Operadores lógicos e matemáticos da linguagem Java

Veja neste artigo quais são os operadores matemáticos e lógicos da linguagem Java, compreendendo a ordem de procedência dos operadores e entendendo quando utilizá-los.

Aritmética

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"**.

Operação	**Operador**	**Expressão algébrica**	**Expressão Java**
Adição	+	a + 1	a +1
Subtração	-	b -2	b -2
Multiplicação	*	cm	c * m
Divisão	/	d / e	d / e
Resto	%	f mod g	f % g

Observação importante: a divisão de inteiros produz um quociente do tipo inteiro, quando possuímos o número 1 maior que o número 2 por exemplo, a expressão 9 / 6 o resultado é interpretado como 1 e a expressão 23 / 8 é avaliada como 2, ou seja a parte fracionária em uma divisão de inteiros é descartada, não contendo nenhum arredondamento.

O **módulo (%)** fornece o resto da divisão, na expressão "x % y", o resultado é o restante depois que x é dividido por y, sendo assim na expressão "7 % 4" o resultado é 3 e "17 % 5" o resultado produz 2. Esse operador é mais utilizado com operandos inteiros, mas também pode ser utilizado com outros tipos.

Precedência de operadores

Os operadores possuem regras que são aplicadas nas expressões aritméticas do Java, que são as mesmas seguidas em álgebra. Quando dizemos que os operadores são aplicados da esquerda para a direita, estamos nos referindo à sua **associatividade**.

Operadores de multiplicação, divisão e módulo são aplicadas primeiro. Por exemplo, quando aparecer uma expressão com várias dessas operações, elas serão aplicadas da esquerda para a direita.

As operações de adição e subtração são aplicadas em seguida.

Abaixo uma tabela de referência dos operadores e suas ordens de avaliação

Operador	**Operação**	**Ordem de avaliação(precedência)**
* / %	Multiplicação Divisão Resto	Avaliado primeiro. Se houver vários operadores desse tipo serão avaliados da esquerda para a direita

+ -	Adição Subtração	Avaliado em seguida. Se houver vários operadores desse tipo, serão avaliados da esquerda para a direita.
=	Atribuição	Avaliado por último

Listagem : Avaliação da precedência dos operadores

```
public class Avalia_Precedencia {
	public static void main(String[] args) {
		int a = 30;
		int b = 5;
		int c = 10;
		int total = (a + b + c) / 10;
		System.out.println("O resultado = "+total);
	}
}
```

Operadores de igualdade e operadores relacionais

Uma condição é uma expressão que pode ser **verdadeira** ou **falsa**, ou seja um valor do tipo **Booleano**.

A **instrução de seleção if** permite um programa tomar uma decisão com base no valor de uma condição.

Listagem 2: Exemplo de uma expressão com operador de igualdade

```
public class Operador_Igualdade {
	public static void main(String[] args) {
		int idade = 16;
		if(idade <= 16){
			System.out.println("É menor de idade");
		}
	}
}
```

Na Listagem 3, o exemplo mostra uma condição que determina que a pessoa é menor de idade, ou seja, se a expressão é verdadeira (true). Se a condição for falsa (false), o corpo do if não é executado.

As condições nas instruções if podem ser formadas utilizando os **operadores de igualdade** (== e !=) e **operadores relacionais** (>,<, >=, <=). Tem que ser prestado muita atenção ao operador de igualdade, o qual possui 2 sinais de igual (==) , sendo bem diferente do que possui 1 igual (=) que apenas atribui valores.

Todos os operadores relacionais têm o mesmo nível de precedência e são associados da esquerda para a direita.

Operador de igualdade	Operador de igualdade	Exemplo de condição em Java	Significado da condição em Java
Operadores de igualdade			
=	==	x == y	x é igual a y
?	!=	x!= y	x é diferente de y
Operadores relacionais			
>	>	x > y	x é maior que y

<	<	x < y	x é menor que y
>_	>=	x >= y	x é maior que ou igual a y
<_	<=	x <= y	x é menor que ou igual a y

Listagem 4: Exemplo de operador relacional

```
public class Valor_Menor {
	public static void main(String[] args) {
		int valorA = 14;
		int valorB = 20;
		if(valorA < valorB)
			System.out.printf("%d é menor que %d", valorA, valorB);
	}
}
```