

**Centro de Formação  
São Domingos**

1

**Fundamentos de Programação**

**em**

**Java**

**guiia prático**

**Fr Joaquim José Hangalo**

**Luanda, 2011**

---

# **Guia prático de introdução à programação e à linguagem java com Orientação a Objectos**

---

## **Princípios da programação na linguagem Java**

---

Como qualquer linguagem de programação, a linguagem Java tem regras estritas indicando como os programas devem ser escritos. É bom ter familiaridade com a sintaxe da linguagem, aprendendo a reconhecer e definir os tipos primitivos, identificadores e expressões em Java

### **Antes de começar**

<b>Convenções do Código Java</b>
a) Classes: O nome da classe deve ter a inicial maiúscula e as demais minúsculas. Quando você utiliza dois nomes, as iniciais devem ser maiúsculas.  <code>class MeuPrimeiroPrograma</code>
b) Interfaces: A mesma regra aplicada às classes
c) Métodos: Devem iniciar com letra minúsculas e as iniciais das próximas palavras

- a) Classes: O nome da classe deve ter a inicial maiúscula e as demais minúsculas.  
Quando você utiliza dois nomes, as iniciais devem ser maiúsculas.
- `class MeuPrimeiroPrograma`
- b) Interfaces: A mesma regra aplicada às classes
- c) Métodos: Devem iniciar com letra minúsculas e as iniciais das próximas palavras

devem ser maiúsculas.

### **informeSalario()**

d) Variáveis: Devem iniciar com letra minúsculas e as iniciais das próximas palavras devem ser maiúsculas.

### **clienteCorrente**

e) Constantes: Devem ser todas em maiúsculas, com as palavras separadas por sublinhado.

### **IDADE\_MAXIMA**

f) Utilizar identação de código

g) É boa prática comentar o código!

3

## **Palavras Reservadas**

abstract	double	int	strictfp
boolean	else	interface	super
break	extends	long	switch
byte	final	native	synchronized
case	finally	new	this
catch	float	package	throw
char	for	private	throws
class	goto	protected	transient
const	if	public	try
continue	implements	return	void
default	import	short	volatile
do	instanceof	static	while

## **Regras gerais para escrever um programa em Java**

Um programa em Java é constituído de uma ou mais classes delimitadas por chaves { }, onde uma destas classes, obrigatoriamente possui um método chamada **main()**. As principais regras são:

- ✓ Letras maiúsculas e minúsculas são tratadas como caracteres diferentes;
- ✓ O formato do texto é livre;
- ✓ O método `main(){}` especifica onde o programa começa e termina de ser executado;
- ✓ Todos os comandos são terminados por ponto e vírgula;
- ✓ Todas as variáveis devem ser declaradas;
- ✓ { método começa a ser executado;
- ✓ } método termina de ser executado.

4

## Elementos básicos da Linguagem

---

### Tipos de dados, variáveis e contas

As variáveis e constantes são os elementos básicos que um programa manipula

#### Tipos de Dados

Determinam o conjunto de valores e as possíveis operações realizadas sobre os mesmos; informam a quantidade de memória (em bytes) que uma variável pode ocupar; Cada tipo de dado possui um intervalo de valores permitido.

Na linguagem Java existem dois grupos de tipos de dados:

1. Tipos básicos primitivos, chamados de built-in types que são divididos em três categorias (numéricos, booleanos e caracteres);
2. Tipos compostos os quais são divididos em três categorias (Strings, arrays e personalizados)

#### Tipos de dados Primitivos

Java é uma linguagem de programação fortemente orientada a objetos e, com exceção dos tipos primitivos, qualquer coisa em Java é uma classe/objeto.

## Tipos Numéricicos Inteiros

Tipo	Tamanho em bits	Faixa
byte	8	-128 até +127
short	16	-32,768 até +32,767
int	32	-2,147,483,648 até +2,147,483,647
long	64	-9,223,372,036,854,775,808 até +9,223,372,036,854,775,807

## Tipos Numéricicos Reais

Tipo	Tamanho em bits	Faixa
float	32	-3.40292347E+38 até +3.40292347E+38
double	64	-1.79769313486231570E+308 até +1.79769313486231570E+308

## Tipo caracter

Tipo	Tamanho em bits	Faixa
char	16	UNICODE - 65536 caracteres possíveis

Observação: Um caractere é delimitado por apóstrofos ´caractereº O que causa confusão, pois as Strings são delimitadas por aspas ¨Stringº.

## Caracteres Especiais de Escape:

\u0000' a	caracteres Unicode
\uFFFF'	
\b	retrocesso
\t	tab
\n	avanço de linha
\r	retorno de carro
\º	aspas
\'	apóstrofo
\\"	barra invertida

Observação: A linguagem Java não possui o tipo primitivo string, bastante conhecido em várias outras linguagens. Para manipulação de texto são utilizadas as classes String e StringBuffer.

## Tipo Lógico

Tipo	Faixa
boolean	true ou false

## Quadro Resumo

Tipo	Natureza	Valores permitidos	Tamanho (bytes)
boolean	Lógico	false / true	1
char	Texto	1 caractere (‐xø)	1
String	Texto	cadeira de caracteres (‐Oláö)	É um objeto
byte	Inteiro	-2 <sup>7</sup> até 2 <sup>7</sup> -1	1
short	Inteiro	-2 <sup>15</sup> até 2 <sup>15</sup> -1	2
int	Inteiro	-2 <sup>31</sup> até 2 <sup>31</sup> -1	4
long	Inteiro	-2 <sup>63</sup> até 2 <sup>63</sup> -1	8
float	Ponto Flutuante	-	4
double	Ponto Flutuante	-	8

Em java alguns tipos primitivos têm uma classe contentora (wrapper)

Tipos primitivos	Classe wrapper
boolean	Boolean
int	Integer
double	Double
long	Long
float	Float
--	String
--	Date
char	Char

## Variáveis

Uma variável é um espaço reservado na memória do computador para armazenar um determinado tipo de dado.

Uma ««Variável»» que pode assumir diversos valores; espaço de memória de um certo tipo de dado associado a um nome para referenciar o seu conteúdo

A variável é o núcleo de um objecto. Varáveis são espaços nomeados na memória que armazenam valores e para cada espaço de memória é associado um nome (identificador) e o tipo de valor a ser armazenado, dentre os tipos primitivos temos o tipo numéricos, caracteres, booleans e referencia.

Uma variável representa a unidade básica de armazenamento temporário de dados e é composta por um tipo, um identificador (nome), um tamanho e um valor.

## Declaração de Variáveis

Declaração é a instrução para reservar uma quantidade de memória para um certo tipo de dado, indicando o nome pelo qual a área será referenciada .

### Sintaxe

```
>> tipo de dado nomeDaVariável; ou  
>> tipo nome1, nome2,...,nomen
```

Ex: String nome;

int idade, num;  
nomeDaVariável é também chamado **identificador**

### Regras para nome de identificadores(das variáveis)

- Permite letras e números
- Começa com uma letra ou "\$" ou "\_"
- Possui tamanho ilimitado. Quantos caracteres se quiser (32);

- Não pode ser uma das palavras reservadas. Não se pode definir um identificador com o mesmo nome que uma palavra reservada (ver tabela das palavras reservadas)
- A linguagem Java é "case sensitive", diferencia maiúsculas de minúsculas

**peso != Peso != pEso**

É importante observar certos padrões para a definição de identificadores para variáveis.

O uso de nomes curtos é indicado para reduzir o volume de código a ser escrito. Evitar o uso de caracteres acentuados, espaços e símbolos especiais pode ser aconselhável para tornar o programa potencialmente mais portável.

### Nomenclatura dos nomes das variáveis

- As variáveis são nomeadas com um atributo no singular
- Não são utilizados os símbolos de sublinhado (ó-ö) ou ó
- Os nomes compostos de múltiplas palavras são organizados segundo o padrão CamelCase, com todas as palavras juntas, onde a primeira letra da primeira palavra é minúscula e das restantes palavra a primeira letra de cada uma fica em maiúscula
- Ex: nomeDoAluno, salarioAnual, mediaSemestral

#### Exemplo

```
public class Variaveis {
    public static void main(String[] args) {
        int idade;
        idade = 30;
        System.out.println(idade);
    }
}
```

**O escopo de uma variável** é o bloco de código onde ela é visível. No caso de um atributo (variável membro) o seu escopo é a própria classe, já no caso de uma variável ou argumento (**variável local**) o seu escopo começa na chamada de um método e termina quando o mesmo finaliza.

## Constantes

As constantes podem ser declaradas em qualquer lugar dentro de uma classe ou de um método.

Para definir uma constante no Java, utiliza-se o modificador **final** e, no seu identificador, todas as letras devem ser maiúsculas (de acordo com a convenção de codificação). O seu conteúdo uma vez atribuído não poderá mais ser alterado.

A sintaxe para a definição de constantes é:

[modificador de acesso] **static final** tipo identificador = valor;

Exemplo

```
public static final float PI = 3.141592;
final Boolean DEBUG = false;
```

## Inicialização de variáveis.

Inicializar uma variável é atribuir um valor à variável

Sintaxe: A sintaxe para a inicialização de variáveis é:

*tipo var\_1 = valor\_1 [, var\_2 = valor\_2, ...]*

Onde *tipo* é o tipo de dado, *var\_1* é o nome da variável a ser inicializada e *valor\_1* é o valor inicial da variável.

## Operador e atribuição

Para se inicializar uma variável utiliza-se o operador de atribuição == é importante notar que, em java, o operador de igualdade tem outra notação ==.

```
int idade = 24;
```

```
String nome = "Maria Sapalo"
```

Nenhuma variável em um programa Java pode ser usada antes de ser inicializada. Quando um objeto é criado, as variáveis são inicializadas automaticamente com os seguintes valores:

byte	0
------	---

short	0
int	0
long	0L
float	0.0F
double	0.0D
char	\u0000\0
boolean	false
Referências a Objectos	null

*Todas as variáveis locais devem ser iniciadas manualmente!*

Variáveis Locais: variáveis declaradas no escopo de métodos ou em estruturas de controle.

## Conversão entre tipos primitivos (Casting)

É possível transformar um tipo primitivo em outro, através de uma operação chamada typecast, colocando o tipo destino da transformação, entre parêntesis, antes da expressão a ser convertida. Esse tipo de conversão é chamado de conversão explícita. Por outro lado existe a conversão implícita, a qual é realizada pela MVJ, mesmo sem possuir operador de typecast, isso quando o tipo de retorno é menos abrangente que o tipo que o recebe.

```
int x = 8;
float y = x;
x= (int)y;
```

O casting do tipo com precisão menor para um tipo com maior precisão, temos uma promoção (widening conversions)

O casting do grande para o pequeno, acarreta um rebaixamento (narrowing conversions)  
 Abaixo estão relacionados todos os casts possíveis na linguagem Java. A indicação **Impl.** quer dizer que aquele cast é implícito e automático, ou seja, não é preciso indicar o cast explicitamente.

<b>PARA:</b>	<b>byte</b>	<b>short</b>	<b>char</b>	<b>int</b>	<b>long</b>	<b>float</b>	<b>double</b>
<b>DE:</b>							
<b>byte</b>	----	<i>Impl.</i>	(char)	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>short</b>	(byte)	----	(char)	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>char</b>	(byte)	(short)	----	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>int</b>	(byte)	(short)	(char)	----	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>long</b>	(byte)	(short)	(char)	(int)	----	<i>Impl.</i>	<i>Impl.</i>
<b>float</b>	(byte)	(short)	(char)	(int)	(long)	----	<i>Impl.</i>
<b>double</b>	(byte)	(short)	(char)	(int)	(long)	(float)	----

<b>Programa(s)</b>	introducao.TiposPrimitivos.java
<b>Demonstração</b>	
	<pre>/*  * Demonstração de Tipos Primitivos  */ package introducao;  public class TiposPrimitivos {      public static void main ( String[] args ) {         /* Declarando tres variaveis do tipo float */         float nota1, nota2, nota3;         /* Inicializando as tres variáveis float */         nota1 = 10;         nota2 = 7;         nota3 = 3;         /* Declarando e inicializando três variaveis float*/         float n1 = 5, n2 = 2, n3 = 9;          int idade = 28;         char sexo = 'M'; //Lembrar de usar apóstofros         double salario = 3000.00;         boolean achou = false;         /*          * quando uma variável é declarada como final a mesmo não pode ter seu          * conteúdo alterado          */         final char FLAG = 'F';         /*          * Erro, pois literais numéricas com casas decimais são double          */         // nota1 = 8.5         nota1 = 8.5f;           // conversão explicita para float         nota2 = 7.0f;           // conversão explicita para float         nota3 = (float) 9.0;    // conversão explicita para float     } }</pre>

```
System.out.println("Dados do Aluno");
System.out.print("Idade \t= " + idade + "\n ");
System.out.println("Sexo \t= " + sexo);
System.out.println("Notas \t= " + nota1 + "\t" + nota2 + "\t" + nota3);
System.out.println("Achou \t= " + achou);
System.out.println("Flag \t= " + FLAG);
/* Conversão de double para int */
double d = 34.78;
int i = (int) d;
System.out.println("double : " + d + "\n int : " + i);

}
```

12

## Comentários

Comentários em Java podem ser expressos em três formatos distintos. Na primeira forma, uma

barra dupla`//` marca o início do comentário, que se estende até o fim da linha. A segunda forma é o comentário no estilo C tradicional, onde o par de símbolos `/*` marca o início de comentário, que pode se estender por diversas linhas até encontrar o par de símbolos `*/`, que delimita o fim do comentário.

O outro estilo de comentário é específico de Java e está associado à geração automática de documentação do *software* desenvolvido. Nesse caso, o início de comentário é delimitado pelas sequências de início `/**` e de término `*/`. O texto entre essas sequências será utilizado pela ferramenta javadoc para gerar a documentação do código em formato hipertexto.

Para gerar esse tipo de documentação, os comentários devem estar localizados imediatamente antes da definição da classe ou membro (atributo ou método) documentado. Cada comentário pode conter uma descrição textual sobre a classe ou membro, possivelmente incluindo *tags* HTML, e directrizes para o programa javadoc. As directrizes para javadoc são sempre precedidas por `@`, como em `@see nomeClasseOuMembro` que gera na documentação um *link* para a classe ou membro especificado, precedido pela frase `See also`.

A documentação de uma classe pode incluir as directrizes `@author`, que inclui o texto na sequência como informação sobre o autor do código na documentação; e `@version`, que inclui na documentação informação sobre a versão do código.

A documentação de um método pode incluir as directrizes @param, que apresenta o nome e uma descrição de cada argumento do método; @return, que apresenta uma descrição sobre o valor de retorno; e @exception, que indica que excepções podem ser lançadas pelo método.

### Quadro Resumo

```
// este é um comentário monolinha; termina no fim desta linha
/*
este é multilinha; só termina quando aparecer o delimitador final */
/** este é um comentário de documentação para a ferramenta javadoc */
```

## Operadores

Veremos agora os operadores da linguagem Java, que agregam importantes funcionalidades aos programas.

Eles possuem uma ordem de precedência na execução da expressão.

Para alterar a ordem de precedência deve-se, agrupar as expressões com parênteses.

Os operadores são utilizadores para criar expressões aritméticas ou lógicas

Uma expressão é um arranjo de operadores e operandos. A cada expressão válida é atribuído um valor numérico.

Exemplo:  $4 + 6$  é uma expressão cujo valor é 10.

Em java os operadores podem ser de atribuição, aritméticos, de atribuição aritmética, incrementais, relacionais, lógicos e condicionais.

### Atribuição

A operação de atribuição é a operação mais simples do java. Consiste de atribuir um valor ou o valor de uma expressão a uma variável.

Sintaxe:

*identificador = valor ou expressão;*

onde *identificador* é o nome de uma variável e *expressão* é uma expressão válida (ou outro identificador).

- Exemplo:
- `a = 1;`
- `delta = b * b - 4. * a * c;`
- `i = j;`

`=`

o símbolo "`=`" representa o comando de atribuição.

o símbolo "`==`" representa o sinal de igualdade.

## Atribuição Composta

Para facilitar a programação, Java oferece um tipo de atribuição chamada atribuição composta. Esse tipo de atribuição pode ser utilizado quando se deseja atribuir a uma variável X, o valor de X adicionado a 10, por exemplo.

Exemplos:

```
x += 10; // Equivale x = x + 10;
x += y; // Equivale x = x + y;
a -= b; // Equivale a = a - b;
a *= 3; // Equivale a = a * 3;
```

## Concatenação de Strings

O operador `+` executa uma concatenação de objetos de Strings, gerando uma nova String.

```
String tratamento = "Sra. ";
String nomeCompleto = "Marta Rocha";
String saudação = tratamento + nomeCompleto;
```

## Operadores aritméticos

São utilizadas para criar expressões aritméticas. Expressões que resultam num valor numérico

Operador aritmético	Ação
<code>-</code>	Subtração
<code>+</code>	Adição

*	Multiplicação
/	Divisão
%	Resto inteiro da divisão
-- ++	Decremento/incremento

### Precedência dos operadores aritméticos (Hierarquia nas Operações)

Hierarquia	Operação
1	Parênteses
2	Métodos
3	++ --
4	- (menos unário)
5	* / %
6	+ -

**Observação:** Quando houver duas ou mais operações de mesma hierarquia, o compilador executa-as da **esquerda para a direita**.

### Operadores relacionais ou de comparação

Estes operadores são utilizadores para criar expressões booleanas Ou seja, expressões que retornam falso ou verdadeiro. Estes operadores permitem comparar valores, ou seja, são utilizados principalmente em comandos que possuem condições.

