



CIÊNCIAS DA COMPUTAÇÃO

FACULDADE DE CIÊNCIAS | UNIVERSIDADE AGOSTINHO NETO

1º Ano

FUNDAMENTOS DE PROGRAMAÇÃO

Docente:

- ✓ Mateus Padoca Calado - PhD

Monitores:

- ✓ Nsimba Kiafuka
- ✓ Mariano Calelua



Tema - 02

Variáveis, Expressões, Operadores, e instrução de Seleção

Conteúdo



Cap. II

Variáveis, Expressões e Operadores

Estrutura de um programa em java

Sintaxe - Estrutura básica em Java

```
public class NomeDoPrograma {  
    public static void main(String[] args) {  
        // Instruções...  
    }  
}
```

- ❑ **NomeDoPrograma:** é um nome sugestivo que constitui a nomenclatura da class. A primeira letra deve ser maiúscula.
- ❑ **main:** é método principal do programa.
- ❑ As chavetas **{}** definem o início (**{**) e o fim (**}**) de um bloco.

Estrutura de um programa em java

- ❑ Java é uma linguagem Case Sensitive, isto é, faz distinção entre maiúsculas e minúsculas.
- ❑ Cada instrução deve ser seguida por ponto e vírgula (;).

Comentários

- ❑ Comentários: são trecho de texto explicativo que visam facilitar a interpretação dos códigos.
- ❑ Java permite três tipos de comentários:
 - De uma linha – `//` escreve-se aqui o comentário
 - De múltiplas linhas - `/*` escreve-se aqui o comentário `*/`
 - De documentação - `/**` escreve-se aqui o comentário . `*/`

Exemplo de Comentários

```
/**  
 * class destinada a resolver  
 * os exemplos de fundamentos  
 */  
public class Exemplo{  
  
    public static void main(String[] args) {  
  
        /* exemplo de comentários  
        com múltiplas linhas*/  
  
    } // fim do método main  
  
} // fim da class Exemplo
```

