

# Curso Java COMPLETO

**Capítulo: Estrutura sequencial**

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

## Expressões aritméticas

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

# Variáveis e tipos primitivos em Java

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

## Visão geral

- Um programa de computador em execução lida com dados
- Como esses dados são armazenados?
- Em **VARIÁVEIS!**

# As três operações básicas de programação

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

Um programa de computador é capaz de realizar essencialmente três operações:



## Saída de dados

**Programa → Usuário**



Dispositivo de SAÍDA



Também chamada de  
**ESCRITA:**

"O programa está escrevendo dados."

## Saída de dados em Java

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

- Comentários de linha:

- Começam com //

- Atalhos:

- Importar classes: CTRL + SHIFT + O
- Autoindentação: CTRL + SHIFT + F
- sysout CTRL + espaço

# Processamento de dados em Java, Casting

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

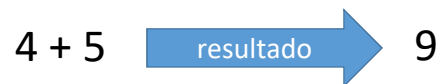
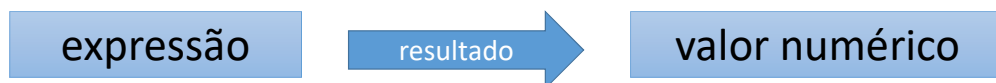
## Exemplo 5

```
double a;  
int b;  
  
a = 5.0;  
b = a;  
  
System.out.println(b);
```

## Entrada de dados em Java

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

## Expressões aritméticas



## Operadores aritméticos

**C, C++,  
Java, C#** →

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão ("mod")

**Precedência:** 1º lugar: \* / %  
2º lugar: + -

## Exemplos de expressões aritméticas

$$2 * 6 / 3 \quad \text{Resultado} = 4$$

$$3 + 2 * 4 \quad \text{Resultado} = 11$$

$$(3 + 2) * 4 \quad \text{Resultado} = 20$$

$$60 / (3 + 2) * 4 \quad \text{Resultado} = 48$$

$$60 / ((3 + 2) * 4) \quad \text{Resultado} = 3$$

## Exemplos com o operador "mod"

$$14 \% 3 \quad \text{Resultado} = 2$$

$$19 \% 5 \quad \text{Resultado} = 4$$

**Pois:**

$$\begin{array}{r|l} 14 & 3 \\ \hline 2 & 4 \end{array}$$

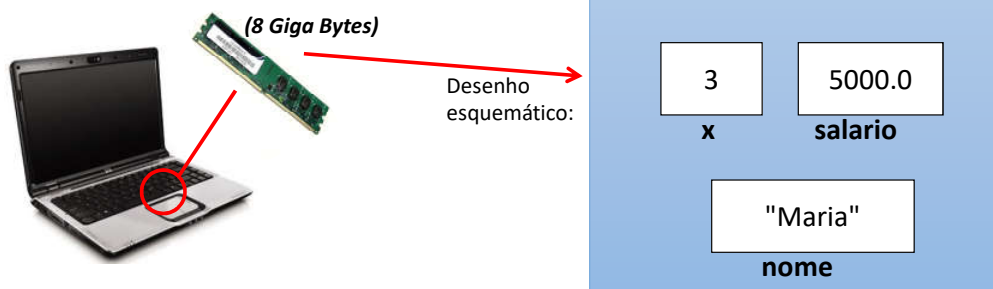
$$\begin{array}{r|l} 19 & 5 \\ \hline 4 & 3 \end{array}$$



# Variáveis

## Definição informal:

Em programação, uma variável é uma porção de memória (RAM) utilizada para armazenar dados durante a execução dos programas.



