# Operadores

## Operador de atribuição

Atribuir um valor a uma variável parece bastante direto; você simplesmente atribui o material no lado direito do '=' à variável à esquerda. Abaixo, a instrução 1 que atribui o valor 10 à variável x e a instrução 2 está criando um objeto String chamado name e atribuindo o valor "Abel" a ele.

Instrução 1: x =10;

Instrução 2: String name = new String ("Abel");

A atribuição pode ser de vários tipos. Vamos discutir cada um em detalhes.

### Atribuição Primitiva

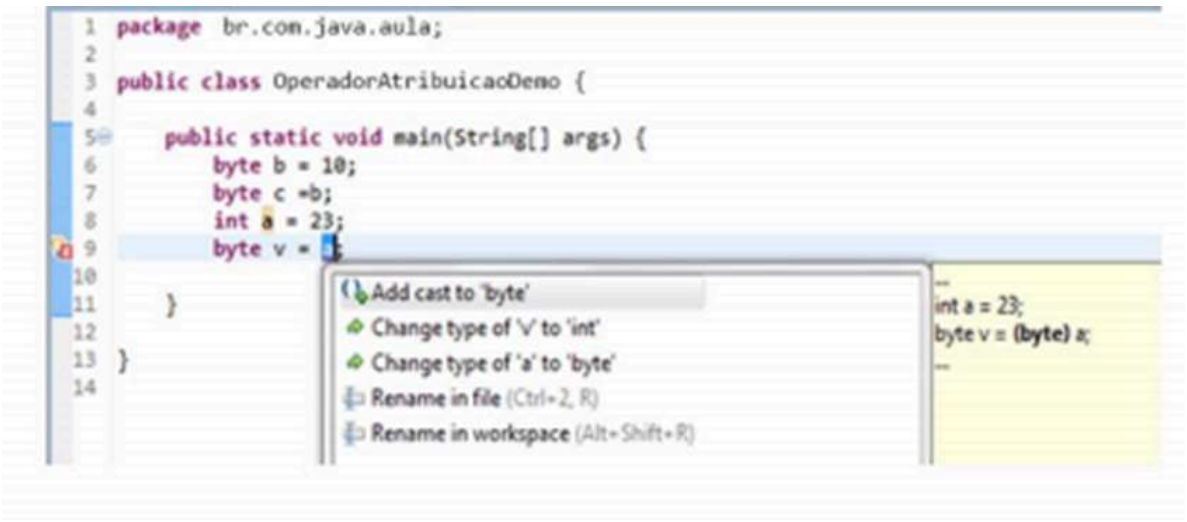
O sinal de igual (=) é usado para atribuir um valor a uma variável. Podemos atribuir uma variável primitiva usando um literal ou o resultado de uma expressão.

```
int x = 7; // atribuição literal

int y = x + 2; // atribuição com uma expressão

int z = x * y; // atribuição com uma expressão com literal
```

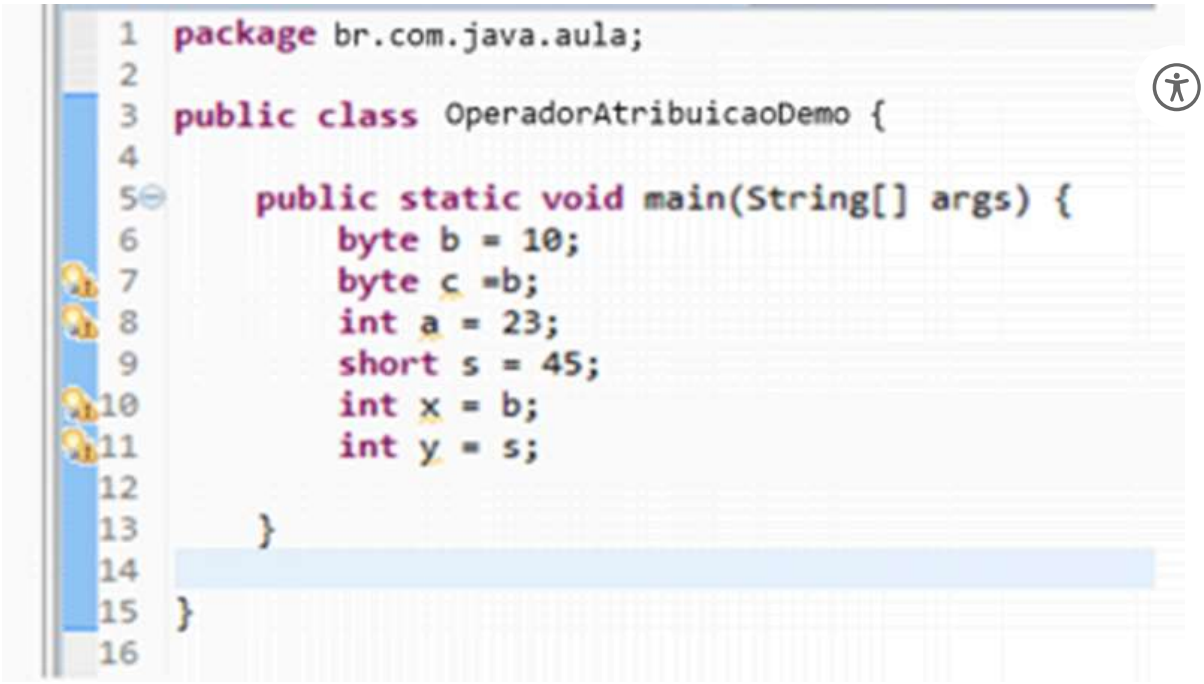
### Casting de tipos primitivos (Primitive Casting)

