

Laço com while

Transcrição

Finalmente chegamos nos laços, a última estrutura de controle de fluxo, básica e primordial em todas as linguagens! Criaremos uma classe específica para aprendermos sobre a estrutura de laço de repetição, o `TestaWhile`.

O `while` é uma palavra chave e, dentro dos parênteses, obrigatoriamente recebe uma expressão booleana, assim como o `if`. Por isso, precisaremos incluir algo lá dentro, que nos devolva `true` ou `false`.

```
public class TestaWhile {  
  
    public static void main(String[] args) {  
        int contador = 0;  
        while(contador <= 10) {  
            System.out.println(contador);  
            contador = contador + 1;  
        }  
    }  
}
```

Ao salvarmos e rodarmos o código, serão impressos os números de 0 a 10, como gostaríamos!

O `while` é uma instrução muito simples - lembrando que é preciso sempre inicializar e declarar a variável a ser utilizada, neste caso, em `contador`. Para reforçarmos algo que já foi visto, o escopo, poderemos imprimir `contador` novamente após o `while`:

```
public class TestaWhile {  
  
    public static void main(String[] args) {  
        int contador = 0;  
        while(contador <= 10) {  
            System.out.println(contador);  
            contador = contador + 1;  
        }  
        System.out.println(contador);  
    }  
}
```

A partir do qual se obtém a impressão de 0 a 11!

É claro que cabem outras condições booleanas no lugar de `contador <= 10`. Não é muito comum utilizarmos o formato `contador = contador + 1`; quando operamos sobre a própria variável, uma vez que existe uma forma mais sucinta, herdada do C:

```
contador += 1;
```

Não é que seja "igual a mais um"! Queremos somar 1 nele mesmo. É uma sintaxe estranha, mas indica exatamente o mesmo que `contador = contador + 1;`. Para o mesmo efeito, existe ainda o `++`:

```
contador++;
```

Esta, na verdade, é a forma mais comum de se somar o valor de si mesmo mais uma vez, e usar `++contador;` (o pré-incremento) também traria o mesmo resultado. Há casos em que existem diferenças, mas por ora não nos preocuparemos com isso.

É muito mais importante entendermos o escopo, que a variável precisa ser inicializada antes de se fazer qualquer ação com ela, pois isso não acontece automaticamente em condições temporárias, e que o `while` é o sistema de laço mais simples de todos.