#### Operadores relacionais

Operador	Ação
>	Maior que
$\geq$	Maior ou igual a
<	Menor que
$\leq$	Menor ou igual
$=$	Igual a

$!=$	Diferente de
------	--------------

$3 == 3, 4 <= 5, 5 >= 4, 3 < 4, 4 > 3, 4 != 3$

## Operadores Lógicos

São utilizadores para combinar o valor lógicos de duas ou mais expressões booleanas. São operadores utilizados em comandos que tem mais de uma condição.

16

Operadores lógicos

Operador lógica	Ação
$&&$ ou $\&$	AND (e)
$  $ ou $ $	OR (ou)
!	NOT (não)

**Observação:**  $\&&$  e  $||$  são chamados *Short Circuit Operator*, ou seja, quando o resultado das condições não puder mais ser alterado, há um truncamento nas próximas condições.

## Precedência (Hierarquia dos operadores)

Precedência dos operadores relacionais e lógicos

Hierarquia	Operação
1	!
2	$>$ $\geq$ $<$ $\leq$
3	$==$ $!=$
4	$\&&$
5	$  $

**Observação:** As expressões que utilizam operadores relacionais e lógicos retornam **0** (zero) para falso e **!0** (não zero) para verdadeiro, ou seja:

**true** é diferente de 0 ( $!= 0$ ) e **false** é igual 0 ( $== 0$ )

**Tabela de verdade dos operadores `&&`, `||` e `não`**

p	q	p&&q	p  q	!q	!p
0	0	0	0	1	1
0	1	0	1	0	1
1	0	0	1	1	0
1	1	1	1	0	0

### Exercícios

Qual é o resultado da expressão

a) <code>!0 &amp;&amp; 0    0</code>	b) <code>!(0 &amp;&amp; 0)    0</code>
<code>10&gt;5 &amp;&amp; !(10&lt;9)    3&lt;=4</code>	

### Operadores de incremento e decremento

Incremento

`++`

Decremento

`--`

As seguintes operações são equivalentes:

`x++;`       $x = x + 1;$

`x--;`       $x = x - 1;$

Altera o valor da variável de n unidades.

Caso a unidade não seja informada o padrão 1(um) será utilizado.

Os operadores (`++` ou `--`) podem ser colocados antes ou depois do operando. Quando precede seu operando, Java efetua a **operação** de incremento ou decremento antes de

utilizar o valor do operando. Quando o operador vier depois do operando, Java utiliza o **valor do operando** antes de incrementá-lo ou decrementá-lo.

Quando o operador é colocado depois da variável tem-se um pós incremento ou pós decremento. Quando o operador é colocado antes da variável tem-se um pré incremento ou pré decremento. O quadro seguinte ilustra o funcionamento do mecanismo

Operador	Uso	Descrição
++	op++	Incremente op de 1; avalia o valor de op antes de incrementa-lo
++	++op	Incrementa op de 1; avalia o valor de op depois de incrementa-lo
--	op--	Decrementa op de 1; avalia o valor de op antes de decrementa-lo
--	--op	Decrementa op de 1; avalia o valor de op depois de decrementa-lo

**Observe-se antes de mais nada que `++x` é diferente de `x++`!**

```
Se
  x = 10;
  y = ++x;
  /* x=x+1;    y=x; */
então
  x = 11 e
  y = 10
```

```
porém Se
  x = 10;
  y = x++;
  /* y=x;  x=x+1 */
então
  x = 11 e
  y = 10
```

**Quadro exemplo de incremento e decremento**

Valor inicial		Expressão	Valor Final		Interpretação
X	Y		X	Y	
10	10	$y = x++$	11	10	$y = x \rightarrow x = x+1$
10	11	$y = ++x$	11	11	$x = x+1 \rightarrow y = x$
10	10	$y = x --$	9	10	$y = x \rightarrow x = x - 1$
10	9	$y = -- x$	9	9	$x = x - 1 \rightarrow y = x$

### Linearização de expressões

Uma expressão em Java é qualquer combinação válida de operadores (aritméticos, relacionais, lógicos), constantes, variáveis e métodos.

Exemplo:  $c = Math.sqrt (a) + b / 3.4;$

A linearização consiste em transformar expressões matemáticas tradicionais em expressões em java.

Ex.

$\frac{3 + 5}{2 + 2}$	A expressão pode ser transformada em $(3+5)/((2+2)/2)$
-----------------------	---

### Tabela de precedências de todos os operadores

Operador	Acção
----------	-------

Precedência: de cima para baixa, da direita para a esquerda.

Pós-fixados	[ ] . (parâmetros) x++ x--
Pré-fixados Unários	++x --x +x -x ~ !
Criação e Type Cast	new (type)
Multiplicação	* / %
Adição	+ -
Rotação (shift)	<< >> >>>
Relacionais	< <= > >= instanceof
Igualdade	== !=
AND binário	&
XOR binário	^
OR binário	
AND lógico	&&
OR lógico	
Condicional	?:
Atribuição	= += -= *= /= %= <<= >>= >>>= &= ^=  =

## Métodos de entrada e saída de dados

A seguir são mostrados alguns métodos que permitem fazer entrada de dados via teclado e saída de dados via monitor (tela).

**Entrada de dados via teclado utilizando a classe Scanner do pacote java.util**

Para utilizar a classe Scanner numa classe java deve-se proceder da seguinte maneira:

1 ó importar o pacote java.util

```
import java.util.Scanner;
```

2 ó Instanciar e criar um objecto Scanner:

```
Scanner ler = new Scanner(System.in);
```

3 ó Ler valores através do teclado

Métodos de leitura da classe Scanner

3.1 ó Ler um valor inteiro: int n = ler.nextInt();

3.2 ó Ler um valor real: float preco = ler.nextFloat();

3.3 ó Ler um valor real: double salario = ler.nextDouble();

3.4 ó Ler uma String: String palavra = ler.next();

é usado na leitura de palavras simples, ou seja, que não usam o caracter espaço (ou barra de espaço)

3.5 ó Ler uma String: String palavra = ler.nextLine();

é usado na leitura de palavras compostas, por exemplo Patinho Feio

21

Na leitura consecutiva de valores numéricos e String deve-se esvaziar o buffer do teclado antes da leitura de outro valor, por exemplo

```
System.out.println("Escreva um numero");
Int n = ler.nextInt();
ler.nextLine(); // esvazia o buffer do teclado
System.out.println("Escreva um texto");
String palavra = ler.nextLine();
```

```
import java.util.Scanner;

public class TestScanner8 {

    public static void main(String[] args) {
        Scanner input = new Scanner (System.in);
        String s;
        int x;

        System.out.print("Escreva um Inteiro: ");
        x = input.nextInt();

        System.out.println("O Valor digitado e: " + x);
        System.exit(0);
    }
}
```

**Entrada e saida de dados via teclado utilizando a classe JOptionPane do pacote javax.swing**

Para utilizar a classe JOptionPane numa classe java deve-se proceder da seguinte maneira:

1º importar o pacote javax.swing

```
import javax.swing.JOptionPane;
```

**A Classe Java *JOptionPane* oferece caixas de diálogo predefinidas que permitem aos programas exibir mensagens aos utilizadores;**

```
import javax.swing.*;           // carrega toda a biblioteca swing

public class TesteJOptionPane {

    public static void main(String[] args) {
        String s;

        s = JOptionPane.showInputDialog("String: ");
        int x = Integer.parseInt(s);
        JOptionPane.showMessageDialog(null, "Inteiro: " + x,
                                     "Resultado", JOptionPane.PLAIN_MESSAGE);
        // JOptionPane.showMessageDialog(null, "Inteiro: " + x);
        // funciona também, pois os dois últimos argumentos podem ser suprimidos
        System.exit(0);
    }
}
```

### Sintaxe:

```
JOptionPane.showMessageDialog(null,      õx      =      ö      +      x,      õTítuloö,
JOptionPane.PLAIN_MESSAGE);
```

O primeiro argumento é **null**, significando o vazio. O primeiro argumento do método *showMessageDialog()* é utilizado para posicionamento da janela. Ao ser null é ignorado e a janela é apresentada no centro da tela.

**õx = õ + x** significa que sairá na caixa de dialogo x = 5 (por exemplo)

**õTítuloö** significa o título da caixa de diálogo

**JOptionPane.PLAIN\_MESSAGE** significa caixa sem ícone

## Tipos de Ícones

Tipo de Ícone	Tipo de Mensagem
Ícone de erro	JOptionPane.ERROR_MESSAGE
Ícone de informação	JOptionPane.INFORMATION_MESSAGE
Ícone de advertência	JOptionPane.WARNING_MESSAGE
Ícone de pergunta	JOptionPane.QUESTION_MESSAGE
Sem ícone	JOptionPane.PLAIN_MESSAGE

23

A linha `System.exit( 0 )` é necessária em programas com interface gráfica, terminando o aplicativo Java.

O retorno Zero (`0`) para o método `exit()` indica que o programa finalizou com sucesso.

Valores diferentes de zero significam erros na execução e podem ser tratados por aplicativos que chamaram o programa Java.

A classe `System` faz parte do pacote padrão `java.lang`, que dispensa a importação (comando `import`) por ser acrescida aos seus programas por *default*

É possível converter `String` para qualquer tipo primitivo. Para tal utilizam-se os métodos das classes contentoras

- ó `int - Integer.parseInt( string )`
- ó `float - Float.parseFloat( string )`
- ó `double - Double.parseDouble( string )`

**Entrada de dados via teclado em terminal textual usando as bibliotecas `BufferedReader` e `InputStreamReader` .**

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class TesteReader {

    public static void main(String[] args) throws IOException {

        BufferedReader userInput = new BufferedReader
            (new InputStreamReader(System.in));
        System.out.print("Nome: ");
        String nome = userInput.readLine();
```

```

        Integer n = nome.length();
        System.out.println("Nome: " + nome + " tem " +
                           nome.length() + " caracteres");
        System.exit(0);
    }
}

```

## Apresentação(escrita) de dados no monitor

Saída de dados via tela usando o método `System.out.print()`, `System.out.println()` ou `System.out.printf()`.

24

O método `printf`, herado do c, está incorporado no pacote de classes `System.out`.

```
System.out.printf("%2d * %d = %3d\n", i, n, (i*n));
```

*obs:*

É `%2d` ó será substituído por um valor decimal usando 2 posições da tela.

É `\n` - pula uma linha

É `System.out.printf("%0,2d- %s\n", (i+1), nome);`

É *obs.*

É `0,2` ó coloca zeros a esquerda, por exemplo: 01, 02, 03, 04, ..., 09, 10

É `%d` - será substituído por um valor decimal (valores inteiros)

É `%i` - será substituído por um valor inteiro

É `%ld` ó será substituído por um valor inteiro longo (long)

É `%f` - será substituído por um valor real (%8.2f reservando 8 posições da tela das quais 2 serão usadas para as casas decimais)

É `%c` - será substituído por um caractere

É `%s` - será substituído por uma cadeia de caracteres

```

import java.util.Scanner;

public class TesteSaida {
    public static void main(String[] args) {
        Scanner entrada = new Scanner (System.in);
        String s;
        int x;
        System.out.print("Inteiro: ");
        s = entrada.nextLine();
        x = Integer.parseInt(s);
        System.out.println("Valor: " + x);
        System.out.printf("Valor: %03d\n ", x);
        System.exit(0);
    }
}

```

## Métodos para funções matemáticas padrões

Java possui uma classe: **Math** que possui diversos métodos para resolver funções matemáticas.

A classe `java.lang.Math` de Java oferece uma série de métodos estáticos para exponenciação, trigonometria, funções de máximo e mínimo, etc. Como todos esses métodos são estáticos, é possível referenciar-los pelo nome da classe sem ter de criar uma instância da mesma.

Principais métodos da classe Math:

Método	Descrição
<code>abs(x)</code>	Valor absoluto de x (este método também tem versões para valores float, int e long)
<code>ceil(x)</code>	Arredonda x para o menor inteiro não menor que x
<code>cos(x)</code>	Cosseno trigonométrico de x (x em radianos)
<code>exp(x)</code>	Método exponencial
<code>floor(x)</code>	Arredonda x para o maior inteiro não maior que x
<code>log(x)</code>	Logarítmico natural de x (base e)
<code>max(x, y)</code>	Maior valor entre x e y (este método também tem versões para valores float, int e long)
<code>min(x, y)</code>	Menor valor entre x e y (este método também tem versões para valores float, int e long)
<code>pow(x, y)</code>	x elevado à potência y
<code>sin(x)</code>	Seno trigonométrico de x (x em radianos)
<code>sqrt(x)</code>	Raiz quadrada de x
<code>tan(x)</code>	Tangente trigonométrica de x (x em radianos)
etcí	í

## Exercícios

*Algumas fórmulas apresentadas nos enunciados estão na sua forma MATEMÁTICA e não na forma como devem ser escritas nos programas.*

*As indicações entre parênteses são sugestões de nomes de variáveis(escrever os nomes das variáveis conforme a convenção).*

*Elaborar o diagrama de blocos e/ou o pseudocódigo (português estruturado) em todos os casos(em comentário no programa).*

- 1 - Faça um programa para ler um valor inteiro e que imprima o seu antecessor e sucessor.
- 2 - Faça um programa para ler um valor real e exiba o próximo inteiro.
- 3 - Crie um programa que receba três nomes quaisquer, por meio da linha de execução do programa, e que os imprima na tela da seguinte maneira: o primeiro e o último nome serão impressos na primeira linha um após o outro, o outro nome ( o segundo) será impresso na segunda linha, tudo no mesmo comando.
- 4 - Faça um programa que receba a quantidade de três produtos e o valor de três produtos, no seguinte formato: Quantidade1 Valor1 Quantidade2 Valor2 Quantidade3 Valor3. O programa deve calcular estes valores seguindo a fórmula total = Quantidade1 x Valor1 + Quantidade2 x Valor2 + Quantidade3 x Valor3. O valor total deve ser apresentado no final da execução do programa.
- 5 - Qual é a saída quando o programa abaixo é executado:

```
public class Alfabeto {
    public static void main(String args[]) {
        int i = 65;
        char c = (char) i;
        System.out.println("Após o " + c + " vem o " + (c + 1));
    }
}
```

- 6 - Faça um programa que receba a largura e o comprimento de um lote de terra e mostre a área total existente em metros quadrados.
- 7 - Faça um programa que leia dois valores inteiros e imprima o resto da divisão do primeiro pelo segundo sem usar o %.

8 - Crie um programa que receba quatro valores quaisquer e mostre a média entre eles, o somatório entre eles e o resto da divisão do somatório por cada um dos valores.

9 - Faça um programa para imprimir a tabela abaixo apenas com a instrução  
System.out.println();

Nome Arquivo	Título	Localização
<hr/>		
Cap_1.doc	ÓO que é Javaö	c:\Livro\p1
Cap_11.doc	ÓMultithredingö	c:\Livro\p1
Cap_14.doc	ÓEventosö	c:\Livro\p2
Ap_1.doc	ÓReferênciaö	c:\Livro\Ap

10 - Qual das afirmativas são verdadeiras a respeito do código abaixo? (marque uma ou mais)

