CURSO DE PROGRAMAÇÃO EM JAVA

Aula 4

Operadores e Expressões

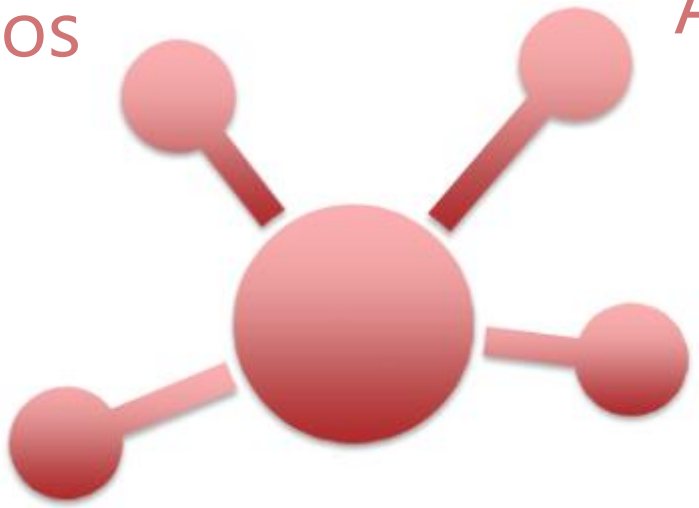
1.
Operadores

São símbolos que representam atribuições, cálculos e ordem dos dados.

Tipos de Operadores

Aritméticos

Atribuição



Lógicos

Relacionais

2.
Operadores Aritméticos

Usados para representar as operações matemáticas

Operadores Aritméticos

Operador	Ação
-	Subtração, também menos unário
+	Adição
*	Multiplicação
/	Divisão
%	Módulo da divisão (resto)
--	Decremento
++	Incremento

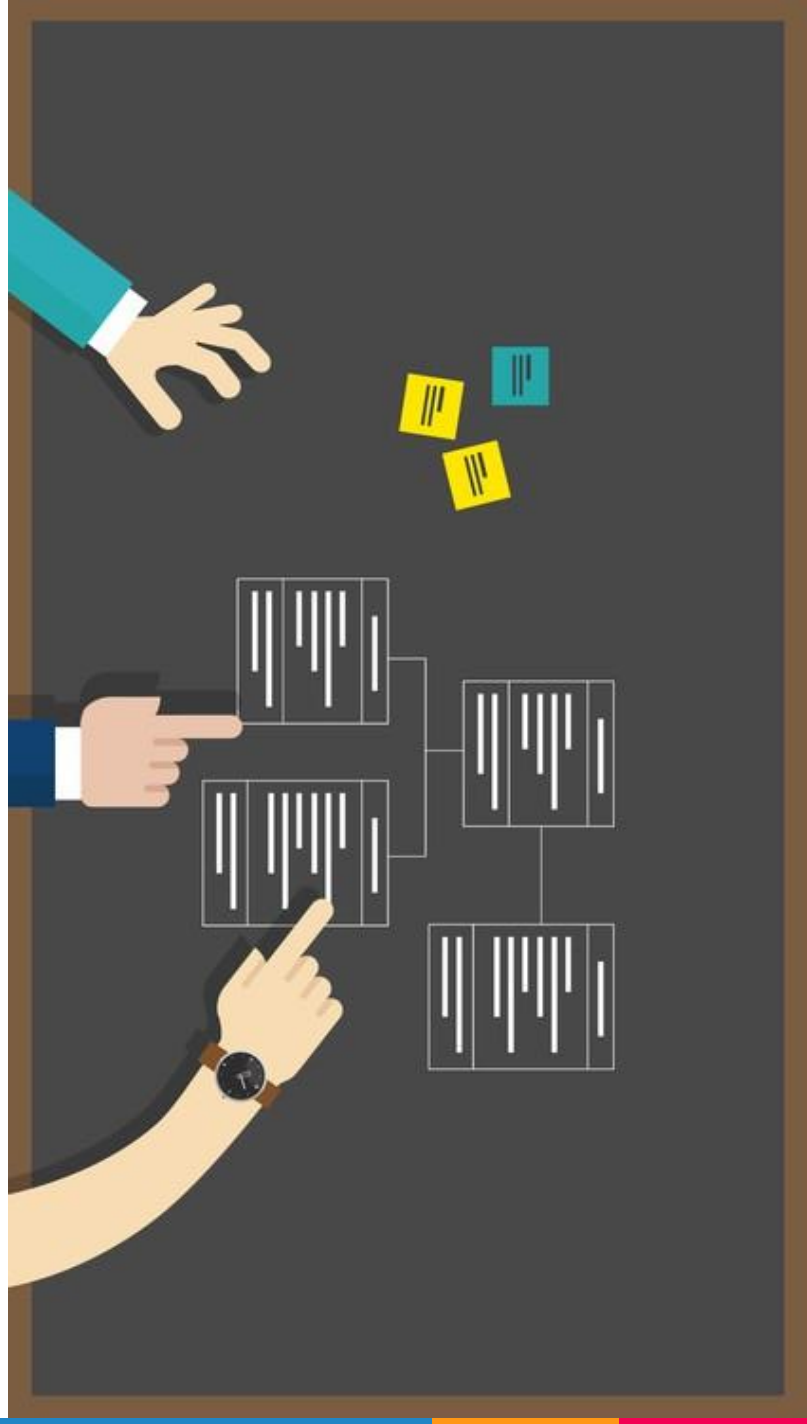
Ordem de prioridade


```
int a = 4;
int b = 3 + 2*a--;
```

Operadores Aritméticos

Os operadores aritméticos são operadores **binários***, ou seja, funcionam com dois operandos. Por exemplo, a expressão “a + 1” contém o operador binário “+” (mais) e os dois operandos “a” e “1”.

