

Material IsiFLIX para uso exclusivo de
Nelson de Campos Nolasco
nelsonnolasco@gmail.com

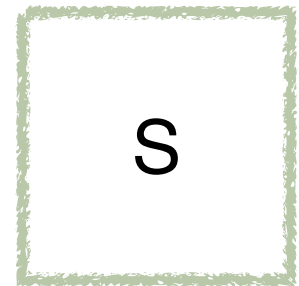
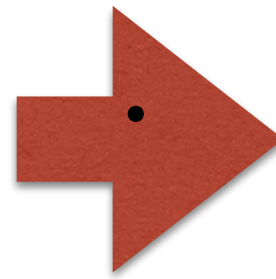
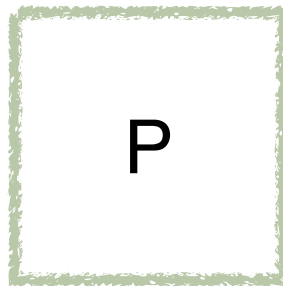
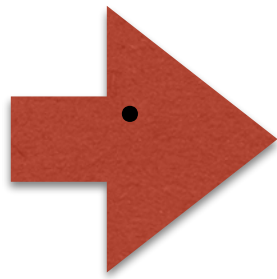
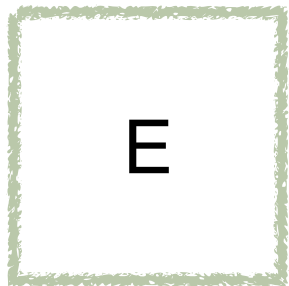


Programação Orientada a Objetos em Java

Aula 2 - Fundamentos da Linguagem Java

Prof. Dr. Francisco Isidro
isidro@professorisidro.com.br

[isi] Já sabemos que...



[isi] Tipos de Dados Primitivos

- Booleanos
- Caracteres
- Inteiros
- Ponto flutuante

Obs.: Além dos tipos de dados primitivos as variáveis em Java podem ser instâncias de qualquer classe definida (Por isso que existe a Orientação a Objetos)

[isi] Booleanos ou Lógicos

- Variáveis do tipo **boolean** podem assumir os valores true ou false.
 - valor *default* para um atributo booleano de uma classe, se não especificado, é false.
 - *Variáveis booleanas e variáveis inteiras, ao contrário do que ocorre em C e C++, não são compatíveis em Java.*
 - *Assim, não faz sentido atribuir uma variável booleana a uma variável inteira ou usar um valor inteiro como uma condição de um teste.*
- Exemplo de declaração e uso:

```
boolean deuCerto = true;
```

[isi] Caracteres

- Uma variável do tipo **char** contém um caracter Unicode, ocupando 16 bits de armazenamento em memória.
 - um valor literal do tipo caracter é representado entre aspas simples (apóstrofes), como em:

```
char umCaracter = 'A';
```
 - Valores literais de caracteres podem também ser representados por seqüências de escape, como em '\n' (nova linha)

[isi] Inteiros

- Valores numéricos inteiros em Java podem ser representados por variáveis do tipo **byte**, **short**, **int** ou **long**.
- Todos os tipos contêm valores inteiros com sinal.
- O valor *default* para atributos desses tipos é 0.
 - **byte**: ocupam 8 bits de armazenamento interno (-128 a +127).
 - **short**: ocupam 16 bits de armazenamento interno (-32.768 a +32.767).
 - **int**: ocupam 32 bits de armazenamento interno (-2.147.483.648 a +2.147.483.647).
 - **long**: ocupam 64 bits de armazenamento interno (-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807).
 - Constantes literais do tipo long podem ser identificadas em código Java através do sufixo l ou L, como em:

```
long valorQuePodeCrescer = 100L;
```
 - *Ao contrário do que ocorre em C, não há valores inteiros sem sinal (unsigned) em Java. Combinações da forma long int ou short int são inválidas em Java.*

[isi] Ponto Flutuante

- Valores reais, com representação em ponto flutuante, podem ser representados por variáveis de tipo **float** ou **double**.
- O valor *default* para atributos desses tipos é 0.0.
 - **float**: ocupam 32 bits de armazenamento interno com nove dígitos significativos de precisão.
 - **double**: ocupam 64 bits de armazenamento interno com 18 dígitos significativos de precisão.
 - Constantes literais do tipo float podem ser identificadas no código Java pelo sufixo f ou F; do tipo double, pelo sufixo d ou D.