Tipos de dados

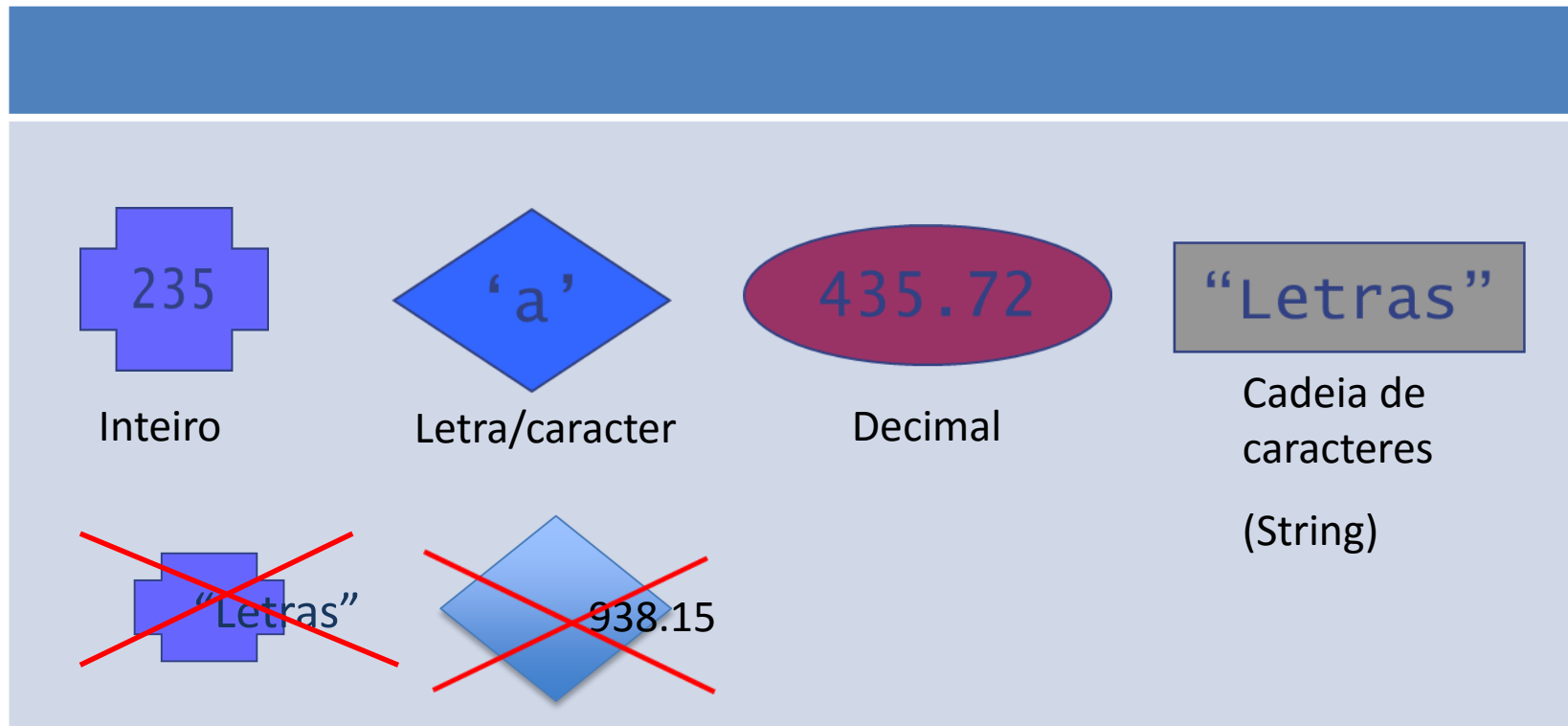
- ❑ Um tipo de dados é uma abstracção de algo e define o domínio de valores e o tamanho em byte (conjunto de 8 bits) que determinada variável ocupará em memória.
- ❑ A linguagem de programação java compreende dois grupos de tipos de dados : primitivos e referência.
- ❑ **Tipos Primitivos**
 - **char** – um caracter
 - **int** – número inteiro (existem três outros tipos de inteiros)
 - **float ou double** – número decimal
 - **boolean** – verdadeiro ou falso
- ❑ **Tipos por Referência**
 - Object
 - String – cadeia de caracteres
 - Tipos definidos pelo utilizador
 - outros

Tipos primitivos em JAVA

Tipo	Tamanho/ Formato	Valores literais	Domínio
<i>(números inteiros)</i>			
byte	8 bits	10, ...	[-128, 127]
short	16 bits	234, ...	[-32768, 32767]
int	32 bits	176, ...	[-2147483648, 2147483647]
long	64 bits	8374L, ...	[-9223372036854775808, ...07]
<i>(números decimais)</i>			
float	32-bit	3.14f, 200.482F, ...	[+/-1.4E-45, +/- ~3.40E38]
double	64-bit	18.0, 1.8e1, 18.0d, ...	[+/-4.9E-324, +/- ~1.78E308]
<i>(outros tipos)</i>			
char	16 bits/Unicode	'A', '.', '£', ...	[..., '!', ..., 'ÿ', ...] ou [\u0000, \uffff]
boolean	(não definido)	false e true	{false, true}

Variáveis

- ❑ Servem para guardar informação
- ❑ Guardam dados que têm de ser de um tipo definido



Variáveis

❑ Antes da primeira utilização :

- Tem de se declarar quais as variáveis que passam a existir nesse programa, qual o seu nome e tipo - *declaração*
- Deve também ser dado um valor inicial a cada variável - *inicialização*

❑ Após a declaração:

- Podem ser utilizadas para guardar dados (valores)
 - Colocação de um valor numa variável chama-se atribuição.

