



## Operadores Básicos – Java

**Operadores aritméticos:** "+", "-", "/", "\*", "%" **Precedência:** "()" -> "\*", "/", "%" -> "+", "-"

**Operadores lógicos:** "||", "&&", "!=", "!"

**Operador de atribuição:** "="

**Operador de comparação:** "==", ">", "<"

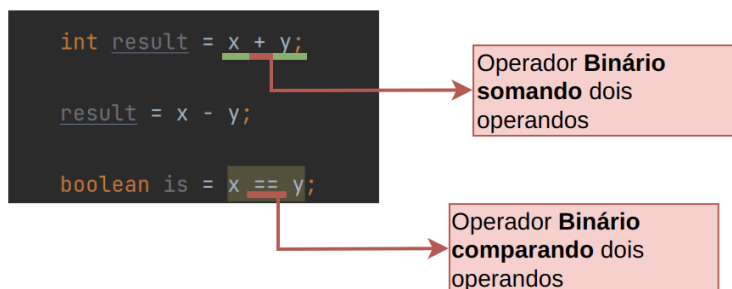


## Operador Unário, binário, Ternário – Java

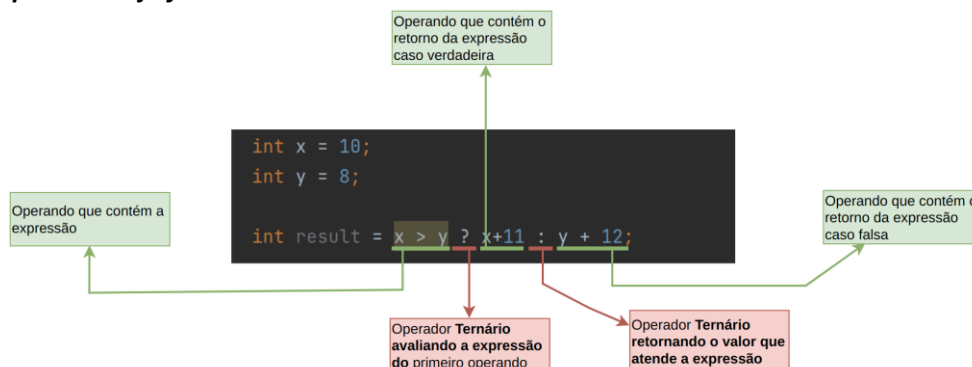
Operador **unário** é aquele que antecede ou precede um **único operando** em uma ação, tipo **incrementar/decrementar uma ÚNICA variável**:



Operador **binário** é o que age sobre **dois operandos**, **somar, subtrair, comparar** e etc:



E o operador **ternário** é o que age sobre **três operandos**, geralmente **onde o primeiro operando é uma expressão**, o **segundo é o valor caso a expressão seja verdadeira** e o **terceiro é o valor caso a expressão seja falsa**:

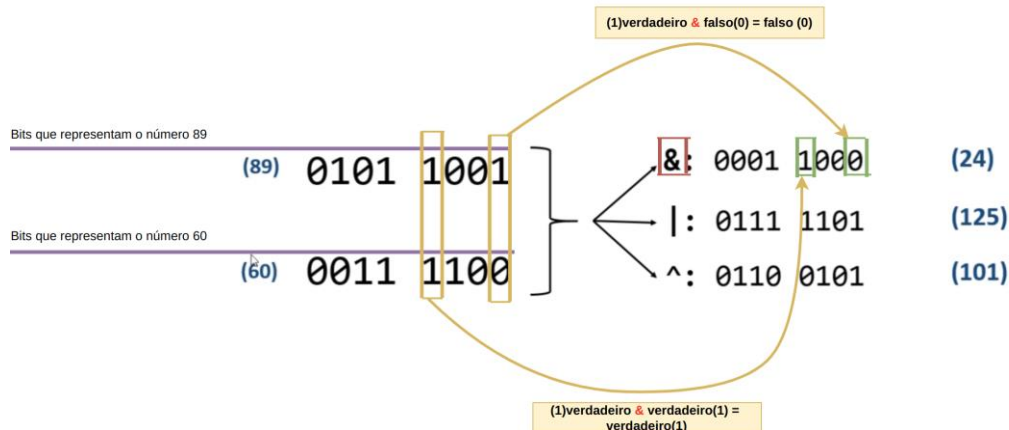




## Operador Bitwise – Java

São operadores que fazem operações lógicas “bit a bit” LITERALMENTE, em valores. São eles “& (e)”, “| (ou)”, “^ (ou exclusivo)”.

Um **número é representado por uma sequência de bits 0 e 1**, os **operados bitwise** então **comparam cada bit do número** e **forma uma outra sequência de bits resultantes dessa comparação**:



| = vai pegar **bit a bit** e fazer a operação de “ou”:

verdadeiro (1) | falso (0) = verdadeiro(1)

falso (0) | falso (0) = falso (0)

verdadeiro (1) | verdadeiro (1) = verdadeiro (1)

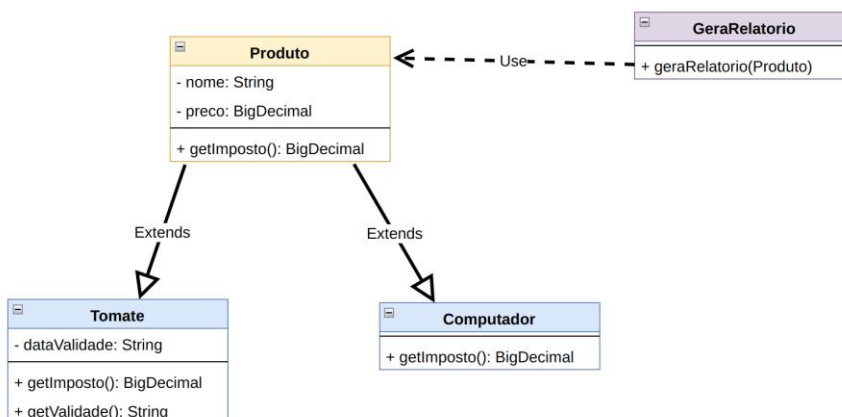
^ = vai pegar **bit a bit** e fazer a operação de “ou exclusivo”.

Esse tipo de operador é usado em programação de baixo nível, nunca usei eles na prática.



## Operador instanceof – Java

O operador **instanceof** serve para você saber se uma **variável** é de um **tipo específico de objeto**. Por exemplo, se eu tiver um esquema de herança onde eu tenha uma **classe abstrata que represente todos os tipos de produtos** e **classes específicas que são produtos**.



Como eu posso gerar um relatório que inclui uma **linha específica de validade** caso o produto passado seja de um tipo **perecível**? Posso usar o *instanceof*:



```
List<Produto> produtos = new ArrayList<>(Arrays.asList(
    new Tomate(nome: "Tomate", new BigDecimal(val: "12.8"), dataValidade: "12/12/2022"),
    new Computador(nome: "Dell", new BigDecimal(val: "6600.8"))
));

gerarRelatorio(produtos);
```

```
private static void gerarRelatorio(List<Produto> produtos) {
    produtos.forEach(produto -> {
        System.out.println("Nome produto: " + produto.getNome());
        System.out.println("Preço: " + produto.getPreco());
        System.out.println("Imposto: " + produto.getImposto());
        if(produto instanceof Tomate tomate){
            System.out.println("Validade: " + tomate.getDataValidade());
        }
        System.out.println("=====");
    });
}
```



```
Nome produto: Dell
Preço: 6600.0
Imposto: 990.000
=====
```



```
Nome produto: Tomate
Preço: 12.8
Imposto: 1.200
Validade: 12/12/2022
=====
```