```
public class CarTeste {
    public static void main(String args[]) {
        byte b1 = 3;
        byte b2 = 5;
        byte b = (byte) b1 ó b2;
        System.out.println(b);
    }
}
```

O código não compila porque deveria usar uma conversão (cast) explícito para converter int para byte.

O código não compila porque o tipo primitivo byte não aceita valores negativos.

O código compila, executa e imprime: -2

O código compila, executa e imprime: 2

O código compila, executa e não imprime nada

11 - Leia um número inteiro e escreva seu sucessor e antecessor.

12 - Leia dois números e escreva o dividendo, divisor, quociente e resto.

13 - Calcule a dívida do cheque especial ao ser quitada 6 meses depois a uma taxa de 5% de juros. O valor do empréstimo deve ser informado.

14 - Dada a base e a altura de um retângulo, calcule o perímetro, a área e a diagonal.

15 - Dada o lado de um quadrado, calcule o perímetro, a área e a diagonal.

16 - Dado o raio de um círculo, calcule o perímetro e a área.

- 17 - Dados os três lados de um paralelepípedo, calcule o perímetro, a área, o volume e a diagonal.
- 18 - Dados dois catetos de um triangulo retângulo, calcule a hipotenusa.
- 19 - Dada a razão de uma PA e seu primeiro termo, calcular o 20º termo.
- 20 - Dada a razão de uma PG e seu primeiro termo, calcular o 20º termo.
- 21 - Dado um horário, calcule quantos minutos e segundos transcorreram desde o início do dia.
- 22- Dado o valor do salário-mínimo e um determinado salário, calcule quantos salários-mínimos estão contidos nele.

## Estruturas de controlo de fluxo

A ordem de execução normal de comandos num programa em Java é sequencial. As estruturas de controlo de fluxo permitem modificar essa ordem natural de execução através dos mecanismos de escolha ou de iteração.

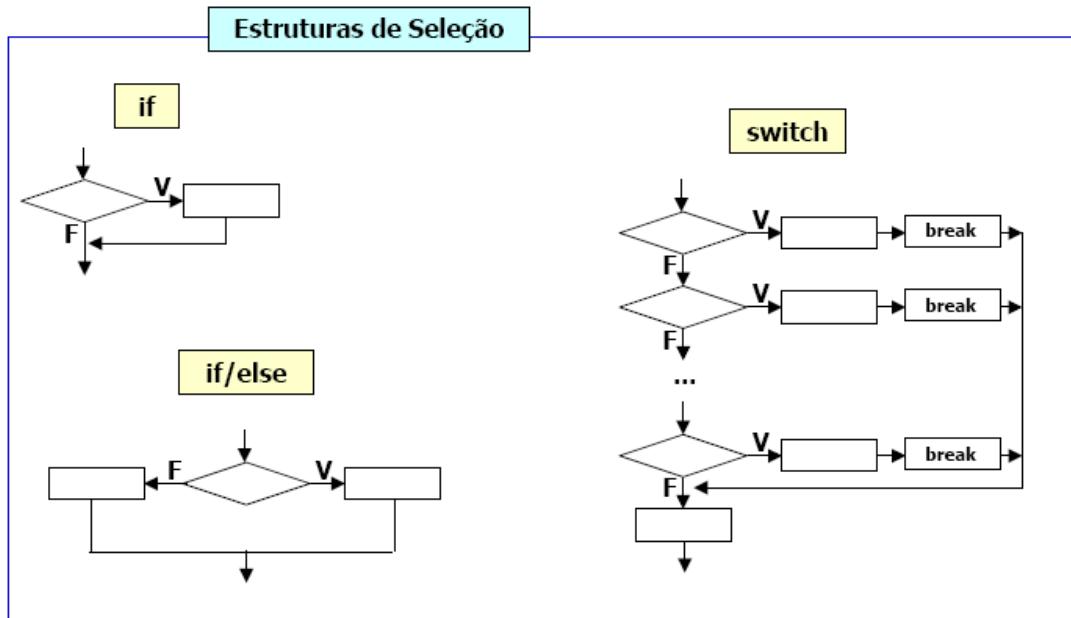
## Controlo de Fluxo

O Controle de fluxo é a habilidade de ajustar a maneira como um programa realiza as suas tarefas. Por meio de estruturas e instruções especiais, essas tarefas podem ser executadas selectivamente, repetidamente ou excepcionalmente. Não fosse o controlo de fluxo, um programa poderia executar apenas uma única sequência de tarefas, perdendo completamente uma das características mais interessantes da programação: a dinâmica.

Em Java podemos classificar as estruturas de controlo de fluxo em basicamente quatro categorias:

Categoria	Estrutura
Tomada de decisões	if-else, switch-case
Laços ou repetições	for, while, do-while
tratamento de exceções	try-catch-finally, throw
Outros	break, continue, return, label

## Estruturas ou comandos de seleção ou decisão



As estruturas de decisão permitem que um programa possa realizar diferentes alternativas de sequencias de instruções durante a sua execução. Dependendo do valor de uma expressão ou de uma variável boolena, o programa executa uma ou outra sequencia de instruções.

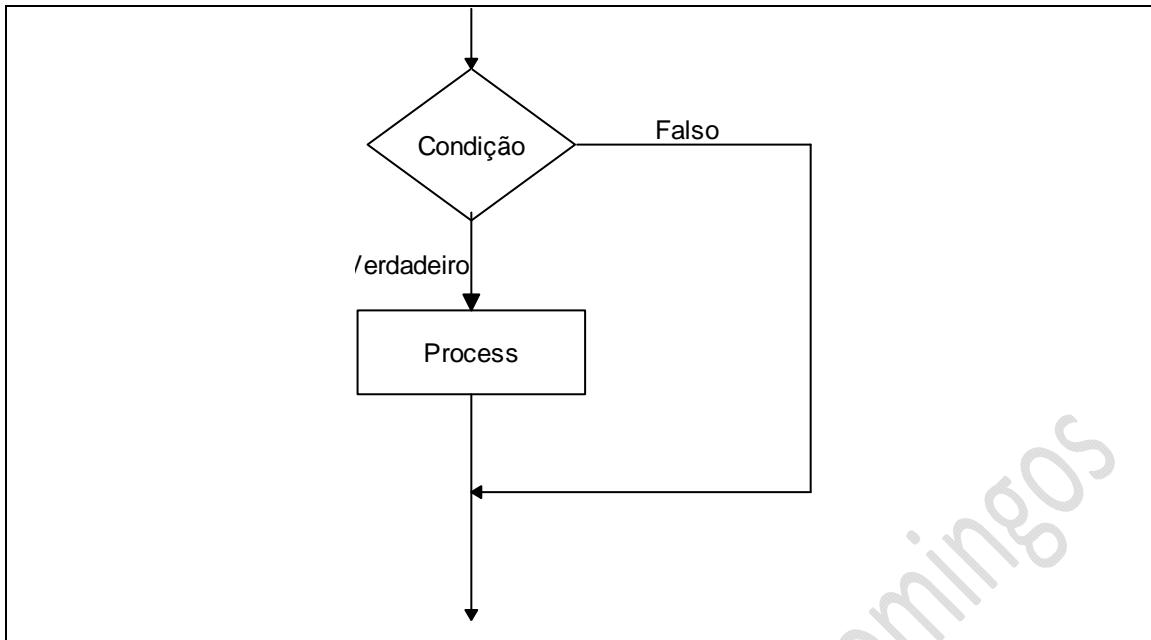
Existem três categorias distintas: selecção simples, selecção dupla e selecção múltipla, as quais são abordadas a seguir.

## Estrutura de decisão simples

### Estrutura de decisão simples

Sintaxe

```
if(<expressão boolena>)
    instrução
ou
if(<expressão boolena>){
    <sequencia de instruções>
}
```



Quando o computador encontra e executa este comando, o que é realizado em primeiro lugar é o teste da expressão booleana. Se ela resultar em verdadeiro, então a instrução ou sequência de instruções é executada. Caso contrário, o programa avança e executa as instruções posteriores ao comando if.

```

import java.util.Scanner;

public class TrocaValorDasVariaveis{

    //Imprimir os numeros em ordem crescente
    public static void main(String[] args){
        int variavel1, variavel2, variavelAuxiliar;
        Scanner teclado = new Scanner(System.in);
        System.out.println("\nDigite o valor para a primeira variável");
        variavel1=teclado.nextInt();
        System.out.println("\nDigite o valor para a segunda variável");
        variavel2 = teclado.nextInt();
        System.out.println("\n\n Os valores foram inseridos na seguinte ordem:\t"+ "1º-\t"+variavel1+"\t2º->\t"+variavel2);
        if(variavel1> variavel2){
    
```

```

variavelAuxiliar = variavel1;
variavel1 = variavel2;
variavel2=variavelAuxiliar;
}
System.out.println("\n\n Valores em ordem crescente:\t"+1°->\t"+variavel1+"\t2°-
>\t"+variavel2);
}
}

```

32

## Estrutura de decisão dupla

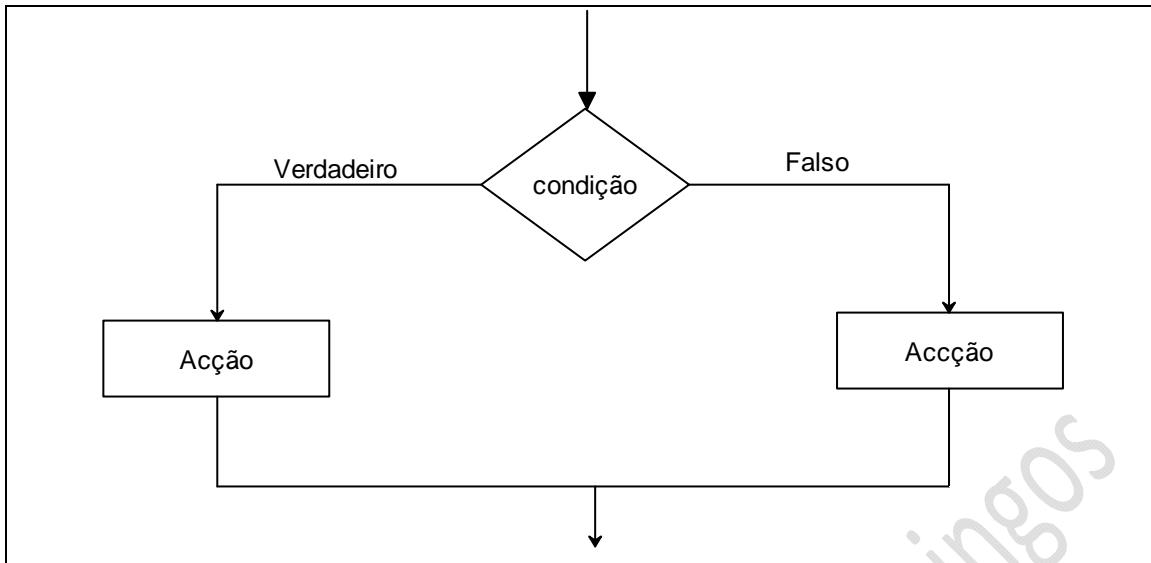
O comando de decisão ou selecção dupla pode ser comparado com a bifurcação de um caminho. É preciso escolher uma das alternativas e segui-la.

Em programação, o comando de selecção dupla serve para executar sempre uma de duas alternativas de sequencias de instruções, de acordo com a verificação de uma condição. A sintaxe da estrutura é mostrada a seguir.

```

if(<expressão booleana>){
    <sequencia de instruções 1>
}
else{
    <sequencia de instruções 2>
}

```



Primeiramente, o computador testa a expressão booleana. Caso resulte em *verdadeiro*, então a sequencia de instruções 1 é executada. Caso contrário, a sequencia de instruções 2 é executada. Assim, apenas uma e sempre uma das sequencias de instruções é executada durante uma mesma realização do comando de selecção dupla.

Após a operação de uma das sequencias de instruções, o programa continuar a sua execução na primeira linha de código após o conjunto de instruções da estrutura de selecção.

```

import java.util.Scanner;
public class SituacaoAluno{

    public static void main(String[] args){
        float nota1, nota2, nota3, media;
        int matricula;
        Scanner teclado = new Scanner(System.in);

        System.out.println("\nEscreva numero de matricula do aluno:\t");
        matricula=teclado.nextInt();
        System.out.println("\nEscreva a primeira nota do aluno:\t");
        nota1=teclado.nextFloat();
        System.out.println("\nEscreva a segunda nota do aluno:\t");
        nota2=teclado.nextFloat();
    }
}
  
```

```

System.out.println("\nEscreva a terceira nota do aluno:\t");
nota3=teclado.nextFloat();

media = (nota1+nota2+nota3)/3;

if(media <9.5)
    System.out.println("O aluno Nº:\t"+matricula+"\tDeve fazer o Exame. Ele teve
media:\t"+media);
else
    System.out.println("O aluno Nº:\t"+matricula+"\tFicou aprovado com a
media:\t"+media);
}

}

```

34

Em muitos casos, além de ser elegante, o uso da estrutura de selecção aninhada pode ser mais eficiente.

Nos exemplos seguintes verifica-se o facto de que ambos os programas lêem três números digitados pelo utilizador e identificam o maior deles, porém, um dos programas faz menos verificações que o outros, ganhando eficiência.

```

import java.util.Scanner;
public class OrdenaTresNumeros{

public static void main(String[] args){
int numero1, numero2, numero3;

Scanner teclado = new Scanner(System.in);

System.out.println("\nDigite o primeiro numero:\t");
numero1 = teclado.nextInt();

```

```
System.out.println("\nDigite o segundo numero:\t");
numero2 = teclado.nextInt();
System.out.println("\nDigite o terceiro numero:\t");
numero3 = teclado.nextInt();

System.out.println("\nOrdenacao do numeros");

if((numero1>numero2)&&(numero1>numero3)){

    System.out.println("O maior numero digitado foi:\t"+numero1);
}

if((numero2>numero1)&&(numero2>numero3)){

    System.out.println("O maior numero digitado foi:\t"+numero2);
}

if((numero3>numero1)&&(numero3>numero2)){

    System.out.println("O maior numero digitado foi:\t"+numero3);
}

}
```

```
import java.util.Scanner;
public class OrdenaTresNumerosSegundo{

    public static void main(String[] args){
        int numero1, numero2, numero3;
```

```
Scanner teclado = new Scanner(System.in);

System.out.println("\nDigite o primeiro numero:\t");
numero1 = teclado.nextInt();

System.out.println("\nDigite o segundo numero:\t");
numero2 = teclado.nextInt();
System.out.println("\nDigite o terceiro numero:\t");
numero3 = teclado.nextInt();

System.out.println("\nOrdenacao do numeros");

if(numero1>numero2){
    if(numero1>numero3){
        System.out.println("O maior numero digitado foi:\t"+numero1);
    }
    else{
        System.out.println("O maior numero digitado foi:\t"+numero3);
    }
}
else{
    if(numero2>numero3){
        System.out.println("O maior numero digitado foi:\t"+numero2);
    }
    else{
        System.out.println("O maior numero digitado foi:\t"+numero3);
    }
}
```

O programa do exemplo a seguir recebe do utilizador três números inteiros e verifica se eles podem ser tamanhos dos lados de um triângulo. Caso verdadeiro, o programa ainda indica a classificação do triângulo.

```
import java.util.Scanner;
public class Triangulos{

    public static void main(String[] args){
        Scanner teclado = new Scanner(System.in);
        int lado1, lado2, lado3;
        int soma1, soma2, soma3;

        System.out.println("Introduza o lado 1 do triangulo:\t");
        lado1=teclado.nextInt();
        System.out.println("Introduza o lado 2 do triangulo:\t");
        lado2=teclado.nextInt();
        System.out.println("Introduza o lado 3 do triangulo:\t");
        lado3=teclado.nextInt();
        soma1 = lado1+lado2;
        soma2 = lado1+lado3;
        soma3 = lado2 +lado3;

        if((lado1==lado2) &&(lado2==lado3)){
            System.out.println("Triangulo equilatero");
        }
        else{
            if((soma1>lado3)&&(soma2>lado2)&&(soma3>lado1)){
                if((lado1!=lado2)&&(lado1!=lado3)&&(lado2!=lado3)){
                    System.out.println("Triangulo escaleno\n");
                }
                else{
                    System.out.println("Triangulo isosceles\n");
                }
            }
            else{
                System.out.println("Os tres lados nao formam um triangulo");
            }
        }
    }
}
```

```
    }  
}  
}  
}
```

38

## Operador condicional ou ternário (?:)

O Java possui um operador ternário (três operandos), designado por operador condicional, que é indicado para abreviar `if` `else` simples.

É uma alternativa útil ao `if` `else`

A sua sintaxe é dada por

`<expressão booleana>? expressão1:expressão2;`

Se a expressão booleana retornar `true`, o resultado será `expressão1`, senão o resultado será a `expressão2`

```
int valor1 = 120, valor2=0;  
int maior;  
maior = (valor1>valor2)? 100:200;  
System.out.println("O maior valor é: "+maior);
```

O programa a seguir gera dois reais aleatórios e utiliza o operador condicional para determinar qual é o menor e qual é o maior.


## Comando de selecção múltipla

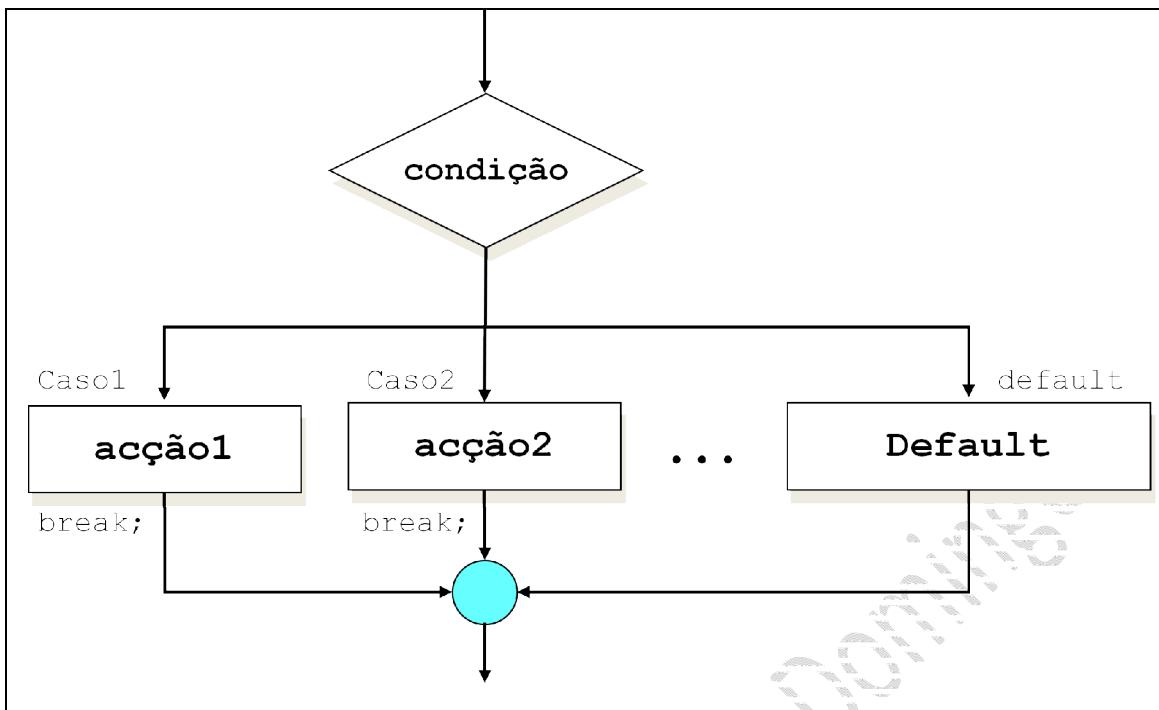
O comando de selecção `if` é bastante poderoso e largamente utilizado. Entretanto, ele não é o único comando de selecção existente. Há um comando que, embora seja menos

abrangente que o comando if, resolve muitas situações de forma clara e elegante. Trata-se do comando de seleção múltipla.

O comando de seleção múltipla sempre pode ser substituído por um ou mais comandos if aninhados, mas ele torna o código muito mais claro quando se quer executar várias sequências de comandos diferentes, sendo que cada sequência deva corresponder a um determinado valor de uma expressão.

#### Forma geral

```
switch(<expressão>){  
    case <valor1>:  
        <sequencia de instruções 1>  
        break;  
    case <valor2>:  
        <sequencia de instruções 2>  
        break;  
    case <valorN>:  
        <sequencia de instruções N>  
        break;  
    default:  
        <sequencia de instruções>  
}
```



40

Na estrutura do comando de selecção múltipla, a expressão tradicionalmente é uma expressão aritmética ou uma variável do tipo numérico(a partir da versão 7 do JDK a expressão pode ser uma String), e associa-se o valor da expressão ao valor de um determinado case. Assim, apenas a sequencia de instruções do bloco case correspondente será executada. É importante observar que cada sequencia de instruções é encerrada pelo comando break.

A cláusula defoult é opcional. Ela marca a sequencia de instruções padrão, que deve ser executada quando nenhuma comparação obtiver sucesso. Se não estiver presente, nada será executado nesse caso.

Na linguagem java, cada case deve servir para comparar o resultado da expressão com um valor específico. Não se permite comparações com o resultado de outras expressões. No exemplo seguinte utiliza-se o comando switch na construção de um programa que simula o funcionamento de uma urna electrónica.