[isi] Conversões de Tipos de Dados

- String para int?
- int para String?
- float para int?
- float para String?
- double para int?
- String para double?..



java.lang.Integer

java.lang.Float

java.lang.Double

java.lang.String



Conversões de Tipos de Dados

- String para double
 - `Double.parseDouble(String)`
- String para int
 - `Integer.parseInt(String)`
- String para Float
 - `Float.parseFloat(String)`
- int, float, double para String
 - `toString()` ou `toString(tipo)`
 - `String.valueOf(int ou float ou double)`

[isi] Estruturas de Controle

- Antes de escrever um programa
 - Entendimento completo do problema
 - Abordagem cuidadosamente planejada para resolvê-lo
 - Entender os tipos de blocos de construção disponíveis
 - Empregar princípios comprovados de construção de programas

[isi]

o principal, antes
de começar...

CAAAAAARMAAAAAAAAAAAA

sim, mantenha a calma

[isi] Estruturas de Controle

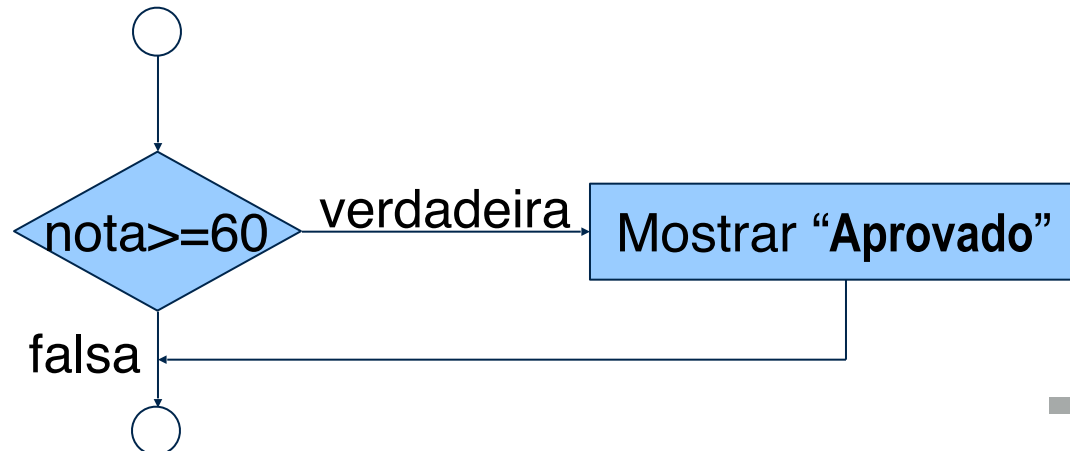
- As instruções em um programa:
 - São executadas na ordem em que são escritas
 - Execução sequencial
- Transferência de controle (Java)
 - Estrutura de seqüência
 - Estrutura de seleção (3 tipos)
 - Estrutura de repetição (3 tipos)

[isi] Estruturas de Controle

- Estruturas de Seleção

- *if (estrutura de seleção única)*
 - Executa uma ação indicada somente quando a condição for verdadeira
 - Exemplo: suponha que a nota de aprovação em um exame seja 60 (em 100).