## Declaração de variáveis

### Sintaxe:

```
<tipo> <nome> = <valor inicial>;
```

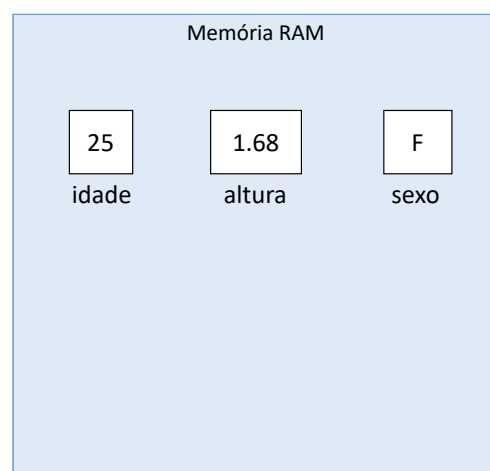
(opcional)

### Exemplos:

```
int idade = 25;  
double altura = 1.68;  
char sexo = 'F';
```

### Uma variável possui:

- Nome (ou identificador)
- Tipo
- Valor
- Endereço



# Tipos primitivos em Java

Descrição	Tipo	Tamanho	Valores	Valor padrão
tipos numéricos inteiros	byte	8 bits	-128 a 127	0
	short	16 bits	-32768 a 32767	0
	int	32 bits	-2147483648 a 2147483647	0
	long	64 bits	-9223372036854770000 a 9223372036854770000	0L
tipos numéricos com ponto flutuante	float	32 bits	-1,4024E-37 a 3,4028E+38	0.0f
	double	64 bits	-4,94E-307 a 1,79E+308	0.0
um caractere Unicode	char	16 bits	'\u0000' a '\uFFFF'	'\u0000'
valor verdade	boolean	1 bit	{false, true}	false

**String** - cadeia de caracteres (palavras ou textos)

Veja: [unicode-table.com](http://unicode-table.com)

Exemplo: 'a' = '\u0061'

Um bit pode armazenar 2 valores possíveis (0 ou 1)

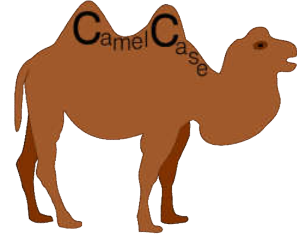
## Cada bit = 2 posibilidades

8 bits:

[illegible]

## Nomes de variáveis

- Não pode começar com dígito: use uma letra ou \_
- Não pode ter espaço em branco
- Não usar acentos ou til
- Sugestão: use o padrão "camel case"



Errado:

```
int 5minutos;  
int salário;  
int salário do funcionario;
```

Correto:

```
int _5minutos;  
int salario;  
int salarioDoFuncionario;
```

## Resumo da aula

- Conceito informal
- Declaração de variáveis: `<tipo> <nome> = valor;`
- Tipos primitivos:
  - **Números inteiros:** byte, short, int, long
  - **Números com ponto flutuante:** float, double
  - **Valor verdade:** boolean
  - **Um caractere Unicode:** char
- Tipo String: cadeia de caracteres (palavras, textos)
- Nomes de variáveis / padrão camel case

## Entrada de dados

**Usuário** → **Programa**  
(dentro de variáveis)



Dispositivo de ENTRADA



Também chamada de  
**LEITURA:**

"O programa está lendo dados."

## Processamento de dados

**É quando o programa realiza os cálculos**



O processamento de  
dados se dá por um  
comando chamado  
**ATRIBUIÇÃO**

```
media = (x + y) / 2.0;
```

## Saída de dados

**Programa → Usuário**



Dispositivo de SAÍDA



**Também chamada de  
ESCRITA:**

"O programa está escrevendo dados."

Para escrever na tela um texto qualquer

**Sem quebra de linha ao final:**

```
System.out.print("Bom dia!");
```

**Com quebra de linha ao final:**

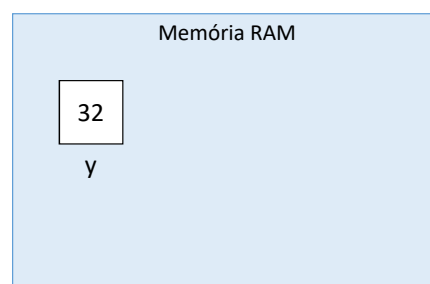
```
System.out.println("Bom dia!");
```

## Para escrever o conteúdo de uma variável de algum tipo básico

Suponha uma variável tipo **int** declarada e iniciada:

```
int y = 32;
```

```
System.out.println(y);
```



