10

Mais laços com break

Transcrição

Feitas as tabuadas do vídeo anterior, vamos testar mais laços encadeados e ver como eles podem se comunicar? Criaremos para isto a classe `TestaLacos2`, para a qual copiaremos e colaremos o conteúdo de `TestaLacos`. Desta vez, substituiremos `multiplicador` por `linha`, enquanto `contador` passará a ser `coluna`. E não faremos mais multiplicações, e sim com que apareçam `10` linhas e `10` colunas. A partir do código abaixo, o que vocês acham que acontecerá?

```
public class TestaLacos2 {
    public static void main(String[] args) {
        for(int linha = 0; linha < 10; linha++) {
            for(int coluna = 0; coluna < 10; coluna++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Na aba "Console", será mostrado algo não muito interessante:

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

Uma grande quantidade de laços encadeados acaba não sendo esteticamente aprazível e, às vezes, queremos que um laço se comunique com outro. Para que os asteriscos formem uma matriz triangular, por exemplo, acrescentaríamos ao código um `if` para quando `coluna` for maior que `linha`, fazendo com que o laço pare de ser executado e saia dali para ir à próxima linha do `for`, externo.

Bem como em outras linguagens, existe um comando no Java, a palavra chave `break`, que "corta" a execução do laço mais interno, isto é, mais próximo de onde ela mesma se encontra, resultando exatamente no efeito que buscamos:

```
public class TestaLacos2 {
    public static void main(String[] args) {
        for(int linha = 0; linha < 10; linha++) {
            for(int coluna = 0; coluna < 10; coluna++) {
                if(coluna > linha) {
                    break;
                }
```

```
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Ao salvarmos e rodarmos o código, teremos:

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

No exemplo acima, poderíamos obter o mesmo efeito usando a condicional `if` sem as chaves, pois o `break` ocupa apenas uma linha, como seria possível também com `for` e `while`. No entanto, por boa prática, e visando à legibilidade e convenção, optaremos por usar as chaves sempre que possível.

E no segundo `for`, poderíamos ter substituído `coluna < 10` por `coluna <= linha`, modificando-se a instrução para não usarmos o `break`. Assim, o código completo ficaria da seguinte maneira:

```
public class TestaLacos2 {
    public static void main(String[] args) {
        for(int linha = 0; linha < 10; linha++) {
            for(int coluna = 0; coluna <= linha; coluna++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Há muitos exercícios a serem feitos e, mesmo que isso seja trivial para você, que já conhece outra linguagem de programação, ou esteja revendo comandos mais básicos, eles são interessantes para fixar erros de compilação. Senão, quando o conteúdo ficar mais complexo, as chances de se debater por aquilo que já deveria estar bem sedimentado serão maiores.

Portanto, não menospreze a sintaxe básica do Java! Se tiver dúvidas, use nosso fórum, com participação de instrutores e alunos, veja as dúvidas, busque se aprofundar cada vez mais.

Pratique bastante, pois no próximo curso encararemos os desafios de migrarmos da melhor forma de uma programação procedural, imperativa, para a tal da Orientação a Objetos. Muito obrigado!