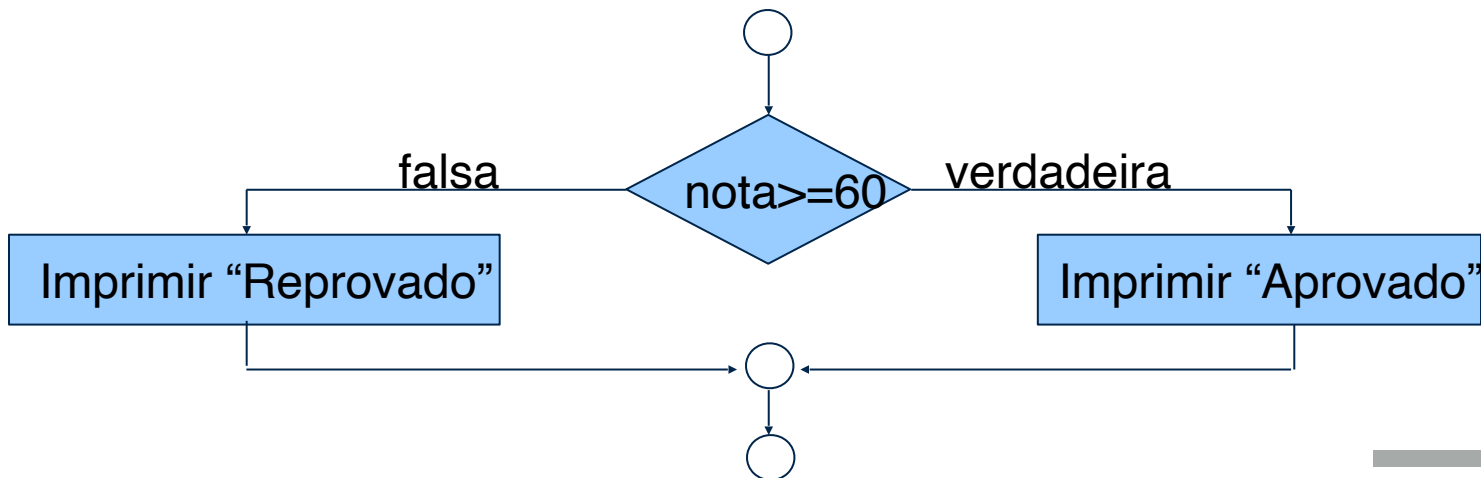
```
if ( nota >= 60 )  
    System.out.println ( "Aprovado" );
```



[isi] Estruturas de Controle

- Estruturas de seleção (continuação)
 - `if/else` (*estrutura de seleção dupla*)
 - Especifica ações separadas que serão executadas quando a condição for verdadeira e quando for falsa.
 - Considere o exemplo anterior:

```
if ( nota >= 60 )  
    System.out.println ( "Aprovado" );  
else  
    System.out.println ( "Reprovado" );
```



[isi] Operador AND (&&)

AND

Condicao 1 (faz sol?)	Condicao 2 (tem combustivel?)	Resultado
V	V	V
V	F	F
F	V	F
F	F	F

[isi] Operador OR (||)

OR

Condicao 1	Condicao 2	Resultado
V	V	V
V	F	V
F	V	V
F	F	F

Estruturas de Controle

- Exemplo estrutura if/else aninhadas: imprimirá **A** para as notas ≥ 90 , **B** para as notas no intervalo de 80 a 89, **C** para as notas no intervalo de 70 a 79, **D** para as notas no intervalo de 60 a 69 e **F** para as demais notas

```
if ( nota >= 90 )
    System.out.println ( "A" );
else
    if ( nota >= 80 )
        System.out.println ( "B" );
    else
        if ( nota >= 70 )
            System.out.println ( "C" );
        else
            if ( nota >= 60 )
                System.out.println ( "D" );
            else
                System.out.println ( "F" );
```

Dica: se existirem vários níveis de recuo, cada nível deve ser recuado pela mesma quantidade adicional de espaço.



Estruturas de Controle - Olha a pegadinha!

