02

Use break e continue

Em qualquer estrutura de laço podemos aplicar os controladores **break** e **continue**. O `break` serve para parar o laço totalmente. Já o `continue` interrompe apenas a iteração atual. Vamos ver alguns exemplos:

```
int i = 1;
while (i < 10) {
    i++;
    if (i == 5)
        break; // sai do while com i valendo 5
    System.out.println(i);
}
System.out.println("Fim");
```

COPIAR CÓDIGO

Ao executar o `break`, a execução do `while` para completamente. Temos a seguinte saída:

```
2
3
4
Fim
```

COPIAR CÓDIGO

Vamos comparar com o `continue`:

```
int i = 1;
while (i < 10) {
    i++;
    if (i == 5)
        continue; // vai para a condição com o i valendo 5
    System.out.println(i);
}
```

COPIAR CÓDIGO

Neste caso, iremos parar a execução da iteração apenas quando o valor da variável for igual a 5. Ao encontrar um `continue`, o código volta ao início da iteração, ao ponto do loop. Nossa saída agora é a seguinte:

```
2
3
4
6
7
8
9
10
Fim
```

COPIAR CÓDIGO

Isto é, o `break` quebra o laço atual, enquanto o `continue` vai para a próxima iteração do laço.

Tome cuidado, pois um laço que tenha um `while` infinito do tipo `true` e que contenha um ::break:: é compilável, já que o compilador não sabe se o código poderá parar, possivelmente sim:

```
while(true) {
    if(1==2) break;
    System.out.println("em loop infinito compilável");
}
```

COPIAR CÓDIGO

Os controladores de laços, `break` e `continue` podem ser aplicados no `for` . O `break` se comporta da mesma maneira que no `while` e no `do .. while` , parar o laço por completo. Já o `continue` faz com que a iteração atual seja abortada, executando em seguida a parte de ::atualização:: do `for` , e em seguida a de ::condição::. Vamos ver o exemplo a seguir:

```
for (int i = 1; i < 10; i++) {
    if (i == 8) {
        break; // sai do for sem executar mais nada do laço.
    }
    if (i == 5) {
        // pula para a atualização sem executar o resto do
corpo.
        continue;
    }
    System.out.println(i);
}
```

COPIAR CÓDIGO

A saída desse código é :

```
1
2
```

```
3
4
6
7
```

COPIAR CÓDIGO

Rótulos em laços (labeled loops)

Às vezes, encontramos a necessidade de "encaixar" um laço dentro de outro. Por exemplo, um `for` dentro de um `while` ou de outro `for`. Nesses casos, pode ser preciso manipular melhor a execução dos laços encaixados com os controladores de laços, `break` e `continue`.

```
for (int i = 1; i < 10; i++) { //laço externo
    for (int j = 1; j < 10; j++) { // laço interno
        if (i * j == 25)    {
            break; // qual for será quebrado?
        }
    }
}
```

COPIAR CÓDIGO

Quando utilizamos o `break` ou o `continue` em laços encaixados, eles são aplicados no laço mais próximo. Por exemplo, nesse código, o `break` irá "quebrar" o `for` mais interno. Se fosse preciso "quebrar" o `for` mais externo, como faríamos?

Labeled statements

Podemos adicionar ::labels:: (rótulos) a algumas estruturas de código, e usá-los posteriormente para referenciarmos essas estruturas. Para declarar um label usamos um nome qualquer (mesma regra de nomes de variáveis etc.) seguido de dois pontos (`:`). Por exemplo, podemos dar um label para um `for` como o que segue:

```
externo: //label
for(int i=0; i<10;i++){
    //código
}
```

COPIAR CÓDIGO

Podemos usar esses ::labels:: para referenciar para qual loop queremos que o `break` ou o `continue` seja executado:

```
externo: for (int i = 1; i < 10; i++) {
  interno: for (int j = 1; j < 10; j++) {
    if (i * j == 25)    {
        break externo; // quebrando o for externo
    }
    if (i * j == 16) {
        continue interno; // pulando um iteração do for
interno
    }
  }
}
```

COPIAR CÓDIGO

Label http

O código a seguir imprime os valores de 1 a 10. Mas como ele compila sendo que temos uma URI logo antes do laço `for` ?

```
http://www.caelum.com.br
    for (int i = 1; i <= 10; i++) {
        System.out.println(i);
    }
}
```

COPIAR CÓDIGO

Um rótulo ou label pode estar presente antes de um ::statement:: qualquer, mas só podemos utilizar um ::statement:: de `break` ou `continue` caso o rótulo esteja referenciando um `for` , `while` ou `switch` :

```
void rotuloEmQualquerLugar() {
    rotulo: System.out.println("oi");
}

void rotuloEmQualquerLugarComBreakNaoCompila() {
    rotulo: System.out.println("oi");
    if(1<10) continue rotulo; // erro de compilação
}
```

COPIAR CÓDIGO

Cuidado, mesmo dentro de um `for` ou similar, o `continue` e o `break` só funcionarão se forem relativos a um label dentro do qual estão, e do tipo `for` , `do...while` , `switch` ou `while` . Vale lembrar que `switch` só aceita `break` .

```
void rotuloEmQualquerLugarComBreakNaoCompila() {
    rotulo: System.out.println("oi");
    for(int i=0;i<10;i++) {
        break rotulo; // não compila
    }
}
void rotuloEmOutroLaco() {
    rotulo:
    for(int i=0;i<10;i++) {
        System.out.println("oi");
    }
    for(int i=0;i<10;i++) {
        break rotulo; // não compila
    }
}
```

COPIAR CÓDIGO

Rótulos podem ser repetidos desde que não exista conflito de escopo:

```
void rotulosRepetidos() {
    rotulo: for (int i = 0; i < 10; i++) {
        break rotulo;
    }
    rotulo: for (int i = 0; i < 10; i++) {
        break rotulo;
    }
}
void rotulosRepetidosNestedNaoCompila() {
    rotulo: for (int i = 0; i < 10; i++) {
        rotulo: for (int j = 0; j < 10; j++) {
            break rotulo;
        }
```

```
        }
    }
```

COPIAR CÓDIGO

Não há conflito de nome entre rótulos e variáveis, pois seu uso é bem distinto. O compilador sabe se você está referenciando um rótulo ou uma variável:

```
class A {
    int rotulo = 15;
    void rotulosENomesDeVariaveisNaoConflitam() {
        rotulo: for (int i = 0; i < 10; i++) {
            int rotulo = 10;
            break rotulo;
        }
    }
}
```

COPIAR CÓDIGO

Um mesmo statement pode ter dois labels:

```
    void rotulosNoMesmoStatement() {
        primeiro: segundo: for (int i = 0; i < 10; i++) {
            System.out.println(i);
        }
    }
```

COPIAR CÓDIGO

Tome bastante cuidado com `breaks` e `continues` que são de `switch` mas parecem ser de `fors` :

```
class TestaLacos {
    public static void main(String[] args) {

        for(int i = 0; i < 4; i++) {
            System.out.println("Estou antes do switch");
            mario:
            guilherme: switch(i) {
                case 0:
                case 1:
                    System.out.println("Caso " + i);
                    for(int j = 0; j < 3; j++) {
                        System.out.println(j);
                        if(j==1) break mario;
                    }
                case 2:
                    System.out.println("Estou em i = " + i);
                    continue;
                case 3:
                    System.out.println("Cheguei no 3");
                    break;
                default:
                    System.out.println("Estranho...");
                    break;
            }
            System.out.println("Estou apos o switch");
        }
    }
}
```

COPIAR CÓDIGO