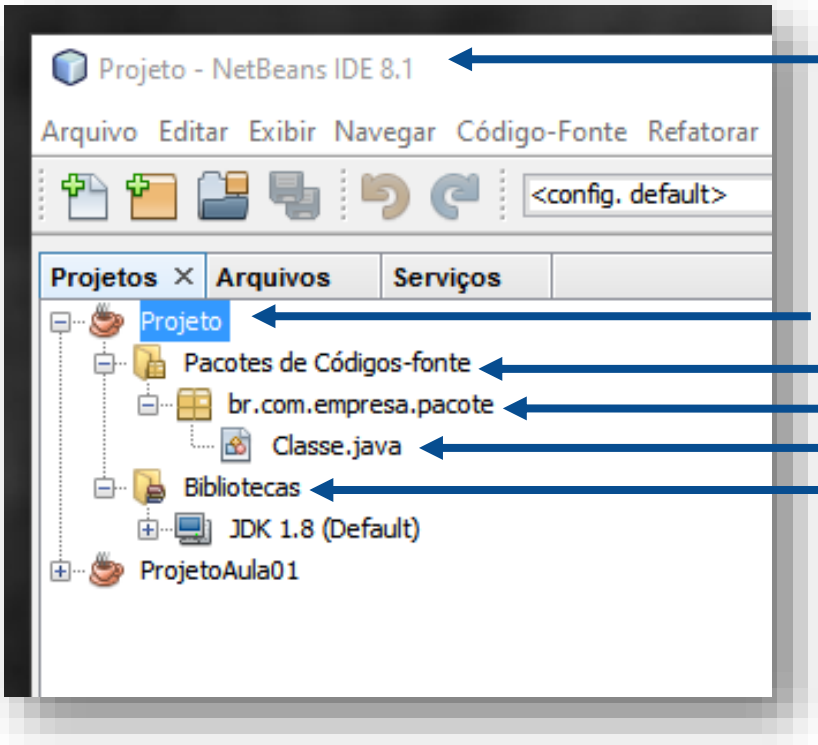
PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

JAVA – VISÃO GERAL

PROJETO

Organização básica de um projeto



IDE Netbeans

Projeto Java

Localização principal das classes e recursos da aplicação

Localização principal das classes e recursos da aplicação

Classe Java

Bibliotecas auxiliares

Funções de um projeto

- Delimitar pacotes e classes referentes a um grupo lógico de funcionalidades;
- Agrupar todos (ou pelo menos a maior parte dos) elementos necessários para geração de um executável de aplicação;
- Permitir a montagem rápida de um ambiente de manutenção ou evolução da aplicação;
- Facilitar o gerenciamento de código por uma IDE.

PACOTES

O que são pacotes?

- Agrupam elementos referentes a uma mesma funcionalidade ou outra característica;
- Facilitam a localização e a referência de classes;
- Organizam o projeto de forma lógica e coesa;

Exemplos de pacotes

```java
/* Declaração de um pacote */
package br.com.empresa.pacote;

/**
 *
 * @author Fabio
 */
public class Classe {

}
```

```java
/*
 * Pacote de classes de modelo
 */
package br.com.empresa.modelo;

/**
 *
 * @author Fabio
 */
public class Cliente {

}
```

```java
/*
 * Armazena as classes de controle de visão
 */
package br.com.empresa.view;

/**
 *
 * @author Fabio
 */
public class ProductViewController {

}
```

```java
/*
 * Armazena classes utilitáris
 */
package br.com.empresa.util;

/**
 *
 * @author Fabio
 */
public class ConversionUtils {

}
```

CLASSES

Separando funcionalidades em classes

- Classes permitem agrupar funcionalidades lógicas e coesas num único elemento;
- São primariamente constituídas de variáveis e métodos;
- São agrupadas em pacotes e podem fazer referências a outras classes;
- Representam aspectos fundamentais de orientação a objetos.

Exemplos de classes

```java
/* Declaração de um pacote */
package br.com.empresa.pacote;

/**
 *
 * @author Fabio
 */
public class Classe {

}
```

```java
/*
 * Pacote de classes de modelo
 */
package br.com.empresa.modelo;

/**
 *
 * @author Fabio
 */
public class Cliente {

}
```

```java
/*
 * Armazena as classes de controle de visão
 */
package br.com.empresa.view;

/**
 *
 * @author Fabio
 */
public class ProductViewController

}
```

```java
/*
 * Armazena classes utilitáris
 */
package br.com.empresa.util;

/**
 *
 * @author Fabio
 */
public class ConversionUtils

}
```

Estrutura de uma classe

```
/*
 * Pacote de classes de modelo
 */
package br.com.empresa.modelo;

/**
 * Classe de cliente
 */
public class Cliente {
    static String primeiroNome = "João";
    static String sobrenome = "da Silva";

    public static String obterNomeCompleto(){
        return primeiroNome + " " + sobrenome;
    }
}
```

Pacote da classe

Modificador de acesso. Torna a classe visível para todas as demais classes.

Declara uma classe

Nome da classe

Variáveis da classe

Métodos da classe

Modificadores de acesso

- “public” é um modificador de acesso em Java;
- Utilizamos modificadores de acesso para controlar a visibilidade de um membro (como um atributo ou método) ou da própria classe para as demais classes
- Utilizamos o modificador “public” para tornar uma classe ou um membro visível para as demais classes do projeto

Outros modificadores de acesso

- “private” – apenas os demais membros da própria classe podem ver o elemento marcado como “private”
- “protected” – o membro fica disponível apenas para demais classes do mesmo pacote ou para classes (independente do pacote) que usem herança para estender da classe onde o membro se encontra

Outros modificadores de acesso

- “padrão” – caso não seja definido um modificador, o membro da classe assume o padrão de acesso (“default”), estando disponível apenas para a classe que contém o membro ou outras classes do mesmo pacote da classe onde o membro se encontra.

