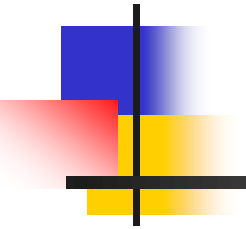
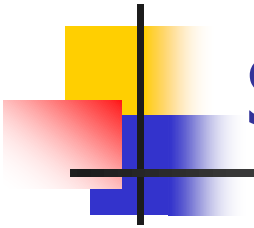


Sintaxe Java – Operadores



Sumário

- Operadores Relacionais
- Operadores Lógicos
- Operadores Bitwise
- Precedência de Operadores

Operadores e Expressões

- O Java oferece um conjunto amplo de operadores para atribuições, operações aritméticas, lógicas, relacionais e bitwise;
- Expressões são construções que podem combinar valores literais, constantes, variáveis, chamadas de métodos e operadores para a obtenção de um resultado numérico ou lógico;

```
SE (a > b) ENTAO
    a ← b * c + (c / 3)
SENAO
  SE (a = b) E (b < c) ENTAO
    b ← 50 * a / 4
```

Operadores de Atribuição e Aritméticos

- Atribuição: =
  - i = 0;
  - y = a*x + b;
- Aritméticos:

Operador	Significado	Exemplo
+	Adição	a + b
-	Subtração	a – b
*	Multiplicação	a * b
/	Divisão	a / b
%	Resto da divisão inteira	a % b
-	Sinal negativo	-a
+	Sinal positivo	+a
++	Incremento unitário	++a ou a++
--	Decremento unitário	--a ou a--

Pré-incremento e Pós-incremento

Pré e Pós Incremento

```
int x = 0, y;
y = ++x;
```

Primeiro incrementa e depois utiliza a variável
y=1

```
int x = 0, y;
y = x++;
```

Primeiro usa a variável e depois incrementa
y=0

```
int x = 1, y;
y = x++ + ++x;
```

Qual é o resultado?
Teste e veja.

Exemplo de Operações Aritméticas

```
public class Aritmetica
{
        public static void main (String args[])
        {
                //Declaração e inicialização das duas variáveis
                int a = 5;
                int b = 2;

                //Exemplos de operações sobre variáveis
                System.out.println ("a = " + a);
                System.out.println ("b = " + b);
                System.out.println (" -b = " + (-b));
                System.out.println ("a + b = " +  (a + b));
                System.out.println ("a - b = " + (a - b));
                System.out.println ("a * b = " + (a * b));
                System.out.println ("a / b = " + (a / b));
                System.out.println ("(float) a / b = " + ((float) a/ b));
                System.out.println ("a % b = " + (a % b));
                System.out.println ("a++ =  " + (a++));
                System.out.println ("--b = " + (--b));
                System.out.println ("a = " + a);
                System.out.println ("b = " + b);
        }
}
```

Altere para pré-incremento e verifique qual a diferença

Promoção e coerção

- Misturar tipos mais simples com tipos mais complexos em uma mesma expressão faz com que o compilador automaticamente converta todos os tipos para o mais complexo. Exemlo:

```
int i=3;
long l=12;
float f = 1.5f;
double d = 2.25;
double res = (i+l)*f/d;
//Resultado em res é 10, mas do tipo double
```

- Esta ação automática do compilador chama-se promoção

Promoção e coerção

- Quando ocorre o contrário, ou seja, a conversão de tipos mais complexos para tipos mais simples, o programador o faz explicitamente.

```
double d = 3.47;
//x recebe 3, parte fracionária é truncada
int x = (int) d;
//Resultado é convertido para float
Float f = (float) (2*d);
```

- Esta ação chama-se coerção e é explícitada pelo programador

Operadores Relacionais

- Os operadores relacionais são aqueles que permitem a comparação de valores, resultando em um tipo lógico;

Operador	**Significado**	**Exemplo**
==	Igual	a == b
!=	Diferente	a != b
>	Maior	a > b
>=	Maior ou igual	a >= b
<	Menor	a < b
<=	Menor ou igual	a <= b

Exemplo de Operadores Relacionais

```
public class Relacional
{
        public static void main (String args[])
        {
                int a = 15;
                int b = 12;
                System.out.println ("a = " + a);
                System.out.println ("b = " + b);
                System.out.println ("a == b -> " + (a == b));
                System.out.println ("a != b -> " + (a != b));
                System.out.println ("a < b -> " + (a < b));
                System.out.println ("a > b -> " + (a > b));
                System.out.println ("a <= b -> " + (a <= b));
                System.out.println ("a >= b -> " + (a >= b));
        }
}
```

Neste ponto altere o valor de b para 15 e verifique o resultado

Operadores Lógicos

- Operadores lógicos são aqueles que conectam logicamente o resultado de diferentes expressões aritméticas ou relacionais;

