, editar a chamada do método para o novo.

```
// ------------------------------------------------  definicao de metodo auxiliar

 /**
  * somar dados em matrizes.
  * @return nova matriz com dados somados
  * @param tabela1   - matriz com dados
  * @param tabela2   - matriz com dados
  */
  public static int [ ][ ] somar
                    ( int [ ][ ] tabela1,
                      int [ ][ ] tabela2 )
{
 // definir dados
    int linhas, colunas;
    int x, y;
    int [ ][ ] nova = null;

 // testar existencia de dados
    if ( tabela1 == null || tabela2 == null )
    {
      IO.println ( "ERRO: Matriz vazia." );
    }
    else
    {
     // testar se tamanhos validos
        if ( tabela1.length == 0 ||
             tabela2.length == 0 ||
             tabela1.length != tabela2.length ||
             tabela1[0].length != tabela2[0].length )
       {
         IO.println ( "ERRO: Tamanho(s) invalido(s)." );
       }
        else
       {
```

```
        // reservar espaco na nova matriz para os dados
            linhas = tabela1.length;
            colunas = tabela1[0].length;
        // reservar espaco para os dados
            nova = new int [ linhas ][ colunas ];
        // testar o espaco disponivel
          if ( nova == null )
          {
            IO.println ( "ERRO: Nao ha' espaco." );
          }
          else
          {
           // repetir para cada dado no arquivo
              for ( x = 0; x < linhas; x = x + 1 )
              {
                 for ( y = 0; y < colunas; y = y + 1 )
                 {
                  // somar dados em cada posicao da matriz
                    nova [ x ][ y ] = tabela1 [ x ][ y ]
                                    + tabela2 [ x ][ y ];
                 } // fim for
              } // fim for
          } // fim se
       } // fim se
    } // fim se

  // retornar nova matriz
     return ( nova );
} // fim somar ( )

/**
  * recuperar, somar e mostrar dados de arquivo.
  */
public static void teste04 ( )
{
 // 1. definir dados
    int [ ][ ] tabela1 = null;
    int [ ][ ] tabela2 = null;
    int [ ][ ] tabela3 = null;

 // 2. identificar
   IO.println ( );
   IO.println ( "Recuperar, somar e mostrar dados de arquivo" );
   IO.println ( );

 // 3. montar dados
    tabela1 = lerDoArquivo ( "DADOS1.TXT" );
    tabela2 = lerDoArquivo ( "DADOS2.TXT" );
```

```
// 4. testar a existencia de dados
   if ( tabela1 == null || tabela2 == null )
   {
    IO.println ( "ERRO: Matriz nula." );
   }
    else
    {
     if ( tabela1.length == 0 ||
          tabela2.length == 0 ||
          tabela1.length != tabela2.length ||
          tabela1[0].length != tabela2[0].length )
    {
     IO.println ( "ERRO: Tamanho invalido." );
    }
     else
    {
     IO.println ( "Somar matrizes: " );
     mostrar ( tabela1 );
     IO.println ( "\n+" );
     mostrar ( tabela2 );
     IO.println ( );
     IO.println ( "Resultado:" );
      tabela3 = somar ( tabela1, tabela2 );
      mostrar ( tabela3 );
    } // fim se
   } // fim se

  // 5. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
 } // fim teste04 ( )
```

OBS.:
Só poderão ser somadas matrizes com mesma quantidade de linhas e colunas.

14.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

15.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

16.) Copiar a versão atual do programa para outra nova – Exemplo0165.java.

