

K

Instruções break e continue rotuladas

K.1 Introdução

No Capítulo 5, discutimos as instruções `break` e `continue` do Java que permitem aos programadores alterar o fluxo de controle nas instruções de controle. O Java também fornece as instruções `break` e `continue` rotuladas para situações em que um programador precise alterar convenientemente o fluxo de controle nas instruções de controle aninhadas. Este apêndice demonstra as instruções `break` e `continue` rotuladas com exemplos que utilizam instruções `for` aninhadas.

K.2 A instrução break rotulada

A instrução `break` apresentada na Seção 5.7 permite a um programa sair das instruções `while`, `for`, `do...while` ou `switch` nas quais a instrução `break` aparece. Às vezes, essas instruções de controle são aninhadas em outras instruções de repetição. Um programa precisaria sair da instrução de controle aninhada inteira em uma única operação, em vez de esperar que complete a execução normalmente. Para sair dessas instruções de controle aninhadas, você pode utilizar a **instrução break rotulada**. Quando executada em uma instrução `while`, `for`, `do...while` ou `switch`, ela resulta em uma saída imediata dessa instrução de controle e de quaisquer instruções envolventes. A execução do programa é retomada com a primeira instrução depois da **instrução rotulada** envolvente. A instrução que segue o rótulo pode ser de repetição ou um bloco no qual uma instrução de repetição aparece. A Figura K.1 demonstra a instrução `break` rotulada em uma instrução `for` aninhada.

O bloco (linhas 7–26 na Figura K.1) inicia com um **rótulo** (um identificador seguido por dois-pontos) na linha 7; aqui utilizamos o rótulo `stop:`. O bloco é colocado entre chaves (linhas 8 e 26) e inclui o `for` aninhado (linhas 10–22 e a instrução de saída na linha 25). Quando a instrução `if` na linha 15 detecta que `row` é igual a 5, a instrução `break` na linha 16 executa. Essa instrução termina as duas instruções `for` nas linhas 13–19 e sua instrução `for` envolvente nas linhas 10–22. Em seguida, o programa prossegue imediatamente para a primeira instrução depois do bloco rotulado — nesse caso, o fim do método `main` é alcançado e o programa termina. O `for` externo executa completamente o seu corpo apenas quatro vezes. A instrução de saída na linha 25 nunca executa, porque ela está no corpo do bloco rotulado e o `for` externo nunca completa.

```
1 // Fig. K.1: 5.13 BreakLabelTest.java
2 // Instrução break rotulada saindo de uma instrução for aninhada.
3 public class BreakLabelTest
4 {
5     public static void main( String args[] )
6     {
7         stop: // bloco rotulado
8         {
9             // conta 10 linhas
10            for ( int row = 1; row <= 10; row++ )
11                {
12                    // conta 5 colunas
13                    for ( int column = 1; column <= 5 ; column++ )
14                        {
```

Figura K.1 A instrução `break` rotulada saindo de uma instrução `for` aninhada. (Parte 1 de 2.)

2 Apêndice K Instruções break e continue rotuladas

```
15         if ( row == 5 ) // se a linha for 5,
16             break stop; // pula p/ final do bloco de parada
17
18         System.out.print( "*" );
19     } // fim do for interno
20
21     System.out.println(); // gera a saída de um caractere nova linha
22 } // fim do for externo
23
24 // a linha seguinte é pulada
25 System.out.println( "\nLoops terminated normally" );
26 } // fim do bloco rotulado
27 } // fim de main
28 } // fim da classe BreakLabelTest
```

```
* * * * *
* * * * *
* * * * *
* * * * *
```

Figura K.1 A instrução break rotulada saindo de uma instrução for aninhada. (Parte 2 de 2.)



Boa prática de programação K.1

Muitos níveis de instruções de controle aninhadas podem tornar um programa difícil de ler. Como uma regra geral, tente evitar a utilização de mais de três níveis de aninhamento.

K.3 A instrução continue rotulada

A instrução continue apresentada na Seção 5.7 prossegue para a próxima iteração (repetição) do while, for ou do...while imediatamente envolvente. A **instrução continue rotulada** pula as instruções restantes no corpo dessa instrução e quaisquer números de instruções de repetição envolventes e prossegue para a próxima iteração da **instrução de repetição rotulada** envolvente (um for, while ou do...while precedido por um rótulo). Nas instruções while e do...while rotuladas, o programa avalia o teste de continuação de loop do loop rotulado logo após a instrução continue executar. Em um for rotulado, a expressão de incremento é executada e o teste do loop de continuação é avaliado. A Figura K. 2 utiliza uma instrução continue rotulada em um for aninhado para permitir que a execução continue para a próxima iteração do for externo.

```
1 // Fig. K.2: ContinueLabelTest.java
2 // Instrução continue rotulada terminando uma instrução for aninhada.
3 public class ContinueLabelTest
4 {
5     public static void main( String args[] )
6     {
7         nextRow: // rótulo-alvo da instrução continue
8
9         // conta 5 linhas
10        for ( int row = 1; row <= 5; row++ )
11        {
12            System.out.println(); // gera a saída de um caractere nova linha
13
14            // conta 10 colunas por linha
15            for ( int column = 1; column <= 10; column++ )
16            {
17                // se a coluna for maior que a linha, inicia a próxima linha
18                if ( column > row )
19                    continue nextRow; // próxima iteração do loop rotulado
20            }
```

Figura K.2 Instrução continue rotulada terminando uma instrução for aninhada. (Parte 1 de 2.)

```
21         System.out.print( "*" );
22     } // fim do for interno
23 } // fim do for externo
24
25     System.out.println(); // gera a saída de um caractere nova linha
26 } // fim de main
27 } // fim da classe ContinueLabelTest
```

```
*
* *
* * *
* * * *
* * * * *
```

Figura K.2 Instrução continue rotulada terminando uma instrução for aninhada. (Parte 2 de 2.)

O for rotulado (linhas 7–23) na verdade inicia no rótulo `nextRow`. Quando a instrução `if` na linha 18 no for interno (linhas 15–22) detecta que `column` é maior do que `row`, a instrução `continue` na linha 19 executa, e o controle do programa continua com o incremento da variável de controle `row` do loop for externo. Mesmo que o for interno conte de 1 a 10, o número de caracteres de saída `*` em uma linha nunca excede ao valor de `row`, criando um padrão triangular interessante.