

# Introdução a JAVA

Variáveis, tipos, expressões,  
comandos e blocos

# + Roteiro

- Variáveis e tipos
- Operadores
  - aritméticos, lógicos, relacionais e bit-a-bit
  - Atribuição
- Comandos básicos
  - Ler, Escrever, Condicional, Laço



# Variáveis

- *Dois grupos de variáveis*
  - *Tipos primitivos*
  - *Referências para Objetos*
- *Java possui oito tipos primitivos*
  - *byte, short, int, long, float, double, char, boolean*

# + Variáveis

- Declaração:

```
int x,y;  
short s1,s2;  
char c;  
float nota1;
```

- Variáveis ficam na pilha armazenam valor
  - Possuem tamanho fixo



## + Literais

- Valores usados diretamente no código
  - Possuem, implicitamente, um tipo
- Inteiros (int)
  - 10 ; 832 ; 2 ; -832 ; 131
- Ponto flutuante (double)
  - 3.2 ; 4.3 ; 1232.1232 ; 32.2 ; 3.2f
- Caractere
  - 'a' ; '\u0041' ; '\u0065'
- Booleano
  - true ; false

# + Tipos Primitivos

Palavra	Descrição	Tamanho/formato
<b><i>Inteiros</i></b>		
byte	Inteiro de um byte	<b>8 bits</b>
short	Inteiro pequeno	<b>16 bits</b>
int	Inteiro	<b>32 bits</b>
long	Inteiro longo	<b>64 bits</b>
<b><i>Números reais</i></b>		
float	Ponto flutuante de precisão simples	<b>32 bits</b>
double	Ponto flutuante de precisão dupla	<b>64 bits</b>
<b><i>Outros tipos</i></b>		
char	Um caractere	<b>16 bits – Unicode</b>
boolean	Um valor lógico	<b>true ou false</b>

# + Variáveis

- Nomes de variáveis
  - Podem conter caracteres, dígitos, `_` e `$`
  - **Não** podem começar com dígitos
  - **Não** podem conter espaços
  - Maiúsculas **diferentes** de minúsculas
- Exemplos

`nota ; x ; y ; raio ; mediaTotal`

`media_total ; media$total ; media l`

`media2 ; media_1 ; media$l`

`nomePai ; NomePai ; nome_mãe`

# + Operadores

Operador	Função
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto de Divisão
&&	E
	OU
==	IGUAL
!=	DIFERENTE
&	E
	OU
^	OU Exclusivo
>>	Desl. Esquerda
<<	Desl. Direita



# + Operadores Aritméticos

- Operações entre elementos do mesmo tipo
  - Mantém o tipo dos elementos
  - Mínimo tipo resultante é **int**
- Operações entre elementos de tipos diferentes
  - Há promoção do tipo “**menor**” ao “**maior**”
  - **byte, short, char** para **int**
  - **int** para **long**
  - **int, long, float** para **double**

## + Incremento e decremento

- Mesma semântica de C/C++

```
int a = 10;
```

```
int b = 20;
```

```
int c, d;
```

```
c = a++; // c recebe 10
```

```
d = ++b; // d recebe 21
```

**a e b** são incrementados em 1,  
porém **a** só é incrementado *depois* de usado,  
enquanto **b** e incrementado *antes* de ser usado

## + Operadores relacionais

- O resultado é sempre um valor lógico (booleano)
- Comparam valores de expressões
- Exemplos

$x > 10$

$y \leq 10$

$a > b$

$c \neq d$

# + Operadores lógicos

- Resultado é sempre booleano
- Compara valores (expressões) booleanos
- Exemplos

`a > b && x < y;`

`nota >= 0 && nota <= 10;`

`a < 500 || b > 1000;`

curto-circuito

Avaliação termina se resultado puder ser determinado antes de avaliar toda expressão

```
int x = 10; int y = 20;
if (x > 0 || ++x < 20)
    y++;
```

Qual o valor de **x** e **y**?