17.) Editar mudanças no nome do programa e versão.
Acrescentar outra função para somar e retornar dados em matrizes, escalados por constante.
Na parte principal, editar a chamada do método para o novo.

```
// ------------------------------------------------  definicao de metodo auxiliar

 /**
  * somar dados em matrizes.
  * @return nova matriz com dados somados
  * @param tabela1   - matriz com dados
  * @param constante - constante para escalar dados
  * @param tabela2   - matriz com dados
  */
  public static int [ ][ ] somar
                    ( int [ ][ ] tabela1,
                      int constante,
                      int [ ][ ] tabela2 )
 {
  // definir dados
     int linhas, colunas;
     int x, y;
     int [ ][ ] nova = null;

  // testar existencia de dados
     if ( tabela1 == null || tabela2 == null )
     {
       IO.println ( "ERRO: Matriz vazia." );
     }
     else
     {
      // testar se tamanhos validos
        if ( tabela1.length == 0 ||
             tabela2.length == 0 ||
             tabela1.length != tabela2.length ||
             tabela1[0].length != tabela2[0].length )
        {
          IO.println ( "ERRO: Tamanho(s) invalido(s)." );
        }
        else
        {
         // reservar espaco na nova matriz para os dados
            linhas = tabela1.length;
            colunas = tabela1[0].length;
         // reservar espaco para os dados
            nova = new int [ linhas ][ colunas ];
         // testar o espaco disponivel
            if ( nova == null )
            {
              IO.println ( "ERRO: Nao ha' espaco." );
            }
            else
            {
```

```
            // repetir para cada dado no arquivo
              for ( x = 0; x < linhas; x = x + 1 )
              {
                 for ( y = 0; y < colunas; y = y + 1 )
                 {
                    // somar dados em cada posicao da matriz
                       nova [ x ][ y ] = tabela1 [ x ][ y ]
                                       + constante * tabela2 [ x ][ y ];
                 } // fim for
              } // fim for
           } // fim se
        } // fim se
     } // fim se

  // retornar nova matriz
     return ( nova );
} // fim somar ( )

/**
  * recuperar, somar e mostrar dados de arquivo.
  */
public static void teste05 ( )
{
 // 1. definir dados
    int [ ][ ] tabela1 = null;
    int [ ][ ] tabela2 = null;
    int [ ][ ] tabela3 = null;

 // 2. identificar
    IO.println ( );
    IO.println ( "Recuperar, somar e mostrar dados de arquivo" );
    IO.println ( );

 // 3. montar dados
    tabela1 = lerDoArquivo ( "DADOS1.TXT" );
    tabela2 = lerDoArquivo ( "DADOS2.TXT" );
```

```
// 4. testar a existencia de dados
   if ( tabela1 == null || tabela2 == null )
   {
    IO.println ( "ERRO: Matriz nula." );
   }
    else
    {
     if ( tabela1.length == 0 ||
          tabela2.length == 0 ||
          tabela1.length != tabela2.length ||
          tabela1[0].length != tabela2[0].length )
    {
     IO.println ( "ERRO: Tamanho invalido." );
    }
     else
    {
     IO.println ( "Somar matrizes: " );
     mostrar ( tabela1 );
     IO.println ( "\n+" );
     mostrar ( tabela2 );
     IO.println ( );
     IO.println ( "Resultado:" );
      tabela3 = somar ( tabela1, -1, tabela2 );
      mostrar ( tabela3 );
    } // fim se
   } // fim se

 // 5. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
 } // fim teste05 ( )
```

OBS.:
Só poderão ser comparadas as matrizes com mesma quantidade de linhas e colunas.

18.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

19.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

20.) Copiar a versão atual do programa para outra nova – Exemplo0166.java.

