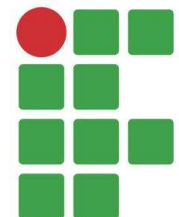


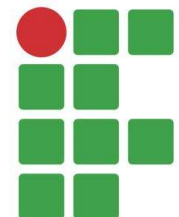
Estrutura Sequencial

Prof. André Chaves Lima

andrelima@iftm.edu.br



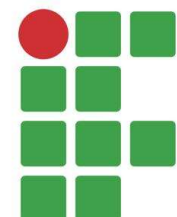
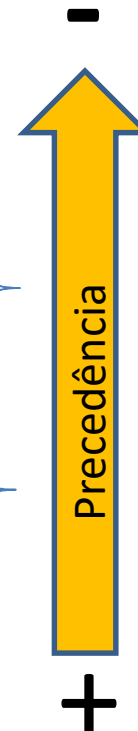
Expressões Aritméticas



Operadores aritméticos








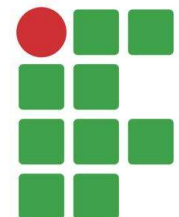
Operadores	Significado
+	Adição
-	Subtração
*	Multiplicação (X)
/	Divisão (÷)
%	Resto da divisão ("mod")



Exemplo



- $2 * 6 / 3$  Resultado = 4
- $3 + 2 * 4$  Resultado = 11
- $(3 + 2) * 4$  Resultado = 20
- $60 / (3 + 2) * 4$  Resultado = 48
- $60 / ((3 + 2) * 4)$  Resultado = 3



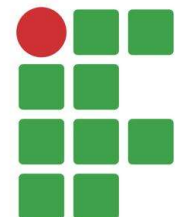
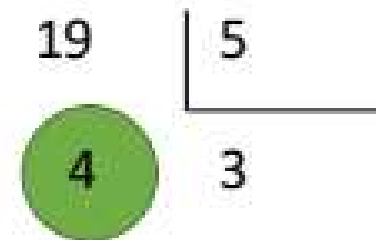
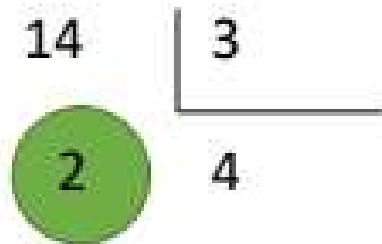


Exemplo “mod” (%)

• $14 \% 3$  Resultado = 2

• $19 \% 5$  Resultado = 4

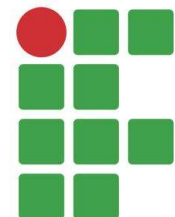
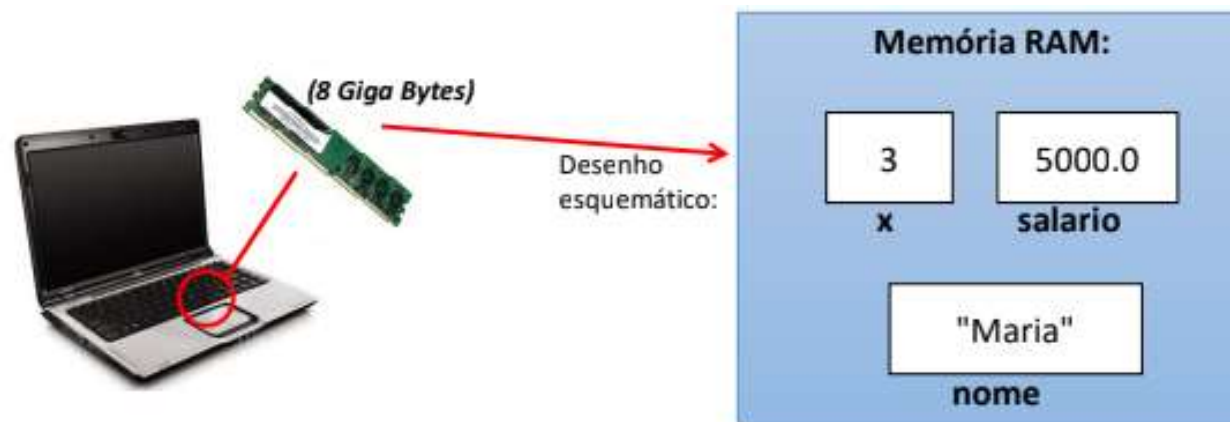
• Pois