## + Operadores bit-a-bit

- Operam sobre inteiros e booleanos
- Resultado do mesmo tipo dos operandos
- Não realizam curto-circuito
- Exemplos

```
x = a&b;
```

```
y = x | y&w | z;
```

```
z = x<<1;
```

```
w = z >>2;
```



# + Atribuição

- Em JAVA a atribuição é um operador
  - retorna um valor
  - pode ser usada em expressões
- Exemplo:

```
x = 10;
```

```
y = 20+x;
```

```
a = b = c = 0;
```

```
z = 4+(x=10*y)*(5+y)
```



## + Atribuição

- Os operadores possuem uma forma de atribuição

**var op= expr;**

- Exemplo:

**x+=10; // mesmo que x=x+10;**

**x\*=y+z; // mesmo que x=x\*(y+z);**

**a%=2; // mesmo que a=a%2;**

- Atribuição é válida para os seguinte operadores:

**+ - \* / % & | ^ << >>**

# + Precedência

- Parênteses determinam precedência

$x = (nota1*2 + nota2*3) / 5;$

- Tabela de precedência:

<b>++ -- !</b>	<i>Operadores unários</i>
<b>* / %</b>	<i>Operadores Multiplicativos</i>
<b>+ -</b>	<i>Operadores Aditivos</i>
<b>&lt;&lt; &gt;&gt;</b>	<i>Deslocamento de Bits</i>
<b>&lt; &gt; &lt;= &gt;=</b>	<i>Operadores Relacionais</i>
<b>== !=</b>	<i>Operadores de Igualdade</i>
<b>&amp;</b>	<i>And Bit-a-bit</i>
<b>^</b>	<i>Xor Bit-a-bit</i>
<b> </b>	<i>Or Bit-a-bit</i>
<b>&amp;&amp;</b>	<i>And Lógico</i>
<b>  </b>	<i>Or Lógico</i>
<b>= op=</b>	<i>Atribuições</i>



# + Conversão

- Mesma sintaxe de C/C++
  - (tipo)exp
- Antes de usar o valor converte seu tipo a (tipo)
  - int x;
  - float f = 10;
  - x = (int)f;

# + Comandos

- Atribuição
  - Não é comando, mas é usado como
- Escrever
  - Classe **System**
  - atributo **out**
- Ler
  - Classe **Scanner**

Condicional

if-else, switch-case

Repetição

while, do-while, for

Blocos de comandos

Agrupar mais de um comando

# + Escrever

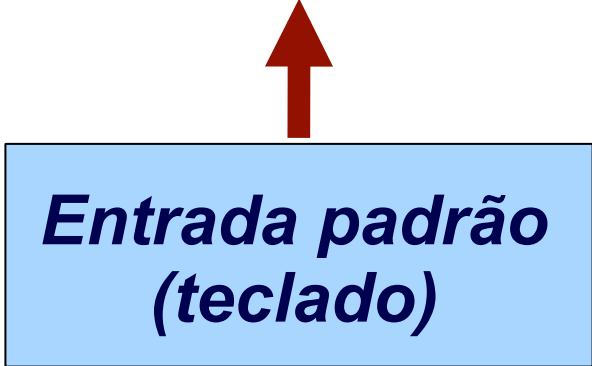
- Atributo **out** da classe **System**
  - metodo **print** e **println**;
- Exemplos:
  - **System.out.println("Teste");**
    - Escreve Teste e avança uma linha
  - **System.out.print("Teste");**
    - Escreve Teste e não avança linha
- Pode-se usar o "+"
  - **System.out.println("Numero:" + n);**



# + Ler

- Necessário criar objeto da classe Scanner
  - Definir variável
    - `java.util.Scanner sc;`
  - Criar Objeto para ler do teclado
    - `sc = new java.util.Scanner(System.in);`
  - Usar para ler dados
    - `int x = sc.nextInt();`

20



***Entrada padrão  
(teclado)***

The diagram consists of a light blue rectangular box with a black border at the bottom. Inside the box, the text "Entrada padrão (teclado)" is written in a bold, italicized, dark blue font. A thick, dark red arrow points vertically upwards from the top center of the box towards the Scanner class mentioned in the text above.

# + Ler

- Tipos primitivos

- `nextByte()`
- `nextShort()`
- `nextInt()`
- `nextLong`
- `nextFloat()`
- `nextDouble()`
- `nextBoolean()`

- Objetos

- `next()`
- `nextLine()`
- `nextBigDecimal()`
- `nextBigInteger()`

## + Exemplo

```
public class Main {  
    public static void main(String[] args) {  
        java.util.Scanner sc;  
        sc = new java.util.Scanner(System.in);  
        int a, b, c;  
        a = sc.nextInt();  
        b = sc.nextInt();  
        c = a + b;  
        System.out.println(a + "+" + b + "=" + c);  
    }  
}
```

## + Bloco de comandos

- Usado onde se espera um comando
  - if, while, ...
- Mesma sintaxe de C/C++
  - Usa-se **{ e }** para **inicio e final de bloco**



## + if-else

- Sintaxe

```
if (expressão)  
    comando;  
else  
    comando;
```

```
if (media >=6.0)  
    System.out.println("Aprovado");  
else  
    System.out.println("Não aprovado");
```