21.) Editar mudanças no nome do programa e versão.
Acrescentar função para comparar duas matrizes.
Na parte principal, editar a chamada do método para o novo.

```
// ---------------------------------------------  definicao de metodo auxiliar

 /**
  * comparar dados em matrizes.
  * @return se matrizes iguais, ou nao
  * @param tabela1 - matriz com dados
  * @param tabela2 - matriz com dados
  */
  public static boolean comparar
                    ( int [ ][ ] tabela1,
                      int [ ][ ] tabela2 )
{
 // definir dados
    boolean resposta = true;
    int linhas, colunas;
    int x, y;

 // testar existencia de dados
    if ( tabela1 == null || tabela2 == null )
    {
      IO.println ( "ERRO: Matriz vazia." );
    }
    else
    {
     // testar se tamanhos validos
        if ( tabela1.length == 0 ||
             tabela2.length == 0 ||
             tabela1.length     != tabela2.length ||
             tabela1[0].length != tabela2[0].length )
       {
         IO.println ( "ERRO: Tamanho(s) invalido(s)." );
       }
        else
       {
        // repetir para cada posicao das matrizes
          linhas = tabela1.length;
          colunas = tabela1[0].length;
           for ( x = 0; x < linhas; x = x + 1 )
           {
              for ( y = 0; y < colunas; y = y + 1 )
              {
               // comparar cada posicao das matrizes
                  resposta = resposta &&
                                ( tabela1 [ x ][ y ] == tabela2 [ x ][ y ] );
              } // fim for
           } // fim for
       } // fim se
    } // fim se

  // retornar nova matriz
     return ( resposta );
} // fim comparar ( )
```

```
/**
 * recuperar e comparar dados de arquivos.
 */
public static void teste06 ( )
{
 // 1. definir dados
    int [ ][ ] tabela1 = null;
    int [ ][ ] tabela2 = null;
   String nome1, nome2;
   boolean resposta;
 // 2. identificar
   IO.println ( );
   IO.println ( "Recuperar e comparar dados de arquivos" );
   IO.println ( );
 // 3. montar dados
   nome1 = IO.readString ( "Qual o nome do primeiro arquivo? " );
   nome2 = IO.readString ( "Qual o nome do segundo  arquivo? " );
    tabela1 = lerDoArquivo ( nome1 );
    tabela2 = lerDoArquivo ( nome2 );
 // 4. testar a existencia de dados
   if ( tabela1 == null || tabela2 == null )
   {
     IO.println ( "ERRO: Matriz nula." );
   }
   else
   {
      if ( tabela1.length == 0 ||
           tabela2.length == 0 ||
           tabela1.length     != tabela2.length ||
           tabela1[0].length != tabela2[0].length )
     {
       IO.println ( "ERRO: Tamanho invalido." );
     }
     else
     {
       IO.println ( "Somar matrizes: " );
       mostrar ( tabela1 );
       IO.println ( "\n+" );
       mostrar ( tabela2 );
       IO.println ( );
       IO.println ( "Resultado:" );
        resposta = comparar ( tabela1, tabela2 );
       if ( resposta )
       {
         IO.println ( "Matrizes iguais." );
       }
       else
       {
         IO.println ( "Matrizes diferentes." );
       } // fim se
     } // fim se
   } // fim se
 // 5. encerrar
   IO.println ( );
   IO.pause ( "Apertar ENTER para continuar." );
} // fim teste06 ( )
```

OBS.:
Só poderão ser comparadas as matrizes com mesma quantidade de linhas e colunas.

22.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

23.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

24.) Copiar a versão atual do programa para outra nova – Exemplo0167.java.

