

Estruturas de Controle Java

Prof. Alexandre Krohn



Roteiro

- Estruturas de Controle Básicas
 - Comandos condicionais
 - Comandos de loops
- Exercícios



Estruturas de Controle

- Comandos condicionais
 - *if*
 - *if else*
 - *if ternário*
 - *switch case*



Comando *if*

- Faz um desvio no fluxo de execução de um método, baseado em uma condição lógica

if (<condicao>) comandos



Comando *if*

- Exemplos:

```
if(a>3)
    b = 2;
```

ou

```
if(a>3) {
    b = 2;
    c = 5;
}
```



Comando *if*

- Exemplos:

```
if(a>3)
    b = 2;
```

ou

```
if(a>3) {
    b = 2;
    c = 5;
}
```

Atenção para esses
Detalhes!



Delimitadores de bloco

- **{** marca o **início** de um bloco de comandos
- **}** marca o **fim** de um bloco de comandos
 - Um bloco pode ser um determinado por um condicional (if), um laço (for, while), um método, ou mesmo uma classe

Blocos : armadilhas

- Exemplos:

```
if(a>3)
    b = 2;
c = 5;
```

≠

```
if(a>3) {
    b = 2;
    c = 5;
}
```


Blocos : armadilhas

- Exemplos:

```
if(a>3)
  b = 2;
c = 5;
```

≠

```
if(a>3) {
  b = 2;
  c = 5;
}
```

b = 2; só é executado se **a > 3**,
c = 5; é sempre executado

Todo o bloco só é executado
se **a > 3**



E a indentação?

- Só serve para orientar o programador
- Não tem efeito algum no programa
- Na verdade, poderia ser feito um programa inteiro em uma linha.



E a indentação?

- Só serve para orientar o programador
- Não tem efeito algum no programa
- Na verdade, poderia ser feito um programa inteiro em uma linha.
- **NÃO façam isso!!!!**



Comando *if else*

- Se a condição for verdadeira, executa o primeiro bloco de comandos, caso contrário executa o segundo

if (<condicao>) comandos

else comandos

Comando *if else*

- Exemplos:

```
if(a>3)
    b = 2;
else
    c = 5;
```

ou

```
if(a>3) {
    b = 2;
} else {
    c = 5;
}
```

ou

```
if(a>3) {
    b = 2;
    d = b + 2;
} else {
    c = 5;
}
```

Comando *if else*

Acostume-se a usar delimitadores de bloco mesmo quando o bloco só possuir uma linha. Isso evitará bugs futuros!

- Exemplos:

```
if(a>3)
    b = 2;
else
    c = 5;
```

ou

```
if(a>3) {
    b = 2;
} else {
    c = 5;
}
```

ou

```
if(a>3) {
    b = 2;
    d = b + 2;
} else {
    c = 5;
}
```


if ternário

- Um *if* ternário tem o comportamento de um *if*, embora sem a palavra *if*
- Exemplo:

```
int a = 5;  
int b = a > 3 ? 2 : 7;
```

A variável b receberá o valor 2 se o valor de a for maior que 3, caso contrário receberá o valor 7.

if ternário

- Pode-se escrever o mesmo código utilizando *if else*
- Exemplo:

```
if (a > 3) {  
    b = 2;  
} else {  
    b = 7;  
}
```

if ternário

- Um *if* ternário tem o comportamento de um *if*, embora sem a palavra *if*
- Exemplo:

```
int a = 5;  
int b = a > 3 ? 2 : 7;
```

A variável b receberá o valor 2 se o valor de a for maior que 3, caso contrário receberá o valor 7.



Comando *switch case*

- Executa um bloco de código se o valor de uma variável é EXATAMENTE igual a um determinado valor

```
switch (variável) {  
    case 1 : comandos  
        break;  
    case 2 : comandos  
        break;  
    default : comandos  
}
```

Comando *switch case*

- Executa um bloco de código se o valor de uma variável é EXATAMENTE igual a um determinado valor

```
switch (variável) {  
    case 1 : comandos  
        break;  
    case 2 : comandos  
        break;  
    default : comandos  
}
```

A **variável** pode ser do tipo inteiro (int), caracter(char), Setring ou enumerações

Comando *switch case*

- Executa um bloco de código se o valor de uma variável é EXATAMENTE igual a um determinado valor

```
switch (variável) {
```

```
    case 1 : comandos  
        break;
```

```
    case 2 : comandos  
        break;
```

```
    default : comandos
```

```
}
```

A **variável** pode ser do tipo inteiro (int), caracter(char), Setring ou enumerações

Os comandos são executados até encontrar uma instrução **break**

Comando *switch case*

- Executa um bloco de código se o valor de uma variável é EXATAMENTE igual a um determinado valor

```
switch (variável) {
```

```
    case 1 : comandos  
        break;
```

```
    case 2 : comandos  
        break;
```

```
    default : comandos
```

```
}
```

A **variável** pode ser do tipo inteiro (int), caracter(char), Setring ou enumerações

Os comandos são executados até encontrar uma instrução **break**

Os comandos depois da cláusula **default** são executados quando nenhum dos valores corresponde ao da variável



Comando *switch case*

- Exemplos

```
int dia = 4;  
switch (day) {  
    case 6:  
        System.out.println("Hoje é sexta-feira");  
        break;  
    case 7:  
        System.out.println("Hoje é sábado");  
        break;  
    default:  
        System.out.println("Só pelo findi!");  
}
```

Comando *switch case*

- Exemplos (Cuidado!)

```
int dia = 4;  
switch (day) {  
    case 6:  
        System.out.println("Hoje é sexta-feira");  
    case 7:  
        System.out.println("Hoje é sábado");  
        break;  
    default:  
        System.out.println("Só pelo findi!");  
}
```

Sem o **break** ao final do caso 6, executará os itens 6 e 7 quando o valor da variável for 6.