- Exemplo:

```
if ( x > 5 )  
    if ( y > 5 )  
        System.out.println("x e y são > 5");  
    else  
        System.out.println("x é <= 5");
```

- o corpo da primeira estrutura if é uma estrutura if/else.

- testa se $x > 5$, se for, a execução continua testando se $y > 5$. Se a segunda condição for verdadeira, exibe **x e y são > 5**

- entretanto se a segunda condição for falsa, o string **x é <= 5** é exibido, embora saibamos que $x > 5$.

```
if ( x > 5 ) {  
    if ( y > 5 )  
        System.out.println("x e y > 5");  
}  
else  
    System.out.println("x é <= 5");
```

- para forçar a estrutura aninhada anterior a ser executada como pretendido a estrutura deve ser escrita como no exemplo acima.

- as { } indicam ao compilador que a segunda estrutura if está no corpo da primeira estrutura if e que o else corresponde à primeira estrutura if.

[isi] Estruturas de Controle

- Para incluir mais instruções no corpo de um if, inclua as instruções entre { e } - **bloco**

```
if ( nota >= 60 )  
    System.out.println ( "Aprovado" );  
else {  
    System.out.println ( "Reprovado" );  
    System.out.println ( "Você deve repetir este curso" );  
}
```

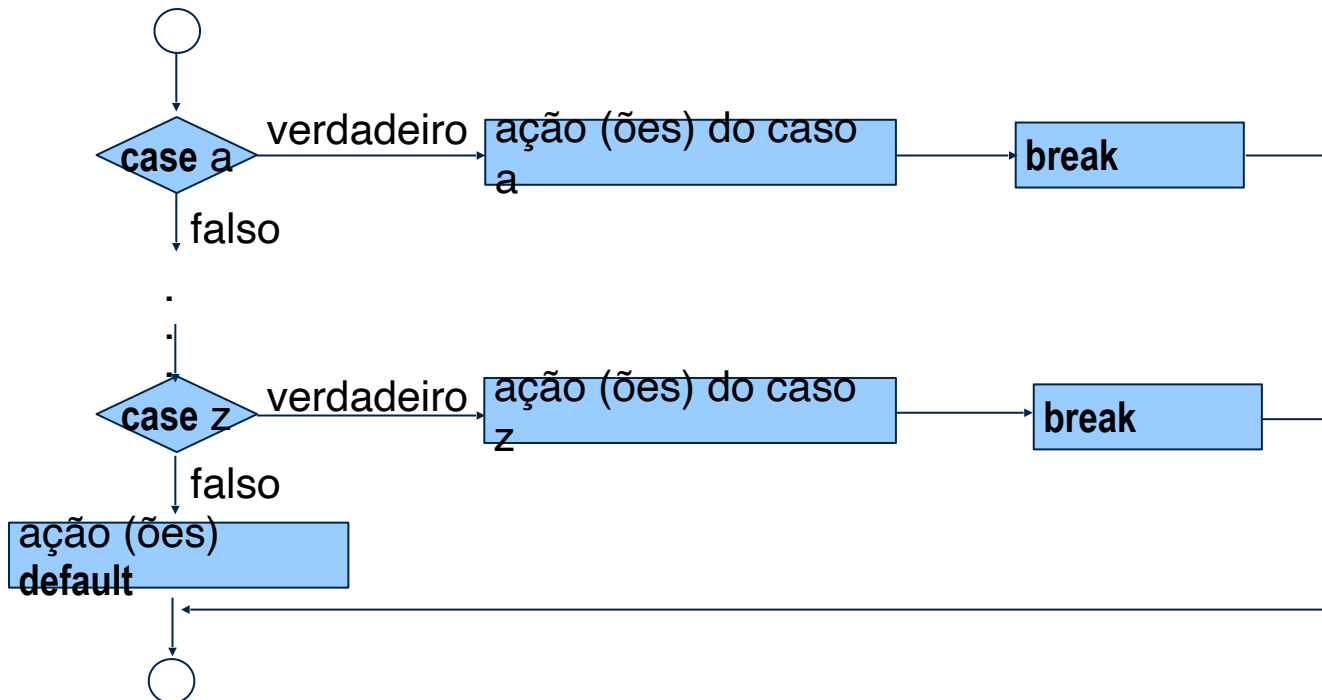
- observe que as chaves cercam as duas instruções na cláusula **else**. Essas chaves são importantes, pois sem elas a instrução