*Com exceção dos operadores de **incremento** e **decremento** que são operadores **unários**.



Exemplos

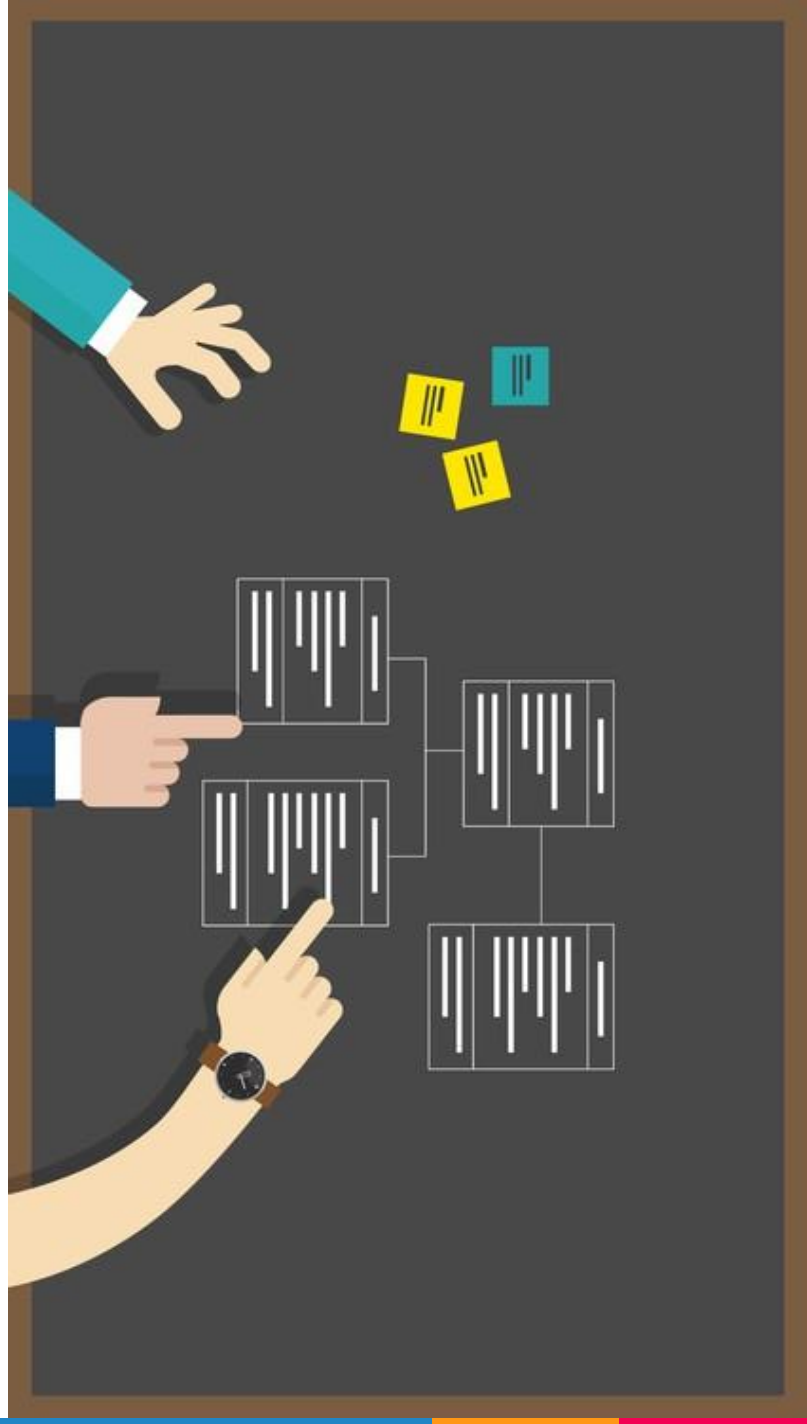
```
int numero = 5;                 //numero passa a valer 5
numero = numero + 8;            //numero passa a valer 13
numero = numero - numero;       //numero passa a valer zero
String x = "Alo";               // x é inicializado com a string "Alo"
String y = "Mundo!";            // y é inicializado com a string "Mundo!"
x = x + ", " + y;               // x passa a valer "Alo, Mundo!"
```

Incremento e Decremento

Tanto o operador de incremento quanto o de decremento podem preceder (pre-fixar) ou vir após (posfixar) o operando.

Por exemplo:

```
x++;      //forma prefixa
++x;      //forma posfixada
```



Incremento e Decremento

```
int x = 5;          // x contém 5
int y, z;           // y e z não foram definidos
y = x++;            // primeiro faz y igual ao valor (anterior) de x, e depois modifica x
z = ++x;            // primeiro modifica x,  e depois atribui a z o novo valor de x
```

3.
Operadores Relacionais

Se referem aos relacionamentos que os valores podem ter uns com os outros.