Variáveis e Tipos primitivos



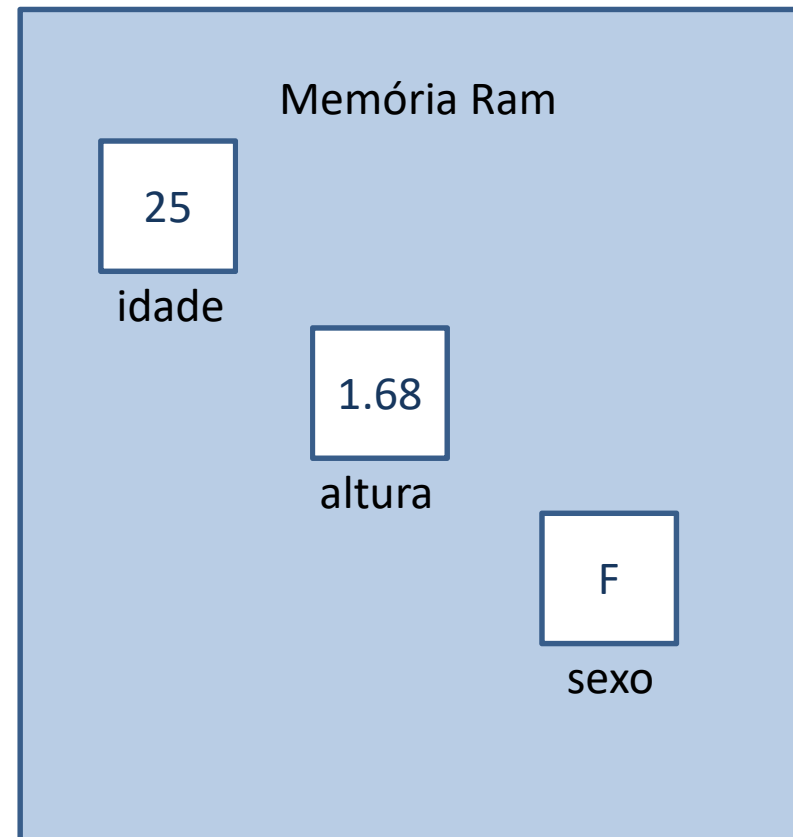
- Variáveis
 - Definição informal:
 - Em programação, uma variável é uma porção de memória (RAM) utilizada para armazenar dados durante a execução dos programas.





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

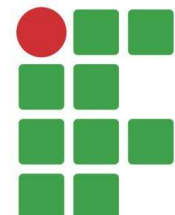


Descrição	Tipo	Tamanho	Valores	Valor padrão
tipos numéricos inteiros	byte	8 bits	-128 a 127	0
	short	16 bits	-32768 a 32767	0
	int	32 bits	-2147483648 a 2147483647	0
	long	64 bits	-9223372036854770000 a 9223372036854770000	0L
tipos numéricos com ponto flutuante	float	32 bits	-1,4024E-37 a 3,4028E+38	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

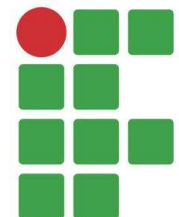
Exemplo: 'a' = '\u0061'





Tipos primitivos

- Um bit pode armazenar 2 valores possíveis (0 ou 1)
- Cada bit = 2 possibilidades
- 8 bits:
- $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$ possibilidades



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”



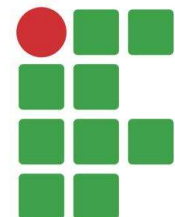
Errado:

```
int 5minutos;  
int salário;  
int salário do funcionário;
```

Correto:

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

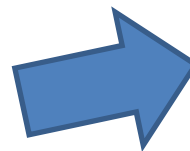
As três operações básicas



Entrada de Dados (Leitura)



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

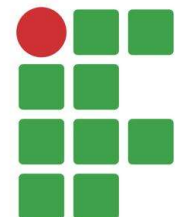


Dispositivo de ENTRADA



Também chamada de
LEITURA:

"O programa está lendo dados."