25.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para dizer se uma matriz é nula (possui todos os valores iguais a zero).
Na parte principal, incluir uma chamada para esse método.

```
// ------------------------------------------------  definicao de metodo auxiliar

  /**
   * comparar se dados em matrizes sao nulos.
   * @return se dados na matriz sao iguais a zero, ou nao
   * @param tabela - matriz com dados
   */
   public static boolean eNula
                       ( int [ ][ ] tabela )
   {
  // definir dados
     boolean resposta = true;
     int linhas, colunas;
     int x, y;
```

```
// testar existencia de dados
   if ( tabela == null )
   {
     IO.println ( "ERRO: Matriz vazia." );
   }
   else
   {
    // testar se tamanhos validos
       if ( tabela.length     == 0 ||
            tabela[0].length == 0 )
      {
        IO.println ( "ERRO: Tamanho invalido." );
      }
       else
      {
       // repetir para cada posicao das matrizes
          linhas = tabela.length;
          colunas = tabela[0].length;
           for ( x = 0; x < linhas; x = x + 1 )
           {
              for ( y = 0; y < colunas; y = y + 1 )
              {
               // comparar cada posicao das matrizes
                  resposta = resposta &&
                                ( tabela [ x ][ y ] == 0 );
              } // fim for
           } // fim for
      } // fim se
    } // fim se

  // retornar resultado
      return ( resposta );
} // fim eNula ( )
```

```
/**
 * recuperar e comparar dados de arquivo.
 */
public static void teste07 ( )
{
 // 1. definir dados
    int [ ][ ] tabela = null;
   String nome;
   boolean resposta;

 // 2. identificar
   IO.println ( );
   IO.println ( "Recuperar e comparar dados de arquivos" );
   IO.println ( );

 // 3. montar dados
   nome = IO.readString ( "Qual o nome do arquivo? " );
    tabela = lerDoArquivo ( nome );

 // 4. testar a existencia de dados
    if ( tabela == null )
   {
     IO.println ( "ERRO: Matriz vazia." );
   }
   else
   {
      if ( tabela.length     == 0 ||
           tabela[0].length == 0 )
     {
       IO.println ( "ERRO: Tamanho invalido." );
     }
     else
     {
       IO.println ( "Testar se matriz nula: " );
       mostrar ( tabela );
       IO.println ( );
       IO.println ( "Resultado:" );
        resposta = eNula ( tabela );
       if ( resposta )
       {
         IO.println ( "Matriz nula." );
       }
       else
       {
         IO.println ( "Matriz nao e' nula." );
       } // fim se
     } // fim se
   } // fim se

 // 5. encerrar
   IO.println ( );
   IO.pause ( "Apertar ENTER para continuar." );
} // fim teste07 ( )
```

OBS.:
Só deverá ser verificada a matriz que possuir dados (não ser vazia).

26.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

27.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

28.) Copiar a versão atual do programa para outra nova – Exemplo0168.java.

29.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para dizer se uma matriz é identidade
(todo valor na diagonal principal é igual a 1, e nas outras posições iguais a zero).
Na parte principal, incluir chamadas para esse método.

```
// ----------------------------------------------  definicao de metodo auxiliar

  /**
   * comparar se dados em matrizes sao nulos,
   * exceto na diagonal principal.
   * @return se matriz igual 'a identidade
   * @param tabela - matriz com dados
   */
   public static boolean eIdentidade
                    ( int [ ][ ] tabela )
   {
  // definir dados
     boolean resposta = true;
     int linhas, colunas;
     int x, y;
```

```
// testar existencia de dados
   if ( tabela == null )
   {
     IO.println ( "ERRO: Matriz vazia." );
   }
    else
   {
    // testar se tamanhos validos
       if ( tabela.length     == 0 ||
            tabela[0].length == 0 )
     {
       IO.println ( "ERRO: Tamanho invalido." );
     }
     else
     {
      // repetir para cada posicao das matrizes
        linhas = tabela.length;
        colunas = tabela[0].length;
         for ( x = 0; x < linhas; x = x + 1 )
         {
            for ( y = 0; y < colunas; y = y + 1 )
            {
             // comparar cada posicao das matrizes
               if ( x == y )
               {
                // na diagonal
                  resposta = resposta &&
                                ( tabela [ x ][ y ] == 1 );
               }
               else
               {
                // fora da diagonal
                  resposta = resposta &&
                                ( tabela [ x ][ y ] == 0 );
               } // fim se
            } // fim for
         } // fim for
     } // fim se
   } // fim se

  // retornar resultado
     return ( resposta );
} // fim eIdentidade ( )
```

```
/**
 * recuperar e comparar dados de arquivo.
 */
public static void teste08 ( )
{
 // 1. definir dados
    int [ ][ ] tabela = null;
    String nome;
    boolean resposta;

 // 2. identificar
    IO.println ( );
    IO.println ( "Recuperar e comparar dados de arquivos" );
    IO.println ( );

 // 3. montar dados
    nome = IO.readString ( "Qual o nome do arquivo? " );
     tabela = lerDoArquivo ( nome );

 // 4. testar a existencia de dados
     if ( tabela == null )
    {
      IO.println ( "ERRO: Matriz vazia." );
    }
     else
    {
       if ( tabela.length     == 0 ||
            tabela[0].length == 0 )
      {
       IO.println ( "ERRO: Tamanho invalido." );
      }
       else
      {
       IO.println ( "Testar se matriz identidade: " );
       mostrar ( tabela );
       IO.println ( );
       IO.println ( "Resultado:" );
        resposta = eIdentidade ( tabela );
       if ( resposta )
       {
         IO.println ( "Matriz identidade." );
       }
       else
       {
         IO.println ( "Matriz nao e' identidade." );
       } // fim se
      } // fim se
    } // fim se

 // 5. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim teste08 ( )
```

