

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

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

```
/**
 * Exemplo0201
 *
 * @author
 * @version 01
 */

// ----- dependencias

import IO.*;

// ----- definicao de classe auxiliar

/**
 * Classe para tratar matriz de objetos.
 */
class Matriz
{
    /**
     * armazenador generico dos dados
     */
    public Object [ ][ ] tabela;

    /**
     * construtor padrao
     */
    public Matriz ( )
    {
        tabela = null;
    } // fim construtor padrao
} // fim da classe Matriz
```

```

// ----- definicao da classe principal

public class Exemplo0201
{
// ----- definicao de metodo auxiliar

/**
 * Testar definições de matriz usando classe.
 */
public static void metodo01 ( )
{
// 1. definir dados
    Matriz a1 = null;           // nao existe objeto
    Matriz a2 = new Matriz ( ); // existe objeto (sem dados, no momento)
// 2. identificar
    IO.println ( "Definicoes de matriz" );
// 3. testar as definicoes de matriz
    if ( a1 == null )
    {
        IO.println ( "Matriz a1 nula (inexistente)" );
    }
    else
    {
        IO.println ( " Matriz a1 nao nula (existente)" );
    } // fim se
    if ( a2 == null )
    {
        IO.println ( " Matriz a2 nula (inexistente)" );
    }
    else
    {
        IO.println ( " Matriz a2 nao nula (existente)" );
    } // fim se
// 5. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo01( )

// ----- definicao do metodo principal

/**
 * main() – metodo principal
 */
public static void main ( String [ ] args )
{
// identificar
    IO.println ( "EXEMPLO0201 - Programa em Java" );
    IO.println ( "Autor: _____" );
// executar o metodo auxiliar
    metodo01 ( );
// encerrar
    IO.pause ( "Apertar ENTER para terminar." );
} // fim main( )
} // fim class Exemplo0201

```

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 – Exemplo0202.java.

05.) Editar mudanças no nome do programa e versão.

Acrescentar novos métodos à classe para lidar com a quantidade de elementos.

```
/**
 * construtor alternativo.
 */
public Matriz ( int linhas, int colunas )
{
    if ( linhas <= 0 || colunas <= 0 )
    {
        IO.println ( "ERRO: quantidade invalida." );
    }
    else
    {
        tabela = new Object [ linhas ][ colunas ];
    } // fim se
} // fim construtor alternativo

/**
 * informar a quantidade de posicoes reservadas (linhas).
 */
public int lines ( )
{
    int linhas = 0;

    if ( tabela != null )
    {
        linhas = tabela.length;
    }
    return ( linhas );
} // fim lines ( )
```

```

/**
 * informar a quantidade de posicoes reservadas (colunas).
 */
public int columns ( )
{
    int colunas = 0;

    if ( tabela != null )
    {
        colunas = tabela[0].length;
    }
    return ( colunas );
} // fim columns ( )

```

Na parte principal, criar um segundo método para testes.

```

// 1. definir dados
Matriz a1 = null;
Matriz a2 = new Matriz ( );
Matriz a3 = new Matriz ( 3, 3 );

// 3. testar as definicoes de matriz
if ( a3 == null )
{
    IO.println ( " Matriz a3 nula" );
}
else
{
    IO.println ( " Matriz a3 nao nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
} // fim se

```

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 – Exemplo0203.java.

09.) Editar mudanças no nome do programa e versão.

Acrescentar método à classe para mostrar o conteúdo de matriz na tela.

```
/**
 * exibir dados em matriz.
 */
public void printMatrix ( )
{
    // definir dados
    int x, y,
        linhas, colunas;
    // identificar
    IO.println ( );
    // testar se a matriz foi montada
    if ( tabela == null )
    {
        IO.println ( "ERRO: Matriz vazia." );
    }
    else
    {
        // obter dimensoes da matriz
        linhas = lines( );
        colunas = columns( );
        // mostrar matriz
        IO.println ( "Matriz com "+linhas+"x"+colunas+" posicoes:" );
        // repetir para cada posicao na matriz
        for ( x = 0; x < linhas; x = x + 1 )
        {
            for ( y = 0; y < colunas; y = y + 1 )
            {
                // mostrar o valor armazenado
                IO.print ( "\t"+matriz [ x ][ y ] );
            } // fim repetir
            IO.println ( );
        } // fim repetir
    } // fim se
} // fim printMatrix ( )
```