Processamento de dados

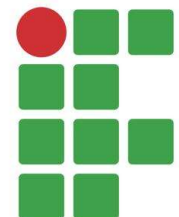


- É quando o programa realiza cálculos



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

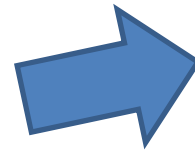
```
media = (x + y) / 2.0;
```



Saída de dados



Programa → Usuário



Dispositivo de SAÍDA



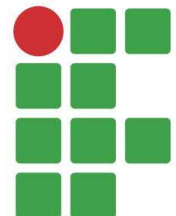
Também chamada de
ESCRITA:

"O programa está escrevendo dados."

Saída de dados em Java



- 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!");`**



Saída de dados em Java



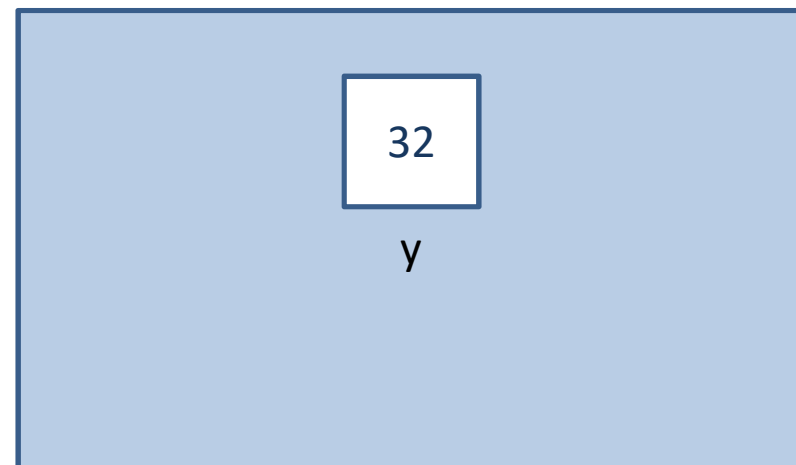
- Para escrever o conteúdo de uma variável de algum tipo básico:

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

```
int y = 32;
```

Memória Ram

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



Saída de dados em Java



- Para escrever o conteúdo de uma variável com ponto flutuante (numero com virgula)

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

```
Double x = 10.35784;
```

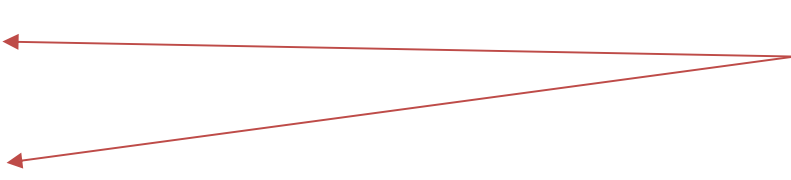
%n = quebra de linha
(independente de
plataforma)

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

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

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

Localidade
do sistema



Saída de dados em Java

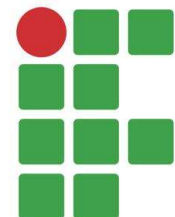


- Para concatenar vários elementos em um mesmo comando de escrita:

Regra geral é usar o *print* e *println*

elemento1 + elemento2 + elemento3 + ... + elementoN

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



Saída de dados em Java



- Para concatenar vários elementos em um mesmo comando de escrita:

Regra geral para o ***printf***

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

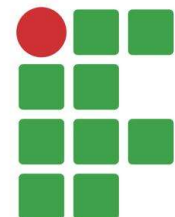
```
System.out.printf("RESULTADO = %.2f metros%n", x);
```