System.out.println ("Você deve repetir este curso");

estaria fora do corpo da parte **else** da estrutura if e seria executada independentemente da nota ser ou não menor que 60.

Estruturas de Controle

- **switch** (*estrutura de seleção múltipla*)
 - Trata uma série de condições nas quais uma variável ou expressão particular é comparada com valores que ela pode assumir e diferentes ações são tomadas
 - **break** depois das instruções para cada **case**: faz com que o controle saia imediatamente da estrutura **switch**
 - Só pode comparar com expressões constantes integrais





```
System.out.println(num + " x 1 = " + (num * 1));  
System.out.println(num + " x 2 = " + (num * 2));  
System.out.println(num + " x 3 = " + (num * 3));  
System.out.println(num + " x 4 = " + (num * 4));  
System.out.println(num + " x 5 = " + (num * 5));  
System.out.println(num + " x 6 = " + (num * 6));  
System.out.println(num + " x 7 = " + (num * 7));  
System.out.println(num + " x 8 = " + (num * 8));  
System.out.println(num + " x 9 = " + (num * 9));  
System.out.println(num + " x 10 = " + (num * 10));
```

[isi] Estruturas de Controle

- Estruturas de repetição

- `while`

- Especifica que uma ação deve ser repetida enquanto alguma condição permanecer verdadeira

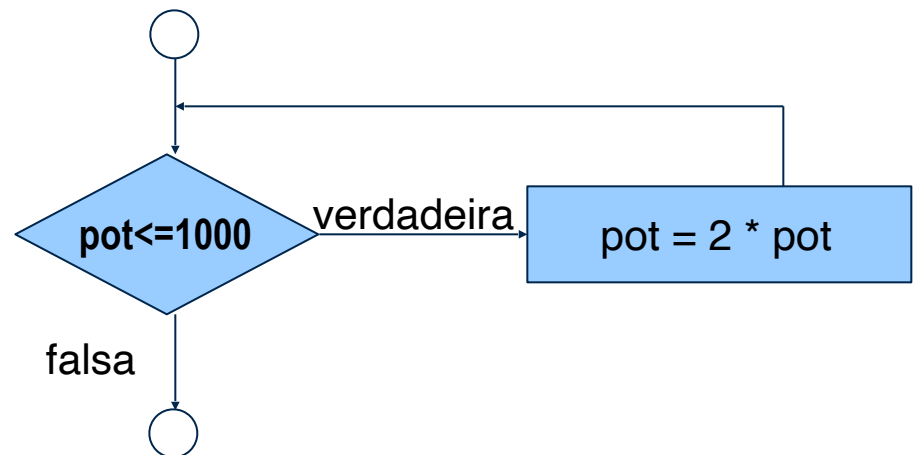
`while` (*condição*)

instrução

- Exemplo: segmento de programa projetado para encontrar a primeira potência de 2 maior que 1000

```
int pot = 2;
```

```
while (pot <= 1000)  
    pot = 2 * pot;
```



[isi] Teste de Mesa

POT pot=2 while (pot < 1000)

2 pot = 2 * pot;