Estruturas de Controle

- Laços de Repetição
 - *for*
 - *while*
 - *do ... while*



Laço *for*

- Executa um bloco até uma condição ser atingida, permitindo controle total da inicialização e incremento

for (<inicialização>, <condição de parada>; <incremento>)
comandos

Laço *for*

- Executa um bloco até uma condição ser atingida, permitindo controle total da inicialização e incremento

```
for (<inicialização>, <condição de parada>; <incremento>) {  
    comandos  
}
```

Melhor usar os delimitadores {}
Sempre!

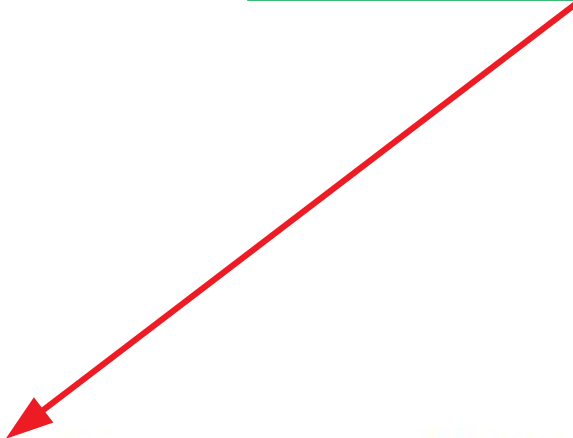
Laço *for*

- Exemplos

```
int a = 5;  
int i;
```

```
for(i = 0; i < 3; i++) {  
    a = a + 1;  
}
```

Repete o laço enquanto
a condição
for verdadeira



Laço *for*

- Exemplos

Variável **declarada** e
Inicializada aqui existe somente
Dentro do laço

```
int a = 5;
```

```
for(int i = 0; i < 3; i++) {  
    a = a + 1;  
}
```

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



Laço *for*

- Exemplos

Tudo o que estiver
entre os delimitadores é repetido

```
for(int i = 0; i < 3; i++) {  
    a = a + 1;  
    System.out.println(a);  
}
```

Laço *for*

- Exemplos (loop infinito)

```
for(;;) {  
    a = a + 1;  
}
```



Laço *while*

- Executa um bloco enquanto uma condição é verdadeira

while (<condição de parada>)
comandos



Laço *while*

- Executa um bloco enquanto uma condição é verdadeira

while (<condição de parada>)
comandos

while (<condição de parada>) {

comandos

}

Aqui o uso de delimitadores
também é recomendado

Laço *while*

- Exemplos:

```
int a = 5;
```

```
while(a < 10) {  
    a++;  
}
```

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

```
int a = 5;
```

```
while(a < 10) {  
    a++;  
    System.out.println(a);  
}
```



Laço *while*

- Exemplos:

```
int a = 5;

while(a < 10) {
    a++;
}

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

```
int a = 5;

while(a < 10) {
    a++;
    System.out.println(a);
}
```



Aqui o System.out.println()
executa somente uma vez



Laço *while*

- Exemplos:


```
int a = 5;

while(a < 10) {
    a++;
}

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

```
int a = 5;

while(a < 10) {
    a++;
    System.out.println(a);
}
```



Aqui o System.out.println()
executa 5 vezes

Laço do ... *while*

- Executa um bloco enquanto uma condição é verdadeira

do

comandos

while (<condição de parada>);

do {

comandos

} *while* (<condição de parada>);

Aqui o uso de delimitadores
também é recomendado

Laço do ... *while*

- Executa um bloco enquanto uma condição é verdadeira

do

comando

while (<condição de parada>);

do {

comandos

} while (<condição de parada>);

No **do ... while** os delimitadores
são obrigatórios



Laço do ... *while*

- Executa um bloco enquanto uma condição é verdadeira

do

comando

while (<condição de parada>;

do {

comandos

} *while* (<condição de parada>;

E o bloco termina com
ponto e vírgula

Laço do ... *while*

- Exemplo

```
int a = 5;
```

```
do {  
    a++;  
    System.out.println(a);  
} while(a < 10);
```

while vs do ... while

- **while** testa a condição **antes** de executar o bloco, e portanto **pode não executar nada**, dependendo da condição.
- **do ... while** testa a condição **depois** da execução do bloco, e por isso o bloco é executado sempre **pelo menos uma vez**



Dúvidas?





Atividades

- Execute as atividades presentes no documento

03.Lista.de.Exercícios.POO.pdf



Referências

- **Rafael Santos, Introdução A Programação Orientada A Objetos: USANDO JAVA**

Próximos passos



- Herança de Classes