%f = ponto flutuante

%n = quebra de linha

MAIS INFORMAÇÕES:

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



Saída de dados em Java



- Para concatenar vários elementos em um mesmo comando de escrita:

Regra geral para o ***printf***

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

%f = ponto flutuante

%d = inteiro

%s = texto (String)

%n = quebra de linha

```
String nome = "Maria";
```

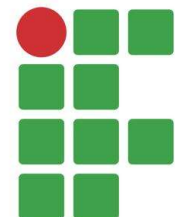
```
int idade = 31;
```

```
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>



Exercício de fixação



Em um novo programa, inicie as seguintes variáveis:

```
String produto1 = "Computador";  
String produto2 = "Mesa de Escritório";
```

```
int idade = 30;  
int código = 5290;  
char genero = 'F';
```

```
double preco1 = 2100.0;  
double preco2 = 650.5;  
double medicao = 53.234567;
```

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

Produtos:

Computador, seu preço é R\$ 2100,00

Mesa de Escritório, seu preço é R\$ 650,50

Record: 30 anos de idade, código: 5290 e gênero: F

Medição com oitos casas decimais: 53,23456700

Medição com três casas decimais: 53,235

Processamento de dados em Java



- Comando de atribuição

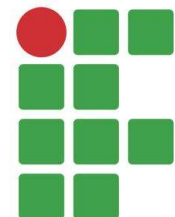
- **Sintaxe:**

<variável> = <expressão>;

↑
Lê-se “recebe”

REGRA:

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



Processamento de dados em Java



- **Exemplo 1**

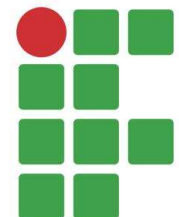
```
int x, y;
```

```
x = 5;
```

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

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

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



Processamento de dados em Java



- **Exemplo 3**

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

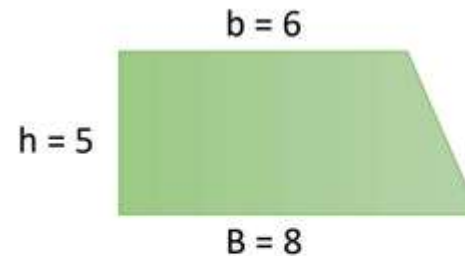
```
b = 6.0;
```

```
B = 8.0;
```

```
h = 5.0;
```

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

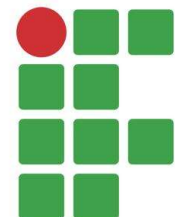
```
System.out.println(area);
```



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

No exemplo:

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



Processamento de dados em Java



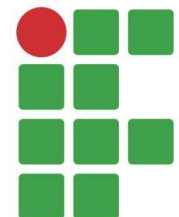
```
double b, B, h, area;  
  
b = 6.0;  
B = 8.0;  
h = 5.0;  
  
area = (b + B) / 2.0 * h  
  
System.out.println(area);
```

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



Processamento de dados em Java



- **Exemplo 3**

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