Outros modificadores

- Além dos modificadores de acesso, Java também possui outros modificadores não relacionados a acessibilidade de membros
- O modificador “static” é um deles
- Utilizamos o modificador “static” para tornar possível o uso de membros de classes sem a necessidade de criação de objetos destas

Outros modificadores

- Ou seja, num cenário onde temos a seguinte classe:

```
public class MinhaClasse1 {

    public static void metodoEstatico() {
        //Faz alguma coisa
    }

    public void metodoNaoEstatico() {
        //Faz outra coisa
    }
}
```

- O método “metodoEstatico” pode ser acessado diretamente, apenas acionando o nome da classe + ponto + o método (“MinhaClasse1.metodoEstatico()”)

Outros modificadores

```
public class MinhaClasse1 {

    public static void metodoEstatico() {
        //Faz alguma coisa
    }

    public void metodoNaoEstatico() {
        //Faz outra coisa
    }
}
```

- Já o método “metodoNaoEstatico” só pode ser acessado em uma instância da “MinhaClasse1”. Ou seja, é necessário instanciar uma variável do tipo “MinhaClasse1” (dando “new” na classe, por exemplo) e só aí será possível acionar o método “metodoNaoEstatico”.

Outros modificadores

```
public static void main(String[] args) {
    //Acionando um método estático
    MinhaClasse1.metodoEstatico();

    //Acionando um método não-estático
    //ERRADO - NÃO COMPILA
    MinhaClasse1.metodoNaoEstatico();
    //CERTO
    MinhaClasse1 minhaClasse1 = new MinhaClasse1();
    minhaClasse1.metodoNaoEstatico();
}
```

Classe executável

- Em Java, uma classe executável precisa, obrigatoriamente, ter um método chamado “main”, público, estático, sem retorno e com um parâmetro correspondente a um vetor de Strings. Este método será o método executado quando você executar uma classe (caso não esteja utilizando um framework ou outro ambiente específico, como web ou Android)

```
public class MinhaClasse2 {
    public static void main(String[] args) {

    }
}
```

ESCOPO

Delimitando escopo de elementos

```java
/*
 * Pacote de classes de modelo
 */
package br.com.empresa.modelo;

/**
 * Classe de cliente
 */
public class Cliente {
    static String primeiroNome = "João";
    static String sobrenome = "da Silva";

    public static String obterNomeCompleto() {
        return primeiroNome + " " + sobrenome;
    }
}
```

Delimitador de escopo

Delimitador de escopo

VARIÁVEIS

O que são variáveis?

- Armazenam dados;
- Precisam de uma declaração de tipo;
- Podem ser consultadas ou modificadas;
- Podem ter diferentes escopos e só poderão ser acessadas no mesmo escopo ou em escopos menores;
- Normalmente nomeadas por substantivos;
- Seguem o padrão de nomenclatura “camelCase”.

Declaração de variáveis

```java
/*
 * Pacote de classes de modelo
 */
package br.com.empresa.modelo;

/**
 * Classe de cliente
 */
public class Cliente {
    static String primeiroNome = "João";
    static String sobrenome = "da Silva";
    static int idade = 22;

    public static String obterNomeCompleto() {
        String espaco = " ";
        return primeiroNome + espaco + sobrenome;
    }
}
```

Variáveis em escopo de classe ou escopo global
(em Java, chamadas de ATRIBUTOS)

Variáveis em escopo de método

Tipos de variáveis, atribuições e exemplos

```
//Armazena valores de texto
String variavelDoTipoTexto = "";
//Armazena caracteres
char variavelDeUmCaracter = 'a';
//Armazena números inteiros de até 32 bits com sinal
int variavelInteira = 0;
//Armazena números inteiros de -32768 até 32.767
short variavelInteiraCurta = 0;
//Armazena números inteiros de até 64 bits com sinal
long variavelInteiraLonga = 0L;
//Armazena números de ponto flutuante de até 32 bits com sinal
float variavelDePontoFlutuante = 0.0F;
//Armazena números de ponto flutuante de até 64 bits com sinal
double variavelDePontoFlutuanteComDuplaPrecisao = 0.0;
//Armazena "verdadeiro" (1) e "falso" (0)
boolean variavelDeVerdadeiroEFalso = false;
//Armazena o valor de um byte
byte variavelDeUmByte = 127;
```