A conversão permite converter valores primitivos de um tipo para outro. Precisamos fornecer conversão quando estamos tentando atribuir primitiva de maior precisã primitiva de menor precisão, por exemplo. Se tentarmos atribuir variável int (que está no intervalo da variável de bytes) à variável de bytes, o compilador lançará uma exceção chamada "possível perda de precisão ". O IDE Eclipse irá sugerir a solução, bem como mostrado abaixo. Para evitar esse problema, devemos usar conversão de tipo que instruirá o compilador para a conversão de tipo.

byte v = (byte) a;




Nos casos em que tentamos atribuir variáveis de contêiner menores a variáveis de contêineres maiores, não precisamos de conversão explícita. O compilador cuidará dessas conversões de tipo. Por exemplo, podemos atribuir variável de bytes ou variável curta a um int sem nenhuma conversão explícita.



```java
package br.com.java.aula;

public class OperadorAtribuicaoDemo {

    public static void main(String[] args) {
        byte b = 10;
        byte c =b;
        int a = 23;
        short s = 45;
        int x = b;
        int y = s;

    }

}
```

**Atribuindo literais muito grandes para uma variável**

Quando tentamos atribuir um valor de variável muito grande (ou fora do intervalo) para uma variável primitiva, o compilador lançará a exceção "possível perda de precisão" se tentarmos fornecer conversão explícita, em seguida, o compilador o aceitará, mas diminuiu o valor usando o método de complemento de dois. Vamos dar um exemplo do byte que possui espaço de armazenamento de 8 bits e intervalo -128 a 127. No programa abaixo, estamos tentando atribuir 129 valor literal ao tipo primitivo de byte, que está fora do intervalo de bytes, para que o compilador o converta em -127 usando o método de complemento de dois.

```java
public class Transforme {
  public static void main(String[] args) {
    byte b = (byte) 129;
    System.out.println("Valor de b= " + b);
  }
}
```

**Resultado:**

```
Valor de b = -127
```

**Atribuição de variável de referência**

Podemos atribuir um objeto recém-criado à variável de referência do objeto, como abaixo

String s = new String ("Abel");

Funcionário e = new Funcionário ();

A primeira linha fará as seguintes coisas,

· Cria uma variável de referência denominada s do tipo String

· Cria um novo objeto String na memória heap

· Atribui o objeto String recém-criado às variáveis de referência

Você também pode atribuir nulo a uma variável de referência do objeto, o que significa simplesmente que a variável não está se referindo a nenhum objeto. A instrução abaixo cria espaço para a variável de referência Funcionário, mas não cria um objeto Funcionário real.

Funcionário a = null;

**Operadores de atribuição composta**

Em algum momento, precisamos modificar o mesmo valor da variável e atribuí-la novamente à mesma variável de referência. Java permite combinar operadores de atribuição e adição usando um operador abreviado. Por exemplo, a instrução anterior pode ser escrita como:

i + = 8; // É o mesmo que i = i + 8;

O + = é chamado de operador de atribuição de adição. Outros operadores de atribuição são mostrados na seguinte tabela:

| Operador | Nome | Exemplo | Equivalente |
|---|---|---|---|
| += | Atribuição de adição | i += 5; | i = i + 5; |
| -= | Atribuição de subtração | j -= 10; | j = j - 10; |
| *= | Atribuição de multiplicação | k *= 2; | k = k * 2; |
| /= | Atribuição de divisão | x /= 10; | x = x / 10; |
| %= | Atribuição de resto da divisão | a %= 4; | a = a % 4; |

**Operadores aritméticos Java**

Podemos usar operadores aritméticos para realizar cálculos com valores em programas. Operadores aritméticos são usados em expressões matemáticas da mesma maneira que são usados em álgebra. Um valor usado em ambos os lados de um operador é chamado de operando. Por exemplo, na declaração da expressão 47 + 3, os números 47 e 3 são operandos. Os operadores aritméticos são exemplos de operadores binários porque requerem dois operandos. Os operandos dos operadores aritméticos devem ser do tipo numérico. Você não pode usá-los em tipos booleanos, mas pode usá-los em tipos de caracteres, pois o tipo de caracteres em Java é, essencialmente, um subconjunto de int.

int a = 47 + 3;

Em Java, você precisa estar ciente do tipo de resultado de um operador aritmético binário (dois argumentos).