- Else é opcional
  - Refere-se ao último **if**
- expressão **DEVE** ser booleana



## + while e do-while

- Sintaxe

```
while(expressão)  
    comando;
```

```
do  
    comando;  
while (expressão)
```

```
int i = 1;  
int soma = 0;  
while (i < 10)  
    soma +=i++;  
System.out.println(soma);
```

```
int i = 1;  
int soma = 0;  
do{  
    soma +=i;  
    i++;  
} while (i <20);  
System.out.println(soma);
```

## + for

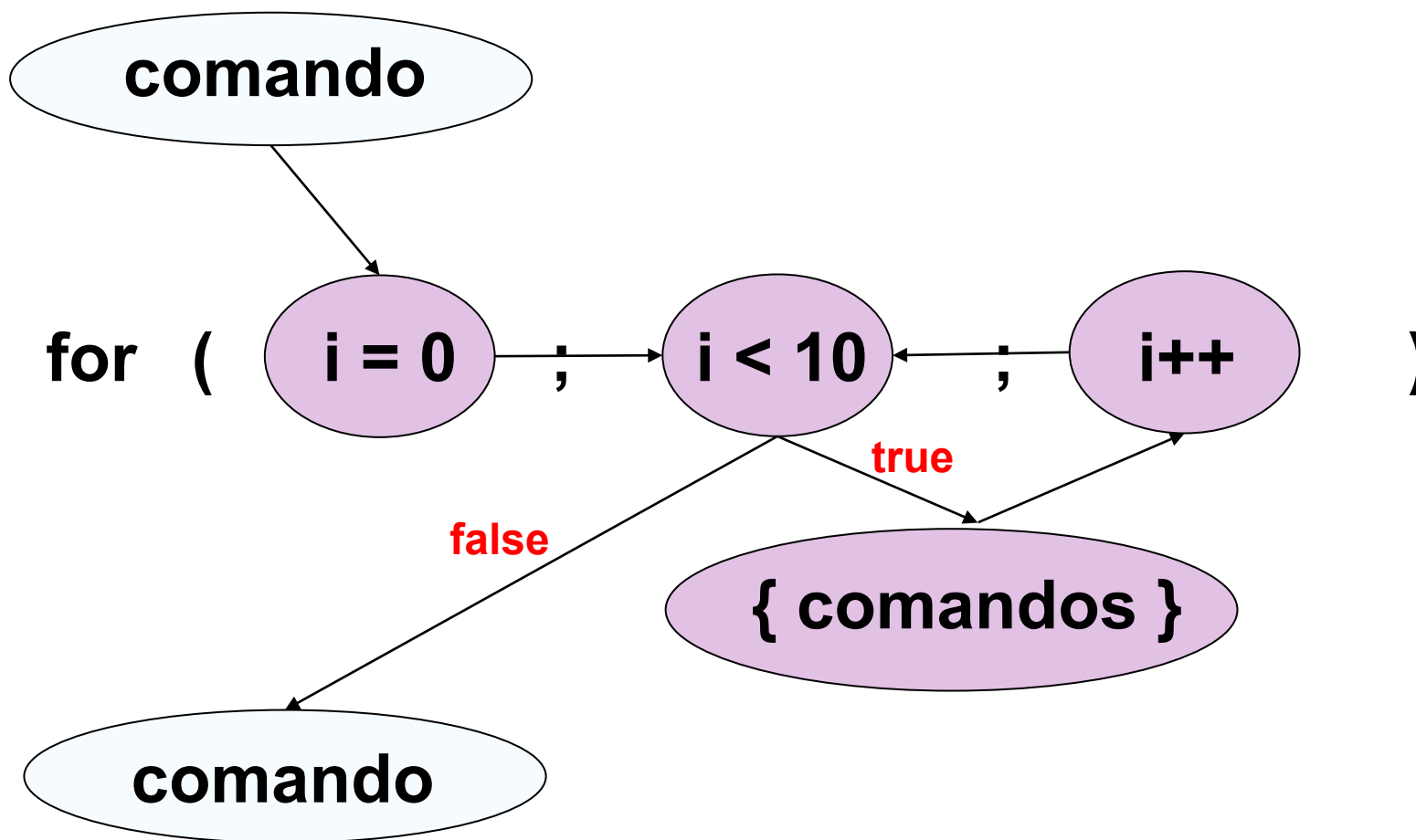
- Sintaxe

```
for (inicializações ; expressão ; passo )  
    comando;
```

```
int soma = 0;  
for (int i = 0 ; i < 10 ; i++)  
    soma+=i;  
System.out.println("A soma é "+soma);
```

# + for

- Execução



# + Dúvidas