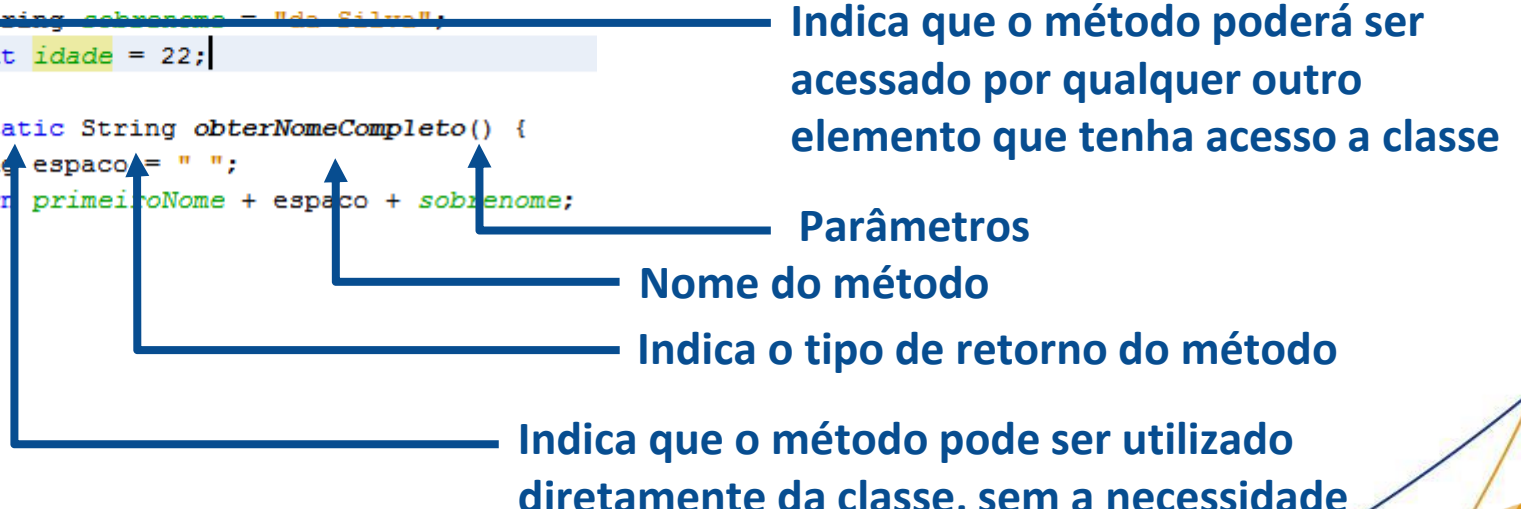
MÉTODOS

O que são métodos?

- Elementos de execução;
- Normalmente nomeados por verbo + substantivos;
- Representam trechos de código que manipulam variáveis e executam ações que podem ou não gerar resultado;
- Separam funções lógicas de elementos e permitem chama-las de outros locais;
- Seguem o padrão de nomenclatura “camelCase”

Declaração de métodos

```java
/*
 * Pacote de classes de modelo
 */
package br.com.empresa.modelo;

/**
 * Classe de cliente
 */
public class Cliente {
    static String primeiroNome = "João";
    static String sobrenome = "da Silva";
    static int idade = 22;

    public static String obterNomeCompleto() {
        String espaco = " ";
        return primeiroNome + espaco + sobrenome;
    }
}
```



Exemplos de métodos

```java
/**
 * Classe de cliente
 */
public class Cliente {

    //Método sem retorno e sem parâmetros
    public static void exemplo() {
        int val = exemploComRetornoEParametros(0, "nada");
    }

    //Método com retorno e sem parâmetros
    public static String exemploComRetorno() {
        return "";
    }

    //Exemplo sem retorno e com parâmetros
    public static void exemploComParametros(int param1, String param2) {
        exemplo();
    }

    //Exemplo com retorno e com parâmetros
    public static int exemploComRetornoEParametros(int param1, String param2) {
        return 0;
    }
}
```

OPERAÇÕES ARITMÉTICAS

Principais operadores matemáticos

- + Adição (e junção de Strings)
- - Subtração
- * Multiplicação
- / Divisão
- % Resto

Exemplos de operadores matemáticos

```java
/**
 * Classe de cliente
 */
public class Cliente {

    public static void exemplo() {
        int valor1 = 10;
        int valor2 = 5;

        //Soma
        int soma = valor1 + valor2;

        //Subtração
        int subtracao = valor1 - valor2;

        //Multiplicação
        int multiplicacao = valor1 * valor2;

        //Divisão
        int divisao = valor1 / valor2;

        //Resto da divisão
        int restoDivisao = valor1 % valor2;
    }
}
```

Lendo dados do console em Java

- Para permitir a digitação de valores no console em Java, utilize a classe “Scanner” e seus métodos de leitura, como “readLine”, “readInt” ou “readBoolean”

```
Scanner scan = new Scanner(System.in);
String valorString = scan.nextLine();
double valorDecimal = scan.nextDouble();
int valorInteiro = scan.nextInt();
boolean valorVerdadeiroFalso = scan.nextBoolean();
```

Escrevendo dados no console em Java

- Para exibir dados no console em Java, utilize a classe e os métodos padrão de saída de dados, “System.out.println(“VALOR A EXIBIR”);”

```
System.out.println("Exibindo dados no console");
System.out.println("Mais dados");
System.out.println("Olá mundo");
```

Exercícios

1) Construa um programa que **receba o valor de três notas por parâmetro de programa** Java (parâmetros do método main), **calcule a média aritmética** (X + Y + Z) / 3 **e exiba o resultado no console**. Componha o programa de forma que a média seja calculada **em um método separado**, chamado pelo "main" quando o programa iniciar. **(0,5 pt)**

DESVIOS CONDICIONAIS

O que são desvios condicionais?

- Permitem decidir se um trecho de código deve ser executado apenas caso atenda determinado(s) critério(s), permitindo a tomada de decisão através do uso de condições;
- Pode ser utilizado através das instruções “if”, “if...else”, “if...else if...else” ou “switch...case”;
- Requer o uso de operadores de comparação == (igual), != (diferente), > (maior), < (menor), >= (maior ou igual), <= (menor ou igual), && (e) e || (ou).