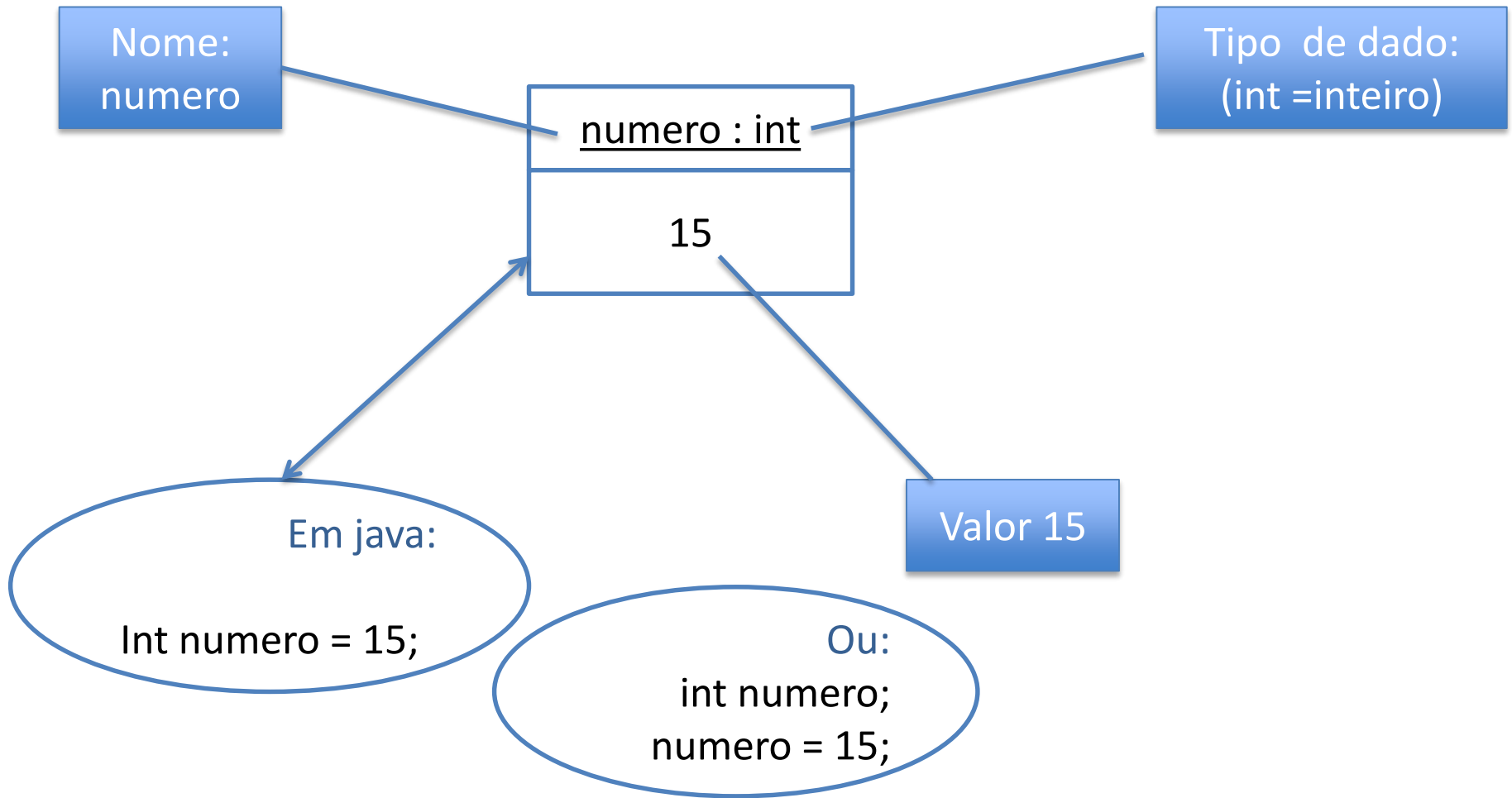
❑ Em Java as variáveis podem ser declaradas em qualquer ponto do programa, sendo válidas em todo escopo onde foram declaradas.

