

Aula 8



Objetivos

- Estrutura de repetição *do/while*;
- Instrução *break* e *continue*;
- Operações lógicas;

Aula 8



Estrutura de repetição *do/while*

Tem semelhança com a estrutura *while*

No começo de cada laço, é testado uma condição:

while (condição)

Na estrutura *do/while*, testa a condição de continuação de laço depois de executar o corpo do laço.

Do

instrução

while (condição);

Aula 8

//Uso da estrutura de repetição do/while

//Pacote Java core

```
import java.awt.Graphics;
```

//Pacote de extensão Java

```
import javax.swing.JApplet;
```

```
public class TesteDoWhile extends JApplet {
```

// Desenha linhas no plano de fundo do applet

```
public void paint( Graphics g )
```

```
{
```

// Chamada de versão hierarquica do método paint

```
super.paint( g );
```

```
int contator = 1;
```

```
do {
```

```
g.drawOval( 110 - contator * 10, 110 - contator * 10, contator * 20, contator * 20 );
```

```
++contator;
```

```
} while ( contator <= 10 ); // fim da estrutura do/while
```

```
} // fim do método paint
```

```
} // fim da classe TesteDoWhile
```



Aula 8



Instruções *break* e *continue*

Neste tipo de instrução *break* pode ser utilizado a alteração de fluxo de controle, isto é, quando executada em uma estrutura *while*, *for*, *do/while* ou *switch*, causa a saída imediata dessa estrutura. A execução continua com a primeira instrução depois da estrutura.

É comum utilizar a instrução *break* para escapar num início de um laço ou pular o restante de uma estrutura *switch*.

```

//Extensão do pacote Java
import javax.swing.JOptionPane;

public class TesteBreak {

// Início da execução do método mais da aplicação Java
public static void main( String args[] )
{
    String output = "";
    int contador;

// repete 10 vezes o loop
for ( contador = 1; contador <= 10; contador++ ) {

// se contador é 5, termina o loop
if ( contador == 5 )
    break; // break loop unicamente se o contador == 5

    output += contador + " ";

} // fim da estrutura for

output += "\nSaiu do loop com contador = " + contador;
JOptionPane.showMessageDialog( null, output );

System.exit( 0 ); // término do aplicativo

} // fim do método main

} // fim da classe TesteBreak

```



Aula 8



Instruções *break* e *continue*

Alguns programadores consideram que *break* e *continue* violam a programação estruturada. Como os efeitos dessas instruções podem ser obtidos através de técnicas de programação estruturada, esses programadores não utilizam *break* e *continue*.

As instruções *break* e *continue*, quando utilizadas adequadamente, têm desempenho mais rápido que as técnicas estruturadas correspondentes.

Aula 8



Instruções rotuladas *break* e *continue*

Neste tipo de instrução *break* pode interromper apenas uma estrutura *while*, *for*, *do/while* ou *switch* que envolve imediatamente. Isto possibilita interromper um conjunto aninhado de estruturas, você pode utilizar a instrução rotulada *break*. Esta instrução, quando executada em uma estrutura *while*, *for*, *do/while* ou *switch*, ocasiona a saída imediata dessa estrutura e de um número qualquer de estruturas de repetição.



```
// Uso da instrução break com rótulo
// Extensão do pacote Java
import javax.swing.JOptionPane;

public class TesteRotuloBreak {

    // Método main inicia a execução da aplicação Java
    public static void main( String args[] )
    {
        String saida = "";

        stop: { // rótulo do bloco

            // conta 10 linhas
            for ( int linha = 1; linha <= 10; linha++ ) {

                // conta 5 colunas
                for ( int coluna = 1; coluna <= 5 ; coluna++ ) {

                    // se a linha é 5, pule para o fim do bloco "stop"
                    if ( linha == 5 )
                        break stop; // pule para o fim do bloco "stop"

                    saida += " * ";

                } // fim da estrutura interna

                saida += "\n";

            } // fim da estrutura externa
        }
    }
}
```

Aula 8



```
// the following line is skipped
saida += "\nTérmino normal do loop";

} // fim do bloco rotulado

JOptionPane.showMessageDialog(
    null, saida, "Teste do break com rotulo",
    JOptionPane.INFORMATION_MESSAGE );

System.exit( 0 ); // término da aplicação

} // fim do método main

} // fim da classe TesteRotuloBreak
```

Aula 8



Operadores lógicos

Até agora foi visto em termos de condições simples:

`contador <= 10`

`total > 1000`

`numero != valorsentinela`

Essas condições foram expressas em termos dos operadores relacionais:

`>`

`<`

`>=`

`<=`

Aula 8



Operadores lógicos

E os operadores de igualdade `==` e `!=` .

Java fornece os seguintes operadores lógicos:

- **&&** (E lógico);
- **&** (E lógico booleano);
- **||** (OU lógico);
- **|** (OU lógico booleano inclusivo);
- **^** (OU lógico booleano exclusivo);
- **!** (NÃO lógico, também chamado de negação lógica).

Aula 8



Operadores lógicos

TABELA-VERDADE PARA OPERADOR **&&** (E lógico)

expressão01	expressão02	expressão01 && expressão02
false	false	false
false	true	false
true	false	false
true	true	true

Aula 8



Operadores lógicos

TABELA-VERDADE PARA OPERADOR || (OU lógico)

expressão01	expressão02	expressão01 expressão02
false	false	false
false	true	true
true	false	true
true	true	true

Aula 8



Operadores lógicos

TABELA-VERDADE PARA OPERADOR LÓGICO BOOLEANO OU(^)

expressão01	expressão02	expressão01 ^ expressão02
false	false	false
false	true	true
true	false	true
true	true	false

Aula 8



Operadores lógicos

TABELA-VERDADE PARA OPERADOR ! (NÃO lógico)

expressão	!expressão
false	true
true	false