A instrução “if”

```
/**
 * Classe de cliente
 */
public class Cliente {

    public static void exemplo() {
        int valor = 0;

        //Verifica se o valor é igual a zero
        if (valor == 0) {
            //Faça algo
        }
    }
}
```

Declaração da instrução “if”

Condição

Código a executar caso “verdadeiro”

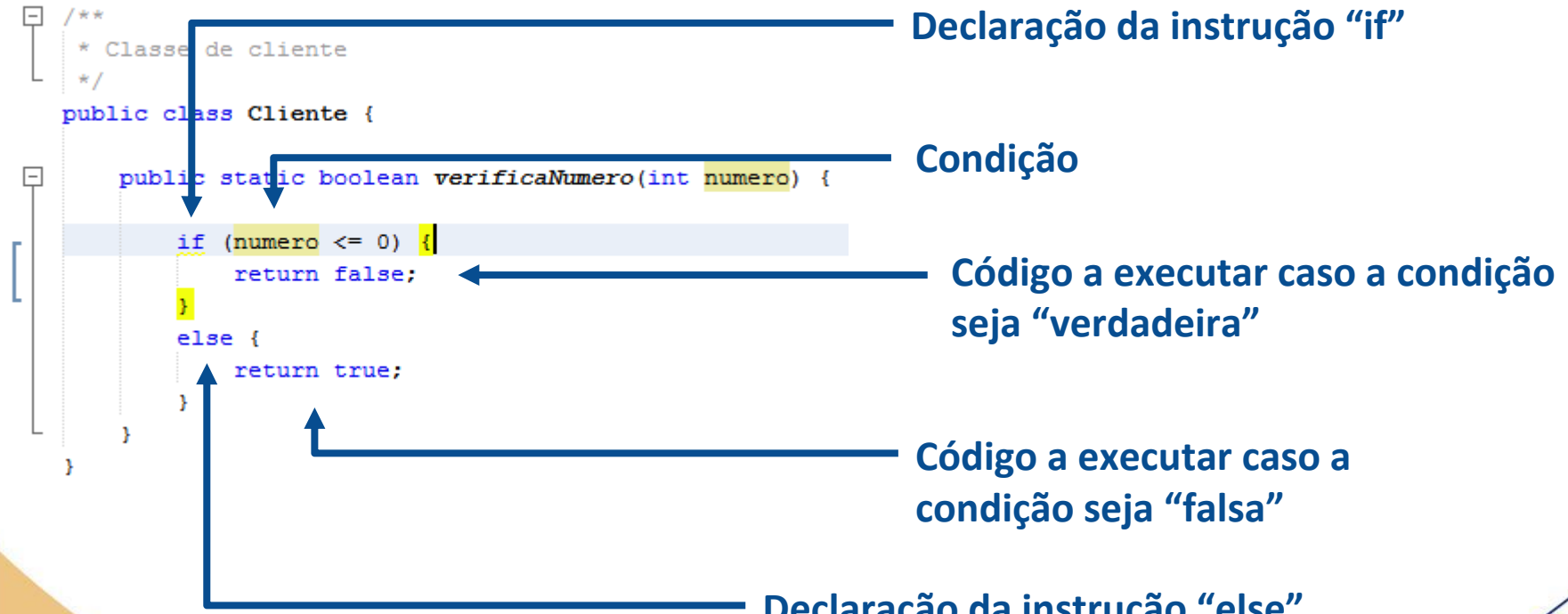
Exemplos da instrução “if”

```
public static boolean verificaNumero(int numero) {
    boolean result = false;

    if (numero > 0) {
        result = true;
    }

    return result;
}
```

```
public static float limitaValor(float valor) {
    float valorFinal = valor;

    if (valorFinal > 100000.00) {
        valorFinal = 100000.00f;
    }

    return valorFinal;
}
```

```
boolean andando = false;
float velocidadeAtual = 100f;

void freiar() {
    if (andando) {
        velocidadeAtual--;
    }
}
```

A instrução “if...else”