❑ O primeiro caractere do nome de uma variável deve ser uma letra, cifrão(\$) ou um sublinhado(_) e os caracteres subsequentes devem ser letras, números, cifrão, ou sublinhados.

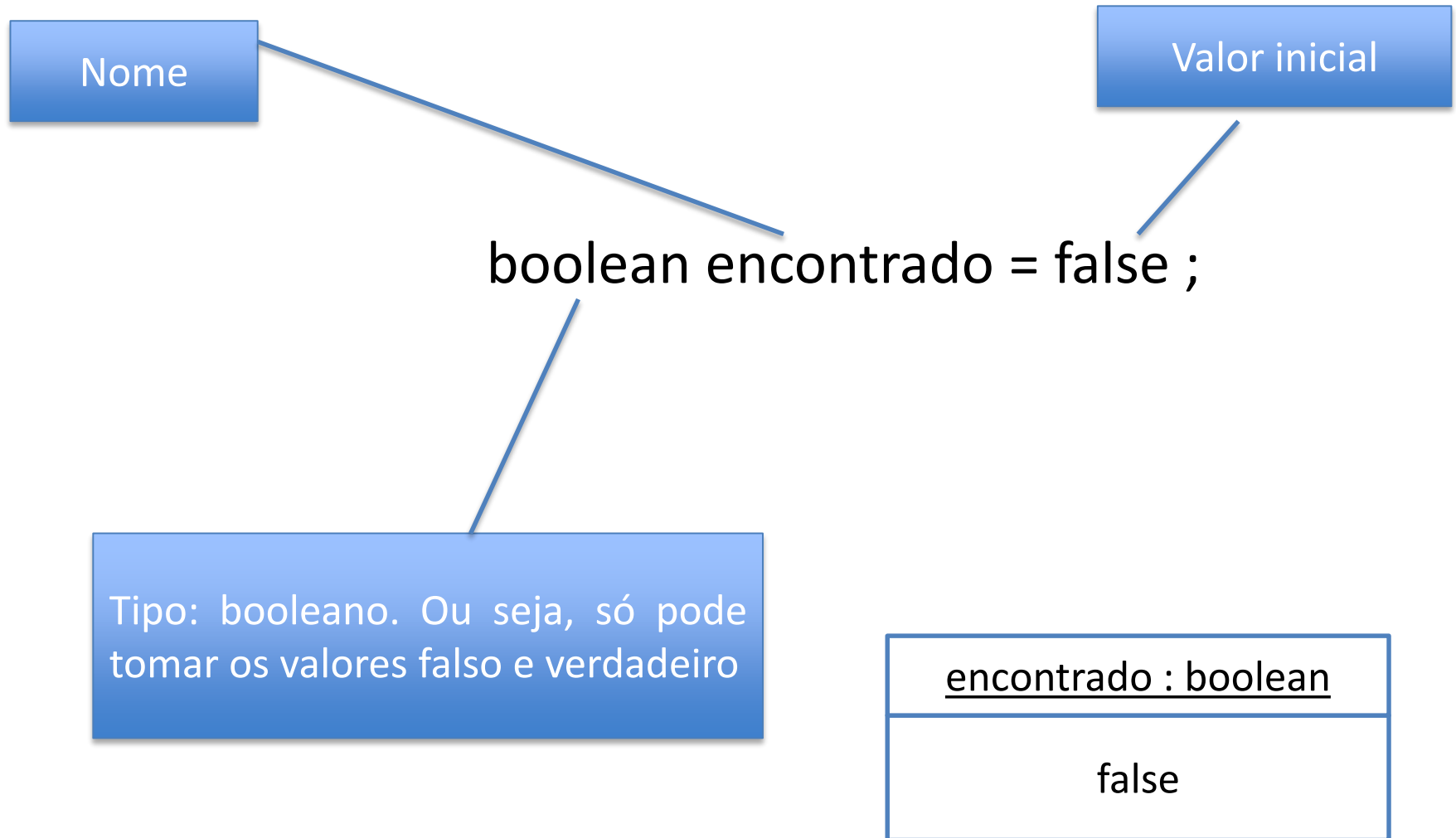
❑ Sintaxe

- `<tipoDeDados> <nomeDaVariavel>;`

Variáveis (representação gráfica)