30.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

31.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

32.) Copiar a versão atual do programa para outra nova – Exemplo0169.java.

33.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para dizer se uma matriz é transposta de outra (tem os mesmos valores nas colunas, ao invés das linhas).
Na parte principal, incluir chamadas para essa função.

```
// ----------------------------------------------  definicao de metodo auxiliar

 /**
  * comparar dados em matrizes.
  * @return se matrizes transpostas, ou nao
  * @param tabela1 - matriz com dados
  * @param tabela2 - matriz com dados
  */
  public static boolean eTransposta
                  ( int [ ][ ] tabela1,
                    int [ ][ ] tabela2 )
{
 // definir dados
    boolean resposta = true;
   int linhas, colunas;
   int x, y;
```

```
// testar existencia de dados
   if ( tabela1 == null || tabela2 == null )
   {
     IO.println ( "ERRO: Matriz vazia." );
   }
    else
   {
    // testar se tamanhos validos
       if ( tabela1.length == 0 ||
            tabela2.length == 0 ||
            tabela1.length     != tabela2[0].length ||
            tabela1[0].length != tabela2.length )
      {
        IO.println ( "ERRO: Tamanho(s) invalido(s)." );
      }
       else
      {
       // repetir para cada posicao das matrizes
          linhas = tabela1.length;
          colunas = tabela1[0].length;
           for ( x = 0; x < linhas; x = x + 1 )
           {
              for ( y = 0; y < colunas; y = y + 1 )
              {
               // comparar cada posicao das matrizes
                  resposta = resposta &&
                                ( tabela1 [ x ][ y ] == tabela2 [ y ][ x ] );
              } // fim for
           } // fim for
      } // fim se
   } // fim se

  // retornar nova matriz
     return ( resposta );
} // fim eTransposta ( )
```

```
/**
 * recuperar e comparar dados de arquivo.
 */
public static void teste09 ( )
{
 // 1. definir dados
    int [ ][ ] tabela1 = null;
    int [ ][ ] tabela2 = null;
   String nome1, nome2;
   boolean resposta;
 // 2. identificar
   IO.println ( );
   IO.println ( "Recuperar e comparar dados de arquivos" );
   IO.println ( );
 // 3. montar dados
   nome1 = IO.readString ( "Qual o nome do primeiro arquivo? " );
   nome2 = IO.readString ( "Qual o nome do segundo  arquivo? " );
    tabela1 = lerDoArquivo ( nome1 );
    tabela2 = lerDoArquivo ( nome2 );
 // 4. testar a existencia de dados
    if ( tabela1 == null || tabela2 == null )
   {
     IO.println ( "ERRO: Matriz nula." );
   }
    else
   {
     if ( tabela1.length == 0 ||
          tabela2.length == 0 ||
          tabela1.length     != tabela2.length ||
          tabela1[0].length != tabela2[0].length )
    {
      IO.println ( "ERRO: Tamanho invalido." );
    }
     else
    {
      IO.println ( "Testar se matriz e' transposta: " );
      mostrar ( tabela1 );
      IO.println ( );
      mostrar ( tabela2 );
      IO.println ( );
      IO.println ( "Resultado:" );
      resposta = eTransposta ( tabela1, tabela2 );
      if ( resposta )
      {
        IO.println ( "Matrizes transpostas." );
      }
      else
      {
        IO.println ( "Matrizes nao sao transpostas." );
      } // fim se
    } // fim se
   } // fim se
 // 5. encerrar
   IO.println ( );
   IO.pause ( "Apertar ENTER para continuar." );
} // fim teste09 ( )
```