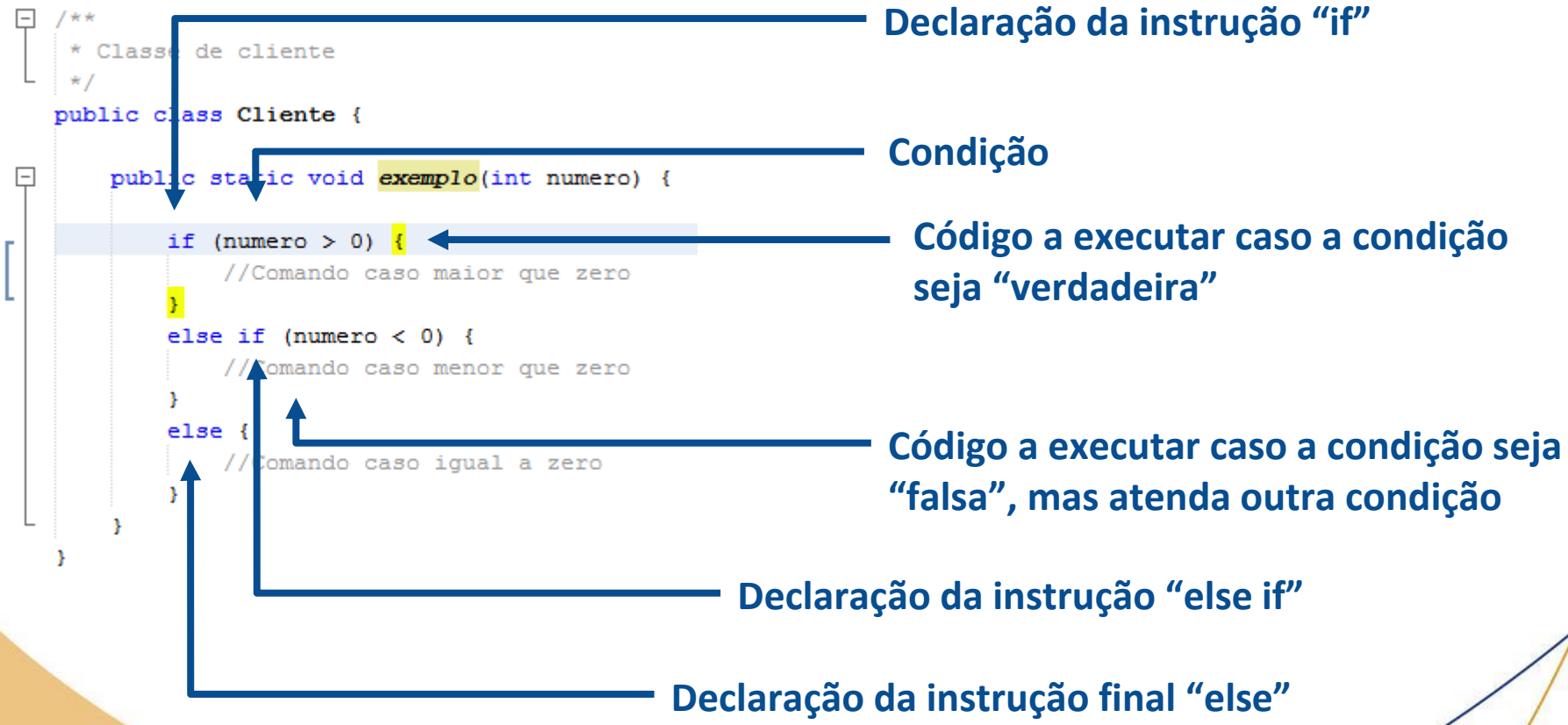


Exemplos da instrução “if...else”

```java
/**
 * Classe de cliente
 */
public class Cliente {

    public static boolean parOuImpar(int valor) {

        if (valor % 2 == 0) {
            return true;
        }
        else {
            return false;
        }
    }
}
```

```java
/**
 * Classe de cliente
 */
public class Cliente {

    public static String verificarString(String valor) {

        if (valor == null || valor.trim().isEmpty()) {
            return "Não selecionado";
        }
        else {
            return valor;
        }
    }
}
```

A instrução “if...else if...else”



Exemplo da instrução “if...else if...else”

```java
/**
 * Classe de cliente
 */
public class Cliente {

    public static boolean verificaNumero(int numero) {

        if (numero > 0) {
            return true;
        }
        else if (numero < 0) {
            if (numero % 2 == 0) {
                return true;
            }
            else {
                return false;
            }
        }
        else {
            return false;
        }
    }
}
```

A instrução “switch...case”

```
/**
 * Classe de cliente
 */
public class Cliente {

    public static void exemplo(int i) {

        switch (i) {
            case 1: {
                System.out.println("One: " + i);
                break;
            }
            case 2:
            case 3: {
                System.out.println("Two or three: " + i);
            }
            default: {
                System.out.println("Default case: " + i);
            }
        }

    }
}
```

Declaração da instrução “switch”

Valor a ser testado

Caso a variável seja de um determinado valor

Interrompe e não executa as verificações dos demais casos

Código a executar caso a condição seja “verdadeira”

Código a executar caso nenhum caso seja “verdadeiro”

Declaração da instrução “default”

Exemplo da instrução “switch...case”

```
/**
 * Classe de cliente
 */
public class Cliente {

    public static String formatMonth(int month) {
        String monthString;
        switch (month) {
            case 1:  monthString = "Janeiro";
                     break;
            case 2:  monthString = "Fevereiro";
                     break;
            case 3:  monthString = "Março";
                     break;
            case 4:  monthString = "Abril";
                     break;
            case 5:  monthString = "Maio";
                     break;
            case 6:  monthString = "Junho";
                     break;
            case 7:  monthString = "Julho";
                     break;
            case 8:  monthString = "Agosto";
                     break;
            case 9:  monthString = "Setembro";
                     break;
            case 10: monthString = "Outubro";
                     break;
            case 11: monthString = "Novembro";
                     break;
            case 12: monthString = "Dezembro";
                     break;
            default: monthString = "Mês inválido";
                     break;
        }

        return monthString;
    }
}
```

Exercícios

3) Leia os parâmetros **diaUtil (boolean)** e **emFerias (boolean)** **durante** a execução do programa.

Com base nos dados, faça um programa que **informe se o usuário pode dormir até mais tarde**. Isto só pode acontecer caso não seja um dia útil ou ele esteja de férias. **(1 pt)**

REPETIÇÕES

O que são repetições?

- Permitem executar determinado código múltiplas vezes, dependendo de condições;
- Mantém o programa “preso” em determinada instrução até que uma situação ocorra;
- Podem ser interrompidas a qualquer momento com a instrução “break”;
- Podem ser utilizadas com as instruções “while”, “do...while” e “for”.