Na parte principal, acrescentar um terceiro método para testar a exibição de dados em matriz.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo03 ( )
{
    // 1. definir dados
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    if ( a3 == null )
    {
        IO.println ( "Matriz a3 nula" );
    }
    else
    {
        // mostrar informacoes sobre matriz
        IO.println ( "Matriz a3 nao nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
        // mostrar dados no matriz
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a3.printMatrix ( );
    } // fim se
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo03( )
```

OBS.:

Observar que todos os dados serão nulos, pois esse é o valor padrão para objetos vazios.

- 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 – Exemplo0204.java.
- 13.) Editar mudanças no nome do programa e versão.
Acrescentar à classe um método para ler valores do teclado e guardá-los em matriz.

```

/**
 * ler valores do teclado e guardar em uma matriz.
 * @param message - com texto a ser mostrado na tela
 */
public void readMatrix ( String message )
{
    // definir dados
    int  x, y,
        linhas = lines( ),
        colunas = columns( );
    String linha;

    // testar se quantidade valida
    if ( linhas <= 0 || colunas <= 0 )
    {
        IO.println ( "ERRO: Tamanho invalido." );
    }
    else
    {
        // mostrar mensagem antes de ler dados
        IO.println ( message );
        // obter o tamanho da matriz
        linhas  = this.lines ( );
        colunas = this.columns ( );
        // repetir para cada posicao na matriz
        for ( x = 0; x < linhas; x = x + 1 )
        {
            for ( y = 0; y < colunas; y = y + 1 )
            {
                // ler linha do teclado
                linha = IO.readLine ( );
                // armazenar em um posicao da matriz
                // como objeto em String
                tabela [ x ][ y ] = linha;
            } // fim repetir
        } // fim repetir

    } // fim se
} // fim readMatrix ( )

```

Na parte principal, acrescentar um método para testar ler e exibir de dados em matriz.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo04 ( )
{
    // 1. definir dados
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    if ( a3 == null )
    {
        IO.println ( "Matriz a3 nula" );
    }
    else
    {
        // mostrar informacoes sobre matriz
        IO.println ( "Matriz a3 nao nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
        // ler dados e guardar no matriz
        a3.readMatrix ( "Entrar com dados na matriz:" );
        // mostrar dados na matriz
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a3.printMatrix ( );
    } // fim se
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo04( )
```

OBS.:

Todas as posições serão lidas e mostradas.

- 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 – Exemplo0205.java.

17.) Editar mudanças no nome do programa e versão.

Acrescentar outro método para ler certa quantidade de dados e guardar em matriz.

```
/**
 * ler valores inteiros e guardar em uma tabela.
 * @param message - com texto a ser mostrado na tela
 * @param n - quantidade de dados a serem lidos
 */
public void readIntMatrix ( String message, int m, int n )
{
    // definir dados
    int x, y,
        linhas = lines( ),
        colunas = columns( );
    String linha;

    // testar se quantidade valida
    if ( linhas <= 0 || colunas <= 0 ||
        m <= 0 || m > linhas ||
        n <= 0 || n > colunas )
    {
        IO.println ( "ERRO: Quantidade invalida." );
    }
    else
    {
        // mostrar mensagem antes de ler dados
        IO.println ( message );
        // repetir para cada posicao na matriz
        for ( x = 0; x < m; x = x + 1 )
        {
            for ( y = 0; y < n; y = y + 1 )
            {
                // ler linha do teclado
                linha = IO.readLine ( );
                // armazenar em uma posicao da tabela
                // valor convertido para inteiro
                tabela [ x ][ y ] = IO.getInt ( linha );
            } // fim repetir
        } // fim repetir
    } // fim se
} // fim readIntMatrix ( )
```