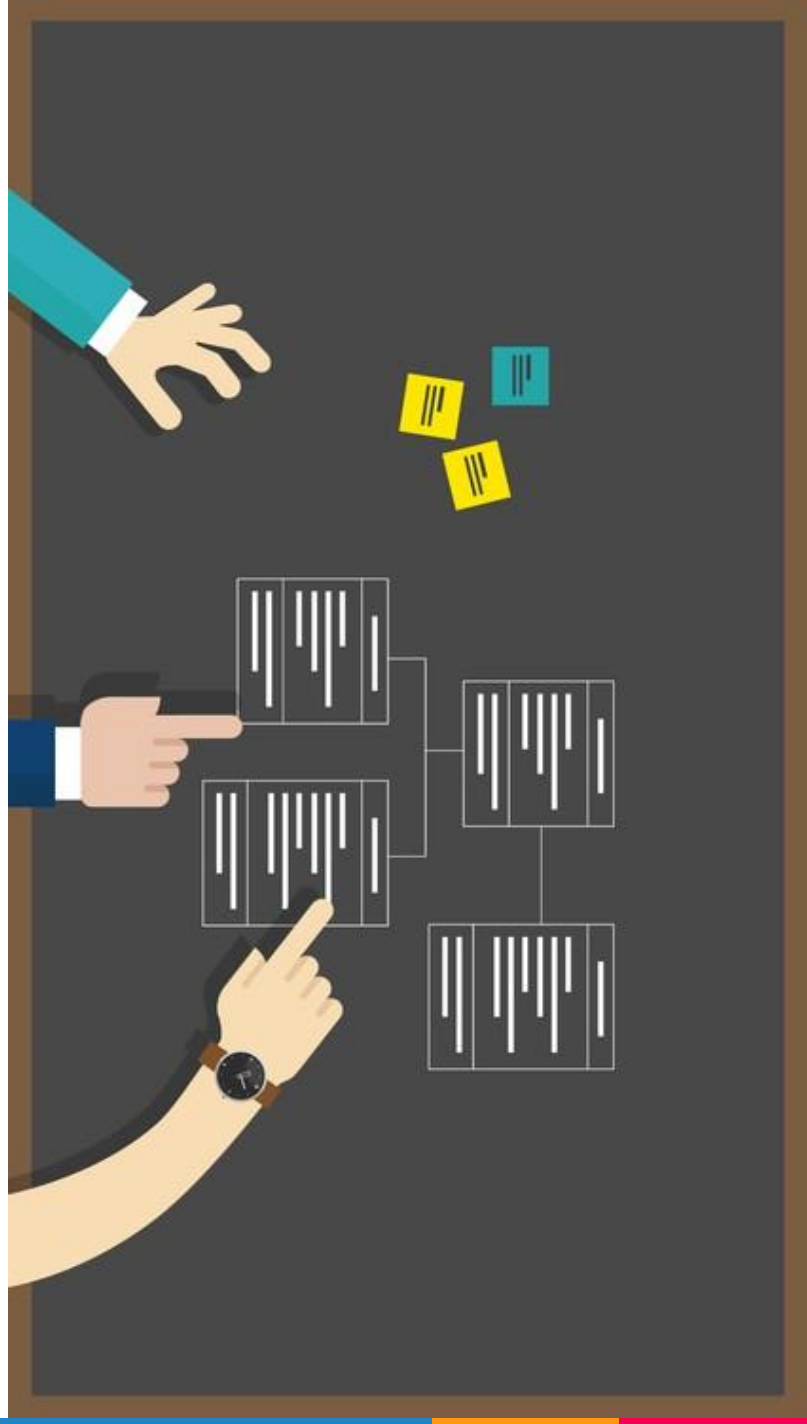
Operadores Relacionais

Operador	Ação
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
==	Igual a
!=	Diferente de

Operadores Relacionais

São utilizados para comparar variáveis ou expressões, resultando em um valor lógico (verdadeiro ou falso). Frequentemente trabalham com os operadores lógicos.

Em Java, podemos comparar todos os objetos para ver se são iguais ou diferentes com o uso de == e !=. No entanto os operadores de comparação >, <, <= ou >= só podem ser aplicadas aos tipos que dão suporte ao relacionamento sequencial, por exemplo aos tipos numéricos e ao tipo char.



Exemplos

```
boolean variavel;
 variavel=(4<4);   //variavel recebe "falso"
 variavel=(4<=4);  //variavel recebe "verdadeiro"
 variavel=(-1>-3); //variavel recebe "verdadeiro"
 variavel=(-4>=0); //variavel recebe "falso“
 variavel=(-5==5);    //variavel recebe "falso"
 variavel=(2!=45674); //variavel recebe "verdadeiro“
```

4.