OBS.:
Só poderão ser comparadas matrizes com igual quantidade de linhas e colunas.

34.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

35.) Executar o programa. Observar as saídas. Registrar os dados e os resultados.

36.) Copiar a versão atual do programa para outra nova – Exemplo0170.java.

37.) Editar mudanças no nome do programa e versão.
Acrescentar uma função para multiplicar matrizes.
Na parte principal, incluir chamadas para esse método.

```
// ---------------------------------------------  definicao de metodo auxiliar

 /**
  * multiplicar dados em matrizes.
  * @return nova matriz com o produto
  * @param tabela1      - matriz com dados
  * @param tabela2      - matriz com dados
  */
  public static int [ ][ ] multiplicar
                    ( int [ ][ ] tabela1,
                      int [ ][ ] tabela2 )
{
 // definir dados
    int linhas, colunas;
    int x, y, z;
     int soma;
     int [ ][ ] nova = null;
    int linhas1, colunas1,
         linhas2, colunas2;
```

```
// testar existencia de dados
   if ( tabela1 == null || tabela2 == null )
   {
     IO.println ( "ERRO: Matriz vazia." );
   }
    else
   {
    // testar se tamanhos validos
       if ( tabela1.length == 0 ||
            tabela2.length == 0 ||
            tabela1.length     != tabela2.length ||
            tabela1[0].length != tabela2[0].length )
      {
        IO.println ( "ERRO: Tamanho(s) invalido(s)." );
      }
       else
      {
       // reservar espaco na nova matriz para os dados
           linhas1    = tabela1.length;
           colunas1 = tabela1[0].length;
           linhas2    = tabela2.length;
           colunas2 = tabela2[0].length;
           nova = new int [ linhas1 ][ colunas2 ];
       // testar o espaco disponivel
           if ( nova == null )
         {
           IO.println ( "ERRO: Nao ha' espaco." );
         }
         else
         {
          // repetir para cada dado no arquivo
              for ( x = 0; x < linhas1; x = x + 1 )
              {
                 for ( y = 0; y < colunas2; y = y + 1 )
                 {
                    soma = 0;
                    for ( z = 0; z < colunas1; z = z + 1 )
                    {
                     // acumular o produto cada posicao da matriz
                        soma = soma +
                                  tabela1 [ x ][ z ] * tabela2 [ z ][ y ];
                    } // fim for
                    nova [ x ][ y ] = soma;
                 } // fim for
            } // fim for
        } // fim se
      } // fim se
    } // fim se

  // retornar nova matriz
     return ( nova );
} // fim multiplicar ( )
```

```
/**
 * recuperar, multiplicar e mostrar dados de arquivos.
 */
public static void teste10 ( )
{
 // 1. definir dados
    int [ ][ ] tabela1 = null;
    int [ ][ ] tabela2 = null;
    int [ ][ ] tabela3 = null;

 // 2. identificar
   IO.println ( );
   IO.println ( "Recuperar, multiplicar e mostrar dados de arquivos" );
   IO.println ( );

 // 3. montar dados
    tabela1 = lerDoArquivo ( "DADOS1.TXT" );
    tabela2 = lerDoArquivo ( "DADOS2.TXT" );

 // 4. testar a existencia de dados
    if ( tabela1 == null || tabela2 == null )
   {
     IO.println ( "ERRO: Matriz nula." );
   }
    else
   {
      if ( tabela1.length == 0 ||
           tabela2.length == 0 ||
           tabela1[0].length != tabela2.length )
     {
       IO.println ( "ERRO: Tamanho invalido." );
     }
      else
     {
       IO.println ( "Multiplicar matrizes: " );
       mostrar ( tabela1 );
       IO.println ( "\n+" );
       mostrar ( tabela2 );
       IO.println ( );
       IO.println ( "Resultado:" );
        tabela3 = multiplicar ( tabela1, tabela2 );
        mostrar ( tabela3 );
     } // fim se
   } // fim se

 // 5. encerrar
   IO.println ( );
   IO.pause ( "Apertar ENTER para continuar." );
} // fim teste10 ( )
```