Na parte principal, acrescentar um método para testar ler e exibir de dados em matriz.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo05 ( )
{
    // 1. definir dados
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    if ( a3 == null )
    {
        IO.println ( "Matriz a3 nula" );
    }
    else
    {
        // mostrar informacoes sobre matriz
        IO.println ( "Matriz a3 nao nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
        // ler dados e guardar no matriz
        a3.readIntMatrix ( "Entrar com dados no matriz:", 2, 2 );
        // mostrar dados na matriz
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a3.printMatrix ( );
    } // fim se
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo05 ( )
```

OBS.:

Notar que a última posição será zero, porque seu valor original não foi alterado.

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 – Exemplo0206.java.

21.) Editar mudanças no nome do programa e versão.

Acrescentar outro método para mostrar apenas certa quantidade de dados guardada em matriz.

```
/**
 * exibir certa quantidade de dados em tabela.
 * @param n - quantidade de dados a serem mostrados
 */
public void printIntMatrix ( int m, int n )
{
    // definir dados
    int x, y,
        linhas = lines( ),
        colunas = columns( ) ;
    String linha;

    // testar se quantidade valida
    if ( linhas <= 0 || colunas <= 0 ||
        m <= 0 || m > linhas ||
        n <= 0 || n > colunas )
    {
        IO.println ( "ERRO: Tabela vazia ou quantidade invalida." );
    }
    else
    {
        // mostrar tabela
        IO.println ( "Tabela com "+linhas+"x"+colunas+" posicoes:" );
        // repetir para cada posicao na tabela
        for ( x = 0; x < m; x = x + 1 )
        {
            for ( y = 0; y < n; y = y + 1 )
            {
                // mostrar o valor armazenado
                IO.print ( "\t" + (int) tabela [ x ][ y ] );
            } // fim repetir
            IO.println ( );
        } // fim repetir
    } // fim se
} // fim printIntMatrix ( )
```

Na parte principal, acrescentar um método para testar ler e exibir de dados em matriz.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo06 ( )
{
    // 1. definir dados
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    if ( a3 == null )
    {
        IO.println ( "Matriz a3 nula" );
    }
    else
    {
        // mostrar informacoes sobre matriz
        IO.println ( "Matriz a3 nao nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
        // ler dados e guardar na matriz
        a3.readIntMatrix ( "Entrar com dados no matriz:", 2, 2 );
        // mostrar dados na matriz
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a3.printIntMatrix ( );
    } // fim se
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo06( )
```

OBS.: Notar que haverá posições não preenchidas.

- 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 – Exemplo0207.java.

- 25.) Editar mudanças no nome do programa e versão.
Acrescentar um método para testar que a atribuição tornará matrizes idênticas.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo07 ( )
{
    // 1. definir dados
    Matriz a2 = null;
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    if ( a3 == null )
    {
        IO.println ( "Matriz a3 nula" );
    }
    else
    {
        // mostrar informacoes sobre matriz
        IO.println ( "Matriz a3 nao nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
        // ler dados e guardar na matriz
        a3.readIntArray ( "Entrar com dados na matriz:", 2, 2 );
        // tornar matrizes idênticas
        a2 = a3;
        // mostrar dados no matriz
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a2.printMatrix ( 2, 2 );
        // ler dados e guardar na matriz original
        a3.readIntMatrix ( "Entrar com dados na matriz:", 3, 3 );
        // mostrar dados na matriz original
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a3.printMatrix ( );
        // mostrar dados na matriz copiada
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a2.printMatrix ( );
    } // fim se
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo07( )
```

OBS.:

Notar que por serem idênticos, a alteração feita em um afetará o outro. Não são cópias !

- 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 – Exemplo0208.java.

29.) Editar mudanças no nome do programa e versão.

Acrescentar um método para copiar dados de um matriz para outra.