Operadores Lógicos

Utilizados quando existe a necessidade de se comparar mais de uma condição

Operadores Lógicos

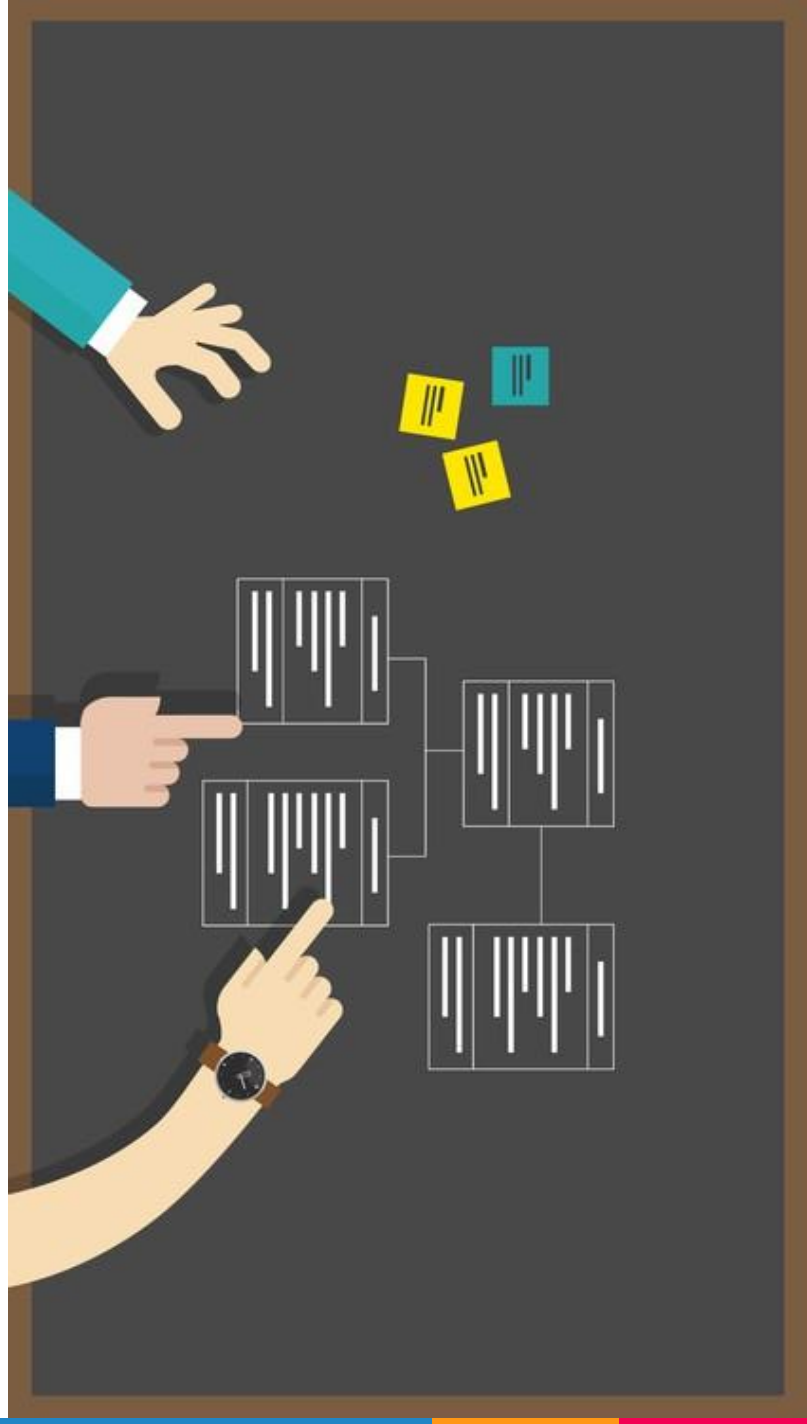
Operador	Significado
&	AND
|	OR
^	XOR (exclusive OR)
||	OR de curto-circuito
&&	AND de curto-circuito
!	NOT

Operadores Lógicos

p	q	p & q	p | q	p ^ q	!p
Falso	Falso	Falso	Falso	Falso	Verdadeiro
Verdadeiro	Falso	Falso	Verdadeiro	Verdadeiro	Falso
Falso	Verdadeiro	Falso	Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro	Falso	Falso

Operadores Lógicos

Quanto aos operadores lógicos, os operandos devem ser do tipo boolean e o resultado de uma operação lógica é do tipo boolean também.