## Para escrever o conteúdo de uma variável com ponto flutuante

Suponha uma variável tipo **double** declarada e iniciada:

```
double x = 10.35784;
```

%n = quebra de linha  
(independente de  
plataforma)

```
System.out.println(x);
```

```
System.out.printf("%.2f%n", x);
```

```
System.out.printf("%.4f%n", x);
```

Localidade do  
sistema

### ATENÇÃO:

Para considerar o separador de decimais como ponto, **ANTES** da declaração do Scanner, faça:

```
Locale.setDefault(Locale.US);
```

Para concatenar vários elementos em um mesmo comando de escrita

Regra geral para **print** e **println**:

elemento1 + elemento2 + elemento3 + ... + elementoN

```
System.out.println("RESULTADO = " + x + " METROS");
```

Para concatenar vários elementos em um mesmo comando de escrita

Regra geral para **printf**:

"TEXT01 %f TEXT02 %f TEXT03", variavel1, variavel2

%f = ponto flutuante

%n = quebra de linha

```
System.out.printf("RESULTADO = %.2f metros%n", x);
```

MAIS INFORMAÇÕES: <https://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

## Para concatenar vários elementos em um mesmo comando de escrita

Regra geral para **printf**:

```
"TEXT01 %f TEXT02 %f TEXT03", variavel1, variavel2
```

%f = ponto flutuante

%d = inteiro

%s = texto

%n = quebra de linha

```
String nome = "Maria";  
int idade = 31;  
double renda = 4000.0;  
System.out.printf("%s tem %d anos e ganha R$ %.2f reais%n", nome, idade, renda);
```

MAIS INFORMAÇÕES: <https://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

## Resumo da aula

- System.out.print
- System.out.println
- System.out.printf
  - %d
  - %f
  - %s
  - %n
- Locale
- Como concatenar vários elementos em um mesmo comando de escrita
- Exemplos



## Exercício de fixação

Em um novo programa, inicie as seguintes variáveis:

```
String product1 = "Computer";
String product2 = "Office desk";

int age = 30;
int code = 5290;
char gender = 'F';

double price1 = 2100.0;
double price2 = 650.50;
double measure = 53.234567;
```

Em seguida, usando os valores das variáveis, produza a seguinte saída na tela do console:

```
Products:
Computer, which price is $ 2100,00
Office desk, which price is $ 650,50

Record: 30 years old, code 5290 and gender: F

Measue with eight decimal places: 53,23456700
Rouded (three decimal places): 53,235
US decimal point: 53.235
```

(correção na próxima página)

```
import java.util.Locale;

public class Main {

    public static void main(String[] args) {

        String product1 = "Computer";
        String product2 = "Office desk";

        byte age = 30;
        int code = 5290;
        char gender = 'F';

        double price1 = 2100.0;
        double price2 = 650.50;
        double measure = 53.234567;

        System.out.println("Products:");
        System.out.printf("%s, which price is $ %.2f%n", product1, price1);
        System.out.printf("%s, which price is $ %.2f%n", product2, price2);
        System.out.println();
        System.out.printf("Record: %d years old, code %d and gender: %c%n", age, code, gender);
        System.out.println();
        System.out.printf("Measue with eight decimal places: %.8f%n", measure);
        System.out.printf("Rouded (three decimal places): %.3f%n", measure);
        Locale.setDefault(Locale.US);
        System.out.printf("US decimal point: %.3f%n", measure);


    }
}
```

## Processamento de dados

Comando de atribuição.