$2*2 = 4$

$2*4 = 8$

$2*8 = 16$

$2*16 = 32$

$2*32 = 64$

$2*64 = 128$

$2*128 = 256$

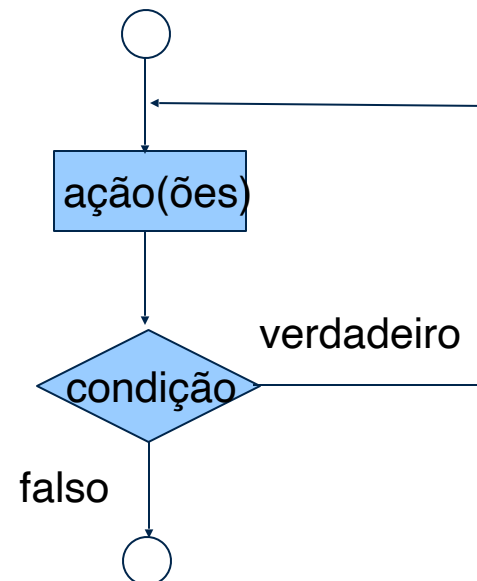
$2*256 = 512$

$2*512 = 1024$

[isi] Estruturas de Controle

- `do/while`
 - Testa a condição de continuação do laço no final do laço
 - O corpo do laço será executado pelo menos uma vez

```
do {  
    instrução  
} while (condição);
```





Estruturas de Controle

- `for`
 - trata de todos os detalhes da repetição controlada por contador e exige o seguinte:
 - o *nome* de uma variável de controle;
 - o *valor inicial* da variável de controle;
 - o *incremento* (ou *decremento*) pelo qual a variável de controle é modificada a cada passagem pelo laço (cada iteração do laço); e
 - A condição que teste o *valor final* da variável de controle.

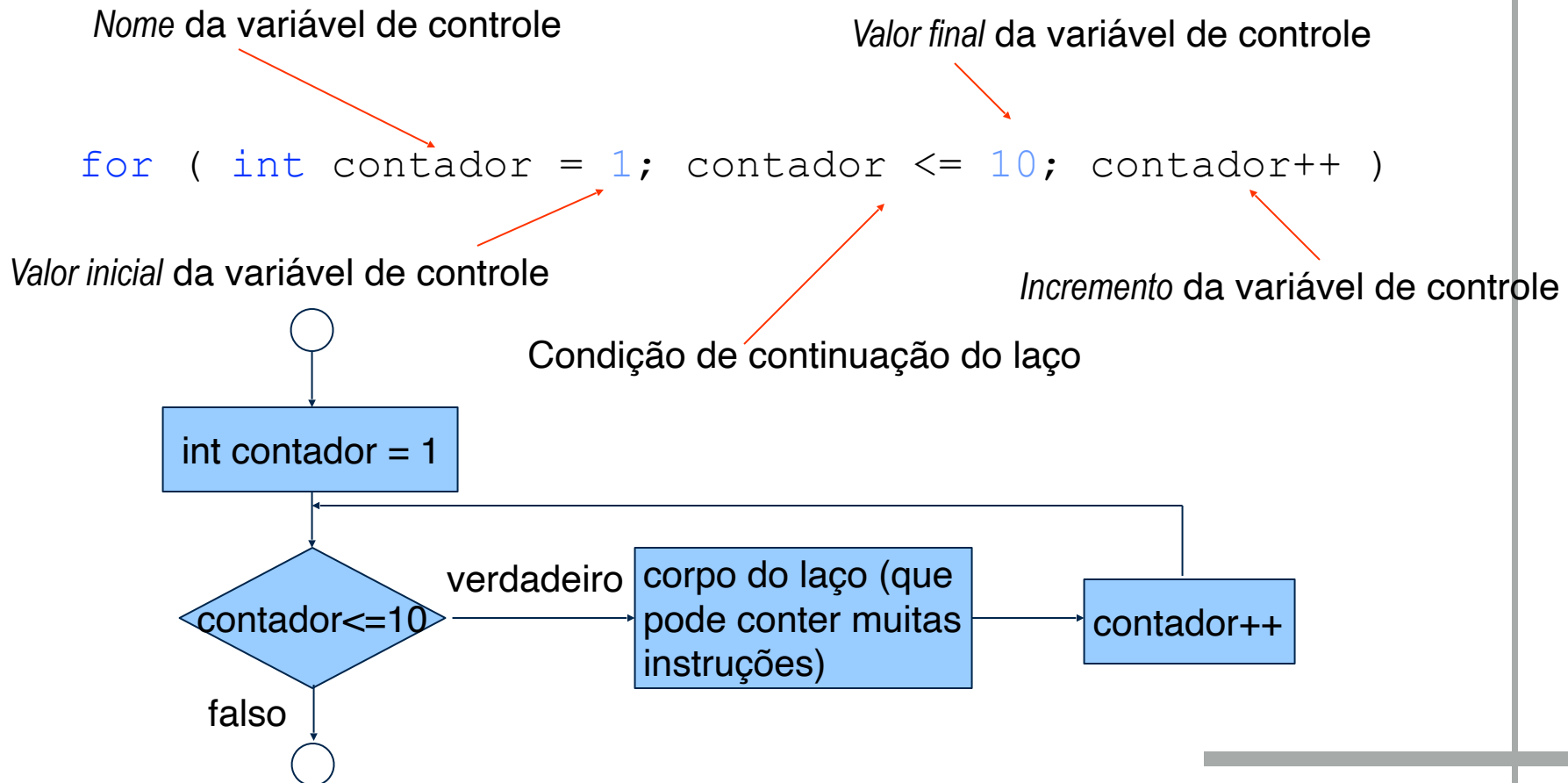
`for` (*expressão1*; *expressão2*; *expressão3*)