Exemplos

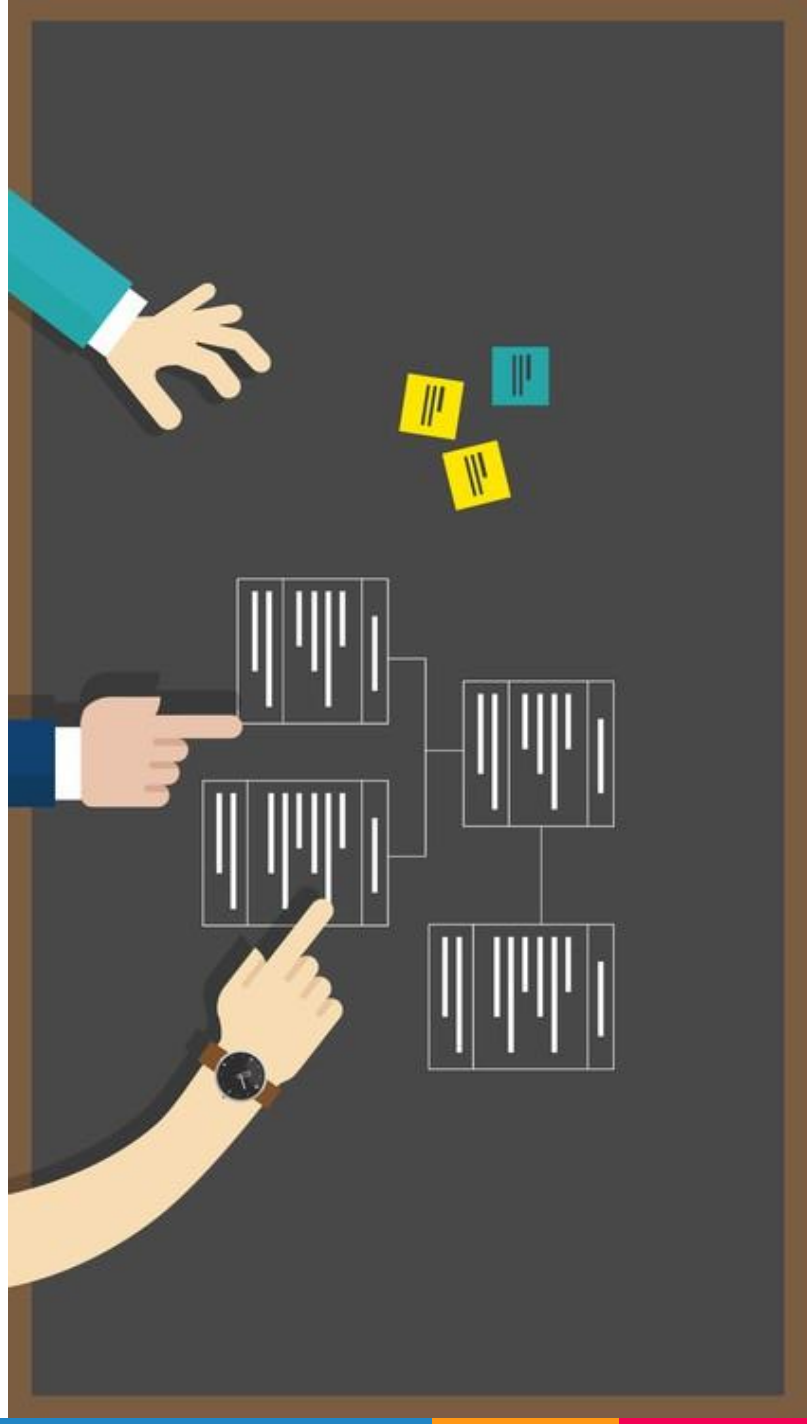
```
// Demonstra os operadores relacionais e lógicos.
class RelLogOps {
  public static void main(String args[]) {
    int i, j;
    boolean b1, b2;

    i = 10;
    j = 11;
    if(i < j) System.out.println("i < j");
    if(i <= j) System.out.println("i <= j");
    if(i != j) System.out.println("i != j");
    if(i == j) System.out.println("this won't execute");
    if(i >= j) System.out.println("this won't execute");
    if(i > j) System.out.println("this won't execute");

    b1 = true;
    b2 = false;
    if(b1 & b2) System.out.println("this won't execute");
    if(!(b1 & b2)) System.out.println("!(b1 & b2) is true");
    if(b1 | b2) System.out.println("b1 | b2 is true");
    if(b1 ^ b2) System.out.println("b1 ^ b2 is true");
  }
}
```

Operadores Lógicos de curto-circuito

Java oferece versões especiais de curto circuito de seus operadores lógicos AND e OR que podem ser usados para produzir código mais eficiente.
A única diferença entre as versões comum e de curto-circuito é que a versão comum sempre avalia cada operando e a versão de curto-circuito só avalia o segundo operando quando necessário.

AND – 1º Falso – Resultado Falso

OR – 1º Verdadeiro – Resultado Verdadeiro



Exemplos

```
// Demonstra os operadores de curto-circuito.
class SCops {
  public static void main(String args[]) {
    int n, d, q;

    n = 10;
    d = 2;
    if(d != 0 && (n % d) == 0)
      System.out.println(d + " is a factor of " + n);

    d = 0; // configura d com zero

    // Já que d é igual a zero, o segundo operando não é avaliado.
    if(d != 0 && (n % d) == 0)
      System.out.println(d + " is a factor of " + n);

    /* Tente a mesma coisa sem o operador de curto-circuito.
       Isso causará um erro de divisão por zero.
    */
    if(d != 0 & (n % d) == 0)
      System.out.println(d + " is a factor of " + n);
  }
}
```

O operador de curto-circuito impede uma divisão por zero.

Agora as duas expressões são avaliadas, permitindo que ocorra uma divisão por zero.