Operador	Significado	Exemplo
&&	E (AND)	a && b
||	OU (OR)	a || b
!	NÃO (NOT)	!a

Exemplo de Operadores Lógicos

```
public class Logico
{
        public static void main (String args[])
        {
                boolean a = true;
                boolean b = false;
                int c = 5;
                int d = 5;

                System.out.println ("a = " + a);
                System.out.println ("b = " + b);
                System.out.println ("a && b = " + (a && b));
                System.out.println ("a || b = " + (a || b));
                System.out.println ("!a = " + (!a));
                System.out.println ("(c==d) && b = " + ((c==d) && b));
                System.out.println ("(c==d) || b = " + ((c==d) || b));
                System.out.println ("(c==d) && !b = " + ((c==d) && !b));
        }
}
```

Operadores de Atribuição Composta

```
ENQUANTO (x > 0)
INICIO
  soma ← soma + 10;
  x ← x – 1;
FIM
```

- Os operadores de atribuição composta simplificam atribuições em que a variável que recebe o resultado faz parte da primeira parte da expressão;
  - **`x = x + 15 ≈ x += 15;`**
  - **`y = y – pos ≈ y -= pos;`**
  - **`total = total * (valor/ref) ≈ total *= (valor/ref);`**
- Todos os operadores aritméticos e bitwise possuem um operador de atribuição composta equivalente;

Operador Ternário

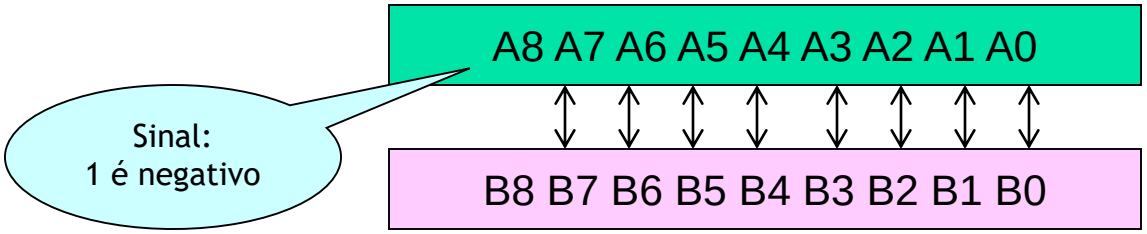
- O operador ternário é um avaliador de condições simples que permite utilizar o valor de duas expressões diferentes conforme o resulta da condição avaliada.

```
<condição> ? <expressão_true> : <expressão_false
```

```
//dois valores
double x = 17.67;
double y = 89.45;
//armazena em max o maior valor entre x e y
double max = x>y ? x : y;
```

Operadores Bitwise

- Os operadores bitwise (bit a bit) são destinados a operações sobre os bits que compõem os seus operandos;



Operador	Significado	Exemplo
&	E bit-a-bit (bitwise and)	a & b
|	OU bit-a-bit (bitwise or)	a | b
^	OU EXCLUSIVO bit-a-bit (bitwise xor)	a ^ b
~	Inversão dos bits	~ a
<<	Rotação à esquerda de N bits	a << N
>>	Rotação à direita de N bits	a >> N
<<<	Rotação à esquerda de N bits sem sinal	a <<< N
>>>	Rotação à direita de N bits sem sinal	a >>> N

Exemplo de Operações Bitwise

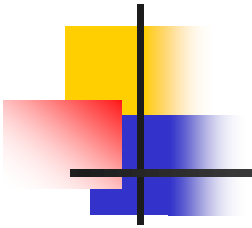
```
public class Bitwise
{
        public static void main (String args[])
        {
                byte a = 0x1F, b = 0x10;

                System.out.println ("a = " + a);
                System.out.println ("b = " + b);
                System.out.println ("a & b = " + (a & b));
                System.out.println ("a | b = " + (a | b));
                System.out.println ("a ^ b = " + (a ^ b));
                System.out.println ("~b = " + (~b));
                System.out.println ("a << 2 = " + (a << 2));
                System.out.println ("-a << 4 = " + (-a << 4));
                System.out.println ("a binario:
"+Integer.toBinaryString(a));
                System.out.println ("-a binario:    " +
                                        Integer.toBinaryString(-a));
                System.out.println ("-a<<4 binario: " +
                                        Integer.toBinaryString(-a<<4));

        }
}
```

Exemplo de Operações Bitwise

- a = 31 (dec) = 00011111 (bin)
- b = 16 (dec) = 00010000 (bin)
  - a & b = 00010000 = 16
  - a | b = 00011111 = 31
  - a ^b = 00001111 = 15
  - a << 2 = 01111100 = 124

Precedência de Operadores

Nível	Operadores
1	[ ] () .
2	++ instanceof new clone –(unário) (cast)
3	* / %
4	+ -
5	<< >> >>>
6	< > <= >=
7	== !=
8	&
9	~
10	|
11	&&
12	||
13	?:
14	=
15	,



Referência Bibliográfica

- JANDL JUNIOR, Peter. Java: guia do programador. São Paulo: Novatec Editora, 2007