```

import java.util.Scanner;

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

```

```
int opcao;

Scanner teclado = new Scanner(System.in);
System.out.println("\n\nURNA ELECTRONICA - O SEU VOTO VAI PARA...\n");
opcao = teclado.nextInt();
switch(opcao){

    case 1:
        System.out.println("Candidato Escolhido: Domingos Marques\n");
        break;

    case 2:
        System.out.println("Candidato Escolhido: Vicente Matamba\n");
        break;

    case 3:
        System.out.println("Candidato Escolhido: Garibaldina Meirelis\n");
        break;

    case 4:
        System.out.println("Candidato Escolhido: Maria do Rosario Hangalo\n");
        break;

    default:
        System.out.println("Numero digitado invalido: Voto anulado\n");

    }
}
}
```

Quando o comando break não termina uma sequencia de instruções cuja comparação resultou verdadeira, a sequencia de instruções seguinte é executada sem que o teste correspondente seja efectuado. Na prática, o que acontece é que quando uma comparação do switch resultar em verdadeiro, todas as sequencias de instruções seguintes são executadas até que o primeiro comando break seja encontrado. É preciso, portanto, ficar atento e finalizar cada sequencia de instruções com o comando break evitando, assim, erros de programação muitas vezes difíceis de detectar.

# Exercícios

*Algumas fórmulas apresentadas nos enunciados estão na sua forma MATEMÁTICA e não na forma como devem ser escritas nos programas.*

*As indicações entre parênteses são sugestões de nomes de variáveis(escrever os nomes das variáveis conforme a convenção).*

*Elaborar o diagrama de blocos e/ou o pseudocódigo (português estruturado) em todos os casos(em comentário no programa).*

42

- 1 Construa um programa que leia o valor de uma conta de luz (CL) e, caso o valor seja maior que AKZ 50,00 apresente a mensagem: õVocê está a gastar muitoö. Caso contrário não exiba mensagem nenhuma.
- 2 Construa um programa que leia o valor de uma conta de luz (CL) e, caso o valor seja maior que AKZ 50,00 apresente a mensagem: õVocê está gastar muitoö. Caso contrário exiba a mensagem: õO seu gasto foi normalö.
- 3 Construa um programa que, tendo como dados de entrada a altura (H) e o sexo (S) de uma pessoa calcule e apresente o seu peso ideal utilizando as seguintes fórmulas:  
Para homens: Peso ideal (P) =  $(72,7 * H) - 58$   
Para mulheres: Peso ideal (P) =  $(62,1 * H) - 44,7$
- 4 Construa um programa que determine quanto será gasto para encher o tanque de um carro (VG), sabendo-se que o preço da gasolina é de AKZ 1,80 e o preço do álcool é de AKZ 1,00. O utilizador fornecerá os seguintes dados: Tipo de carro (TC) (G ó gasolina ou A ó álcool) e Capacidade do tanque (CT), em litros.
- 5 Construa um programa que leia um número inteiro (positivo ou negativo) e apresente o seu módulo (número sem sinal).
- 6 Construa um programa que leia o preço de um produto (P) e apresente a mensagem: õEm promoçãoö, caso o preço seja maior ou igual a AKZ 50,00 e menor ou igual a AKZ 100,00. Caso contrário, deve apresentar a mensagem: õPreço Normalö.
- 7 Construa um programa que, recebendo os valores de vendas do mês de determinado vendedor (VM) e o nome do mesmo (NOME), apresente o nome, quando o valor da venda estiver entre AKZ 10000,00 e AKZ 50000,00 (inclusive).

8 Construa um programa que calcule o novo salário (SAL\_NOVO) de um funcionário. Considere que o funcionário deverá receber um reajuste de 15% caso seu salário (SAL) seja menor que 500. Se o salário for maior ou igual a 500, mas menor ou igual a 1000, o reajuste deve ser de 10%. Caso o salário seja maior que 1000, o reajuste deve ser de 5%.

9 Construa um programa que leia dois números (A e B). Caso A seja igual a B, apresentar a soma dos dois. Caso um seja maior que o outro, apresentar a diferença entre os dois números (sempre lembrando que a diferença entre dois números é SEMPRE positiva).

10 Construa um programa que leia o código de um livro (CL) e apresente a categoria do livro, conforme a tabela abaixo:

Código do Livro (CL)	Categoria
A	Ficção
B	Não-Ficção
Qualquer outro código	Inválido

11 Construa um programa que receba a leitura do termômetro (T). Caso a temperatura esteja abaixo de 100°C, apresentar a mensagem de que a temperatura está muito baixa. Caso a temperatura esteja entre 100°C e 200°C (inclusive), apresentar a mensagem de que a temperatura está baixa. Caso a temperatura esteja acima de 200°C e inferior a 500°C, apresentar a mensagem de que a temperatura está normal. Caso contrário, apresentar a mensagem de que a temperatura está muito alta.

12 Construa um programa que leia a quantidade de dinheiro existente no caixa de uma empresa (CAIXA), a quantidade de produtos a ser comprada (QTD) e o preço de cada unidade (PR). Caso o valor total da compra seja superior a 80% do valor em caixa, a compra deve ser feita a prazo (3x), com juros de 10% sobre o valor total. Caso contrário, a compra deverá ser realizada a vista, onde a empresa receberá 5% de desconto. Apresentar a forma de pagamento escolhida e o valor a ser pago (total a vista ou total a prazo), dependendo da escolha realizada pelo programa.

13 Construa um programa que leia as informações de: horas trabalhadas (HT), valor da hora trabalhada (VH). Calcule e apresente o salário líquido do empregado, baseado nas tabelas abaixo.

OBS: Salário Líquido = Salário Bruto ó INSS ó Imposto de Renda

Salário Bruto = Horas trabalhadas \* Valor da hora trabalhada

INSS = 11% do salário líquido

Imposto de Renda → após descontar o INSS usar esse valor e ler a alíquota do imposto de renda e parcela a deduzir na tabela abaixo

Salário Bruto ó INSS	Alíquota	Valor a Deduzir
Até \$1.257,12	Isento (0%)	
De \$1.257,13 até \$2.512,08	15%	\$188,57
Mais que \$2.512,08	27,5%	\$502,58

44

OBS: Imposto de Renda = Alíquota \* (Salário Bruto ó INSS) ó Valor a Deduzir

Repita o exercício 13, só que agora, a porcentagem de desconto de INSS não é mais fixa. O desconto acontece de acordo com a tabela abaixo:

Salário Bruto	Alíquota
Até \$800,45	7,65%
De \$800,46 até \$900,00	8,65%
De \$900,01 até \$1.334,07	9,00%
De \$1.334,08 até \$2.668,15	11,00%

OBS: Para Salário Bruto acima de \$2.668,15 o valor de desconto é fixo e vale \$293,50

14 Construa um programa que calcule e apresente quanto deve ser pago por um produto considerando a leitura do preço de etiqueta (PE) e o código da condição de pagamento (CP). Utilize para os cálculos a tabela de condições de pagamento a seguir:

Código da condição de pagamento	Condição de pagamento
1	À vista em dinheiro ou cheque, com 10% de desconto
2	À vista com cartão de crédito, com 5% de desconto
3	Em 2 vezes, preço normal de etiqueta sem juros
4	Em 3 vezes, preço de etiqueta com

	acréscimo de 10%
--	------------------

15 Construa um programa que tendo como dados de entrada o preço de um produto (PR) e seu código de origem (CO), apresente o preço e a sua procedência, de acordo com a tabela abaixo:

Código de Origem (CO)	Procedência
1	Sul
2	Sudeste
3	Centro-Oeste
4	Norte
5	Nordeste

45

16 Construa um programa que leia o ano de nascimento de uma pessoa (AN) e mostre a sua idade e, também verifique e mostre se essa pessoa já tem idade para votar (18 anos ou mais) e se já pode conseguir o seu bilhete de identidade (18 anos ou mais).

17 Construa um programa que calcule e apresente a idade REAL de uma pessoa. Será fornecido pelo utilizador:

DN ó dia do nascimento

DH ó dia da data de hoje

MN ó mês do nascimento

MH ó mês da data de hoje

AN ó ano do nascimento

AH ó ano da data de hoje

18 Construa um programa que, dados os comprimentos dos três lados (A, B e C) de um triângulo, verifique o tipo de triângulo formado. Apresentar qual é o tipo. Sabe-se que:

Triângulo do tipo Equilátero ó possui os três lados iguais

Triângulo do tipo Isósceles ó possui dois lados iguais

Triângulo do tipo Escaleno ó possui os três lados diferentes

19 Construa um programa que, dada a idade de um nadador (ID), classifique-o numa das seguintes categorias e apresente a categoria:

Idade (ID)	Categoria
5 até 7 anos	Infantil A
8 até 10 anos	Infantil B
11 até 13 anos	Juvenil A
14 até 17 anos	Juvenil B

Acima de 18 anos	Adulto
------------------	--------

20 Construa um programa que leia o código de um determinado produto (CP) e mostre a sua classificação, utilizando a seguinte tabela:

Código do Produto (CP)	Classificação
1	Alimento não perecível
2, 3 ou 4	Alimento perecível
5 ou 6	Vestuário
7	Higiene Pessoal
8 ou 9	Limpeza e Utensílios Domésticos
Qualquer outro código	Inválido

21c Construa um programa que indique o que o motorista deve fazer de acordo com a cor do semáforo (CS) e distância do cruzamento (DC) fornecida pelo usuário. As condições são:

(V) Vermelho = Parar

(A) Amarelo = se a distância do cruzamento for menor que 5 metros = Passar com cuidado

= se a distância do cruzamento for maior ou igual a 5 metros = Parar

(D) Verde = Passar

22 Construa um programa para calcular o valor a ser pago pelo período de estacionamento do automóvel (PAG). O utilizador entra com os seguintes dados: hora (HE) e minuto (ME) de entrada, hora (HS) e minuto (MS) de saída. Sabe-se que este estacionamento cobra hora cheia, ou seja, se passar um minuto ele cobra a hora inteira. O valor da hora é AKZ 4,00.

23 Construa um programa para calcular o valor a ser pago pelo período de estacionamento do automóvel (PAG). O utilizador entra com os seguintes dados: hora (HE) e minuto (ME) de entrada, hora (HS) e minuto (MS) de saída. Sabe-se que este estacionamento cobra hora cheia, ou seja, se passar um minuto ele cobra a hora inteira.

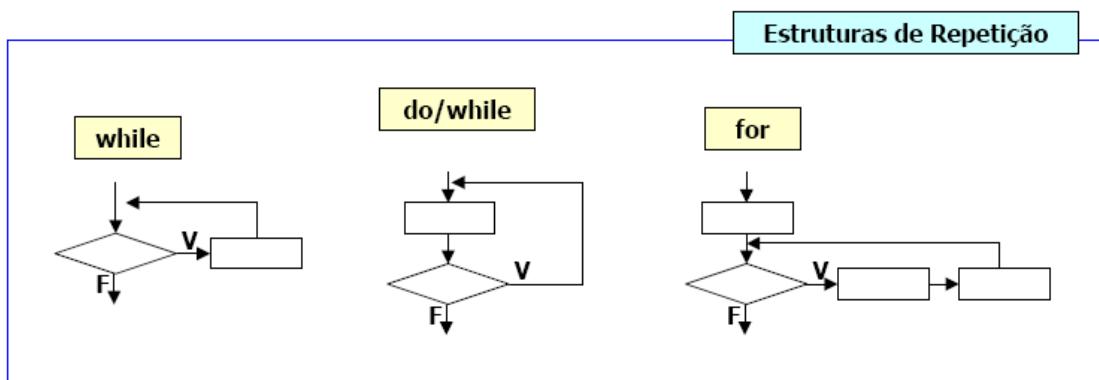
O valor cobrado pelo estacionamento é:

AKZ 4,00 para 1 hora de estacionamento

AKZ 6,00 para 2 horas de estacionamento

AKZ 1,00 por hora adicional (acima de 2 horas)

## Estruturas de repetição (Laços, Ciclos, Loops)



A repetição é uma das tarefas mais comuns da programação, sendo utilizada para realizar contagens, totalizações, a obtenção de múltiplos dados, impressão etc

Um ciclo serve para realizar um processo repetidas vezes. As estruturas de repetição também são chamadas ciclos, laços ou loop. O código incluído entre as chavetas {} (opcionais quando o processo repetitivo consta de apenas uma linha), será executado apenas quando se cumpre determinadas condições. É preciso prestar atenção aos ciclos infinitos, facto que ocorre quando a condição que faz com que o ciclo termine (expressão boolean) não chega a cumprir-se. Trata-se de uma falha muito típica, comum entre os programadores pouco experientes.

## O comando for

### Estrutura de repetição simples

O Java oferece uma poderosa coleção de construções de loop. O mais versátil deles é o loop for.

A estrutura for é controlada por um contador. O contador é inicializado antes da iteração. Depois a condição é testada, no fim de cada iteração, até atingir o limite estabelecido.

O loop for é também conhecido como repetição simples. A repetição simples é a execução consecutiva de uma instrução ou bloco de instruções por um número conhecido e fixo de vezes, o que muitas vezes se denomina laço automático. Cada repetição efectuada é o que se chama iteração.

#### Sintaxe

```
for(inicialização; expressão booleana; iteração)
    instrução
```

ou

```
for(inicialização; expressão booleana; iteração){
    instruções
}
```

O for tem três secções distintas de controlo, todas opcionais, delimitadas por um par de parênteses, nos quais cada secção é separada de outra por um ponto-e-vírgula.

A porção de inicialização ajusta a variável de controlo do loop para um valor inicial. A condição é uma expressão booleana que testa a variável de controlo de loop. Se o resultado for verdadeiro(true), o loop for continua a ser executado. Se for falso(false), o loop é encerrado. A expressão de iteração determina como a variável de controlo do loop é alterada cada vez que o loop é executado. Eis um pequeno programa que ilustra o loop for.

```
import java.util.Scanner;

public class Tabuada {

    public static void main(String args[]) {
        Scanner teclado = new Scanner(System.in);
        System.out.println("Informe um numero inteiro:");
        int n = teclado.nextInt();

        for (int i=1; i<=10; i++) {
            System.out.println(i + " * " + n + " = " + (i*n));
        }
    }
}
```

```
import java.util.Scanner;
public class Factorial
{
```

```

public static void main(String args[]) {
    Scanner teclado = new Scanner(System.in);
    int num,fatorial=1;
    System.out.println("Informe um numero inteiro:");
    num=teclado.nextInt();
    for(int i=num;i>=1;i--)
        fatorial*=i;
    System.out.println("O factorial do numero e:\t"+fatorial);
}

```

50

## Estruturas de Repetição Condicionais

As estruturas de repetição condicionais são os comandos cujo controlo da repetição de execução é feito por meio da avaliação de uma expressão lógica associada ao comando, sendo então chamados de laços condicionais. Essas estruturas são adequadas para permitir a execução repetida de uma ou mais instruções por um número indeterminado de vezes, isto é, um número que não é conhecido durante a programação, mas que se torna conhecido durante a execução do programa, tal como um valor fornecido pelo utilizador, obtido de um ficheiro, ou ainda de cálculos realizados com outros dados. No Java, tais estruturas são o while e do while.

### O comando while

O comando while repete uma instrução ou um bloco de instruções enquanto a expressão de controlo for verdadeira

<pre> inicialização while(condição) {     Instrução } </pre>
--

```
    iteracã  
}
```

51

A condição pode ser qualquer expressão booleana. O corpo do loop será executado enquanto a expressão condicional representada por condição for satisfeita. Quando condição se tornar falsa, o controlo passa para a próxima linha de código imediatamente após o loop. As chavetas { } são desnecessárias se o que estiver a ser repetido for apenas uma única instrução.

Como o loop while avalia a expressão condicional no início do loop, o corpo do loop não será executado nenhuma vez se a condição for inicialmente falsa.