A instrução “while”

```java
/**
 * Classe de cliente
 */
public class Cliente {

    public static void exemplo(int number) {
        while (number < 100) {
            //Comando a ser repetido aqui
        }
    }

}
```

Declaração da instrução “while”

Código a executar enquanto “verdadeiro”

Condição

Exemplos da instrução “while”

```
public static void contar(int number) {

    int count = 1;
    while (count <= number) {
        System.out.println("Contando: " + count);
        count++;
    }

}

public static int incrementFiveBy(int value, int incrementTimes) {
    int result = value;
    int count = 1;
    while (count <= incrementTimes) {
        result += 5;
        count++;
    }

    return result;
}

public static void readValue() {
    Scanner scan = new Scanner(System.in);
    while (true) {
        System.out.println("Sair?");
        String value = scan.nextLine();
        if (value.equalsIgnoreCase("S") || value.equalsIgnoreCase("Sim")) {
            break;
        }
    }
}
```

A instrução “do...while”

```
public static void exemplo(int value) {
    do {
        //Código a ser executado enquanto value < 100
    }
    while (value < 100);
}
```

Declaração da instrução “do”

Código a executar enquanto “verdadeiro”

Condição

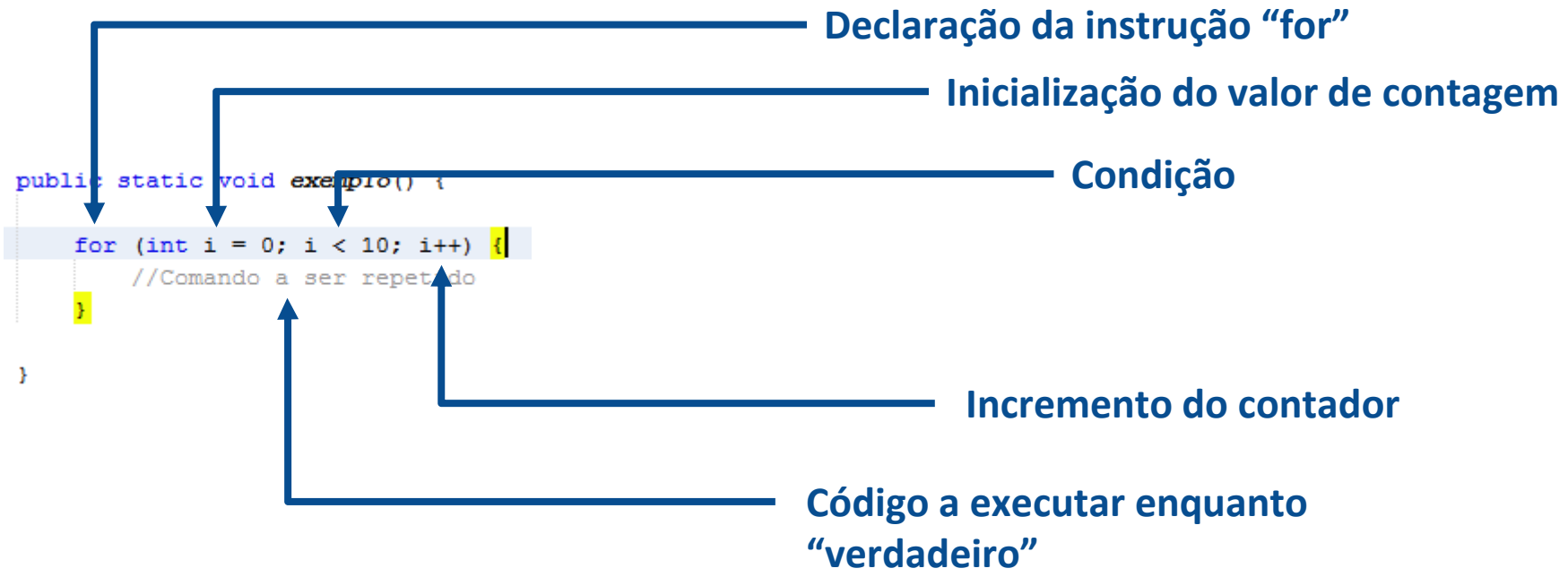
Declaração da instrução “while”

Exemplos da instrução “do...while”

```
public static void readValue() {

    Scanner scan = new Scanner(System.in);
    String resp;
    do {
        System.out.println("Continuar?");
        resp = scan.nextLine();
    }
    while (resp.equalsIgnoreCase("Sim") || resp.equalsIgnoreCase("S"));
}
```

```
public static void printValue() {
    int x = 10;
    do {
        System.out.print("Valor de X: " + x);
        x++;
        System.out.print("\n");
    } while (x < 20);
}
```

A instrução “for”



Exemplos da instrução “for”

```java
public static void exemplo() {

    for (int i = 0; i < 5; i++) {
        System.out.println("i vale : " + i);
    }

}

    public static int factorial(int n) {
        int result = 1;
        for (int i = 2; i <= n; i++) {
            result *= i;
        }
        return result;
    }

public static void numerosImpares(int number) {

    int limit = 50;

    System.out.println("Imprime os números ímpares entre 1 e " + limit);

    for (int i = 1; i <= limit; i++) {
        //Se o resto da divisão por 2 não for 0, então o número é ímpar
        if (i % 2 != 0) {
            System.out.print(i + " ");
        }
    }
}
```

CONVERSÃO DE DADOS

Conversão de valores de variáveis

- Normalmente, utiliza-se as classes de representação de tipos primitivos em Java para conversão de valores;
- Os métodos de conversão normalmente iniciam-se por “parse” e devolvem o tipo de dado convertido;
- Deve-se tomar cuidado com o nível de precisão das variáveis antes de convertê-las, ou poderão ocorrer truncamentos de dados.