**Sintaxe:**

**<variável> = <expressão>;**



Lê-se “recebe”

REGRA:

- 1) A expressão é calculada
- 2) O resultado da expressão é armazenado na variável

## Exemplo 1

```
int x, y;
```

```
x = 5;
```

```
y = 2 * x;
```

```
System.out.println(x);
```

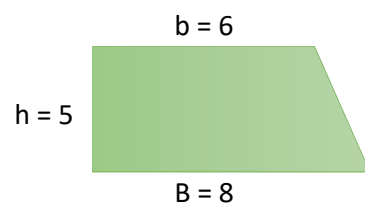
```
System.out.println(y);
```

## Exemplo 2

```
int x;  
double y;  
  
x = 5;  
  
y = 2 * x;  
  
System.out.println(x);  
System.out.println(y);
```

## Exemplo 3

```
double b, B, h, area;  
  
b = 6.0;  
B = 8.0;  
h = 5.0;  
  
area = (b + B) / 2.0 * h;  
  
System.out.println(area);
```



$$area = \frac{(b + B)}{2} \times h$$

**No exemplo:**

$$\begin{aligned} area &= \frac{(6 + 8)}{2} \times 5 \\ &= \frac{14}{2} \times 5 = 7 \times 5 = 35 \end{aligned}$$

```
double b, B, h, area;
```

```
b = 6.0;
```

```
B = 8.0;
```

```
h = 5.0;
```

```
area = (b + B) / 2.0 * h;
```

```
System.out.println(area);
```

Boa prática:

Sempre indique o tipo do número, se a expressão for de ponto flutuante (não inteira).

Para **double** use:

.0

Para **float** use:

f

```
float b, B, h, area;
```

```
b = 6f;
```

```
B = 8f;
```

```
h = 5f;
```

```
area = (b + B) / 2f * h;
```

```
System.out.println(area);
```

Boa prática:

Sempre indique o tipo do número, se a expressão for de ponto flutuante (não inteira).

Para **double** use:

.0

Para **float** use:

f

## Exemplo 4

```
int a, b;  
double resultado;  
  
a = 5;  
b = 2;  
  
resultado = a / b;  
  
System.out.println(resultado);
```

## Casting

É a conversão explícita de um tipo para outro.

É necessário quando o compilador não é capaz de “adivinhar” que o resultado de uma expressão deve ser de outro tipo.

## Exemplo 4

```
int a, b;  
double resultado;  
  
a = 5;  
b = 2;  
  
resultado = a / b;  
  
System.out.println(resultado);
```

## Exemplo 4

```
int a, b;  
double resultado;  
  
a = 5;  
b = 2;  
  
resultado = (double) a / b;  
  
System.out.println(resultado);
```

## Entrada de dados

**Usuário** → **Programa**  
(dentro de variáveis)



Dispositivo de ENTRADA

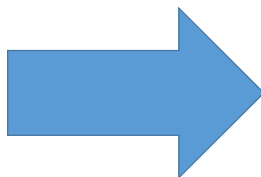


Também chamada de  
**LEITURA:**

"O programa está lendo dados."



32      ENTER



```
int x;
```

Memória RAM

32

x

## Scanner

Para fazer entrada de dados, nós vamos criar um objeto do tipo "Scanner" da seguinte forma:

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

← `import java.util.Scanner;`

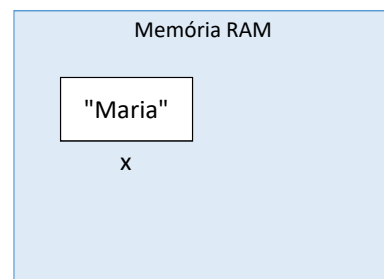
`faça sc.close() quando não precisar mais do objeto sc`

## Para ler uma palavra (texto sem espaços)

Suponha uma variável tipo **String** declarada:

```
String x;
```

```
x = sc.next();
```



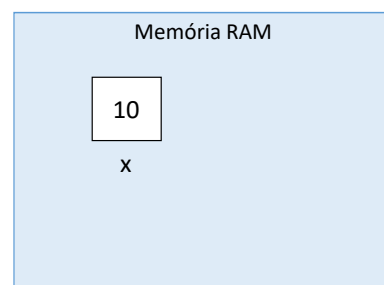


## Para ler um número inteiro

Suponha uma variável tipo **int** declarada:

```
int x;
```

```
x = sc.nextInt();
```