```
public class WhileDemo  
{  
    public static void main(String args[])  
    {  
        int contador=10;  
  
        while(contador>0)  
        {  
            System.out.println(contador+"!");  
            contador--;  
        }//fim de while.  
        System.out.println("Ok!");  
    }//fim de main()  
}//fim de class WhileDemo
```

```
import java.util.Scanner;  
  
// Verifica se um número inteiro é primo.  
  
public class NumeroPrimo {
```

```

public static void main(String args[]) {
    Scanner teclado = new Scanner(System.in);
    System.out.println("Informe um numero inteiro:");
    int n = teclado.nextInt();
    boolean ehPrimo = true;
    int i = 2;
    while ((ehPrimo == true) && (i <= (n / 2))) {
        if ((n % i) == 0)
            ehPrimo = false; // encontrou um divisor (não eh primo)
        else i++; // próximo divisor usando autoincremento
    }
    if (ehPrimo == true)
        System.out.println(n + " \\"eh\\" um numero primo.");
    else
        System.out.println(n + " \\"não eh\\" um numero primo.");
}
}

```

## O comando do..while

Muitas situações é desejável que o corpo do loop seja executado pelo menos uma vez, mesmo que a condição de controlo seja inicialmente falsa. Em outras palavras, há ocasiões em que a condição de término deve ser testada no fim do loop, e não no início.

Para esses casos, a linguagem Java oferece um loop que faz exatamente isso: o do-while. O loop do-while sempre executa o seu corpo pelo menos uma vez, porque a expressão condicional está no fim do loop.

Forma geral
inicialização do { Instrução iteracção }while(condição);

Cada iteração do loop do-while executa primeiro o corpo do loop e depois avalia a expressão condicional representada por condição. Se a expressão for verdadeira, o loop será repetido. Caso contrário, o loop termina. Como acontece com todos os loops da linguagem Java, a condição deve ser uma expressão booleana.

```
public class DemoDoWhile{  
    public static void main(String[] args){  
  
        int a =0;  
        do{  
            System.out.println("Java");  
            a=a+1;  
        }while(a<5);  
    }  
}
```

```
import javax.swing.*;  
  
public class SomaDoWhile{  
    public static void main(String[] args){  
  
        int numero = 10, soma = 0;  
        do{  
            soma += numero;  
            numero++;  
        } while (numero < 10);  
        JOptionPane.showMessageDialog(null,"Total =" + soma);  
    }  
}
```

## Exercícios

*Algumas fórmulas apresentadas nos enunciados estão na sua forma MATEMÁTICA e não na forma como devem ser escritas nos programas.*

*As indicações entre parênteses são sugestões de nomes de variáveis(escrever os nomes das variáveis conforme a convenção).*

*Elaborar o diagrama de blocos e/ou o pseudocódigo (português estruturado) em todos os casos(em comentário no programa).*

1 Construa um programa que apresente o peso total que será carregado por um caminhão. Sabe-se que esse caminhão carrega 25 caixas, com pesos diferentes. Será entrada do programa o peso (P) de cada uma das caixas.

2 Construa um programa que leia a quantidade (Q) e o preço (PR) de 45 produtos diferentes, comprados por uma empresa, e apresente o total gasto por ela.

3 Construa um programa que leia o número de horas trabalhadas diárias (NH) de um funcionário por um período de 30 dias (ele trabalhou todos os 30 dias) e apresente o total de horas trabalhadas por ele nesse período.

4 Construa um programa que leia o número de horas trabalhadas diárias (NH) de um funcionário por um período de 30 dias (ele trabalhou todos os 30 dias) e apresente o salário bruto recebido por ele nesse período, sabendo que o valor do salário base é AKZ 10,00/hora trabalhada.

6 A conversão de graus Fahrenheit para Celsius é obtida pela fórmula  $C = \frac{5}{9}(F - 32)$ . Construa um programa que calcule e apresente TODAS as temperaturas (em Celsius) correspondentes aquelas em Fahrenheit de 1 até 50, ou seja, para cada temperatura em Fahrenheit, variando de 1 até 50, calcular e apresentar uma temperatura em Celsius.

7 Construa um programa que apresente o valor de H, sendo H calculado por:

$$H = 1 + 2 + 3 + 4 + \dots + N$$

O valor de N será apresentado pelo utilizador.

8 Construa um programa que calcule N! (factorial de N), sendo que o valor de N (inteiro) é fornecido pelo utilizador. Sabe-se que:

$$N! = 1 \times 2 \times 3 \times 4 \times \dots \times N$$

OBS:  $0! = 1$  (factorial do número zero é igual a 1 por definição).

Além disso, não deve ser permitido que seja calculado o factorial de número negativo, pois isso não existe.

9 Construa um programa que leia o conjunto de 20 números inteiros e mostre qual foi o maior valor fornecido.

10 Construa um programa que leia o conjunto de 20 números inteiros e mostre qual foi o maior e o menor valor fornecido.

11 Construa um programa que leia a quantidade (Q) e o preço (PR) de vários produtos diferentes, comprados por uma empresa, e apresente o total gasto por ela. O final da lista de produtos deverá ser indicado pelo utilizador (escolha a maneira que preferir).

OBS: Não se esqueça de validar a entrada dos valores, pois não são aceitas quantidades negativas, nem preços negativos.

12 Construa um programa que leia vários números inteiros e positivos, calculando ao final da sequência a soma e a média desses números. A sequência termina quando o utilizador entrar com um valor negativo (esse valor não deve fazer parte de nenhum dos cálculos).

13 Construa um programa que leia vários números inteiros e mostre qual foi o maior valor fornecido. O final da lista de produtos será indicado quando o utilizador entrar com um valor negativo (esse valor não deve fazer parte da comparação de valores).

14 Construa um programa que leia vários números inteiros e mostre qual foi o menor valor fornecido. Para cada valor digitado, deve ser solicitado ao utilizador que ele digite se ele deseja continuar.

OBS: Não se esqueça de validar a resposta do utilizador, pois ele só pode responder ôSö ou ôNö.

15 Construa um programa que leia vários números inteiros e mostre qual foi o maior e o menor valor fornecido. Para cada valor digitado, deve ser solicitado ao utilizador que digite se ele deseja continuar.

OBS: Não se esqueça de validar a resposta do usuário, pois ele só pode responder ôSö ou ôNö.

16 Construa um programa que leia um número e indique se ele é par ou ímpar. O programa só deve levar em consideração valores positivos.

17 Anacleto tem 1,50m e cresce 2 centímetros por ano, enquanto Felisberto tem 1,10 e cresce 3 centímetros por ano. Construa um programa que calcule e apresente quantos anos serão necessários para que Felisberto seja maior que Anacleto.

18 Construa um programa que calcule a área total de uma residência (sala, cozinha, quartos, etc., sendo todos eles retangulares). O utilizador deverá entrar com a largura (L) e o comprimento (C) de cada cômodo da casa. Em seguida deverá ser apresentada uma pergunta, solicitando a confirmação do utilizador para continuar com a entrada de dados (a confirmação será dada quando o utilizador entrar com ôSö). Caso ele entre com o valor ôNö. Deverá ser apresentada a área total da casa.

OBS: Não se esqueça de validar a entrada da resposta do utilizador, que só pode aceitar os caracteres ôSö ou ôNö.

19 Construa um programa que apresente a tabuada de um número N. O valor de N será apresentado pelo utilizador.

20 Num cinema, certo dia, cada espectador respondeu a um questionário, que perguntava a sua idade (ID) e a opinião em relação ao filme (OP), seguindo os seguintes critérios:

Opinião	Significado
A	Ótimo

B	Bom
C	Regular
D	Ruim
E	Péssimo

A entrada de dados sobre a opinião deve ser validada.

O final da pesquisa será indicado quando a idade do utilizador for informada como negativa (idade inexistente).

Construa um programa que, lendo esses dados, calcule e apresente:

Quantidade de pessoas que respondeu a pesquisa

Média de idade das pessoas que responderam a pesquisa

Porcentagem de cada uma das respostas

21 Construa um programa que leia as informações de: horas trabalhadas (HT), valor da hora trabalhada (VH). Calcule e apresente o salário líquido dos empregados da empresa, baseado nas tabelas abaixo.

OBS: Salário Líquido = Salário Bruto ó INSS ó Imposto de Renda

INSS = 11% do salário bruto

Imposto de Renda → após descontar o INSS usar esse valor e ler a alíquota do imposto de renda na tabela a seguir:

Salário Bruto ó INSS	Alíquota	Deduzir
Até \$900	Isento	
De \$900 até \$1800	15%	\$135
Mais que \$1800	27,5%	\$360

Não é conhecido o número de funcionários da empresa. Ao final de cada cálculo, o programa deve perguntar se a pessoa deseja calcular o salário de outro funcionário.

Caso a resposta seja negativa, o programa deve parar.

OBS: Não se esqueça de validar a entrada da resposta do utilizador, que só pode aceitar os caracteres Só ou Nö.

22 Construa um programa que calcule e apresente o total da compra realizada pelo cliente numa loja. São fornecidos ao programa, o preço da etiqueta (PE) de cada um dos produtos comprados e, com a compra encerrada, a condição de pagamento escolhida pelo cliente (CP). Utilize para os cálculos a tabela de condições de pagamento a seguir:

Código da condição de pagamento	Condição de pagamento
1	À vista em dinheiro ou cheque, com 10% de desconto
2	À vista com cartão de crédito, com 5% de desconto
3	Em 2 vezes, preço normal de etiqueta sem juros
4	Em 3 vezes, preço de etiqueta com acréscimo de 10%

Uma compra pode ser composta por mais do que um produto, portanto, deve ser indicado ao programa quando a compra deve ser encerrada (escolha a forma que desejar).

OBS: Não esqueça de validar a entrada do código da condição de pagamento.

23 Construa um programa que indique qual a melhor forma de pagamento para a compra realizada por uma empresa. Essa compra será composta por vários produtos e a entrada de dados deve parar quando o utilizador digitar como quantidade um valor negativo. O programa deve ler a quantidade de dinheiro existente no caixa de uma empresa (CAIXA), a quantidade de cada item comprado (QTD) e o preço de cada produto (PR).

Caso o valor total da compra seja superior a 80% do valor em caixa, a compra deve ser feita a prazo (3x), com juros de 10% sobre o valor total. Caso contrário, a compra deverá ser realizada a vista, onde a empresa receberá 5% de desconto. Apresentar a forma de pagamento escolhida e o valor a ser pago (total a vista ou total a prazo), dependendo da escolha realizada pelo programa.

OBS: Não se esqueça de validar a entrada dos valores, pois não são aceites preços negativos.

24 Construa um algoritmo que calcule o novo salário (SAL\_NOVO) para cada um dos funcionários da empresa. Considere que o funcionário deverá receber um reajuste de 15% caso seu salário (SAL) seja menor que 500. Se o salário for maior ou igual a 500,

mas menor ou igual a 1000, o reajuste deve ser de 10%. Caso o salário seja maior que 1000, o reajuste deve ser de 5%.

O programa deve parar quando for digitado um salário (SAL) com valor negativo, ou seja, inválido.

Além disso, ao final, o programa deve apresentar quanto será gasto a mais pela empresa com esses aumentos.

## Modularização

---

Os programas são escritos a fim de resolverem vários tipos de problemas. Alguns desses problemas exigem mais tempo e maiores esforços do programador. Por isso, é indispensável que o programador utilize de técnicas que visam uma maior organização e consequente entendimento facilitado do programa. Uma das técnicas mais utilizadas é a modularização.

Quando se trabalha em qualquer problema complexo, seja em programação, seja em outra área, o ser humano sente a necessidade de dividir este problema maior em problemas menores. Cada problema menor é então trabalhado de forma a solucionar as questões que lhe cabem e, juntando-se cada uma dessas soluções, tem-se a solução do problema maior. Este processo é chamado de modularização e baseia-se na conhecida técnica de dividir para conquistar.

O programa pode ser dividido em várias partes, sendo que cada parte trata de uma determinada funcionalidade.

Desta forma, o programa tem a sua legibilidade melhorada, uma vez que fica mais fácil entender o que o programa faz por completo ou como o programa faz determinada subtarefa.

A modificabilidade do programa também é melhorada, uma vez que para se alterar uma determinada funcionalidade, é necessário apenas alterar uma pequena parte do código, não sendo necessário modificar vários pontos do código.

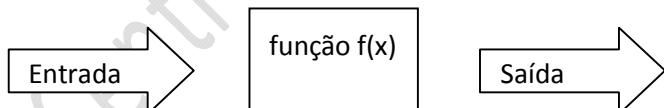
Com as funcionalidades do programa separadas logicamente, o programador tem a oportunidade de reutilizar uma parte do programa para escrever um outro programa que utilize, nalguma das suas tarefas, o mesmo bloco de instruções. A confiabilidade do programa também é aumentada pelo facto dele ser mais fácil de ser entendido e corrigido.

A produtividade para a elaboração de um programa também é aumentada, uma vez que cada parte do programa pode ser trabalhada por uma equipa diferente, além de que cada equipa só precisa de saber o que as partes da outra equipa fazem e não como fazem.

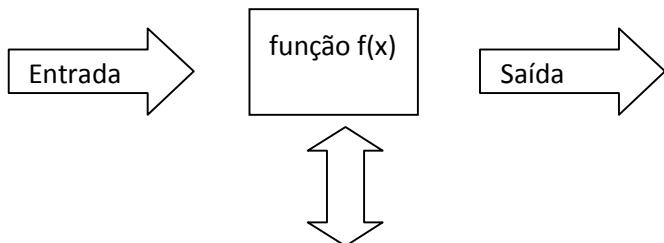
## Métodos

Um método é um trecho de um programa que realiza qualquer operação computacional. Ele efectua parte de uma tarefa que um algoritmo maior deveria executar, ou seja, ele é uma parte do programa, especializado nalguma funcionalidade. Na linguagem Java, o uso de métodos é um dos recursos fundamentais de modularização.

Um método é um conjunto de instruções que recebe alguns valores como dados de entrada e, a partir deles, produz uma valor como saída.



Os dados de entrada são chamados de parâmetros e o valor de saída é chamado de retorno. Durante a execução do método, os dados de entrada são manipuladores, com o auxilio de operadores, de maneira a produzir o resultado esperado. Durante esta manipulação de dados a função pode chamar outras funções, que contêm rotinas que auxiliam na elaboração do resultado final.



### função auxiliar

Os métodos são utilizados para executar operações, funções, ações, etc., dentro das aplicações. Desta forma, eles representam a parte funcional de qualquer sistema, sendo muito importante compreender bem a sua utilização.

Sintaxe para a declaração de um método

```
[modificador]tipo_retorno identificador([parâmetros]){\  
//corpo  
[return]  
}
```

## Partes de um método

Um método é constituído por um cabeçalho, um dicionário de dados, um corpo e comentário

### Cabeçalho

O cabeçalho ou assinatura do método é onde estão definidos o nome do método, os tipos dos seus parâmetros de entrada e o tipo de dado de retorno.

Os parâmetros do métodos, são as informações que ele precisa para executar as suas instruções e os seus tipos devem ser rigorosamente respeitados. Os parâmetros do método podem ser constituídos por uma lista, separada por vírgulas, com os nomes das variáveis (e seus tipos) que receberão os argumentos quando o método for chamado ou executado. Nem todos métodos recebem parâmetros.

Na chamada da função a quantidade é o tipo de parâmetros deve ser rigorosamente respeitada sob pena do compilador detectar erros do tipo not find symbol

O tipo de retorno do método simboliza o tipo de produto gerado por ele, melhor especifica o tipo de dado que será retornado pelo método. O retorno do método é feito pelo comando return (valor).

Se o método diz que retorna um número float, que dizer que o programador deve esperar receber apenas este tipo de dado e tomar as medidas necessárias para manipulá-lo posteriormente.

O nome do método serve para identificá-lo. O programador deve escolher nomes auto-explicativos sobre a funcionalidade do método, ou seja, para o que ele serve. Isto torna o código mais legível e menos suscetível de erros por parte do programador.

### **Exemplo**

```
float calculaMedia(float a, float b);
```

No exemplo acima, a função calculaMedia possui dois parâmetros de entrada do tipo float e retorna um valor também do tipo float, já que o tipo indicado para ele é o float. O primeiro parâmetro recebe o nome a e o segundo recebe o nome b. O método retorna um dado do tipo float, ou seja, retornará um número que contém casas decimais.

### **Dicionário de dados**

O dicionário de dados é onde se faz a declaração de variáveis e constantes usadas no métodos e não declaradas no cabeçalho do método.

Quando se declaram variáveis nos métodos, estas só podem ser utilizadas naquele métodos. Qualquer outro métodos, mesmo aquele que chamar o método, não tem acesso a estas variáveis. Assim, por existirem e serem acessíveis apenas naquele método, são chamadas de variáveis locais.

### **Corpo**

O corpo do método é onde estão contidas as instruções a serem executadas pelo métodos. Nele, podem ser realizadas operações, tais como: atribuições de valores às variáveis, chamadas de outras funções, cálculos de expressões, leitura de dados e apresentação de resultados.

### **Comentários**

Para melhorar a legibilidade e o entendimento do código do método, é recomendável fazer comentários.

Acima de cada método pode-se colocar comentários com as seguintes finalidades: dizer para que serve o método, quais os parâmetros que ele recebe, explicando como devem ser estes parâmetros (unidade de medida, relevância para o método, etc.), restrições para aplicações, entre outras.

No dicionários de dados, os comentários podem ser utilizados para explicar o significado de alguma variável cujo nome não é suficientemente significativo.

No corpo do método, os comentários podem ser utilizados para explicar o que foi feito num determinado conjunto de instruções cujo entendimento não é fácil ou que não tem utilidade aparente.

Em java, os comentários dos métodos devem ser feitos da mesma maneira que a indicada anteriormente, entre /\* e \*/ ou após //

### Exemplo

```
/*

```

Método para calcular a distância entre dois pontos no plano cartesiano .

Dados de entrada :

float x1: a coordenada x do primeiro ponto .

float y1: a coordenada y do primeiro ponto .

float x2: a coordenada x do segundo ponto .

float y2: a coordenada y do segundo ponto .

Dados de saída :

dist : retorna a distância entre os pontos passados

```
*/

```

```
float distancia ( float x1 , float y1 , float x2 , float y2){
```

```
    float dx , dy , dist ;
```

```
    dx = x2 - x1;
```

```
    dy = y2 - y1;
```

```
    dist = Math.sqrt (dx*dx + dy*dy);
```

```
    return dist ;
```

```
}
```

## Escopo das variáveis ó variáveis locais e variáveis globais

Um método pode ter variáveis locais, que são variáveis declaradas dentro do método.

As variáveis locais podem ser utilizadas apenas dentro do método onde foram declaradas.

Os parâmetros do método também são locais a ele.

Métodos diferentes podem ter parâmetros ou variáveis locais com nomes comuns.

Nesse caso, cada método tem os seus próprios parâmetros e variáveis locais, independentemente de os nomes serem os mesmos.

As variáveis globais são as que estão declaradas fora de qualquer método. Estas são chamadas variáveis de instância ou, se tiverem o modificador static, variáveis de classe.

As variáveis assim declaradas podem ser utilizadas em toda a extensão da classe.

### Chamada do métodos

Em programação, quando um método solicita serviços de um outro método, dizemos que foi feita uma chamada ao método. Durante a execução de um programa, podem ser feitas diversas chamadas a um mesmo método.

A chamada de um método em java é feita digitando o nome da função e, em seguida, digitando os dados de entrada necessários entre parentes, se os houver, para que assim o método execute as instruções de que é responsável.