Exemplos de conversão de dados

```java
public static void convert(){
    //Converte uma string em um inteiro
    int i = Integer.parseInt("5");

    //Converte uma string em um double
    double d = Double.parseDouble("5.0");

    //Converte uma string num float
    float f = Float.parseFloat("5.0");

    //Converte uma string num long
    long l = Long.parseLong("5");

    //Converte um inteiro numa string
    String s = String.valueOf(5);
}
```

TRATAMENTO DE ERROS

Blocos “try...catch...finally”

- Código protegido por um bloco “try” executará as instruções dentro do bloco “catch” quando um erro ocorrer;
- O bloco “finally” sempre será executado, independentemente da ocorrência ou não de erros;
- A execução de código é interrompida no bloco “try” assim que ocorrer um erro numa instrução. As demais instruções do bloco abaixo da que ocasionou o erro não serão executadas;
- É possível encadear blocos de “try...catch...finally”.

Exemplo de “try...catch”

```
try {
    // Construtor pode lançar FileNotFoundException
    FileReader reader = new FileReader("someFile");
    int i=0;
    while(i != -1){
        //reader.read() pode lançar IOException
        i = reader.read();
        System.out.println((char) i );
    }
    reader.close();
    System.out.println("Fim do arquivo");
} catch (FileNotFoundException e) {
    //Fazer alguma coisa para tratar o erro
} catch (IOException e) {
    //Fazer alguma coisa para tratar o erro
}
finally {
    //Fazer alguma coisa SEMPRE, independente da ocorrência de erros
}
```

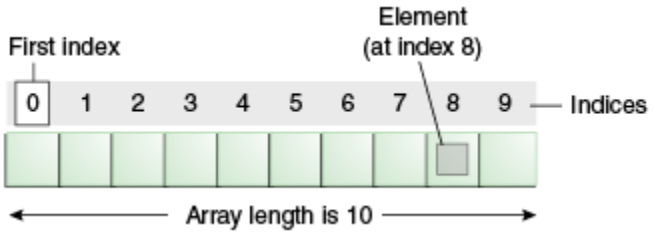
Exercícios

5) Crie um programa que **leia números do usuário até que seja digitado um caractere não numérico**. A cada leitura, o programa deve dizer se o número é **par ou ímpar. (1 pt)**

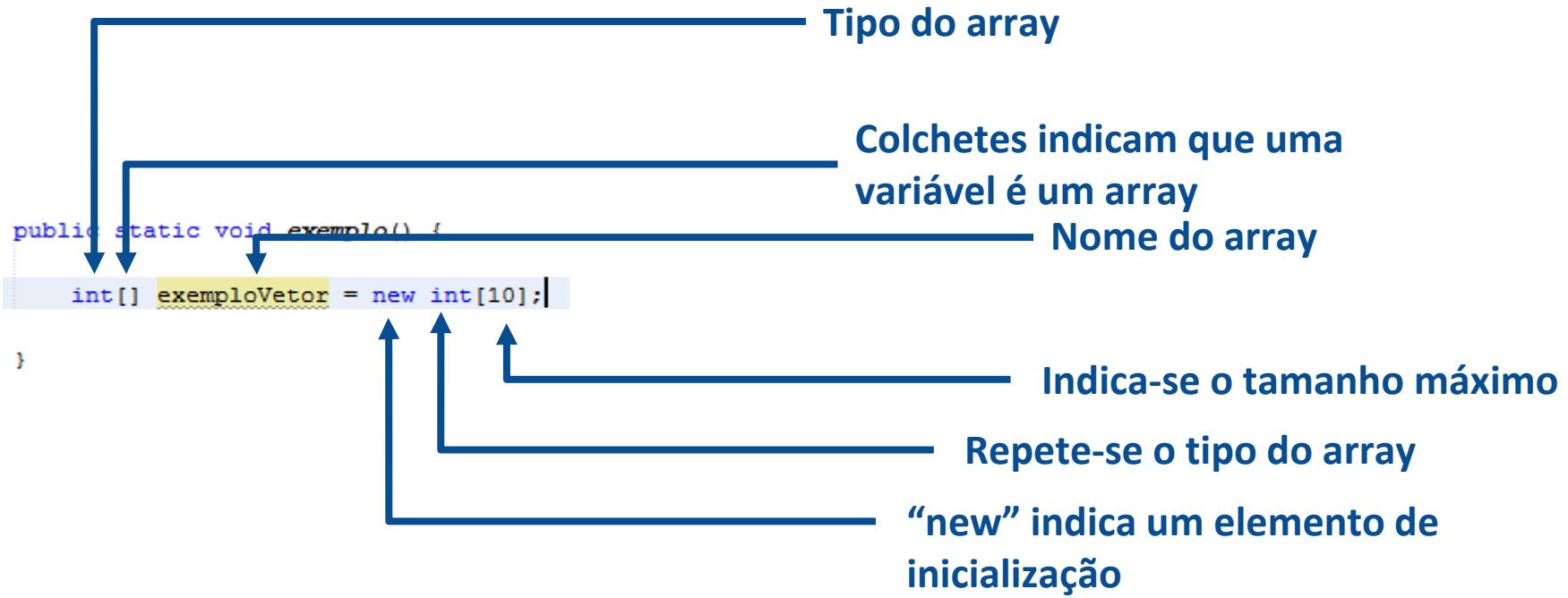
VETORES

O que são vetores (ou arrays)?

