02

Utilize o if e if/else

Imagine um programa que aceita comandos do usuário, ou seja, um sistema interativo. De acordo com os dados que o usuário passar, o programa se comporta de maneiras diferentes e, consequentemente, pode dar respostas diferentes.

O programador, ao escrever esse programa, deve ter recursos para definir o comportamento para cada possível comando do usuário, em outras palavras, para cada situação. Com isso, o programa será capaz de tomar decisões durante a execução com o intuito de mudar o fluxo de execução.

As linguagens de programação devem oferecer aos programadores maneiras para ::controlar o fluxo de execução dos programas::. Dessa forma, os programas podem tomar decisões que afetam a sequência de comandos que serão executados.

if / else

A maneira mais simples de controlar o fluxo de execução é definir que um determinado trecho de código deve ser executado quando uma condição for verdadeira.

Por exemplo, suponha um sistema de login. Ele deve verificar a autenticidade do usuário para permitir ou não o acesso. Isso pode ser implementado com um **if/else** do Java.

```
boolean autentico = true;
```

```
if (autentico) {
    System.out.println("Usuario aceito");
} else {
    System.out.println("Usuario incorreto");
}
```

COPIAR CÓDIGO

A sintaxe do `if` é a seguinte:

```
if (CONDICAO) {
    // CODIGO 1
} else {
    // CODIGO 2
}
```

COPIAR CÓDIGO

A condição de um `if` **sempre** tem que ser um valor booleano:

```
if(1 - 2) { } // erro, numero inteiro

if(1 < 2) {} //ok, resulta em true

boolean valor = true;
if (valor == false) {} // ok, mas resulta em false

if (valor) {} // ok, valor é boolean
```

COPIAR CÓDIGO

Atenção dobrada ao código a seguir:

```
int a = 0, b = 1;

if(a = b) {
    System.out.println("iguais");
}
```

COPIAR CÓDIGO

Esta é uma pegadinha bem comum. Repare que não estamos fazendo uma **comparação** aqui, e sim, uma **atribuição** (um único `=`). O resultado de uma atribuição é sempre o valor atribuído, no caso, um inteiro. Logo, este código não compila, pois passamos um inteiro para a condição do `if`.

A única situação em que um código assim poderia funcionar é caso a variável atribuída seja do tipo `boolean`, pois o resultado da atribuição será `boolean`:

```
boolean a = true;

if(a = false) {
    System.out.println("Falso!");
}
```

COPIAR CÓDIGO

Neste caso, o código compila, mas não imprime nada. Após a atribuição, o valor da variável `a` é `false`, e o `if` não é executado.

Caso só tenhamos um comando dentro do `if` ou `else`, as chaves são opcionais:

```
if(!resultado)
```

```
    System.out.println("Falso!");
else
    System.out.println("Verdadeiro!");
```

COPIAR CÓDIGO

Caso não tenhamos nada para ser executado em caso de condição `false`, não precisamos declarar o `else`:

```
boolean autentico = true;
if (autentico)
    System.out.println("Usuario aceito");
```

COPIAR CÓDIGO

Mas sempre temos que ter algum código dentro do `if`, se não o código não compila:

```
boolean autentico = true;
if (autentico)
else // erro
    System.out.println("Acesso negado");
```

COPIAR CÓDIGO

Na linguagem Java, **não** existe o comando **elseif**. Para conseguir o efeito do "elseif", os `if`s são colocados dentro dos `else`.

```
if (CONDICAO1) {
    // CODIGO 1
} else if (CONDICAO2) {
```

```
    // CODIGO 2
} else {
    // CODIGO 3
}
```

COPIAR CÓDIGO

Grande parte das perguntas sobre estruturas de `if/else` são pegadinhas, usando a indentação como forma de distração:

```
boolean autentico = true;
if (autentico)
    System.out.println("Usuario aceito");
else
    System.out.println("Usuario incorreto");
    System.out.println("Tente novamente");
```

COPIAR CÓDIGO

A mensagem `"Tente novamente"` sempre é impressa, independente do valor da variável `autentico` .

Esse foi um exemplo bem simples, vamos tentar algo mais complicado. Tente determinar o que é impresso:

```
int valor = 100;
if (valor > 200)
if (valor <400)
if (valor > 300)
    System.out.println("a");
else
    System.out.println("b");
```

```
    else
        System.out.println("c");
```

COPIAR CÓDIGO

E então? `"c"` ? Vamos reindentar o código para ver se fica mais fácil:

```
int valor = 100;
if (valor > 200)
    if (valor <400)
        if (valor > 300)
            System.out.println("a");
        else
            System.out.println("b");
    else
        System.out.println("c");
```

COPIAR CÓDIGO

É sempre complicado analisar código não indentado ou mal indentado, e esse recurso é usado extensivamente em várias questões durante a prova, fique esperto!

Unreachable Code e Missing return

Um código Java não compila se o compilador perceber que aquele código não será executado sob hipótese alguma:

```
class Teste {
    public int metodo() {
        return 5;
        System.out.println("Quando isso será executado?");
```

```
    }
}
```

COPIAR CÓDIGO

```
Teste.java:10: unreachable statement
        System.out.println("Quando isso será executado?");
                ^
```

COPIAR CÓDIGO

O código após o `return` não será nunca executado. Esse código não compila. Vamos ver alguns outros exemplos:

```
class Teste {
    public int metodo(int x) {
        if(x > 200) {
            return 5;
        }
    }
}
```

COPIAR CÓDIGO

Este também não compila. O que será retornado se `x` for `<= 200` ?

```
Teste.java:12: missing return statement
    }
        ^
1 error
```

COPIAR CÓDIGO

Vamos modificar o código para que ele compile:

```
class Teste {
    public int metodo(int x) {
        if(x > 200) {
            return 5;
        }
        throw new RuntimeException();
    }
}
```

COPIAR CÓDIGO

Apesar de não estarmos retornando nada caso o `if` seja falso, o Java percebe que nesse caso uma exceção será disparada. A regra é: todos os caminhos possíveis devem retornar o tipo indicado pelo método, ou lançar exceção.

Em um `if` , essa expressão compila normalmente:

```
    if(false) {.... } //compila, apesar de ser unreachable
code
```

COPIAR CÓDIGO

São pequenos detalhes, tome cuidado para não cair nessas pegadinhas.