```
import javax.swing.JOptionPane;
public class EstudoMetodosI {
    public static void main(String[] args) {
        double xa, xb, ya, yb, distancia;
        JOptionPane.showInputDialog(null, "Forneceda as coordendas x e Y (em
quilometros) das cidades A e B, respectivamente");
        xa = Double.parseDouble(JOptionPane.showInputDialog("Coordenada xa "));
        xb = Double.parseDouble(JOptionPane.showInputDialog("Coordenada xb "));
        ya = Double.parseDouble(JOptionPane.showInputDialog("Coordenada ya"));
        yb = Double.parseDouble(JOptionPane.showInputDialog("Coordenada yb"));
        distancia = distancia(xa, ya, xb, yb);
        JOptionPane.showMessageDialog(null, "A distancia é de " + distancia +
"Quilometros");
    }
    static double distancia(double x1, double y1, double x2, double y2) {
        double dx, dy, distancia;
        dx = x2 - x1;
```

```
dy = y2 - y1;  
distancia = Math.sqrt(dx * dx + dy * dy);  
return distancia;  
}  
}
```

## Passagem de parâmetros

65

Os parâmetros de um método são os dados de entrada necessários para que ele possa realizar com sucesso o seu trabalho.

Cada parâmetro de um método possui um tipo, declarado no cabeçalho do método.

Quando é feita uma chamada ao método, podem ser passados como parâmetros os valores de variáveis do programa principal, que obrigatoriamente devem ser do mesmo tipo dos parâmetros do método chamado.

É importante perceber que, quando é feita uma passagem de parâmetros, apenas os valores das variáveis são passados para o método chamado e não as variáveis em si. Esse método de passagem de parâmetros é chamado passagem por cópia, ou ainda, passagem por valor.

Em Java, parâmetros de tipos primitivos sempre são passados por valor.

A primeira implicação dessa regra é a não necessidade de ser utilizar variáveis como entrada para o outro método, podendo-se utilizar directamente um valor.

```
distancia = distancia(10, -15, 26, -23);
```

Outra implicação do facto de que apenas valores são passados como parâmetros para um método é a possibilidade de mudar o conteúdo das variáveis que são utilizadas como entrada para o método. No exemplo a seguir a variável *a*, do método *dobraValor*, recebe apenas o valor da variável *x* da função principal. Como isto, a variável *x* permanece com o seu valor anterior à chamada da função *dobraValor*, apenas a variável *a* dessa função tem o seu valor alterado.

```
import javax.swing.JOptionPane;
```

```
public class EstudoMetodos {  
    public static void main(String[] args) {  
        mutiplica(3, 3, 3);  
        // metodo distancia  
        double xa, xb, ya, yb, distancia;  
        JOptionPane.showMessageDialog(null, "Forneceda as coordendas x e Y (em  
        quilometros) das cidades A e B, respectivamente");  
        xa = Double.parseDouble(JOptionPane.showInputDialog("Coordenada xa "));  
        xb = Double.parseDouble(JOptionPane.showInputDialog("Coordenada xb "));  
  
        ya = Double.parseDouble(JOptionPane.showInputDialog("Coordenada ya "));  
        yb = Double.parseDouble(JOptionPane.showInputDialog("Coordenada yb "));  
        distancia = distancia(xa, ya, xb, yb);  
        JOptionPane.showMessageDialog(null, "A distancia e de " + distancia +  
        "Quilometros");  
        // metodo dobraValor  
        int x = 10;  
        dobraValor(x);  
    }  
    static double distancia(double x1, double y1, double x2, double y2) {  
        double dx, dy, distancia;  
        dx = x2 - x1;  
        dy = y2 - y1;  
        distancia = Math.sqrt(dx * dx + dy * dy);  
        return distancia;  
    }  
    static int dobraValor(int a) {  
        System.out.println("Valor Original:\t" + a);  
        a = 2 * a;  
  
        System.out.println("\nO seu dobro eh:\t" + a);  
  
        return a;  
    }  
}
```

```

}

static void imprimeNome() {
    System.out.println("Maria do Rosario Hangalo");
}

static void mutiplica(float a, float b, float c) {
    System.out.println("Resultado:\t" + (a * b * c));
}

static void saudacao() {
    JOptionPane.showMessageDialog(null, "Bem vindos ao Centro de Formação \n
São Domingos");
}

}

```

67

## Passagem de parâmetros por referencia.

Ao se passar um vector como parâmetro por valor, os seus elementos se comportam como se fossem passados por referencia.

Isto significa que se forem alterado o valor de um elemento do vector dentro do método, essa alteração vai ocorrer também no elemento do vector passado como parâmetro.

```

public class InverteVector {
    public static void main(String[] args) {
        int[] valores = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        System.out.println("Disposicao Orginam dos elementos");
        for (int i = 0; i < valores.length; i++) {
            System.out.println("Elemento na posicao" + i + ":\t" + valores[i]);
        }
        System.out.println("\nVector Invertido");
        inverte(valores);
        for (int i = 0; i < valores.length; i++) {
            System.out.println("Elemento na posicao" + i + ":\t" + valores[i]);
        }
    }
    static void inverte(int[] v) {

```

```

int k = v.length - 1;
for (int i = 0; i < v.length / 2; i++) {
    int t = v[i];
    v[i] = v[k - i];
    v[k - i] = t;
}
}

```

68

Após uma chamada ao método inverte, os elementos do vector são invertidos.

## Retorno de valores

Esta secção trata da forma como os dados dos métodos são retornados, isto é, como se produz uma saída do método.

Um método pode produzir, no decorrer da execução das suas instruções, uma valor final (uma saída), que deverá ser passado para o programa ou mesmo método que o chamou.

O valor retornado pelo método será atribuído a alguma variável do programa que chamou este método ou então utilizado numa expressão ou ainda passado para um método de impressão no ecrão.

Em Java, deve-se declarar o tipo de dado retornado pelo método no cabeçalho do método e utiliza-se o comando `return` para especificar a variável ou o valor que dever ser retornado. Após a sua execução, a execução do método corrente termina.

```

import javax.swing.*;
import java.text.*;

public class ConverteDistancias {

    public static void main(String[] args) {
        double milhas, kms;
        milhas = Double.parseDouble(JOptionPane.showInputDialog("Introduza a
distancia em milhas"));
    }
}

```

```
kms = conversor(milhas);
mostraMsg(milhas, kms);
}

static double conversor(double milhas) {
    double kms;
    kms = 1.6093 * milhas;
    return kms;
}

static void mostraMsg(double milhas, double kms) {
    //definir o formato dos numeros decimais
    DecimalFormat df = new DecimalFormat("0.00");

    JOptionPane.showMessageDialog(null, "Distancia em Milhas:" +
        df.format(milhas) + "\n" + "Distancia em Kms:" + df.format(kms));
}

}
```

## Métodos sem lista de parâmetros

Até agora os métodos apresentados continham dados de entrada, ou seja, uma lista de parâmetros. Mas há casos em que isto não é necessário, pois a função executa as suas instruções sem precisar de dados de entrada.

```
static void imprimeNome(){
    System.out.println("Maria do Rosario Hangalo");
}
```

## Métodos sem retorno de dados

Assim como há métodos que não têm nenhum parâmetro, também há funções que não retornam nenhum tipo de dados. Em java, para traduzir esta situação utiliza-se a palavra reservada void como tipo de dado retornado.

```
static void mutiplica(float a, float b, float c){  
    System.out.println("Resultado:\t"+(a*b*c));  
}
```

Há também a opção do método não ter nenhum parâmetro de entrada e nenhum dado de retorno

```
static void saudacao(){  
    JOptionPane.showMessageDialog(null, "Bem vindos ao Centro de Formação \n São  
    Domingos");  
}
```

## Separação das classes

No processo de modularização podemos separar as classes que contêm os métodos de propósitos gerais da classe que contém o método main.

```
public class Calculadora {  
  
    float num1;  
    float num2;  
    float resultado;  
  
    void setNum1(float n1) {  
        this.num1 = n1;  
    }  
  
    void setNum2(float n2) {  
        this.num2 = n2;
```

```
}

float getNum1() {
    return num1;
}

float getNum2() {
    return num2;
}

float getResultado() {
    return resultado;
}

void setSoma(float n1, float n2) {
    this.resultado = (n1 + n2);
}

void setSubtracao(float n1, float n2) {
    this.resultado = (n1 - n2);
}

void setDivisao(float n1, float n2) {
    this.resultado = (n1 / n2);
}

void setMultiplicacao(float n1, float n2) {
    this.resultado = (n1 * n2);
}

void menu() {
    System.out.println("@@@@@@@Calculadora@@@@@@@");
    System.out.println("Escolha a operação");
    System.out.println("1-Soma");
    System.out.println("2-Divisão");
    System.out.println("3-Multiplicação");
    System.out.println("4-Subtração");
}
}
```

```
import java.util.Scanner;  
  
public class Principal {  
  
    public static void main(String args[]) {  
        int escolha = 0;  
        Scanner tec = new Scanner(System.in);  
  
