

Java – Estruturas de Controle

Tito Kenzo



Objetivos da Aula

- if /else
- switch case
- for
- while

Expressões de Controle de Fluxo

- Expressões Condicionais
 - if, else
 - switch
- Expressões de Repetição
 - for
 - while
 - do / while

Expressões Condicionais

- **if / else**

- Permitem a execução seletiva de trechos de programa de acordo com os valores de algumas expressões.

```
if (condição)    //"condição" pode ser uma expressão
{
    declarações ou blocos
}
else            //caso contrário
{
    declarações ou blocos
}
```

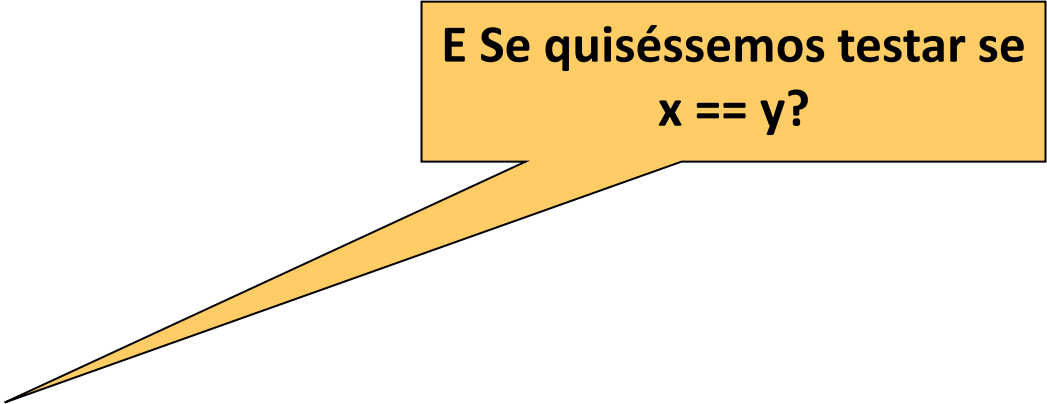
Expressões Condicionais

- **if / else**

- Exemplo:

```
...  
int x = 10;  
int y = 15;  
int menor;  
if (x > y) {  
    menor = y;  
} else {  
    menor = x  
}  
...
```

//se x < y ou se x == y



**E Se quiséssemos testar se
x == y?**

Expressões Condicionais


- **if / else**

- Operador Ternário

expressao ? declaracao1 : declaracao2;

declaracao1 e declaracao2 são obrigadas a retornar o mesmo tipo de dado (não pode ser void).

```
int x = 10;  
int y = 15;  
int menor;  
if (x > y) {  
    menor = y;  
} else {  
    menor = x;  
}
```

 **menor = (x > y) ? y : x;**

Expressões Condicionais

- **if / else**

```
public class NumeroRadomico {  
    static int max = 100, maxPermitido = 100;  
    public static void main (String args[]) {  
        int numAleatorio = (int) ((Math.random() * max) + 1);  
        if (numAleatorio > maxPermitido)  
            System.out.println ("Valor fora do intervalo");  
        else  
            System.out.println (numAleatorio);  
    }  
}
```

O uso de delimitadores de bloco **{ }** são necessários apenas se o bloco if ou else tiver mais de uma linha

Expressões Condicionais

- **switch**

- Utilizado caso se tenha vários if e else aninhados.

```
switch (expressao) {  
    case constante1:  
        expressoes;  
        break;  
    case constante2:  
        expressoes;  
        break;  
    default:  
        expressoes;  
        break;  
}
```

<expressao> deve ser um tipo compatível com int (byte, short e char) ou membro da classe String ou enum

<constante> deve ter um tipo compatível com int (byte, short e char)

break força o término do bloco switch. Caso contrário a execução continua sem checagem nos case posteriores.

default especifica o bloco de código que será executado caso a expressão não case com nenhum case.

Expressões Condicionais

- **switch**

```
switch (modeloCarro) {  
    case LUXO:  
        adicionaArCondicionado();  
        adicionaRadio();  
        adicionaRodas();  
        break;  
    case STANDARD:  
        adicionaRadio();  
        adicionaRodas();  
        break;  
    default:  
        adicionaRodas();  
}
```

Expressões Condicionais

- **switch**

```
switch (modeloCarro) {  
    case LUXO:  
        adicionaArCondicionado();  
    case STANDARD:  
        adicionaRadio();  
    default:  
        adicionaRodas();  
}
```

Expressões de Repetição (loop)

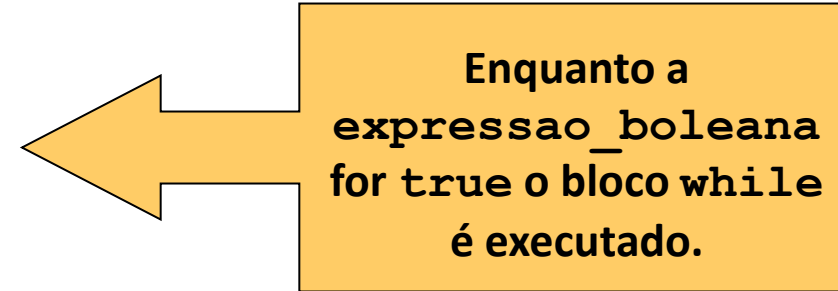
- Repetição (loop) em linguagem de programação é executar um bloco de instruções, repetidamente, enquanto uma condição for verdadeira.
- Java fornece três tipos de loop:
 - for
 - while
 - do – while
- Todo loop é formado por quatro partes:
 - Inicialização → Estado inicial do loop (inicializar todas variáveis)
 - Iteração → Serve para controlar a execução do loop.
 - Corpo → Código a ser executado enquanto condição válida
 - Finalização → Uma expressão é testada para verificar se o loop continuará ou não.