Variáveis (declaração e inicialização)



Constantes

- ❑ O conteúdo de uma constante que é atribuída no momento de declaração não pode ser modificado durante a execução do programa.
- ❑ Constantes em Java são definidas usando o modificador `final`.
- ❑ Por convenção as constantes devem ter todas as suas letras em maiúsculo.
- ❑ Sintaxe: `final tipo_de_dados NOME_DA_CONSTANTE = valor;`

❑ Ex: `final int LIMITE = 200;`

Palavras Reservadas, Variáveis e Tipos

- ❑ Palavras Reservadas de uma Linguagem:
 - Têm um significado especial para o compilador
 - Não podem ser usadas para outras finalidades
- ❑ Variáveis são criadas *sempre* associadas a um *tipo* de dados.
 - Tipo de dados é sempre uma palavra reservada (tipos primitivos)
 - Não pode ser usado para outras finalidades

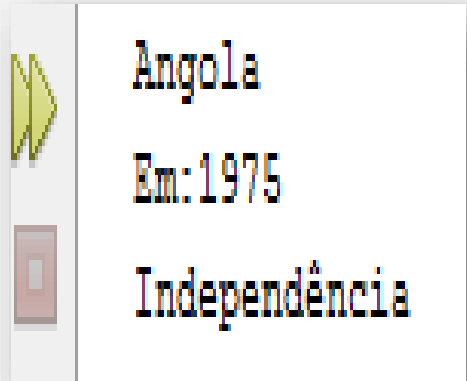
Exemplo: `int` é palavra reservada em JAVA

`float int = 300.0; // ERRO FLAGRANTE!`

Escrita de valores no ecrã

❑ Para escrever(apresentar) informação no monitor utiliza-se a instrução:

- `System.out.print();` //Apenas escrever
- `System.out.println();` // Muda de linha depois de escrever

Exemplo	Resultado
<pre>public class Fundamentos { public static void main(String[] args) { System.out.println("Angola"); int x = 75; System.out.print("Em:"); System.out.println("19" + x); System.out.println("Independência"); } }</pre>	

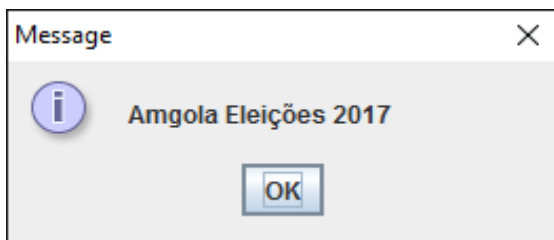
Exibir Textos em Caixa de Diálogo

- ❑ Também é possível exibir as informações caixa de dialogo utilizando a classe **JOptionPane**.

Exemplo

```
import javax.swing.JOptionPane;

public class Fundamentos {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Angola Eleições 2017");
    }
}
```



Leitura de valores do teclado

❑ Utilizaremos a classe Scanner para entrada de valores a partir do teclado obedecendo os seguintes passos:

- Importar a Classe Scanner:
 - `import java.util.Scanner;`
- Criar um objecto de leitura:
 - `Scanner teclado = new Scanner (System.in)`

Leitura de valores do teclado

- ❑ Com o objecto de leitura criado, pode-se ler(receber) os dados digitados por tipo de dado requerido.
- ❑ Ler os dados do teclado:
`tipo a = teclado.nextTipo();`
 - Inteiro (int): `int a = teclado.nextInt();`
 - Float: `float b = teclado.nextFloat();`
 - Double: `double c = teclado.nextDouble();`
 - String (palavra): `String s = teclado.next();`
 - String (linha): `teclado.nextLine();`