        //Instanciar a classe que contem os métodos
```

```
Calculadora calc = new Calculadora();
//chamada do método menu().....
calc.menu();
System.out.println("Escolha uma opção:");
escolha = (tec.nextInt());
switch (escolha) {
    case 1: {
        System.out.println("Informe o primeiro valor a ser somado");
        calc.setNum1(tec.nextFloat());
        System.out.println("Informe o segundo valor a ser somado");
        calc.setNum2(tec.nextFloat());
        calc.setSoma(calc.getNum1(), calc.getNum2());
        System.out.println("O Resultado e : " + calc.getResultado());
        break;
    }
    case 2: {
        System.out.println("Informe o primeiro valor a ser dividido");
        calc.setNum1(tec.nextFloat());
        System.out.println("Informe o segundo valor a ser dividido");
        calc.setNum2(tec.nextFloat());
        calc.setDivisao(calc.getNum1(), calc.getNum2());
        System.out.println("O Resultado e : " + calc.getResultado());
        break;
    }
    case 3: {
        System.out.println("Informe o primeiro valor a ser multiplicado");
        calc.setNum1(tec.nextFloat());
        System.out.println("Informe o segundo valor a ser multiplicado");
        calc.setNum2(tec.nextFloat());
        calc.setMultiplicacao(calc.getNum1(), calc.getNum2());
        System.out.println("O Resultado e : " + calc.getResultado());
        break;
    }
    case 4: {
        System.out.println("Informe o primeiro valor a ser subtraido");
        calc.setNum1(tec.nextFloat());
        System.out.println("Informe o segundo valor a ser subtraido");
        calc.setNum2(tec.nextFloat());
        calc.setSubtracao(calc.getNum1(), calc.getNum2());
        System.out.println("O Resultado e : " + calc.getResultado());
        break;
    }
}
```

# Exercícios

*Algumas fórmulas apresentadas nos enunciados estão na sua forma MATEMÁTICA e não na forma como devem ser escritas nos nossos programas.*

*As indicações entre parênteses são sugestões de nomes de variáveis(escrever os nomes das variáveis conforme a convenção).*

*Elaborar o diagrama de blocos e/ou o pseudocódigo (português estruturado) em todos os casos(em comentário no programa).*

73

1. Escreva um método que receba um número inteiro positivo N como parâmetro e retorne a soma dos N números inteiros existentes entre 1 e esse número.

2. Escreva um método que faça a conversão binário/decimal e vice-versa.

Sugestão: considerar o numero binário como um vector de char de 0 ..1

3. Escrever um método que aceite como entrada um numero inteiro positivo n e mostre a sucessão dos números inteiros positivos de 1 a n.

4. Escreva um método que aceite como entrada um numero inteiro n e visualize a sucessão dos números inteiros positivos impares de 1 a n

5. Escreva um método que aceite como entrada dois números inteiros positivos n e m, com n menor que m, e visualize a sucessão dos números positivos de n a m.

6. Escreva um método que aceite como entrada um numero inteiro n e visualize os números primos menores do que n.

7. Escreva um método que aceite como entrada um numero inteiro n e visualize os números triangulares menores do que n: 1, 3, 6, 10, 15, 21Í [A sequencia dos números triangulares obtém-se partindo da unidade e somando ao numero precedente: 2, 3, 4, 5Í ]

8. Escrever um método que aceite como entrada um numero inteiro n e visualize as potencias cúbicas dos números naturais até à potencia não superior ao valor de n. 1, 8, 27, 64, 124Í

9. Escreva um método que receba um valor inteiro positivo como parâmetro e retorne o factorial do número.

10. Escreva um método que calcule uma potência qualquer com expoente inteiro maior ou igual a zero.

11. Alterar o método anterior de modo que calcule também potências com expoente inteiro negativo.

12 . Escreva um método que calcule, em função da variável x, o valor da expressão:

$$3\sqrt{x^2 + 3} = \frac{x^2 + 3}{2}$$

13. O governo de uma cidade fez uma pesquisa entre os habitantes, colectando dados sobre o salário e o número de filhos.

Escreva uma função que leia esses dados para um número não determinado de pessoas e mostre a média salarial da população, a média de filhos, o maior salário e a percentagem de pessoas com salário até AKZ 12000,00.

14. Escreva um programa que controle o stock de uma empresa. Para isso utilize uma matriz, onde cada coluna representa o código do produto e cada linha a quantidade em stock deste produto. Escreva três funções para controlo de stock: a primeira será responsável pela inserção de produtos no stock, a segunda pela remoção e a terceira para listar todo o stock.

15. Escreva um método que dado um número inteiro retorne uma string com cada dígito deste número por extenso, da forma:

64852 <= número passado como parâmetro seis quatro oito cinco dois <= string a ser retornada

16 Escreva um método que receba um número inteiro. Este método deve verificar se tal número é primo. No caso positivo, a função deve retornar 1, caso contrário zero.

Escreva também um programa para testar tal método.

17. Escreva um método que receba dois números inteiros  $x$  e  $y$ . Esse método deve verificar se  $x$  é divisível por  $y$ . No caso positivo, a função deve retornar 1, caso contrário zero.

Escreva também um programa para testar tal método.

18. Um número é dito regular caso a sua decomposição em factores primos apresenta apenas potências de 2, 3 e 5. Faça um método que verifique se um número é (retorne 1) ou não (retorne 0) regular. Escreva também um programa para testar tal função.

19. Criar um método que determine se um carácter, recebido como parâmetro, é ou não uma letra do alfabeto. O método deve retornar 1 caso positivo e 0 em caso contrário.

Escreva também um programa para testar tal método.

20. Um número é dito ser **capicua** se quando lido da esquerda para a direita é o mesmo que quando lido da direita para a esquerda. O ano 2002, por exemplo, é **capicua**. Então, elabore um método para verificar se um número possui essa característica. Caso o número seja **capicua**, o método deve retornar 1 e 0 em caso contrário. Escreva também um programa para testar tal método.

21. Criar um método que calcule e retorne o **MAIOR** entre dois valores recebidos como parâmetro. Um programa para testar tal método deve ser criado.

22 Criar um método que verifique quantas vezes um número inteiro  $x$  é divisível por um número inteiro  $y$ . O método deve retornar -1 caso não seja possível calcular. Escreva também um programa para testar tal método.

23. Construa uma função que efectue a **TROCA** dos valores de  $a$  por  $b$ , recebidos como parâmetro. Crie também um programa que leia dois valores quaisquer, e imprima os valores após a chamada da função **TROCA**.

24. Construa um método que receba três valores,  $a$ ,  $b$  e  $c$ , retorne o **MAIOR** e o **MENOR** valor desses três. Deve ser criado um programa para utilizar tal método lendo os três valores e imprimindo o maior e o menor valor computado.

25. Construa um método que receba dois valores inteiros  $a$  e  $b$ , retorne o quociente, **div**, e o resto divisão, **mod**, de  $a$  por  $b$ . A função deve retornar:

-1 caso não seja possível realizar as operações e 0 caso seja possível. Um programa para utilizar tal método deve ser criado, tratando o retorno da função.

26. Construa um método que receba cinco valores e determine o 2º e o 4º maior valores dentre eles. Construa também um programa para ler tais valores, e imprimir o resultado obtido com a chamada do método.

27. Construa um método, que receba três coeficientes relativos à uma equação de segundo grau ( $a.x^2 + b.x + c = 0$ ) e calcule as suas raízes através da fórmula de Baskara:

$$\boxed{x} = \frac{-\boxed{a} \pm \sqrt{\boxed{a}^2 - 4\boxed{a}\boxed{c}}}{2\boxed{a}}$$

O método deve levar em conta a possibilidade da existência de nenhuma, uma ou duas raízes. O método deve retornar o número de raízes ou -1 em caso de inconsistência. Os valores das raízes devem ser retornados. Construa também um programa para utilizar o método construído.

28. Crie um método que realize a conversão para Radianos (*rad*) a partir de Graus (*grad*), onde *grad* é passado como parâmetro e *rad* é retornado. Sabe-se que  $180^\circ$  (graus) está para p radianos. Crie também um programa para testar tal método.

29. Crie um método que realize a conversão de Fahrenheit (*F*) para graus Celsius (*C*), onde *F* é passado como parâmetro e *C* é retornado. Sabe-se que os pontos de fusão e ebulição nas escalas Celsius e Fahrenheit são:  $0^\circ\text{C}$  e  $100^\circ\text{C}$ , e  $32^\circ\text{F}$  e  $212^\circ\text{F}$ , respectivamente. Crie também um programa para testar tal método.

30. Crie um método que realize a conversão de Polegadas (*pol*) para Centímetros (*cm*), onde *pol* é passado como parâmetro e *cm* é retornado. Sabe-se que 1 polegada está para 2,54 centímetros. Crie também um programa para testar tal método.

31. Crie um método que realize a conversão de pés (*feet*) para metros (*m*), onde *feet* é passado como parâmetro e *m* é retornado. Sabe-se que 1 metro está para 3,281 pés. Crie também um programa para testar tal método.

32. Crie um método que realize a conversão da escala Kelvin (*K* - escala absoluta) para a escala Fahrenheit (*F*). Sabe-se que  $273\text{K}$  equivale a  $32^\circ\text{F}$  e a cada variação de 10 unidades na escala Kelvin equivale a 18 na escala Fahrenheit. O método deve retornar zero caso não seja possível realizar a conversão e um em caso contrário. Crie também um programa para testar tal método.

33. Escrever um método que receba o salário mensal e a percentagem de reajuste.

Calcular e retornar o valor do novo salário. O salário mensal, reajuste e novo salário devem ser do tipo float.

34. Escrever um método que receba o raio de um círculo, calcular e escrever a sua área.

35. Faça um método que leia uma sequência de notas de alunos e mostre a maior e a menor das notas. O final da leitura será identificado pela introdução de uma nota negativa, que não deve fazer parte do cálculo.

36. Implementar um método `lerValorValido()` que verifica se um valor introduzido pelo utilizador pertence ao conjunto limitado por dois dados valores inteiros, devolvendo o primeiro valor que pertença àquele intervalo;

## Arrays ó Estruturas de dados homogêneas

---

A manipulação de conjuntos de dados na memória é um problema clássico nas linguagens de programação em geral. A maneira existente para resolver esta questão é a utilização de arrays. Um array ou arranjo é uma estrutura de dados armazenada na memória, que possui uma referencia para o acesso e índices para dimensões e elementos. Os arrays podem ser unidimensionais (vectores) ou multidimensionais(matrizes).

Os arrays, em java, são utilizados constantemente para reunir objectos em colecções, nas quais é possível efectua pesquisas, bem como manipulação e ordenação de elementos, de forma simples e eficiente.

Um array é uma sequencia de valores dos mesmo tipo, isto é homogéneas, e os valores memorizados são chamados elementos.

Um array é um tipo de dados que permite o armazenamento de várias informações, na memória RAM, no mesmo instante de tempo e com contiguidade física, ou seja, uma variável possui vários elementos, igualmente distanciados, ou seja, um ao lado do outro na memória principal.

Na linguagem java, os arrays são considerados objectos, sendo, portanto, do tipo referencia. Eles podem armazenar coleccões de tipos primitivos (arrays primitivos) ou de tipos referencia (arrays de objectos).

Cada elemento de um array armazena um valor. Os elementos de um array são referenciados por meio de um índice. O primeiro elemento tem sempre o índice 0 (zero). Portanto, para um array de n elementos, os índices variam de 0 a n-1.

v[0]	v[1]	v[2]	v[3]	v[4]	v[5]
v					

Um elemento é um valor dentro de um array e é acedido pela sua posição dentro deste array.

O valor destas posições pode ser manipulado conforme a necessiade. Cada posição pode ser acedida individualmente, utilizando o operador `[]` e informando a posição. A posição ou índice de um array deve ser um inteiro positivo ou uma expressão do tipo inteiro.

Todo array deve ser inicializado para que o seu tipo assuma valores padrão quando for passado algum valor, ou seja, de uma forma geral, todo objecto (não apenas em arrays) não inicializado armazena null.

Em Java, os arrays têm tamanho fixo e, uma vez criados, não podem ser redimensionados. O campo `length` contém o tamanho do array.

Um array é um objecto da classe `Object`, ou seja, a criação de um array cria, implicitamente, um objecto da classe `java.lang.Object`. Os arrays são objectos e são

criados dinamicamente e podem ser assinados como variáveis do tipo Object. Todos os métodos da class Object podem ser invocados num array.

O tamanho do array é estabelecido quanto ele é criado e após a criação o tamanho da sua estrutura é fixa, portanto não pode ser alterada. Quando se cria um objecto array utilizando o operador new, todos os índices são inicializados automaticamente, sendo 0 (zero) para arrays numéricos, falso para boolean, \u0000 para caracteres, e NULL para objectos.

Os arrays são objectos. Se forem passados como parâmetros para um método, a passagem é sempre por referência do objecto.

Quando é necessário armazenar dados de diferentes tipos numa estrutura simples ou então se precisa utilizar uma estrutura que possa mudar dinamicamente o seu tamanho então tem-se a disposição as colecções (Framework Collection) da API, as quais são simples para este tipo de situação

Valores padrão de arrays não pré-definidos	
Tipo de elemento	Valor Inicial
byte, short, int	0 (zero)
char	\u0000 (Unicode)
long	0L (zero)
float	0.0F (zero)
double	0.0D (zero)
boolean	false
Object Reference	null

## Criação de arrays

Pode-se criar arrays vazios ou com elementos e estes elementos podem ser consultados ou modificados.

Lembrete

Um array é uma colecção de tipos iguais sendo eles primitivos ou não

O tamanho do array pode ou não ser especificado;

Pode-se colocar os colchetes [] antes ou depois do identificador do array;

Pode-se fazer conversão (casting) em elementos de arrays desde que sejam seguidas as regras padrão de casting.

Pode-se copiar topo ou parte de um array para outro arrays, desde que seja do mesmo tipo

A criação de um array dá-se quando da sua declaração.

### Declaração para a criação de um array

#### Sintaxe

```
tipo_de_dado[] identificador = new tipo_de_dado[tamanho];
```

ou

```
tipo_de_dado identificador [] = new tipo_de_dado[tamanho];
```

### Exemplo 1

```
float[] salario;
```

#### exemplo 2

```
double a[], b[], c[];
```

pode ser escrito como: double[] a, b, c;

Esta declaração simples de um array não aloca memória para armazenar os elementos do array, ela apenas indica que se trata de algum tipo de array. Se o programa tentar aceder qualquer elemento deste array antes de se alocar memória para ele, o compilador mostrará um erro de compilação.

78

A declaração do array não cria o objecto. Só cria uma referência (ponteiro) para o array

### Exemplo 2

```
char letra[] = new char[26]; // vetor de 26 valores do tipo char
```

```
Ponto p[] = new Ponto[3]; // vetor de 3 objetos da classe Ponto
```

```
int x[] = {2,8,3,5,1}; // declaração e inicialização
```

```
float[] salario = new float[10];
```

O acesso a um elemento de índice fora do intervalo declarado para o array gera uma exceção (erro em tempo de execução).

Depois de criado, o array possui atributos e métodos como qualquer outro objecto.

## Inicialização de arrays

Após a criação, os elementos do array devem ser inicializados, antes de serem utilizados (na criação, os elementos são inicializados com o valor padrão). Na verdade, quando se cria um novo array ele não tem nada dentro dele e quando é inicializado passa a ter o mínimo dentro dele o que depende de cada tipo de array criado.

Existem diferentes modos de declaração e de inicialização de arrays.

Exemplo	Descrição
<pre>String nome[] = new String[3]; nome[0] = "Zé"; nome[1] = "Jó"; nome[2] = "Lu";</pre>	Declara primeiro e inicializa depois. O tamanho do array é dado na declaração. A alocação de memória é feita pela declaração. A inicialização ocorre depois da memória ter sido alocada.
<pre>String nome[]; int n = 3; nome = new String[n]; nome[0] = "Zé"; nome[1] = "Jó"; nome[2] = "Lu";</pre>	Declara somente o ponteiro. Em seguida cria os objectos. Notar que entre a declaração do ponteiro e a criação dos objectos, o tamanho do array pode ser lido, atribuído ou calculado. Com isso, o array pode ter exactamente o tamanho necessário. A inicialização ocorre depois

	da memória ter sido alocada.
String nome[] = {"Zé","Jó","Lu"};	Declara e inicializa. O tamanho do array é dado pelo número de valores na lista de inicialização. A alocação de memória é feita durante a inicialização.

## Acesso aos elementos de um array

O acesso e a impressão dos elementos de um array pode ser feito de duas formas. Imprimindo os índices desejados em arrays pequenos ou, de forma melhor e eficiente percorrendo o array com uma das estruturas de repetição (for, while, do-while)

```
public class AcessoElementos {

    public static void main(String[] args) {
        //cria um array com posições
        int[] inteiros = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};

        //Laço for para mostrar todos os elementos do array
        System.out.println("Todos os elementos do array");
        for (int i = 0; i < inteiros.length; i++) {
            System.out.print(inteiros[i] + ",");
        }

        System.out.println(); // Pula uma linha

        System.out.println("Elemento 5=" + inteiros[4]);
        System.out.println("Elemento 10=" + inteiros[9]);
        System.out.println("Elemento 15=" + inteiros[14]);
    }
}
```

Também é possível efectuar operações matemáticas sobre os elementos de arrays numéricos. Os índices do array podem ser tratados como variáveis independentes, sendo assim pode-se utilizar estes índices para realizar cálculos

```
public class SomaPares {

    public static void main(String[] args) {
        // cria um array de numeros inteiros

        int[] inteiros = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
        20};
        int soma = 0;
        for (int i = 0; i < inteiros.length; i++) {
            if (inteiros[i] % 2 == 0) {
```

```

        soma += inteiros[i];
    }
}
System.out.print("A soma dos elementos pares do array e =" + soma);
}

}

```

```

public class MeuArray {

    public void mostrarValores(String nome, int vetor[]) {
        for (int i = 0; i < vetor.length; i++) {
            System.out.println(nome + "[" + i + "] = " + vetor[i]);
        }
    }
}

import javax.swing.JOptionPane;

public class LerTamanhoArray {

    public static void main(String[] args) {
        int v[];
        String sTam;
        int tam;
        sTam = JOptionPane.showInputDialog("Tamanho do vetor");
        tam = Integer.parseInt(sTam);
        // Construir um array do tamanho desejado
        v = new int[tam];
        for (int i = 0; i < tam; i++) {
            v[i] = 2 * (i + 1);
        }
        MeuArray mv = new MeuArray();
        mv.mostrarValores("v", v);
    }
}

```

```

import java.util.*;

public class ExemploArray {

    public static void lerArray(int v[]) {
        Scanner in = new Scanner(System.in);
        for (int i = 0; i < v.length; i++) {
            System.out.print("v[" + i + "] = ");
            v[i] = in.nextInt();
        }
    }
}

```

```

    }

public static int somarArray(int v[]) {
    int soma = 0;
    for (int i = 0; i < v.length; i++) {
        soma = soma + v[i];
    }
    return soma;
}

public static void mostrarArray(int v[]) {
    System.out.print("[");
    for (int i = 0; i < v.length; i++) {
        System.out.print(v[i]);
        if (i < v.length - 1) {
            System.out.print(", ");
        }
    }
    System.out.println("]");
}

public static void main(String[] args) {
    int n, x[];
    Scanner in = new Scanner(System.in);
    System.out.print("Tamanho do array: ");
    n = in.nextInt();
    x = new int[n];
    lerArray(x);
    mostrarArray(x);
    int s = somarArray(x);
    System.out.println("Soma = " + s);
}
}

```

## Arrays como parâmetros de métodos

Os arrays podem ser parâmetros de métodos. Na chamada ao método, somente o nome do array é passado. Os arrays, como são objectos (e, portanto, o nome do array é um ponteiro) são sempre passados por referência.

```

public class MeuArray {

    public void mostrarValores(String nome, int vetor[]) {
        for (int i = 0; i < vetor.length; i++) {
            System.out.println(nome + "[" + i + "] = " + vetor[i]);
        }
    }
}

```

```

        }
    }

public class ProgArray {

    public static void main(String[] args) {
        int x[] = {1, 2, 3, 4, 5};
        MeuArray mv = new MeuArray();
        mv.mostrarValores("x", x);
    }
}

```

82

## Recordando:

Existem duas maneiras de passar parâmetros para os métodos: a passagem por valor e a passagem por referência.

Quando um parâmetro é passado por valor, uma cópia do valor do parâmetro é passada para o método chamado. O método chamado trabalha exclusivamente com a cópia. As alterações feitas na cópia pelo método chamado não afectam o valor do parâmetro original.

Quando um parâmetro é passado por referência, o método chamado trabalha directamente com o parâmetro original e, portanto, as alterações feitas pelo método chamado afectam o parâmetro original.

Em Java, o programador não escolhe o tipo de passagem de parâmetros: valores primitivos são passados por valor e referências a objectos são passados por referência.

Notar que:

Os arrays são passados por referência (pois o nome do array é uma variável de referência)

Os elementos do array de tipos primitivos são passados por valor

## Exercícios de consolidação.

1

Criar um programa que receba um array de inteiros e o apresente ordenado utilizando o algoritmo da força bruta

```

public class OrdenaVector {

    public static void main(String[] args) {
        int[] vetor = {5, 40, 20, 10, 50, 30};
        System.out.print("Vector de Entrada: ");
        for (int i = 0; i < vetor.length; i++) {
            System.out.print(vetor[i] + " ");
        }
        System.out.println();
        for (int i = 0; i < vetor[0] - 1; i++) {
            for (int j = i + 1; j < vetor[0]; j++) {
                if (vetor[i] > vetor[j]) {

```

```

        int temp = vetor[i]; // sort
        vetor[i] = vetor[j];
        vetor[j] = temp;
    }
}
System.out.print("Vetor Ordenado: ");
for (int i = 1; i < vetor.length; i++) {
    System.out.print(vetor[i] + " ");
}
System.out.println();
System.out.println("Número de Elementos: " + vetor[0]);
}

}

```

83

## 2

Considere que um array v contém o número de notas dos alunos de FP2 agrupadas em categorias, ou seja: v[0] contém o número de notas de 0.0 a 0.9, v[1] contém o número de notas de 1.0 a 1.9, v[2] contém o número de notas de 2.0 a 2.9, assim por diante, até v[10], que contém o número de notas 10. Mostrar a distribuição de notas por meio de um gráfico de barra de asteriscos (\*).