Expressões de Repetição (loop)

- while

- Sintaxe

```
while (expressao_booleana)
{
    expressoes ou blocos;
}
```



- Exemplo

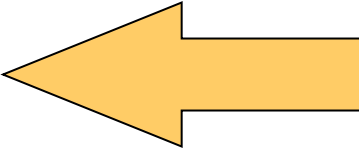
```
int i = 0;                //inicialização
while (i < 5) {            //finalização
    mostraItem(i);         //corpo
    i++;                   //iteração
}
```

Expressões de Repetição (loop)

- do-while

- Sintaxe

```
do {  
    expressoes ou blocos;  
}  
while (expressao_booleana);
```



A condição só será válida após a execução do loop.

O bloco será executado pelo menos uma vez

- Exemplo

```
int i = 0;                //inicialização  
do {  
    mostraItem(i);        //corpo  
    i++;                  //iteração  
} while (i < 5);           //finalização
```

Expressões de Repetição (loop)

- for

- Sintaxe

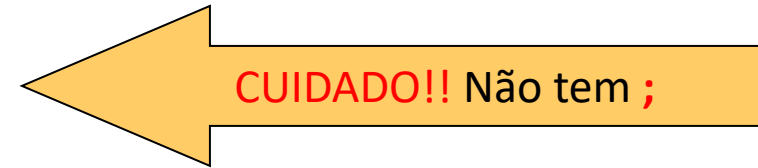
```
for(expr_inicial;teste_booleano;alter_expr){  
    expressões ou blocos  
}
```

- Exemplo

```
for (int i=0; i<17; i++)  
    System.out.println(i);
```

- Exemplo

```
for (int i=0; i<17; i++);  
    System.out.println("erro");
```



Expressões de Repetição (loop)

🕒 Quadro Resumo:

LOOP	EXECUTA	UTILIZAR QUANDO:
for	Zero ou mais vezes	O número de iterações for conhecido com antecedência.
while	Zero ou mais vezes	O número de iterações não for conhecido com antecedência.
do-while	Uma ou mais vezes	O número de iterações não for conhecido com antecedência.

Controles Especiais de Fluxo

- Em alguns casos é necessário terminar a execução normal de um loop antecipadamente.

Em Java isso é possível através do `break` e `continue`.

- `break [label]`

Causa a finalização do loop.

- `continue [label]`

Prematuramente completa a iteração do loop.

- `label : expressao;`

Permite um controle sobre o local para onde a execução será transferida.

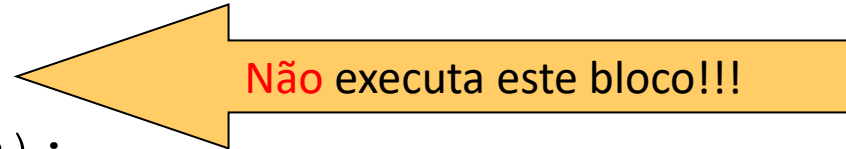
Onde `expressao` deve ser um loop;

Controles Especiais de Fluxo

- break

- Exemplo

```
do {  
    expressoes;  
    if (condição)  
        break;  
    mais_expressoes;  
} while (expressao_boleana);
```



Controles Especiais de Fluxo

- continue

- Exemplo

```
do {  
    expressoes;  
    if (condição)  
        continue;  
    mais_expressoes;  
} while (expressao_boleana);
```

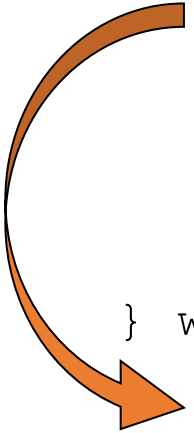


Não executa este bloco!!!

Controles Especiais de Fluxo

- break com label
 - Exemplo

```
outer:do {  
    expressoes;  
    do {  
        if (condição)  
            break outer;  
        mais_expressoes;  
    } while (expressao_booleana);  
    outras_expressoes;  
} while (expressao_booleana);
```



Tarefinha

- Escreva um programa que gere a seguinte saída:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Tarefinha

- Escreva uma aplicação chamada Fatorial, que calcule e imprima o fatorial de 2, 4, 6 e 10.

- $2! = 2 * 1 = 2$
- $4! = 4 * 3 * 2 * 1 = 24$
- $5! = 5 * 4 * 3 * 2 * 1 = 120$

$$n \geq 0$$

$$n! = n * (n-1)!$$

$$0! = 1$$