Leitura de valores de uma janela

- ❑ É possível também usar uma janela para ler valores (p.e., uma frase ou cadeia de caracteres - String) a partir do teclado:

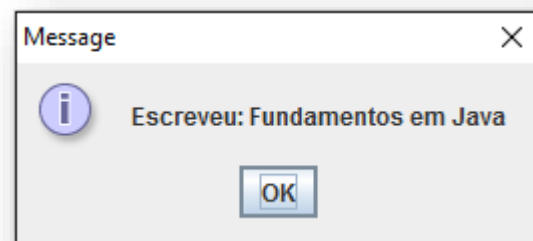
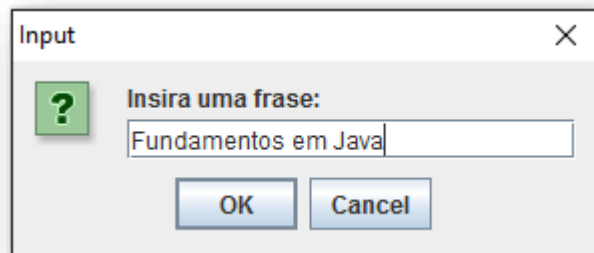
Exemplo

```
import javax.swing.JOptionPane;

public class Janelas {
    public static void main(String[] parametros) {

        String lida = JOptionPane.showInputDialog("Insira uma frase: ");
        JOptionPane.showMessageDialog(null, "Escreveu: " + lida);

    }
}
```



Operações

- ❑ Operações : Conjunto de cálculos sobre os dados.
- ❑ Para fazer as Operações sobre os dados (guardados nas variáveis ou não) é necessário usar os operadores
- ❑ Operadores são:
 - ❖ **Aritméticos**
 - ❖ **Relacionais**
 - ❖ **Lógicos**

Operadores Aritméticos

Operador	Símbolo	Exemplo
Adição	+	$a + b$
Subtração	-	$a - b$
Multiplicação	*	$a * b$
Divisão	/	a / b
Resto da divisão	%	$a \% b$

Operadores relacionais

Operador	Símbolo	Exemplo
Maior	>	a > b
Maior igual	>=	a >= b
Menor	<	a < b
Menor igual	<=	a <= b
Diferente	!=	a != b
Igual	==	a == b

Nota: Estes operadores são usados nas **condições** das *estruturas de selecção e repetição*

Operadores lógicos

Operador	Símbolo	Exemplo
Conjunção	&&	<code>((a > b) && (b < c))</code>
Disjunção		<code>((a > b) (b < c))</code>

❑ **Nota:** Estes operadores são usados entre dois ou mais **Operadores Relacionais**.

- ***&&: lê-se e***
- ***||: lê-se ou***

Operadores de Incremento e Decremento

- ❑ Utiliza-se **++** (**sufixo e prefixo**) para incrementar uma unidade numa variável.
- ❑ Utiliza-se **--** (**sufixo e prefixo**) para decrementar uma unidade numa variável.
 - A expressão $x = x + 1$ equivalem a: **++x** (prefixo) ou **x++** (sufixo).
 - A expressão $x = x - 1$ equivalem a: **--x** (prefixo) ou **x--** (sufixo).

```
int x = 3;  
x++;
```

Resultado: 4

```
int x = 3;  
x--;
```

Resultado: 2

Operadores de Incremento e Decremento

❑ Operadores de Incremento e Decremento

Expressão	Nome da Expressão	
x++	Pós-incremento	Incrementa x por 1, e utiliza o novo valor de x.
++x	Pré-incremento	Utiliza o valor actual de x, e incrementa x por 1.
b--	Pós-decremento	Decrementa b por 1, e utiliza o novo valor de b.
--b	Pré-decremento	Utiliza o valor actual de b, e decrementa b por 1.