5.
Operador de atribuição
=

Operador de atribuição

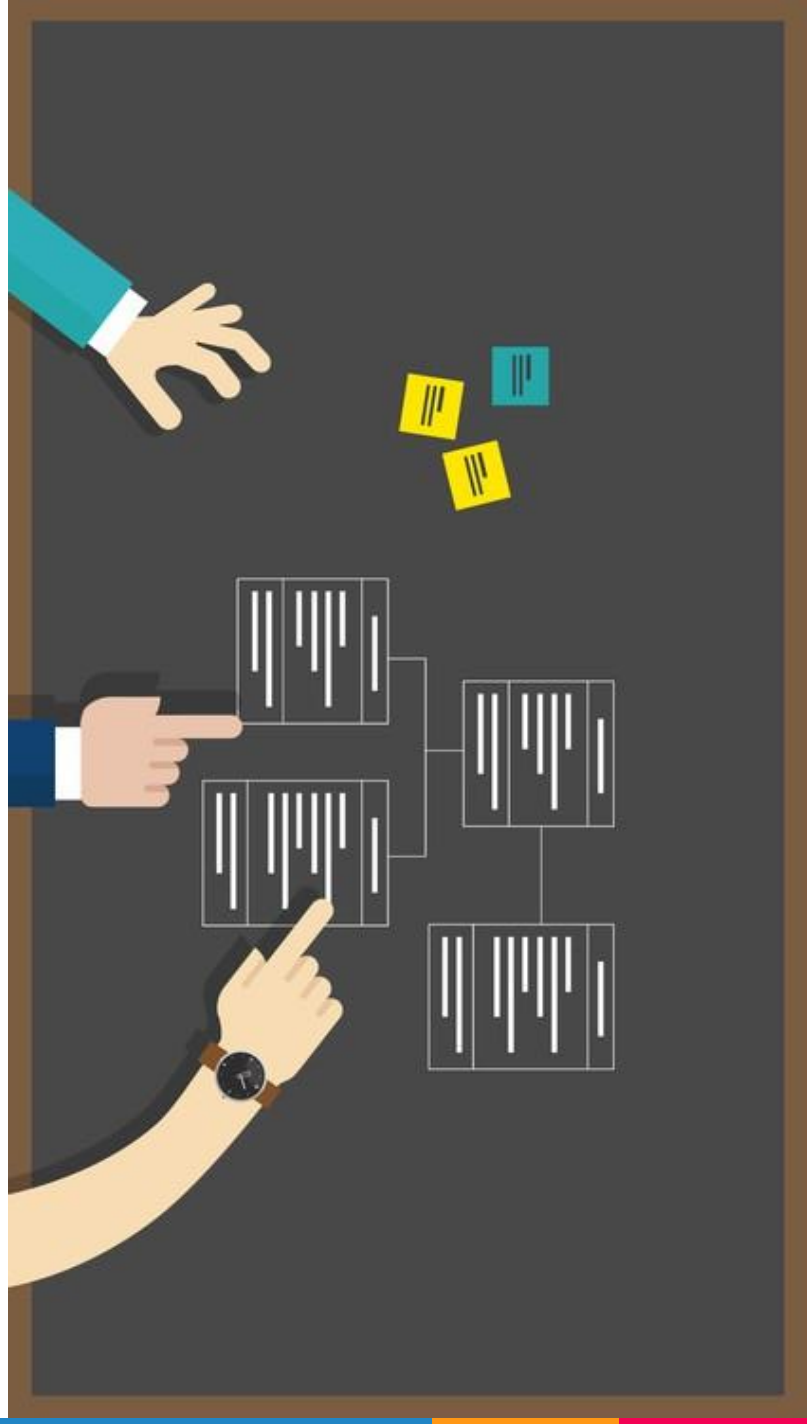
Esse operador funciona em Java de modo muito parecido com que funciona em qualquer outra linguagem de computador.

Var = expressão;

No entanto, o operador de atribuição tem uma propriedade interessante: ele permite a criação de uma cadeia de atribuições. Por exemplo:

```
Int x,y,z;
x = y = z = 100;
```

Isso funciona porque = é um operador que fornece o valor da expressão do lado direito.



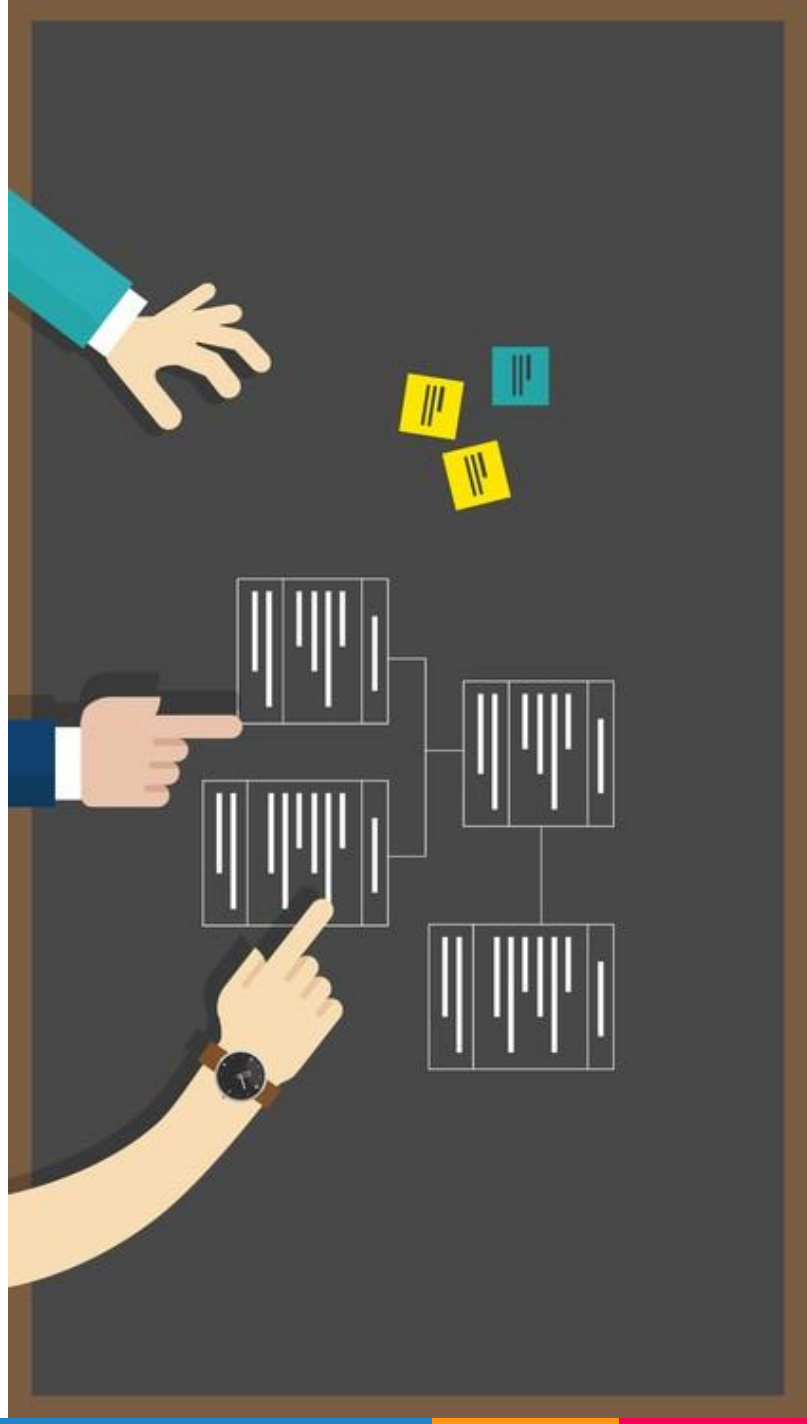
Ternários

O operador ternário serve como uma abreviação de uma expressão do tipo lógica. Utilizando o operador "?", é possível avaliar uma condição lógica e de acordo com o resultado atribuir um valor específico a uma variável.

```
int a = 1 < 3 ? 10 : 11;
```