## Para ler um número com ponto flutuante

Suponha uma variável tipo **double** declarada:

```
double x;
```

```
x = sc.nextDouble();
```

← Localidade do sistema

### **ATENÇÃO:**

Para considerar o separador de decimais como ponto, **ANTES** da declaração do Scanner, faça:

```
Locale.setDefault(Locale.US);
```

## Para ler um caractere

Suponha uma variável tipo **char** declarada:

```
char x;
```

```
x = sc.next().charAt(0);
```

## Para ler vários dados na mesma linha

```
string x;  
int y;  
double z;
```

```
x = sc.next();  
y = sc.nextInt();  
z = sc.nextDouble();
```

## Para ler um texto ATÉ A QUEBRA DE LINHA

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String s1, s2, s3;

        s1 = sc.nextLine();
        s2 = sc.nextLine();
        s3 = sc.nextLine();

        System.out.println("DADOS DIGITADOS:");
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);

        sc.close();

    }
}
```

## ATENÇÃO: quebra de linha pendente

Quando você usa um comando de leitura diferente do `nextLine()` e dá alguma quebra de linha, essa quebra de linha fica "pendente" na entrada padrão.

Se você então fizer um `nextLine()`, aquela quebra de linha pendente será absorvida pelo `nextLine()`.

### Solução:

Faça um `nextLine()` extra antes de fazer o `nextLine()` de seu interesse.

```
int x;
String s1, s2, s3;
```

```
x = sc.nextInt();
s1 = sc.nextLine();
s2 = sc.nextLine();
s3 = sc.nextLine();
```

```
System.out.println("DADOS DIGITADOS:");
System.out.println(x);
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
```

## Resumo da aula

- Scanner
  - next()
  - nextInt()
  - nextDouble()
  - next().charAt(0)
- Locale
- Como ler até a quebra de linha
  - nextLine()
  - como limpar o buffer de leitura

## Funções matemáticas em Java

Prof. Ms. Francisco Carlos da Silva  
francisco.silva@unisal.br

## Algumas funções matemáticas em Java

Exemplo	Significado
<code>A = Math.sqrt(x);</code>	Variável A recebe a raiz quadrada de x
<code>A = Math.pow(x, y);</code>	Variável A recebe o resultado de x elevado a y
<code>A = Math.abs(x);</code>	Variável A recebe o valor absoluto de x

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

        double x = 3.0;
        double y = 4.0;
        double z = -5.0;
        double A, B, C;

        A = Math.sqrt(x);
        B = Math.sqrt(y);
        C = Math.sqrt(25.0);
        System.out.println("Raiz quadrada de " + x + " = " + A);
        System.out.println("Raiz quadrada de " + y + " = " + B);
        System.out.println("Raiz quadrada de 25 = " + C);

        A = Math.pow(x, y);
        B = Math.pow(x, 2.0);
        C = Math.pow(5.0, 2.0);
        System.out.println(x + " elevado a " + y + " = " + A);
        System.out.println(x + " elevado ao quadrado = " + B);
        System.out.println("5 elevado ao quadrado = " + C);

        A = Math.abs(y);
        B = Math.abs(z);
        System.out.println("Valor absoluto de " + y + " = " + A);
        System.out.println("Valor absoluto de " + z + " = " + B);
    }
}
```

Incluindo funções em expressões maiores

$$x = \frac{-b \pm \sqrt{\Delta}}{2.a}$$

$$\Delta = b^2 - 4ac$$

```
delta = Math.pow(b, 2.0) - 4*a*c;
```

```
x1 = (-b + Math.sqrt(delta)) / (2.0 * a);
```

```
x2 = (-b - Math.sqrt(delta)) / (2.0 * a);
```

## Funções matemáticas

- sqrt – raiz quadrada
- pow – potenciação
- abs – valor absoluto
- Exemplos

Maiores informações: `java.lang.Math`

Créditos: Prof. Dr. Nelio Alves