Se um dos operandos for do tipo double, o outro será convertido em double. Caso contrário, se um operando for do tipo float, o outro será convertido em float. E se for do tipo long, o outro será convertido em long. Caso contrário, os dois operandos

são convertidos para o tipo int. Para operadores aritméticos unários (argumento único): Se o operando for do tipo byte, curto ou char, o resultado será um valor tipo int. Caso contrário, um operando numérico unário permanece como está e não é convertido.

As operações aritméticas básicas - adição, subtração, multiplicação e divisão - se comportam como seria de se esperar para todos os tipos numéricos. O operador menos também possui uma forma unária que nega seu único operando. Lembre-se de que quando o operador de divisão é aplicado a um tipo inteiro, não haverá componente fracionário anexado ao resultado.

O programa simples a seguir demonstra os operadores aritméticos. Também ilustra a diferença entre a divisão de ponto flutuante e a divisão inteira.

```
public class OperadorAritmeticoDemo {
  //Demonstrar os operadores aritméticos básicos
  public static void main(String args[]) {
    // aritmética usando números inteiros
    System.out.println("Integer");
    int i = 1 + 1;
    int n = i * 3;
    int m = n / 4;
    int p = m - i;
    int q = -

    System.out.println("i = " + i);
    System.out.println("n = " + n);
    System.out.println("m = " + m);
    System.out.println("p = " + p);
    System.out.println("q = " + q);

    // aritmética usando números do tipo double System.out.println("\nFloating
        Point"); double a =1+1;

    double b = a * 3;
    double c = b / 4;
    double d = c - a;
    double e = -d;

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

**Resultado:**

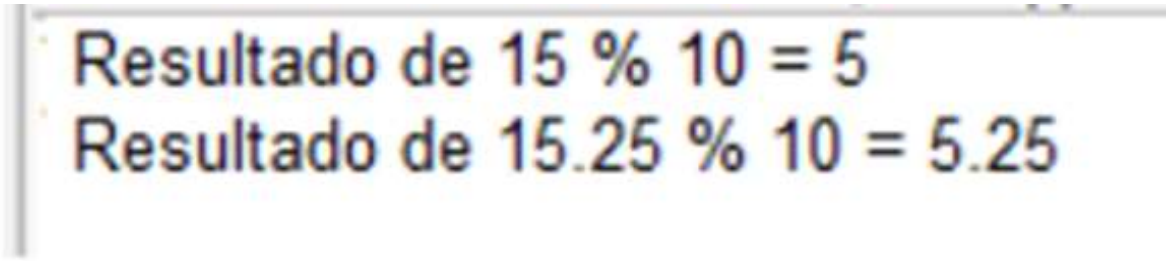
```
Integer
i = 2
n = 6
m = 1
p = -1
q = 1

Floating Point
a = 2.0
b = 6.0
c = 1.5
d = -0.5
e = 0.5
```

**O operador do módulo**

O operador do módulo, %, retorna o resto de uma operação de divisão. Pode ser aplicado a tipos de  ponto flutuante e a números inteiros. O programa de exemplo a seguir demonstra o%:

```
public class RestoDemo {
  public static void main(String[] args) {
    int x = 15;
    int int_resto = x % 10;
    System.out.println("Resultado de 15 % 10=" + int_resto);
    double d = 15.25;
    double double_resto = d % 10;
    System.out.println("Resultado de 15.25 % 10=" + double_resto);
  }
}
```

**Resultado:**




Além disso, há algumas peculiaridades a serem lembradas em relação à divisão por 0:

Um valor inteiro diferente de zero dividido pelo número 0 resultará em ArithmeticException no tempo de execução.

**Operadores aritméticos de atalho (operador de incremento e decremento)**

O operador de incremento aumenta seu operando em um. O operador de decremento diminui seu operando em um. Por exemplo, esta declaração:

x = x + 1;

x ++;

Operador de decréscimo da mesma maneira

x = x - 1;

é equivalente a

x–;

**Atividade Extra**

Vídeo: Fazendo conversões entre tipos primitivos. Link para o vídeo: https://www.youtube.com/watch?v=r-iy10_mSyo

**Referências Bibliográficas**


BARNES, D. J. KOLLING, M. **Programação orientada a objetos com java: uma introdução prática usando o bluej.** 4.ed. Pearson: 2009.

FELIX, R. (Org.). **Programação orientada a objetos.** Pearson: 2017.

MEDEIROS, L. F. de. **Banco de dados: princípios e prática.** Intersaberes: 2013;

ORACLE. Java Documentation, 2021. **Documentação oficial da plataforma Java.** Disponível em: < https://docs.oracle.com/en/java/ >


Ir para exercício