- Armazenam uma quantidade fixa de valores de determinado tipo;
- O tamanho do vetor (ou array) deve ser estabelecido quando este é inicializado;
- Cada item do array é chamado de “elemento” do array e é acessível por um “índice”;



Declarando arrays




```
public static void exemplo() {
    int[] exemploVetor = new int[10];

}
```

Exemplo de declaração e inicialização de arrays

```
public static void exemplo() {

    //declara um array de inteiros
    int[] anArray;

    //Aloca posições para 10 elementos
    anArray = new int[10];

    //Inicializa o primeiro elemento
    anArray[0] = 100;
    //Inicializa o segundo elemento
    anArray[1] = 200;
    //e assim por diante
    anArray[2] = 300;
    anArray[3] = 400;
    anArray[4] = 500;
    anArray[5] = 600;
    anArray[6] = 700;
    anArray[7] = 800;
    anArray[8] = 900;
    anArray[9] = 1000;

    System.out.println("Elemento no índice 0: " + anArray[0]);
    System.out.println("Elemento no índice 1: " + anArray[1]);
    System.out.println("Elemento no índice 2: " + anArray[2]);
    System.out.println("Elemento no índice 3: " + anArray[3]);
    System.out.println("Elemento no índice 4: " + anArray[4]);
    System.out.println("Elemento no índice 5: " + anArray[5]);
    System.out.println("Elemento no índice 6: " + anArray[6]);
    System.out.println("Elemento no índice 7: " + anArray[7]);
    System.out.println("Elemento no índice 8: " + anArray[8]);
    System.out.println("Elemento no índice 9: " + anArray[9]);
}
```

Percorrendo vetores

```
public static void exemplo() {
    int[] vetorIdades = new int[7];

    for (int i = 0; i < 7; i++) {
        vetorIdades[i] = i * 10;
    }
    for (int i = 0; i < 7; i++) {
        System.out.println(vetorIdades[i]);
    }
}
```

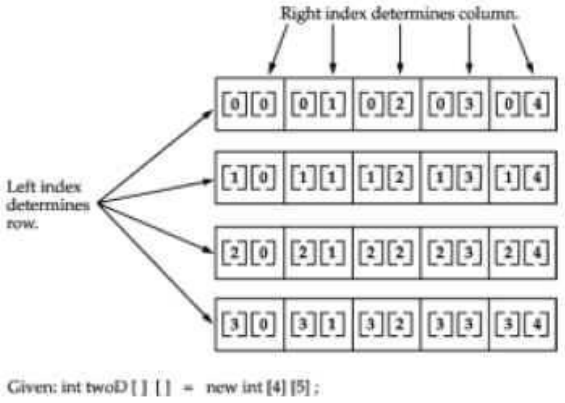
```
void mostrarVetor(int[] array) {
    for (int i = 0; i < array.length; i++) {
        System.out.println(array[i]);
    }
}
```

```
void mostrarVetor(int[] array) {
    for (int x : array) {
        System.out.println(x);
    }
}
```

MATRIZES

O que são matrizes?

- Vetores de múltiplas dimensões;
- Utilizam múltiplos índices para referência de valores, um índice para cada dimensão;
- Todas as “dimensões” devem ser do mesmo tipo;
- Pode ser percorrido por uma repetição para cada dimensão.



Exemplo de declaração e inicialização de matrizes

```
public static void main(String[] args) {

    int[][] aryNumbers = new int[6][5];

    aryNumbers[0][0] = 10;      aryNumbers[1][0] = 20;
    aryNumbers[0][1] = 12;      aryNumbers[1][1] = 45;
    aryNumbers[0][2] = 43;      aryNumbers[1][2] = 56;
    aryNumbers[0][3] = 11;      aryNumbers[1][3] = 1;
    aryNumbers[0][4] = 22;      aryNumbers[1][4] = 33;

    aryNumbers[2][0] = 30;      aryNumbers[3][0] = 40;
    aryNumbers[2][1] = 67;      aryNumbers[3][1] = 12;
    aryNumbers[2][2] = 32;      aryNumbers[3][2] = 87;
    aryNumbers[2][3] = 14;      aryNumbers[3][3] = 14;
    aryNumbers[2][4] = 44;      aryNumbers[3][4] = 55;

    aryNumbers[4][0] = 50;      aryNumbers[5][0] = 60;
    aryNumbers[4][1] = 86;      aryNumbers[5][1] = 53;
    aryNumbers[4][2] = 66;      aryNumbers[5][2] = 44;
    aryNumbers[4][3] = 13;      aryNumbers[5][3] = 12;
    aryNumbers[4][4] = 66;      aryNumbers[5][4] = 11;

    int rows = 6;
    int columns = 5;
    int i, j;

    for ( i = 0; i < rows; i++) {
        for ( j = 0; j < columns; j++) {
            System.out.print(aryNumbers[i][j] + " ");
        }
        System.out.println( "" );
    }
}
```

Percorrendo matrizes

```
void mostrarMatriz(int[] array) {
    int[][] matriz = new int[3][3];

    Scanner entrada = new Scanner(System.in);
    System.out.println("Matriz M[3][3]\n");

    for(int linha=0 ; linha < 3 ; linha++){
        for(int coluna = 0; coluna < 3 ; coluna ++){
            System.out.printf("Insira o elemento M[%d][%d]: ",
                linha+1,coluna+1);
            matriz[linha][coluna]=entrada.nextInt();
        }
    }

    System.out.println("\nA Matriz ficou: \n");
    for(int linha=0 ; linha < 3 ; linha++){
        for(int coluna = 0; coluna < 3 ; coluna ++){
            System.out.printf("\t %d \t",matriz[linha][coluna]);
        }
        System.out.println();
    }
}
```

Exercícios

7) Desenvolva um programa que leia os **nomes dos jogos eletrônicos favoritos de um usuário e suas respectivas notas** de avaliação pessoal, com limite máximo de **10 jogos** ou até que o usuário **digite "sair"** no nome do jogo. Valide se as informações de nota são corretas e, então, **informe o jogo do usuário com maior nota. (2 pt)**

Exercícios

https://www.sp.senac.br


"That's all Folks!"