38.) Compilar o programa novamente.
Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

39.) Executar o programa.
Observar as saídas.
Registrar os dados e os resultados.

Exercícios:
DICAS GERAIS: Consultar o Anexo Java 02 na apostila para outros exemplos.
Prever, realizar e registrar todos os testes efetuados.

01.) Fazer um programa (Exemplo0171) com função
para ler as dimensões (quantidade de linhas e colunas) de uma matriz real do teclado,
bem como todos os seus elementos (apenas valores positivos).
Verificar se as dimensões não são nulas ou negativas.
Para testar, ler dados e mostrá-los na tela por outro método.
DICA: Supor que as dimensões da matriz não passarão de 10.

02.) Fazer um programa (Exemplo0172) com método
para gravar uma matriz real em arquivo.
A matriz e o nome do arquivo serão dados como parâmetros.
Para testar, usar a matriz do problema anterior.
Usar outra função para ler e recuperar a matriz do arquivo, antes de mostrá-la na tela.
DICA: Supor que as dimensões da matriz não passarão de 10.

03.) Fazer um programa (Exemplo0173) com método
para mostrar somente os valores na diagonal principal de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e são iguais.

04.) Fazer um programa (Exemplo0174) com método
para mostrar somente os valores na diagonal secundária de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e são iguais.

05.) Fazer um programa (Exemplo0175) com método
para mostrar somente os valores abaixo da diagonal principal de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e serão iguais.

06.) Fazer um programa (Exemplo0176) com método
para mostrar somente os valores acima da diagonal principal de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e serão iguais.

07.) Fazer um programa (Exemplo0177) com método
para mostrar somente os valores abaixo da diagonal secundária de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e serão iguais.

08.) Fazer um programa (Exemplo0178) com método
para mostrar somente os valores acima da diagonal secundária de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e serão iguais.

09.) Fazer um programa (Exemplo0179) com função
para testar se são todos nulos os valores abaixo da diagonal principal de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e serão iguais.

10.) Fazer um programa (Exemplo0180) com função
para testar se não são nulos os valores acima da diagonal principal de uma matriz real quadrada.
DICA: Supor que as dimensões da matriz não passarão de 10, e serão iguais.

Tarefas extras

E1.) Fazer um programa para ler do teclado as quantidades de linhas e colunas de uma matriz,
e mediante uma função montar e retornar uma matriz com a característica abaixo,
a qual deverá ser gravada em arquivo, após o retorno

					1	2	3	4
		1	2	3	5	6	7	8
1	2	4	5	6	9	10	11	12
3	4	7	8	9	13	14	15	16

E2.) Fazer um programa para ler do teclado as quantidades de linhas e colunas de uma matriz,
e mediante uma função montar e retornar uma matriz com a característica abaixo,
a qual deverá ser gravada em arquivo, após o retorno

					1	5	9	13
		1	4	7	2	6	10	14
1	3	2	5	8	3	7	11	15
2	4	3	6	9	4	8	12	16