

Material de Apoio Aula 7

Controle de fluxo de execução: Iteração – Continuação

O comando `while` permite controlar o número de repetições em um laço testando a condição de prosseguimento antes do início de cada iteração. A estrutura de um laço controlado por `while` pode ser representada da seguinte forma:

```
Inicializa  
while ( Testa ) {  
    Comandos  
    Atualiza  
}
```

Importante: a leitura de um comando `while` é: enquanto for verdadeiro, faça Comandos.

Comando do ... while

O comando do ... `while` também permite controlar a execução de um loop. No entanto, o teste de continuação da repetição é realizado no final de cada iteração através de uma expressão que retorna um valor booleano (`true` ou `false`). Com esta estrutura de controle, o valor `true` permite o retorno para a próxima iteração e o valor `false` interrompe o processo iterativo. Note que, como o teste é realizado após entrar na iteração, portanto, ao menos uma iteração será executada.

A sintaxe do comando `while` é a seguinte:

```
// Sintaxe 1: o loop é representado pela execução de uma única instrução.  
do  
    comando;  
while ( expressão );  
  
// Sintaxe 2: o loop é representado pela execução de um bloco de instruções.  
do {  
    comando;  
    comando;  
    ...  
} while ( expressão );
```

Ou seja,

```
Inicializa  
do {  
    Comandos  
    Atualiza  
} while ( Testa );
```

Importante: a leitura de um comando do ... `while` é: enquanto for verdadeiro, refaça Comandos.

Exemplos:

<ul style="list-style-type: none">• Implemente um método que imprima todos os números inteiros entre 0 e 10.	<ul style="list-style-type: none">• Implemente um método que imprima todos os números inteiros pares maiores que 0 e menores que 10. Assuma que 0 é um número par.	<ul style="list-style-type: none">• Implemente um método que imprima todos os números inteiros maiores que 0 e menores ou iguais a 10 em ordem invertida.
<pre>void imprimeInteiros() { int i; Saida s; s = new Saida(); i = 0; do { s.print(i); i++; } while (i < 10); }</pre>	<pre>void imprimeInteirosPares() { int i; Saida s; s = new Saida(); i = 0; do { s.print(i); i = i + 2; } while (i < 10); }</pre>	<pre>void imprimeInteirosPares() { int i; Saida s; s = new Saida(); i = 10; do { s.print(i); i--; } while (i > 0); }</pre>
<ul style="list-style-type: none">• Implemente um método na classe <code>Desenha</code> que desenhe um quadrado de lado 10 com "*" (asteriscos) na tela (este programa tem um erro de lógica, ver Exercício 1).		
<pre>class Desenha { private Saida s; public Desenha() { s = new Saida(); } public void quadradoLadoDez() { int i, j; i = 1; // Inicializa as variáveis de controle das ... j = 1; // ... iterações do { // Controla número de linhas j = 1; do { // Controla número de colunas s.print("*"); j++; // Imprimiu a coluna j na linha i } while(j <= 10); s.print("\n"); // A linha i foi impressa i++; // Ir para a linha i+1 } while(i <= 10); } // método } // classe</pre>		

Exercícios

1. Compare os testes realizados nos exemplos com aqueles apresentados no exemplo do comando `while` apresentados no material de Apoio 6.
2. (Com `do ... while`) Reimplemente os exercícios de número 2 a 10 do material de Apoio 6 com o comando de controle de repetição `do ... while`.

Comando `for`

O comando `for` é o terceiro comando de controle de iteração disponível em Java. Sua estrutura é a seguinte:

<pre>// Sintaxe 1: o loop é representado pela execução de uma única instrução. for(Inicialização ; Teste ; Atualização) comando;</pre>
<pre>// Sintaxe 2: o loop é representado pela execução de um bloco de instruções. for(Inicialização ; Teste ; Atualização) { comando; }</pre>

De uma maneira geral, o comando `for` é bastante semelhante ao comando `while`, porém com uma sintaxe mais reduzida.

Exemplos:

<ul style="list-style-type: none">• Implemente um método que imprima todos os números inteiros entre 0 e 10.	<ul style="list-style-type: none">• Implemente um método que imprima todos os números inteiros pares maiores que 0 e menores que 10. Assuma que 0 é um número par.	<ul style="list-style-type: none">• Implemente um método que imprima todos os números inteiros maiores que 0 e menores ou iguais a 10 em ordem invertida.
<pre>void imprimeInteiros() { int i; Saida s; s = new Saida(); for(i = 0 ; i <= 10 ; i++) s.print(i); }</pre>	<pre>void imprimeInteirosPares() { int i; Saida s; s = new Saida(); for(i = 0 ; i < 10 ; i= i+2) s.print(i); }</pre>	<pre>void imprimeInteirosPares() { int i; Saida s; s = new Saida(); i = 10; for(i = 10 ; i >= 0 ; i--) s.print(i); }</pre>
<ul style="list-style-type: none">• Implemente um método na classe <code>Desenha</code> que desenhe um quadrado de lado 10 com "*" (asteriscos) na tela (este programa tem um erro de lógica, ver Exercício 1).		
<pre>class Desenha { private Saida s; public Desenha() { s = new Saida(); } public void quadradoLadoDez() { int i, j; for(i = 1 ; i <= 10 ; i++) { // Controla número de linhas for(j = 1 ; j <= 10 ; j++) // Controla número de colunas s.print("*"); s.print("\n"); // A linha i foi impressa } // for i } // método } // classe</pre>		

Exercícios

3. Compare os testes realizados nos exemplos com aqueles apresentados no exemplo do comando `while` apresentados no material de Apoio 6.
4. (com `for`) Imprima, na tela, a tabuada de 0 a 10.
5. (Com `for`) Reimplemente os exercícios de número 2 deste material de apoio utilizando o comando de controle de repetição `for`.