```
/**
 * copiar tabela.
 * @return nova tabela com dados copiados
 */
public Matriz clone ( )
{
    // definir dados
    int x,y,
        linhas, colunas;
    Matriz nova = null;

    // testar existencia de dados
    if ( tabela == null )
    {
        IO.println ( "ERRO: Tabela vazia." );
    }
    else
    {
        // obter as dimensoes da tabela original
        linhas = lines( );
        colunas = columns( );
        // reservar espaco para a nova tabela
        nova = new Matriz ( linhas, colunas );
        // testar a existencia de dados
        if ( nova == null )
        {
            IO.println ( "ERRO: Nao ha' espaco." );
        }
        else
        {
            // repetir para cada posicao na tabela original
            for ( x = 0; x < nova.lines( ); x = x + 1 )
            {
                for ( y = 0; y < nova.columns( ); y = y + 1 )
                {
                    // copiar dado de uma posicao
                    nova.tabela [ x ][ y ] = tabela [ x ][ y ];
                } // fim repetir
            } // fim repetir
        } // fim se
    } // fim se

    // retornar nova tabela
    return ( nova );
} // fim clone ( )
```

Na parte principal, acrescentar um método para testar ler e exibir de dados em matrizes.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo08 ( )
{
    // 1. definir dados
    Matriz a2 = null;
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    if ( a3 == null )
    {
        IO.println ( "Matriz a3 nula" );
    }
    else
    {
        // mostrar informacoes sobre matriz
        IO.println ( "Matriz a3 não nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
        // ler dados e guardar na matriz
        a3.readIntMatrix ( "Entrar com dados na matriz:", 2, 2 );
        // tornar matrizes idênticas
        a2 = a3.clone ( );
        // mostrar dados na matriz
        IO.println ( "Mostrar dados copiados:" );
        a2.printIntMatrix ( 2, 2 );
        // ler dados e guardar na matriz
        a2.readIntMatrix ( "Entrar com dados na matriz:", 3, 3 );
        // mostrar dados na matriz original
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a3.printMatrix ( );
        // mostrar dados na outra matriz
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a2.printMatrix ( );
    } // fim se
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo08( )
```

OBS.:

Notar que a clonagem de dados preservará a individualidade de cada matriz.

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 – Exemplo0209.java.

- 33.) Editar mudanças no nome do programa e versão.
Acrescentar um método para copiar certa quantidade de dados em matriz.

```
/**
 * copiar certa quantidade de dados em tabela.
 * @return nova tabela com dados copiados
 * @param n - quantidade de dados
 */
public Matriz copyMatrix ( int m, int n )
{
    // definir dados
    int x, y,
        linhas = lines( ),
        colunas = columns( );
    Matriz nova = null;

    // testar existencia de dados
    if ( tabela == null ||
        m <= 0 || m > linhas ||
        n <= 0 || n > colunas )
    {
        IO.println ( "ERRO: Tabela vazia." );
    }
    else
    {
        // reservar espaco para a nova tabela
        nova = new Matriz ( m, n );
        // testar a existencia de dados
        if ( nova == null )
        {
            IO.println ( "ERRO: Nao ha' espaco." );
        }
        else
        {
            // repetir para cada posicao na tabela original
            for ( x = 0; x < nova.lines( ); x = x + 1 )
            {
                for ( y = 0; y < nova.columns( ); y = y + 1 )
                {
                    // copiar dado de uma posicao
                    nova.tabela [ x ][ y ] = tabela [ x ][ y ];
                } // fim repetir
            } // fim repetir
        } // fim se
    } // fim se

    // retornar nova tabela
    return ( nova );
} // fim copyMatrix ( )
```