Essa expressão é equivalente a:

```
int a;
if(1 < 3)
   a = 10;
 else
   a = 11;
```



6.
Atribuições Abreviadas

Java oferece simplificações de certas instruções de atribuição

Atribuições Abreviadas

int a=10;

a = a + b, fazemos: a +=b

a = a - b, fazemos: a -=b

a = a * b, fazemos: a *=b

a = a / b, fazemos: a /=b

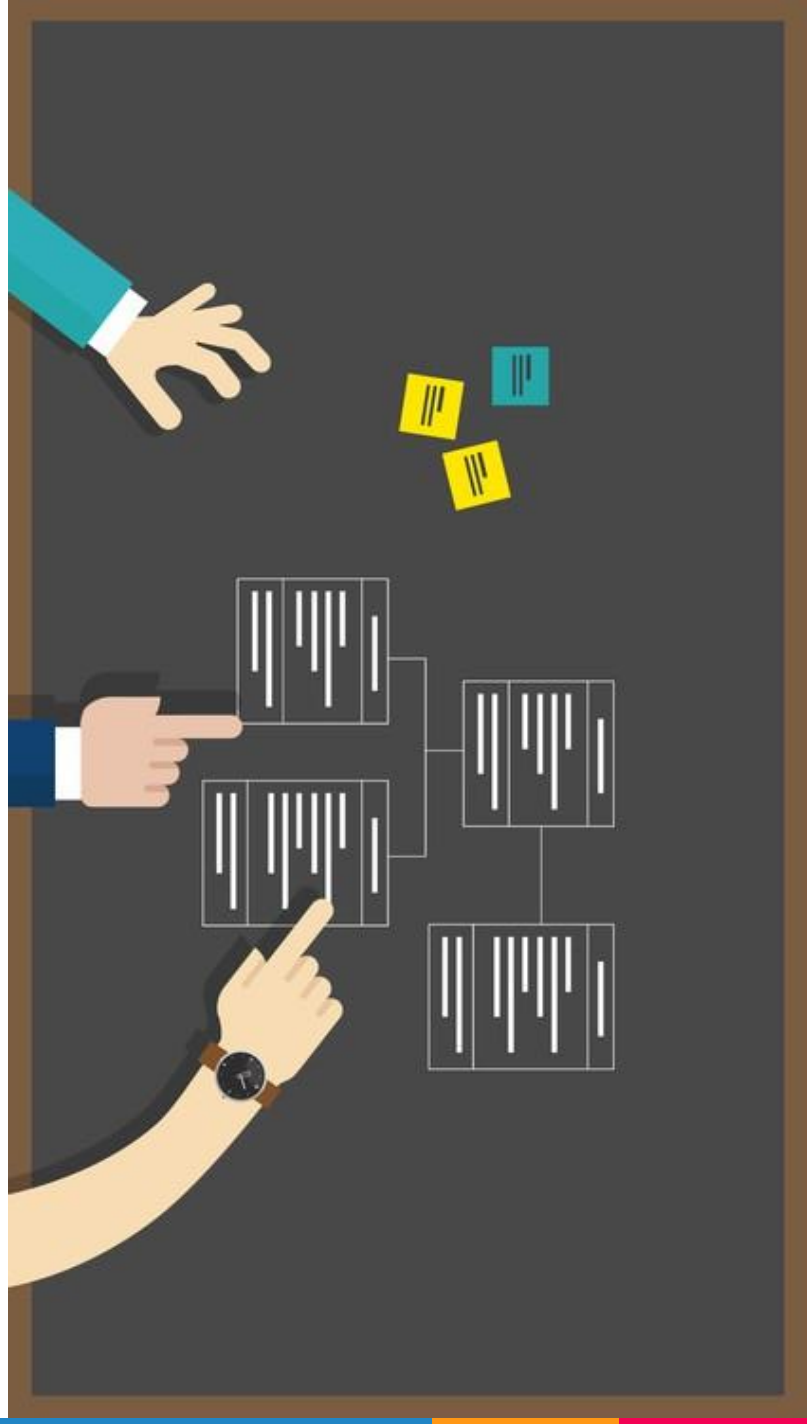
a = a % b, fazemos: a %=b

Atribuições Abreviadas

Essas atribuições abreviadas funciona para todos os operadores binários em Java (os que requerem dois operandos).

var op = expressão;

Como eles combinam uma operação com uma atribuição, são chamados de operadores de atribuição compostos.