```
b = 6f;
```

```
B = 8f;
```

```
h = 5f;
```

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

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

Boa prática

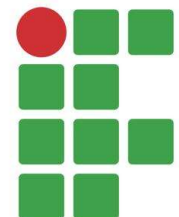
Sempre indique o tipo do número, se a expressão for de ponto flutuante(não inteira).

Para **double** use:

.0

Para **float** use:

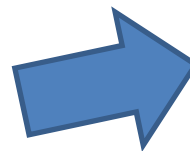
f



Entrada de Dados em Java



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

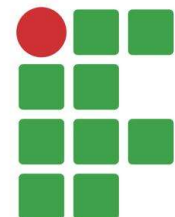


Dispositivo de ENTRADA



Também chamada de
LEITURA:

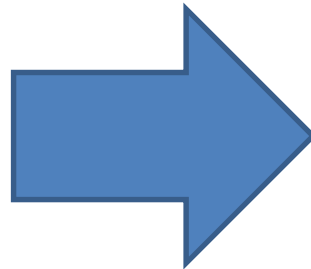
"O programa está lendo dados."



Entrada de Dados em Java

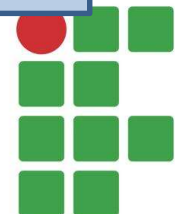
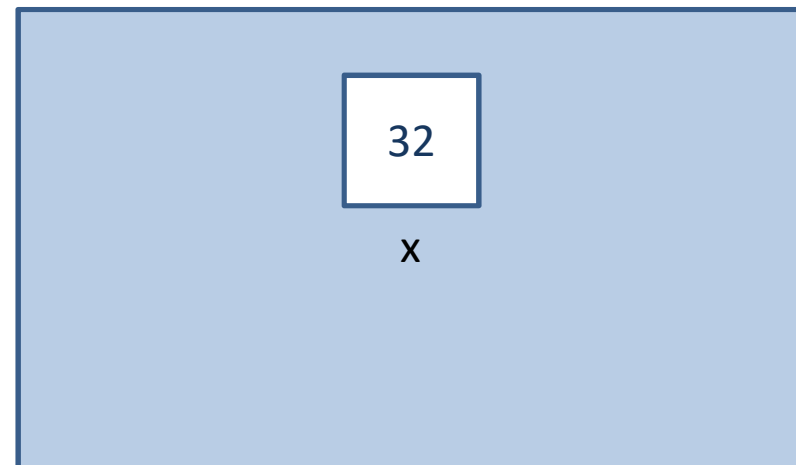


32 ENTER



```
int x;
```

Memória Ram



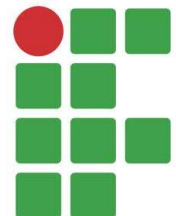
Entrada de Dados em Java



- Scanner
 - Para fazer a 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 o `sc.close()` quando não
precisar mais do objeto `sc`



Entrada de Dados em Java

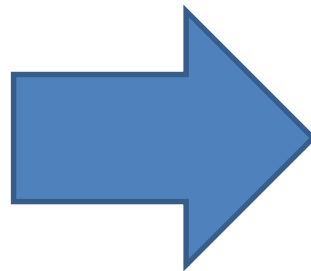


- Para ler uma palavra (texto sem espaços):

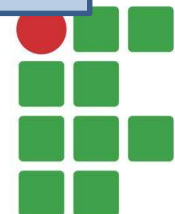
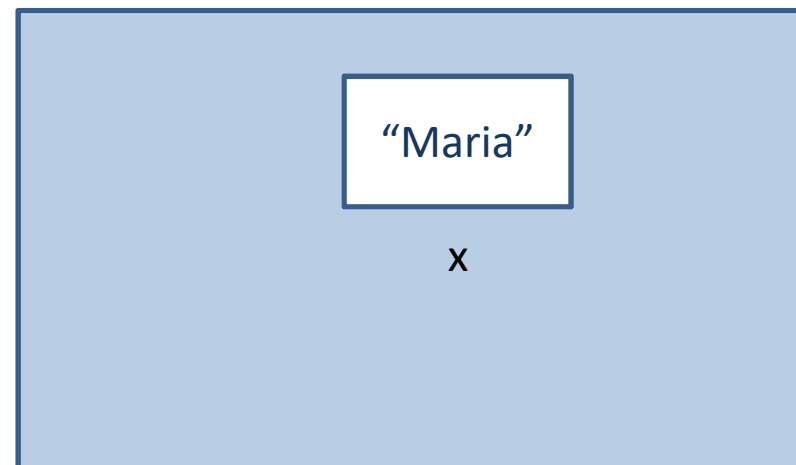
Suponha uma variável tipo **String** declarada:

```
String x;
```

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



Memória Ram



Entrada de Dados em Java

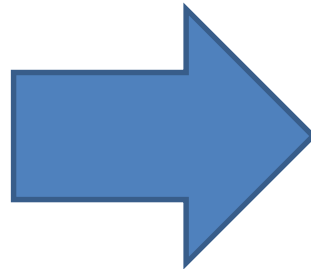


- Para ler um número inteiro:

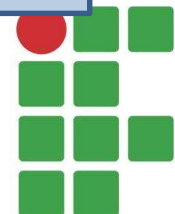
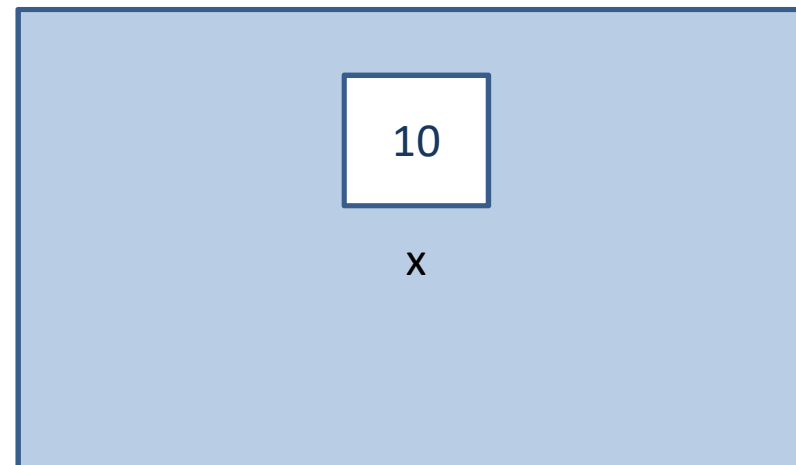
Suponha uma variável tipo **int** declarada:

```
int x;
```

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



Memória Ram



Entrada de Dados em Java



- Para ler um número com ponto flutuante:

Suponha uma variável tipo **double** declarada:

```
double x;
```

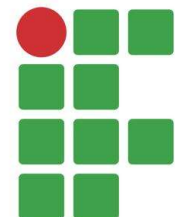
```
x = sc.nextDouble();
```

Localidade do sistema

ATENÇÃO

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

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



Entrada de Dados em Java

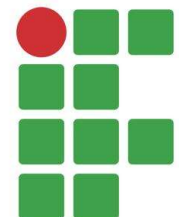


- Para ler um caractere:

Suponha uma variável tipo **char** declarada:

```
char x;
```

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



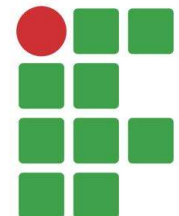
Entrada de Dados em Java



- Para ler vários dados na mesma linha:

```
String x;  
int y;  
double z;
```

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



Entrada de Dados em Java



- Para ler um texto ATÉ A QUEBRA DE LINHA:

```
import java.util.Scanner;

public class Main{

    Run | Debug
    public static void main(String[] args) {

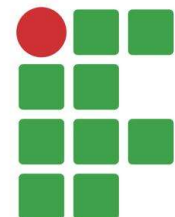
        Scanner sc = new Scanner(System.in);

        String s1, s2, s3;

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

        System.out.println(x: "Dados Digitados:");
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);

        sc.close();
    }
}
```



Entrada de Dados em Java



- **ATENÇÃO:** quebra de linha pendente

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

Se você então fizer um `nextLine()` em seguida aquela quebra de linha pendendo será absorvida pelo `nextLine()`.

Solução:

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

```
import java.util.Scanner;

public class Main{

    Run | Debug
    public static void main(String[] args) {

        int x;
        Scanner sc = new Scanner(System.in);

        String s1, s2, s3;

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

        System.out.println(x: "Dados Digitados:");
        System.out.println(x);
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);

        sc.close();
    }
}
```

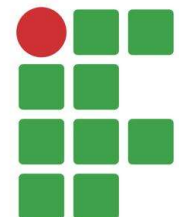
Funções matemáticas em Java



- 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

MAIS INFORMAÇÕES: [java.lang.Math](#)



Funções matemáticas em Java



- 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);
```