02

Operadores de comparação

Comparadores

A comparação entre os valores de duas variáveis é feita através dos operadores de comparação. O mais comum é comparar a igualdade e a desigualdade dos valores. Existem operadores para essas duas formas de comparação.

- `==` - igual
- `!=` - diferente

Além disso, os valores numéricos ainda podem ser comparados em relação à ordem.

- `>` - maior
- `<` - menor
- `>=` - maior ou igual
- `<=` - menor ou igual

Uma comparação pode devolver dois valores possíveis: verdadeiro ou falso. No Java, uma comparação sempre devolve um valor `boolean`.

```
System.out.println(1 == 1);         // true.
System.out.println(1 != 1);         // false.
System.out.println(2 < 1);         // false.
System.out.println(2 > 1);         // true.
System.out.println(1 >= 1);         // true.
System.out.println(2 <= 1);         // false.
```

COPIAR CÓDIGO

Toda comparação envolvendo valores numéricos não considera o tipo do valor. Confira somente se eles têm o mesmo valor ou não, independente de seu tipo:

```
// true.
System.out.println(1 == 1.0);

// true.
System.out.println(1 == 1);

// true. 1.0 float é 1.0 double
System.out.println(1.0f == 1.0d);

// true. 1.0 float é 1 long
System.out.println(1.0f == 1l);
```

COPIAR CÓDIGO

Os valores não primitivos (referências) e os valores ::boolean:: devem ser comparados somente com dois comparadores, o de igualdade (`==`) e o de desigualdade (`!=`).

```
// não compila, tipo não primitivo só aceita != e ==
System.out.println("Mario" > "Guilherme");

// não compila, boolean só aceita != e ==
System.out.println(true < false);
```

COPIAR CÓDIGO

Não podemos comparar tipos incomparáveis, como um `boolean` com um valor numérico. Mas podemos comparar `chars` com numéricos.

```
// não compila, boolean é boolean
System.out.println(true == 1);

// compila, 'a' tem valor numérico também
System.out.println('a' > 1);
```

COPIAR CÓDIGO

Cuidado, é muito fácil comparar atribuição com comparação e uma pegadinha aqui pode passar despercebida, como no exemplo a seguir:

```
int a = 5;
System.out.println(a = 5); // não imprime true, imprime 5
```

COPIAR CÓDIGO

Precisão

Ao fazer comparações entre números de ponto flutuante, devemos tomar cuidado com possíveis problemas de precisão. Qualquer conta com estes números pode causar um estouro de precisão, fazendo com que ele fique ligeiramente diferente do esperado. Por exemplo, `1 == (100.0 / 100)` pode não ser verdadeiro caso a divisão tenha uma precisão não exata.