02

Crie e use laços do tipo while

Outra maneira de controlar o fluxo de execução de um programa é definir que um determinado trecho de código deve executar várias vezes, como uma repetição ou um laço.

Uma linguagem como o Java oferece alguns tipos de laços para o programador escolher. O comando `while` é um deles.

```
int i = 1;
while (i < 10) {
    System.out.println(i);
    i++;
}
```

COPIAR CÓDIGO

A sintaxe do `while` é a seguinte:

```
while (CONDICAO) {
    // CODIGO
}
```

COPIAR CÓDIGO

Assim como no `if`, a condição de um bloco `while` deve ser um booleano. Da mesma maneira, se o bloco de código tiver apenas uma linha, podemos omitir as chaves:

```
int i = 0;
while( i < 10)
    System.out.println(i++);
```

COPIAR CÓDIGO

O corpo do `while` é executado repetidamente até que a condição se torne falsa. Em outras palavras, enquanto a condição for verdadeira.

É necessário tomar cuidado para não escrever um `while` infinito, ou seja, um laço que não terminaria se fosse executado.

```
int i = 1;
//Quando fica false?
while(i < 10){
    System.out.println(i);
}
```

COPIAR CÓDIGO

Em casos em que é explícito que o loop será infinito, o compilador é esperto e não deixa compilar caso tenha algum código após o laço:

```
class A {
    int a() {
        while(true) { //nunca fica false
            System.out.println("Faz algo");
        }
        return 1; // não compila, nunca chegará aqui
    }
}
```

COPIAR CÓDIGO

Mesmo que a condição use uma variável, pode ocorrer um erro de compilação, caso a variável seja final:

```
class A {
    int a() {
        final boolean RODANDO = true;
        while(RODANDO) {
            System.out.println("Faz algo");
        }
        return 1; // não compila, nunca chegará aqui
    }
}
```

COPIAR CÓDIGO

Agora, caso a variável não seja final, o compilador não tem como saber se o valor irá mudar ou não, por mais explícito que possa parecer, e o código compila normalmente:

```
class A {
    int a() {
        boolean rodando = true; // não final
        while(rodando) {
            System.out.println("Faz algo");
        }
        return 1;
        // compila, não tem como saber se o valor de rodando
        // vai mudar
```

```
    }
}
```

COPIAR CÓDIGO

Caso um laço nunca seja executado, também teremos um erro de compilação:

```
//unreachable statement, não compila.
while(false) { //código aqui }

//unreachable statement, não compila.
while(1 > 2) { //código aqui }
```

COPIAR CÓDIGO

Lembre-se que o compilador só consegue analisar operações com literais ou com constantes. No caso a seguir, o código compila, mesmo nunca sendo executado:

```
int a = 1;
int b = 2;
while(a > b){ //compila, mas nunca executa
    System.out.println("OI");
}
```

COPIAR CÓDIGO