Operador de atribuição e concatenação

- ❑ Em Java o operador de atribuição é: **= (igual)**
- ❑ O operador de concatenação é: **+ (mais)**

Ex: `a=5; //(variável a recebe o valor 5);`
`b=a; //(variável b recebe o valor 5);`
`String s ="Linguagem" + "Java";`

❑ Operadores de Atribuição Composta

- A expressão `x = x + 3` equivalem a: **`x += 3`**.
- A expressão `x = x - 3` equivalem a: **`x -= 3`**.

```
int x = 3;  
x += 3;
```

Resultado: 6

```
int x = 3;  
x -= 3;
```

Resultado: 0

Operadores

- ❑ Os operadores Aritméticos e Lógicos obedecem uma ordem de precedências conforme listado:

Operadores	Símbolos	Resultado
Aritméticos	+ - * / %	Numérico
Relacionais	> < >= <= <> ou != ?:	Booleano
Lógicos	! &&	Booleano
Outros	= ++ --	-

Wrappers

- ❑ Na linguagem Java os **Wrapper** são conhecidos como classes especiais que possuem métodos capazes de fazer conversões em variáveis primitivas.
- ❑ Para cada um dos oito tipos primitivos em Java existe, associada, uma **wrapper** class – com variáveis do correspondente tipo primitivo.
- ❑ Entre as funcionalidades das **wrapper** classes encontramos várias rotinas – termo que passaremos a designar por Métodos.

classe Wrapper para cada tipo primitivo

Tipo primitivo	Classe Wrapper
boolean	Boolean
byte	Byte
char	Character
int	Integer
float	Float
double	Double
long	Long
short	Short

Wrappers

Sintaxe

```
tipo v = Tipo.parseTipo();
```

Exemplo

```
// para converter uma String num valor inteiro
    ▪ int i = Integer.parseInt("123");

// converte uma String num inteiro long.
    ▪ long L2 = Long.parseLong("101010");
```

Cast

- ❑ Podemos forçar uma expressão a ser de um determinado tipo utilizando um cast.
- ❑ A forma genérica de um cast é:
 - **(tipo)** expressão
 - onde **tipo** é qualquer tipo de dados válido em Java.
- ❑ **Exemplo**
 - `int x = (int)5/2;`

Conversões **possíveis** entre tipos de dados:

- char «--» números inteiros
- números decimais «--» números inteiros

Conversões **impossíveis** (por enquanto ...):

- boolean «--» qualquer outro tipo
- referência «--» qualquer outro tipo

CAP. II - Exercícios

1. Fazer um programa que soma dois números
2. Fazer um programa que depois de receber três números calcula as quatro operações aritméticas.
3. Implementar um programa que peça o nome e três notas de um aluno. O mesmo deve calcular a média.
4. Calcular o salário líquido de um funcionário. Será informado seu nome, seu salário base e o desconto do INSS (11%, por exemplo).
Formula: $\text{salarioLiquido} = \text{Base} - \text{Base} * \text{Desconto} / 100$.



Cap. III

Estruturas de Selecção (decisão)

Instrução de selecção (decisão)

- ❑ Em Java As estruturas de selecção comportam-se da mesma forma. Estas diferem-se somente pela sintaxe comparada aos pseudocódigos. Estrutura de Decisão simples.

Pseudocódigo	Em Java
<pre>se(condição) então // Instruções fimse</pre>	<pre>If (condição) { // Instruções }</pre>

- ❑ Utilizando esta estrutura, as instruções são somente executadas caso a condição avaliada seja verdadeira. Caso contrário, o programa segue o seu percurso normal; ou seja não faz nada.

Instrução de selecção (decisão)

❑ Estrutura de decisão simples

- Exemplo: Crie um programa em Java que recebe a idade de um indivíduo e imprime “JOVEM” se a idade for maior ou igual a 18 e menor que 45.