instrução

- *Expressão1*: inicializa a variável de controle do laço;
- *Expressão2*: é a condição de continuação do laço;
- *Expressão3*: incrementa a variável de controle, até que a condição de continuação do laço se torne falsa

Estruturas de Controle

- Componentes de um cabeçalho de **for** típico





Estruturas de Controle

- Exemplos com a estrutura for:

- Variável de controle de 1 a 100 em incrementos de 1

```
for ( int i = 1; i <= 100; i++ )
```

- Variável de controle de 100 a 1 em incrementos de -1

```
for ( int i = 100; i >= 1; i-- )
```

- Variável de controle de 7 a 77 em incrementos de 7

```
for ( int i = 7; i <= 77; i += 7 )
```

- Variável de controle de 20 a 2 em incrementos de -2

```
for ( int i = 20; i >= 2; i -= 2 )
```

- Variável de controle assumir a seguinte seqüência de valores: 2, 5, 8, 11, 14, 17, 20.

```
for ( int j = 2; j <= 20; j += 3 )
```

- Variável de controle assumir a seguinte seqüência de valores: 99, 88, 77, 66, 55, 44, 33, 22, 11.

```
for ( int j = 99; j > 0; j -= 11 )
```

[isi] Estruturas de Controle

- **break**

- Quando executada em uma das instruções de repetição (**while**, **for**, **do/while** ou **switch**), causa saída imediata dessa estrutura
- A execução continua com a primeira instrução depois da estrutura

- **continue**

- Quando executada em uma das instruções de repetição (**while**, **for**, **do/while**), pula qualquer instrução restante no corpo da estrutura e prossegue com o teste para a próxima iteração do laço
 - Nas estruturas **while** e **do/while**, o programa avalia o teste de continuação do laço imediatamente depois da instrução **continue** ser executada
 - Em estruturas **for**, a expressão de incremento é executada e depois o programa avalia o teste de continuação do laço



Scanner (objeto de entrada formatada)

```
import java.util.Scanner;
public class LeituraFormatada {
    public static void main (String args[]){
        Scanner teclado = new Scanner(System.in);
        int valor = teclado.nextInt();
        String texto = teclado.nextLine();
        double outroValor = teclado.nextDouble();
        float  valorReal = teclado.nextFloat();
    }
}
```

Scanner é poderoso, porém tem algumas particularidades

Ele assume o idioma padrão do computador!