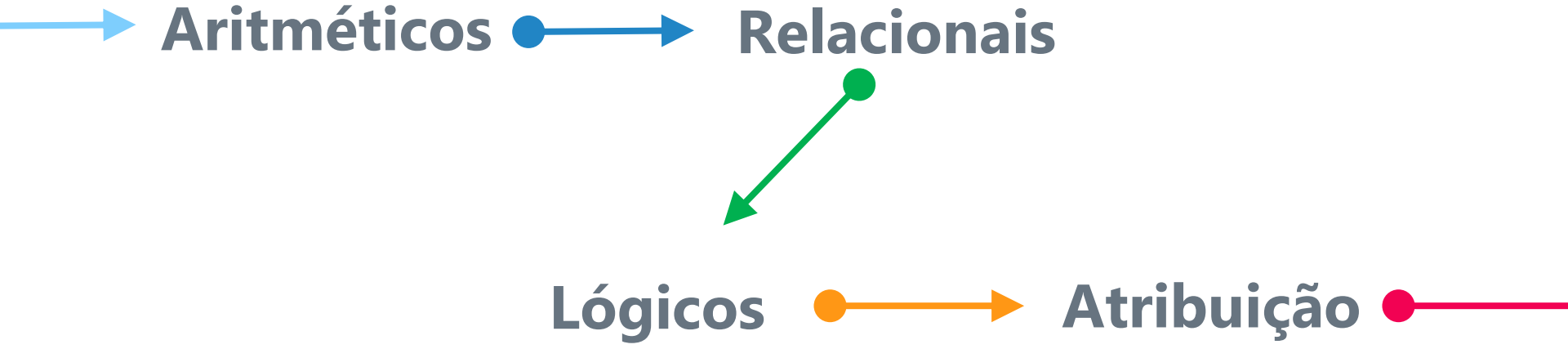
Exemplos

```
public class Atribuicao {
    public static void main(String[] args) {
        int a=1;
        int b=2;
        System.out.println("Valor inicial a = " + a);
        System.out.println("Valor inicial b = " + b);
        System.out.println("Fazendo a +=b");
        a +=b;
        System.out.println("Agora a = " + a);
        System.out.println();

        System.out.println("Fazendo a -=b");
        a -=b;
        System.out.println("Agora a = " + a);
        System.out.println("Fazendo a *=b");
        a *=b;
        System.out.println("Agora a = " + a);
```

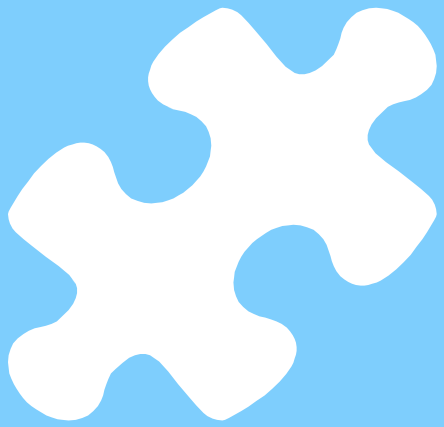
Exemplos

```
System.out.println("Fazendo a +=2 ");
        a +=2;
        System.out.println("Agora a = " + a);
        System.out.println();

        System.out.println("Fazendo a /=b");
        a /=b;
        System.out.println("Agora a = " + a);
        System.out.println();
    }

}
```

Ordem de prioridade




```
a = 1 + 3 == 0 || false;
```



DESAFIO

E aí, vamos praticar?

Operadores de atribuição

Faça um algoritmo que:
a) Obtenha o valor para a variável HT (horas trabalhadas no mês);
b) Obtenha o valor para a variável VH (valor hora trabalhada):
c) Obtenha o valor para a variável PD (percentual de desconto);
d) Calcule o salário bruto => SB = HT * VH;
e) Calcule o total de desconto => TD = (PD/100)*SB;
f) Calcule o salário líquido => SL = SB – TD;
g) Apresente os valores de: Horas trabalhadas, Salário Bruto, Desconto, Salário Liquido.

Entrada:
46
37.24985470297004
2.378793194009624

Saída:
HT = 46
SB = 1713.49
TD = 40.76
SL = 1672.73

Divisão e Resto

Leia um código de cinco algarismos (variável Codigo) e gere o digito verificador (DigitoV) módulo 7 para o mesmo.

Supondo que os cinco algarismos do código são ABCDE, uma forma de calcular o dígito desejado, com módulo 7 é:

DigitoV = resto da divisão de S por 7, onde

S = 6*A + 5*B + 4*C + 3*D + 2*E

Entrada:
SMJJZ

Saída:
6

Divisão e Resto

Dado um número de três algarismos N = CDU (onde C é o algarismo das centenas, D é o algarismo das dezenas e U o algarismo das unidades), considere o número M constituído pelos algarismos de N em ordem inversa, isto é, M = UDC. Gerar M a partir de N (p.ex.: N = 123 -> M = 321).

Entrada:
70816

Saída:
61807

Divisão e Resto

Suponha que uma escola utilize, como código de matrícula, um número inteiro no formato AASDDD, onde:

- Os dois primeiros dígitos, representados pela letra A, são os dois últimos algarismos do ano da matrícula;
- O terceiro dígito, representado pela letra S, vale 1 ou 2, conforme o aluno tenha se matriculado no 1º ou 2º semestre;
- Os quatro últimos dígitos, representados pela letra D, correspondem à ordem da matrícula do aluno, no semestre e no ano em questão.

Crie um algoritmo que leia o número de matrícula de um aluno e imprima o ano e o semestre em que ele foi matriculado.

Entrada:
970296

Saída:
MATRICULA = 296
ANO = 97
SEMESTRE = 0

Obrigado!

Alguma pergunta?

Você pode me contatar em:
ywassef@hotmail.com