Pseudocódigo

```
idade:inteiro
Inicio

    escreva("Digite a sua idade")
    leia(idade)

    se ((idade>=18) e (idade<45)) entao
        escreva("JOVEM")
    Fimse

finalgoritmo
```

Em Java

```
import java.util.Scanner;
public class Exercícios {
    public static void main(String[]args){

        Scanner teclado=new Scanner(System.in);

        System.out.println("Digite a sua Idade");
        int idade=teclado.nextInt();

        if((idade>=18)&&(idade<45)){
            System.out.print("JOVEM");
        }
    }
}
```

Instrução de selecção (decisão)

❑ Estrutura de decisão composta.

Pseudocódigo	Em Java
<pre>se(condição) então // Instruções Senão // Instruções fimse</pre>	<pre>If (condição) { // Instruções }else{ // Instruções }</pre>

- ❑ Esta estrutura de selecção é implementada quando temos instruções a serem executadas caso o teste (condição) seja verdadeiro e outras instruções caso o teste resulte em falso.

Instrução de selecção (decisão)

❑ Estrutura de decisão composta.

- Exemplo: crie um programa em Java que verifica se um número é par ou ímpar.

Pseudocódigo	Em Java
<pre>num:inteiro inicio escreva("Digite um número") leia(num) se (num mod 2=0) entao escreva("Número PAR") senao escreva("Número IMPAR") fimse finalgoritmo</pre>	<pre>import java.util.Scanner; public class Exercícios { public static void main(String[]args){ Scanner teclado=new Scanner(System.in); System.out.println("Digite um número Inteiro"); int num=teclado.nextInt(); if(num % 2 == 0){ System.out.print("Número PAR"); }else{ System.out.print("Número ÍMPAR"); } } }</pre>

Instrução de selecção (decisão)

❑ Estrutura de decisão encadeada.

Pseudocódigo	Em Java
<pre>se(condição)então se(condição)então //Instruções Senão //Instruções fimse; Senão //Instruções fimse;</pre>	<pre>if (condição) { if (condição) { // Instruções } else{ // Instruções } } else{ if (condição) { // Instruções } }</pre>

❑ **Nota:** Não existe uma regra para as estruturas encadeadas. Elas variam de acordo com a necessidade do problema proposto.

Instrução de selecção (decisão)

❑ Estrutura de decisão switch.

- Java tem um comando interno de selecção múltipla **switch**.
- O **switch** testa sucessivamente o valor de uma expressão contra uma lista de constantes inteiras ou de caratere.
- Quando o valor coincide, os comandos associados àquela constante são executados

Instrução de selecção (decisão)

❑ Estrutura de decisão switch

A forma geral do comando switch é:

```
Switch (valor) {  
    case valor1:  
        instruções;  
        break ;  
    case valor2:  
        instruções;  
        break ;  
    case valor3:  
        instruções;  
        break ;  
    ...  
    default:  
        instruções ;  
}
```

Instrução de selecção (decisão)

❑ Estrutura de decisão switch.

- O **default** é opcional e, se não estiver presente, nenhuma acção será realizada se todos os testes falharem.
- Para obter mais eficiência devemos limitar o número de comandos case a uma quantidade menor.
- O comando *break* é um dos comandos de desvio em java.

Cap. III - Exercício

1. Crie um programa que recebe o salário de um trabalhador e o valor da prestação de um empréstimo, se a prestação for maior que 20% do salário imprimir: Empréstimo não concedido, caso contrário imprimir: Empréstimo concedido.
2. Escrever um programa que solicite um salário ao utilizador e mostre o imposto a pagar.
 - Se o salário for negativo ou zero mostre o erro respectivo.
 - Se o salário for maior que 1000 paga 10% de imposto, senão paga apenas 5%.
3. O sistema de avaliação de determinada disciplina, é composto por três provas. A primeira prova tem peso 2, a segunda tem peso 3 e a terceira tem peso 5. Faça um programa em java para calcular a média final de um aluno desta disciplina.