Na parte principal, acrescentar um método para testar ler e exibir de dados em matrizes.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo09 ( )
{
    // 1. definir dados
    Matriz a2 = null;
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    if ( a3 == null )
    {
        IO.println ( "Matriz a3 nula" );
    }
    else
    {
        // mostrar informacoes sobre matriz
        IO.println ( "Matriz a3 nao nula com "+a3.lines( )+"x"+a3.columns( )+" posicoes." );
        // ler dados e guardar na matriz
        a3.readIntMatrix ( "Entrar com dados na matriz:", 3 );
        // tornar matrizes identicas
        a2 = a3.copyMatrix ( 2, 2 );
        // mostrar dados na matriz copiada
        IO.println ( "Mostrar dados copiados:" );
        a2.printIntMatrix ( );
        // mostrar dados na matriz original
        IO.println ( "Mostrar dados lidos e armazenados:" );
        a3.printMatrix ( );
    } // fim se
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo09( )
```

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 – Exemplo0210.java.

37.) Editar mudanças no nome do programa e versão.

Acrescentar um método à classe para exibição automática de todos os dados em matriz.

```
/**
 * exibir dados em matriz automaticamente.
 */
public String toString ( )
{
    // definir dados
    String msg = null;
    int x, y,
        linhas,
        colunas;
    // testar se a tabela foi montada
    if ( tabela != null )
    {
        // obter dimensoes da matriz
        linhas = lines( );
        colunas = columns( );
        // repetir para cada posicao na tabela
        msg = "";
        for ( x = 0; x < linhas; x = x + 1 )
        {
            for ( y = 0; y < colunas; y = y + 1 )
            {
                // guardar valor armazenado
                msg = msg + "\t" + tabela [ x ][ y ];
            } // fim repetir
            msg = msg + "\n";
        } // fim repetir
    } // fim se
    // retornar valores armazenados
    return ( msg );
} // fim toString ( )
```

Na parte principal, acrescentar um método para testar ler e exibir de dados em matrizes.