```

public class GraficoBarras {

    public void mostrarGrafico(int vetor[]) {
        for (int i = 0; i < vetor.length; i++) {
            if (i < 10) {
                System.out.printf("%3.1f-%3.1f: ", (i + 0.0), (i + 0.9));
            } else {
                System.out.printf("%7.1f: ", 10.0);
            }
            for (int j = 0; j < vetor[i]; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}

```

import javax.swing.JOptionPane;

```

public class DistribuicaoNotas {

    public static void main(String[] args) {
        int v[] = new int[11];
        String s;
        int nAlunos, nota, categ;
        for (int i = 0; i < 11; i++) {
            v[i] = 0;
        }
    }
}

```

```

s = JOptionPane.showInputDialog("Número de alunos");
nAlunos = Integer.parseInt(s);
// Ler as notas dos alunos e distribuir nas categorias
for (int i = 0; i < nAlunos; i++) {
    s = JOptionPane.showInputDialog("Nota do aluno " + (i + 1));
    nota = Integer.parseInt(s);
    categ = nota / 10;
    v[categ]++;
}
GraficoBarras gb = new GraficoBarras();
gb.mostrarGrafico(v);
}
}

```

84

## Arryas Multidimensionais

A linguagem java não possui arrays multidimensionais, mas permite declarar um array baseado num outro array, assim consegue-se um array de arrays.

Estes arrays são chamados de array esparço por alguns autores/programadores e são um agrupamento indexado de dados onde nem todas as dimensões são iguais. Arrays esparços podem ser representados em Java através de arrays com dimensões heterogéneas.

A flexibilidade de se colocar os colchetes à esquerda ou a direita do nome da variável não se aplica a outros aspectos da sintaxe dos arrays. Por exemplo, new int[][][10], não é válido.

### *Sintaxe para a declaração e instanciação de arrays de arrays*

tipo\_de\_dado[l][c] identificador = new tipo\_de\_dado[l][c];  
ou  
tipo\_de\_dado identificador [l][c] = new tipo\_de\_dado[l][c];  
Onde l=linha e c = coluna.

Pode-se também utilizar a forma directa que tem a seguinte sintaxe, por exemplo para um array 3x4;

```
int[][] matrizInteiros = { { 1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} };
```

	coluna [0]	coluna [1]	coluna [2]	coluna [3]
Linha[0]	1	2	3	4
Linha[1]	5	6	7	8

Linha[2]	9	10	11	12
----------	---	----	----	----

## Percorrer uma matriz

Para percorrer uma matriz em geral é necessário utilizar mais de uma estrutura de repetição

```
public class VectorMatriz {

    public static void main(String[] args) {
        int[] vetor = {10, 20, 30, 40, 50};
        int[][] matriz = {{1, 2}, {3, 4}, {5, 6}};
        int numLin = 3, numCol = 2;
        System.out.println("Vector Unidimensional");
        for (int i = 0; i < vetor.length; i++) {
            System.out.println("Vector: " + vetor[i]);
        }
        System.out.println("\n\nMatriz Bidimensional de ordem: " + numLin + " x " + numCol);
        for (int i = 0; i < numLin; i++) {
            for (int j = 0; j < numCol; j++) {
                System.out.println("Matriz: " + matriz[i][j]);
            }
        }
    }
}
```

85

### Entrada de dados para um array Unidimensional:

```
final int NUMERO_ELEMENTOS = 7;
int [] vetor = new int [NUMERO_ELEMENTOS];
for (int i = 0;i < vetor.length;i++) {
    s = JOptionPane.showInputDialog("Valor: ");
    vetor[i] = Integer.parseInt(s);
}
í
```

### Entrada de dados para um vector Bidimensional:

```
final int NUMERO_LINHAS = 3;
final int NUMERO_COLUNAS = 4;

í
double [][] matriz = new double [NUMERO_LINHAS][ NUMERO_COLUNAS];
for (int i = 0;i < matriz.length;i++) {
    for (int j = 0;j < matriz[i].length;j++) {
```

```

        s = JOptionPane.showInputDialog("Valor: ");
        matriz[i][j] = Integer.parseInt(s);
    }
}
í

```

É muito comum se cometerem erros quando impressão dos elementos de arrays de arrays. Na impressão em geral é utilizado mais de um for, mas no segundo for é necessário especificar também a linha do array constante do primeiro for conforme de pode ser destacado no trecho anterior.

```

import java.util.*;
public class ExemploMatriz {
    public static void lerMatriz(int m[][]) {
        Scanner in = new Scanner(System.in);
        for (int i = 0; i < m.length; i++) {
            for (int j = 0; j < m[i].length; j++) {
                System.out.print("m[" + i + "," + j + "] = ");
                m[i][j] = in.nextInt();
            }
        }
    }

    //O método modificarMatriz(n,m) multiplica
    //cada elemento de m pelo valor de n.
    public static void modificarMatriz(int n, int m[][]){
        for (int i = 0; i < m.length; i++) {
            for (int j = 0; j < m[i].length; j++) {
                m[i][j] = n * m[i][j];
            }
        }
    }

    public static int[][] somarMatrizes(int a[][], int b[][]){
        int soma[][] = a;
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a[i].length; j++) {
                soma[i][j] = a[i][j] + b[i][j];
            }
        }
        return soma;
    }

    public static void mostrarMatriz(int m[][]){
        for (int i = 0; i < m.length; i++) {
            for (int j = 0; j < m[i].length; j++) {
                System.out.printf("%4d", m[i][j]);
            }
        }
    }
}

```

```

        System.out.println();
    }

}

public static void main(String[] args) {
    int numLins, numCols, mat1[][];
    Scanner in = new Scanner(System.in);
    System.out.print("Número de linhas: ");
    numLins = in.nextInt();
    System.out.print("Número de colunas: ");
    numCols = in.nextInt();
    mat1 = new int[numLins][numCols];
    lerMatriz(mat1);
    System.out.println("\nMatriz 1:");
    mostrarMatriz(mat1);
    int mat2[][] = new int[numLins][numCols];
    mat2 = (int[][]) mat1.clone();
    modificarMatriz(2, mat2);
    System.out.println("\nMatriz 2:");
    mostrarMatriz(mat2);
    int mat3[][] = somarMatrizes(mat1, mat2);
    System.out.println("\nMatriz 3:");
    mostrarMatriz(mat3);
}
}
}

```

Em Java é possível criar métodos que recebem um número não especificado de parâmetros. Um tipo seguido por reticências (...) indica que o método recebe um número variável de parâmetros deste tipo. As reticências podem ocorrer apenas uma vez e devem ser colocadas no fim da lista de parâmetros.

No corpo do método, a lista de parâmetros de comprimento variável é tratada como um vector.

```

public class CalcularMedia {
    public double media(double... valor) {
        //Notar que o parâmetro valor é tratado como um vetor no corpo do método.
        double soma = 0.0;
        for (int i = 0; i < valor.length; i++) {
            soma = soma + valor[i];
        }
        return (soma / valor.length);
    }
}

```

```

import java.util.Locale;
public class TesteMedias {
    public static void main(String[] args) {
        double d1 = 10.0;
        double d2 = 20.0;
        double d3 = 30.0;
        double d4 = 40.0;
        CalcularMedia cm = new CalcularMedia();
        System.out.printf(Locale.US, "Valores: %4.1f, %4.1f, %4.1f, %4.1f\n",
            d1, d2, d3, d4);
    }
}

```

```

        System.out.printf(Locale.US, "Media dos dois primeiros = %5.2f\n",
            cm.media(d1, d2));
        System.out.printf(Locale.US, "Media dos tres primeiros = %5.2f\n",
            cm.media(d1, d2, d3));
        System.out.printf(Locale.US, "Media de todos os quatro = %5.2f\n",
            cm.media(d1, d2, d3, d4));
    }
}

```

## Exercícios de consolidação

**1**

Criar um programa que leia 4 notas de X alunos e armazene os dados numa matriz. Em seguida, mostre as 4 notas e a média obtida dos X aluno.

```

import java.util.Scanner;
public class MediaAlunos {
    public static void main(String[] args) {
        Scanner Entrada;
        Entrada = new Scanner(System.in);
        System.out.print("Digite a quantidade de alunos: ");
        int lin = Entrada.nextInt();
        float[][] mat = new float[lin][4];
        for (int i = 0; i < mat.length; i++) {
            System.out.println("Aluno" + (i + 1));
            System.out.print("Digite nota 1: ");
            mat[i][0] = Entrada.nextFloat();
            System.out.print("Digite nota 2: ");
            mat[i][1] = Entrada.nextFloat();
            System.out.print("Digite nota 3: ");
            mat[i][2] = Entrada.nextFloat();
            System.out.print("Digite nota 4: ");
            mat[i][3] = Entrada.nextFloat();
            System.out.println("=====");
        }
        for (int i = 0; i < mat.length; i++) {
            System.out.println("Aluno" + (i + 1));
            System.out.println("Nota 1: " + mat[i][0]);
            System.out.println("Nota 2: " + mat[i][1]);
            System.out.println("Nota 3: " + mat[i][2]);
            System.out.println("Nota 4: " + mat[i][3]);
            System.out.println("Média: " + ((mat[i][0] + mat[i][1] + mat[i][2] + mat[i][3]) /
4));
            System.out.println("=====");
        }
    }
}

```

**2**

Criar um programa que lê uma matriz e uma linha qualquer e retorne a soma dos elementos dessa linha.

```

import java.util.Scanner;
public class ElementoLinha {
    public static void main(String[] args) {
        Scanner Entrada;
        Entrada = new Scanner(System.in);
        System.out.print("Linhas: ");
        int lin = Entrada.nextInt();
        System.out.print("colunas: ");
        int col = Entrada.nextInt();
        int[][] mat = new int[lin][col];
        for (int i = 0; i < lin; i++) {
            for (int j = 0; j < col; j++) {
                System.out.print("Elemento: ");
                mat[i][j] = Entrada.nextInt();
                System.out.print("\n");
            }
        }
        System.out.print("Digite uma linha (0- " + (lin - 1) + "): ");
        int linha = Entrada.nextInt();
        System.out.println("Elementos da Linha");
        for (int j = 0; j < col; j++) {
            System.out.print(mat[linha][j] + "\t");
        }
    }
}

```

**3**

Cria um programa que le uma matriz quadrada e retorne a soma dos elementos da diagonal principal e abaixo desta.

```

import java.util.Scanner;
public class SomaDiagonalAbaixo {

    public static void main(String[] args) {
        Scanner Entrada;
        Entrada = new Scanner(System.in);
        System.out.print("Ordem da Matriz: ");
        int lin = Entrada.nextInt();
        int[][] mat = new int[lin][lin];
        for (int i = 0; i < lin; i++) {
            for (int j = 0; j < lin; j++) {
                System.out.print("Elemento: ");
                mat[i][j] = Entrada.nextInt();
                System.out.print("\n");
            }
        }
        int soma = 0;
        for (int i = 0; i < lin; i++) {

```

```

        for (int j = 0; j < lin; j++) {
            if (i >= j) {
                System.out.print(mat[i][j] + "\t");
                soma = mat[i][j] + soma;
            }
        }
        System.out.println("A soma dos elementos é: " + soma);
    }
}

```

**4**

Criar um programa que leia uma matriz matA e retorne matB, tal que  $\text{matB} = \text{matA}^T$ .

90

```

import java.util.Scanner;
public class Transposta {
    public static void main(String[] args) {
        Scanner Entrada;
        Entrada = new Scanner(System.in);
        System.out.print("Linhas: ");
        int lin = Entrada.nextInt();
        System.out.print("colunas: ");
        int col = Entrada.nextInt();
        int[][] mat = new int[lin][col];
        for (int i = 0; i < lin; i++) {
            for (int j = 0; j < col; j++) {
                System.out.print("Elemento: ");
                mat[i][j] = Entrada.nextInt();
                System.out.print("\n");
            }
        }
        int[][] mat2 = new int[lin][col];
        for (int i = 0; i < lin; i++) {
            for (int j = 0; j < col; j++) {
                mat2[i][j] = mat[j][i];
            }
        }
        System.out.println("Matriz digitada:");
        for (int i = 0; i < lin; i++) {
            for (int j = 0; j < col; j++) {
                System.out.print(mat[i][j] + "\t");
            }
            System.out.print("\n");
        }
        System.out.println("Matriz transposta:");
        for (int i = 0; i < lin; i++) {
            for (int j = 0; j < col; j++) {
                System.out.print(mat2[i][j] + "\t");
            }
            System.out.print("\n");
        }
    }
}

```

```
}
```

```
}
```

```
}
```

## Exercícios

91

*Algumas fórmulas apresentadas nos enunciados estão na sua forma MATEMÁTICA e não na forma como devem ser escritas nos programas.*

*As indicações entre parênteses são sugestões de nomes de variáveis(escrever os nomes das variáveis conforme a convenção).*

*Elaborar o diagrama de blocos e/ou o pseudocódigo (português estruturado) em todos os casos(em comentário no programa).*

1. Implementar o método lerVector() que preenche um vector com DIM valores reais (DIM >= 1);

2. Implementar o método mostrarVector() que mostra um vector com DIM valores reais (DIM >=1);

**Consulte os livros de álgebra linear e:**

3. Escreva um programa para calcular a transposta de uma matriz.

4. Escreva um programa para calcular o determinante de uma matriz.

5. Escreva um programa para inverter uma matriz.

6. Foi feita uma estatística em algumas cidades angolanas para colectar dados sobre acidentes de trânsito. Foram obtidos os seguintes dados:

ó código da cidade

ó número de veículos de passeio

ó número de acidentes de transito com vítimas

Faça um programa que leia estes dados e mostre:

ó qual é o maior e o menor índice de acidentes de transito e a que cidades pertencem

ó qual a média de veículos nas cidades juntas

ó qual a média de acidentes de trânsito nas cidades com menos de 2.000 veículos de passeio

OBS.: A quantidade de cidades a serem lidas é um número a ser digitado pelo utilizador

7. Faça um programa para contagem de votos numa eleição presidencial onde existem quatro candidatos. Os votos para cada candidato são informados através do número do candidato.

Os dados utilizados para a contagem dos votos obedecem à seguinte codificação:

ó 1,2,3,4 = voto para os respectivos candidatos;

ó 5 = voto nulo;

ó 6 = voto em branco;

O programa deve calcular e mostrar:

ó total de votos para cada candidato;

ó total de votos nulos;

ó total de votos em branco;

Para finalizar a leitura de votos, o utilizador deve digitar 0.

**8.** Faça um programa que imprima uma matriz quadrada de dimensão 2 contendo: o número **1** nos elementos abaixo da diagonal principal o número **0** nos demais elementos

9- Faça um programa que leia uma matriz A de duas dimensões com 4 linha e 5 colunas. Construir uma matriz B de mesma dimensão, sendo que cada elemento da matriz B deverá ser o dobro de cada elemento correspondente da matriz A, com excepção para os valores situados na diagonal principal (posições B[1,1], B[2,2], B[3,3], B[4,4] e B[5,5]) os quais deverão ser o triplo de cada elemento correspondente da matriz. Apresentar, no fim, a matriz B.

10. Faça um programa que leia um vector A com 10 números inteiros, calcule e mostre a soma dos quadrados dos elementos do vector.

11. Dada uma tabela de 4x5 elementos, faça um programa para calcular a soma de cada linha da matriz. Apresente o somatório de cada linha, e o somatório total de todas as linhas.

12. Elaborar um programa que efectue o cálculo de uma tabuada de um número qualquer e armazene os resultados numa matriz A de uma dimensão para 10 elementos. Apresentar os valores armazenados na matriz.

13. Construir um programa que leia 6 elementos (valores inteiros) para as matrizes A e B de uma dimensão do tipo vector.

Construir as matrizes C e D de mesmo tipo e dimensão, sendo que a matriz C deverá ser formada pelos elementos de índice par das matrizes A e B, e a matriz D deverá ser formada pelos elementos de índice ímpar das matrizes A e B. Apresentar as matrizes C e D.

14. Dada uma tabela de 4 x 5 elementos, faça um programa para calcular a soma de cada linha e a soma de todos os elementos da matriz.

15. Escreva um programa que lê uma matriz M(5,5) e calcula as somas:

- a) da linha 4 de M
- b) da coluna 2 de M
- c) da diagonal principal
- d) da diagonal secundária
- e) de todos os elementos da matriz

Escreva estas somas e a matriz.

16. Construir um programa que lê 6 valores, e conta quantos destes valores são negativos. Armazene estes valores num vector. Ao final, imprima a média aritmética destes valores.

17. Implemente um programa que tenha os seguintes métodos:

a) Método que receba por parâmetro um vector de números inteiros e um número inteiro indicando o número de elementos armazenados neste vector. Este método deverá retornar o maior valor armazenado neste vector.

b) Método que recebe por parâmetro um vector de números inteiros e um número inteiro indicando o número de elementos armazenados neste vector. Este método deverá retornar a média dos valores armazenados neste vector.

No método main deverá ser declarado um vector de 100 posições para armazenar a idade dos alunos do primeiro semestre do curso de Engenharia Informática. Peça ao utilizador para digitar as idades e armazene-as no vector até que seja digitada uma idade

negativa ou até que o vector seja totalmente preenchido. Depois apresente a média das idades e a idade do aluno mais velho chamando os métodos implementadas.

18. Um ñQuadrado Mágicoö é uma matriz quadrada NxN na qual são armazenados números inteiros, de tal forma que a soma dos números presentes em cada linha, e a soma dos números presentes em cada coluna, é sempre igual. Um exemplo de quadrado mágico de dimensão 3x3 é apresentado a seguir. Crie um programa que leia um número N(<5), e então leia os números de uma matriz NxN do teclado, verificando se os números digitados formam um ñQuadrado Mágicoö.

1	2	3
2	3	1
3	1	2

19 Elabore um programa para ler o nome completo de dez pessoas armazenando-os numa matriz, e imprimindo-os no seguinte formato: sobrenome, nome.

20. Elabore um programa que manipule os dados para uma sala de 52 alunos. O programa deve receber como entrada (via leitura de teclado):

- Os nomes dos alunos.
- As duas notas parciais.

O programa deve calcular e imprimir, numa mesma linha, para cada aluno:

- O nome e a sua média final.

E também deve apresentar:

- A quantidade de alunos com média maior que a média geral da turma.
- A maior média da turma, e a menor média da turma.
- A quantidade de alunos aprovados (media  $\geq 9.5$ ), reprovados(media < 7) e em exame de recurso (media entre 9 e 7).

22. O Sudoku é um jogo japonês onde deve-se completar uma matriz 9x9 de números inteiros entre 1 e 9, de tal forma que:

- em cada linha, apareça apenas um dígito de cada;
- em cada coluna, apareça apenas um dígito de cada;
- em cada sub-matriz 3x3 apareça apenas um dígito de cada;

Elabore um programa que seja capaz de verificar se uma matriz 9x9 é uma solução para o Sudoku.

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	3	4	5	6	7	8	9	1
5	6	7	8	9	1	2	3	4
8	9	1	2	3	4	5	6	7
3	4	5	6	7	8	9	1	2
6	7	8	9	1	2	3	4	5
9	1	2	3	4	5	6	7	8

Dica: para verificar a presença única de cada dígito em cada linha, coluna ou submatriz, utilize um vector auxiliar de tamanho 10.

... (continua)