```
/**
 * Testar entrada e saída de dados em matriz usando classe.
 */
public static void metodo10 ( )
{
    // 1. definir dados
    Matriz a2 = null;
    Matriz a3 = new Matriz ( 3, 3 );
    // 2. identificar
    IO.println ( );
    IO.println ( "Entrada e saída em matriz" );
    IO.println ( );
    // 3. testar entrada e saída em matriz
    // ler dados e guardar no matriz
    a3.readIntMatrix ( "Entrar com dados na matriz:" );
    // tornar matrizes idênticas
    a2 = a3.copyMatrix( );
    // mostrar dados na matriz copiada
    IO.println ( "Mostrar dados copiados:" );
    IO.println ( ""+a2 );
    // mostrar dados na matriz original
    IO.println ( "Mostrar dados lidos e armazenados:" );
    IO.println ( ""+a3 );
    // 4. encerrar
    IO.println ( );
    IO.pause ( "Apertar ENTER para continuar." );
} // fim metodo10( )
```

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 (Exemplo0211) para acrescentar um método à classe Matriz para ler certa quantidade de dados do teclado e armazenar na matriz a partir de determinada posição.
Incluir um método para testar essa nova característica.
DICA: Testar se as posições iniciais e as quantidades são válidas.

Exemplo: `matriz.readIntMatrix ("Matriz 1:", linhaInicial, colunaInicial, quantidadeNaLinha, quantidadeNaColuna);`

- 02.) Fazer um programa (Exemplo0212) para acrescentar um método à classe Matriz para mostrar certa quantidade de dados armazenados na matriz a partir de determinada posição.
Incluir um método para testar essa nova característica.
DICA: Testar se as posições iniciais e as quantidades são válidas.

Exemplo: `matriz.printIntMatrix ("Matriz 1:", linhaInicial, colunaInicial, quantidadeNaLinha, quantidadeNaColuna);`

- 03.) Fazer um programa (Exemplo0213) para acrescentar um método à classe Matriz para ler dados de arquivo, dado o nome do mesmo, e armazenar em matriz.
Incluir um método para testar essa nova característica.
DICA: Ler o tamanho também do arquivo e reservar o tamanho de acordo.

Exemplo: `matriz.fromFile ("Arquivo1.txt");`

- 04.) Fazer um programa (Exemplo0214) para acrescentar um método à classe Matriz para gravar dados de matriz em arquivo, dado o nome do mesmo.
Incluir um método para testar essa nova característica.
DICA: Gravar o tamanho também do arquivo, primeiro, antes dos outros dados.

Exemplo: `matriz.toFile ("Arquivo2.txt");`

- 05.) Fazer um programa (Exemplo0215) para acrescentar um método à classe Matriz para ler certa quantidade de dados de arquivo, dado o nome do mesmo, e armazenar em matriz.
Incluir um método para testar essa nova característica.
DICA: Ler o tamanho também do arquivo e reservar o tamanho de acordo, apenas se as quantidades forem válidas.

Exemplo: `matriz.fromFile ("Arquivo1.txt", quantidadeDeLinhas, quantidadeDeColunas);`

- 06.) Fazer um programa (Exemplo0216) para acrescentar um método à classe Matriz para gravar em arquivo, dado o nome do mesmo, certa quantidade de dados em matriz. Incluir um método para testar essa nova característica.

DICA: Gravar o tamanho também do arquivo, primeiro, antes dos outros dados; e apenas se as quantidades forem válidas.

Exemplo: `matriz.toFile ("Arquivo3.txt", quantidadeDeLinhas, quantidadeDeColunas);`

- 07.) Fazer um programa (Exemplo0217) para acrescentar um método à classe Matriz para ler certa quantidade de dados de arquivo, dado o nome do mesmo, e armazenar em matriz, a partir de determinada posição desta. Incluir um método para testar essa nova característica.

DICA: Ler o tamanho também do arquivo e reservar o tamanho de acordo, apenas se as posições iniciais e as quantidades forem válidas.

Exemplo: `matriz.fromFile ("Arquivo1.txt", linhaInicial, colunaInicial, quantidadeNaLinha, quantidadeNaColuna);`

- 08.) Fazer um programa (Exemplo0218) para acrescentar um método à classe Matriz para gravar certa quantidade de dados de matriz em arquivo, dado o nome do mesmo, a partir de determinada posição desta.

Incluir um método para testar essa nova característica.

DICA: Gravar o tamanho também do arquivo, primeiro, antes dos outros dados; e apenas se as quantidades forem válidas.

Exemplo: `matriz.toFile ("Arquivo4.txt", linhaInicial, colunaInicial, quantidadeNaLinha, quantidadeNaColuna);`

- 09.) Fazer um programa (Exemplo0219) para acrescentar uma função à classe Matriz para copiar certa quantidade de dados armazenados em matriz a partir de determinada posição.

Incluir um método para testar essa nova característica.

DICA: Testar se as posições iniciais e as quantidades são válidas.

Exemplo: `matriz2 = matriz1.copyMatrix (linhaInicial, colunaInicial, quantidadeNaLinha, quantidadeNaColuna);`

- 10.) Fazer um programa (Exemplo0220) para acrescentar uma função à classe Matriz para testar a igualdade de matrizes.

Incluir um método para testar essa nova característica.

DICA: Testar se dimensões e dados são válidos.

Exemplo: `boolean resposta = matriz1.equals (matriz2);`

Tarefas extras

E1.) Fazer um programa para acrescentar uma função à classe Matriz

para testar se é triangular inferior:

algum valor não-nulo abaixo da diagonal principal,

e todos os elementos acima da diagonal principal iguais a zero.

Mostrar a resposta após o retorno.

DICA: Testar, primeiro, se é uma matriz quadrada.

Supor que todos os seus valores poderão ser convertidos para reais.

Exemplo: `boolean resposta = matriz1.lt ();`

E2.) Fazer um programa para acrescentar uma função à classe Matriz

para testar se é triangular superior:

algum valor não-nulo acima da diagonal principal,

e todos os elementos abaixo da diagonal principal iguais a zero.

Mostrar a resposta após o retorno.

DICA: Testar, primeiro, se é uma matriz quadrada.

Supor que todos os seus valores poderão ser convertidos para reais.

Exemplo: `boolean resposta = matriz1.ut ();`