Operators in Java

By

Dr M. Senthilkumar

What are Operators?

- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations
- Java has rich set of built-in operators

Types of Operators

- Arithmetical operators
- Relational operators
- Logical operators
- Assignment operators
- Conditional operators
- instanceof operator
- dot operator

Arithmetic operators

- Arithmetical operators are: +, -, *, /, and %
- They are used to performs an arithmetic (numeric) operations
- You can use the operators +, -, *, /, and % with both integral and floating-point data values

Arithmetic operators

Operator	Meaning	Variables	Integer Arithmetic	Float Arithmetic	Mixed mode Arithmetic
+	Addition	a + b	10 + 5	10.0 + 5.0	10.0 + 5
-	Subtraction	a - b	10 - 5	10.0 - 5.0	10.0 - 5
*	Multiplication	a * b	10 * 5	10.0 * 5.0	10.0 * 5
/	Division	a / b	10 / 5	10.0 / 5.0	10.0 / 5
%	Modulus (Remainder)	a % b	10 % 5	10.0 % 5.0	10.0 % 5

Relational operators

- The relational operators are used to compare two values
- All relational operators are binary operators and therefore require two operands
- A relational expression returns zero when the relation is false and a non-zero when it is true

Relational operators

Operator	Meaning	Variables	Comparing Integers	Comparing Float	Mixed Mode
<	Less than	a < b	10 < 5	10.0 < 5.0	10.0 < 5
<=	Less than or Equal to	a <= b	10 <= 5	10.0 <= 5.0	10.0 <= 5
>	Greater than	a > b	10 > 5	10.0 > 5.0	10.0 > 5
>=	Greater than or Equal to	a >= b	10 >= 5	10.0 >= 5.0	10.0 >= 5
==	Equal to	a == b	10 == 5	10.0 == 5.0	10.0 == 5
!=	Not Equal to	a != b	10 != 5	10.0 != 5.0	10.0 != 5

Logical operators

Operator	Meaning	Variables
&&	Logical AND	a > b && a>c
||	Logical OR	n < 10 || n > 50
!	Logical NOT	!a

Expression1	Expression 2	&& Result	|| Result
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Assignment operator

- The assignment operator '=' is used for assigning a variable to a value
- This operator takes the expression on its RHS and places it into the variable on its LHS

- Variable = Expression;
- c = a + b;

Shorthand Assignment Operators

Operator	Example	Equivalent to
+ =	A += 2	A = A + 2
- =	A -= 2	A = A – 2
% =	A %= 2	A = A % 2
/=	A /= 2	A = A / 2
*=	A *= 2	A = A * 2

Increment and Decrement Operators

- Java provides two special operators: '++' and '--' for incrementing and decrementing the value of a variable by 1
- The increment/ decrement operator cannot be used with constant
- Increment and decrement operators are classified as pre-increment and post-increment

Increment and Decrement Operators

- The syntax of the increment operator is:
  - Pre-increment: ++variable
  - Post-increment: variable++
- The syntax of the decrement operator is:
  - Pre-decrement: —variable
  - Post-decrement: variable—

Increment and Decrement Operators

- In Prefix form first variable is first incremented/ decremented, then evaluated
- In Postfix form first variable is first evaluated, then incremented / decremented.

- **++a**
- **a++**

Conditional operator

- The conditional operator ?: is called ternary operator as it requires three operands.
- The format of the conditional operator is :

  Conditional_ expression ? expression1 : expression2;
- If the value of conditional expression is true then the expression1 is evaluated, otherwise expression2is evaluated.

Conditional operator

```
int a = 5;
int b = 6;
big = (a > b) ? a : b;
```

- The condition evaluates to false, therefore big gets the value from b and it becomes 6.

Bitwise Operators

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise X-OR
~	Bitwise Complement
<<	Bitwise Shift Left
>>	Bitwise Shift Right
>>>	Bitwise Shift Right with Zero fill

The instanceof operator

- It is an Object reference operator

Person instanceof Student

The dot operator

- It is used to access the instance variable or method of an object

```
Person.age
Person.salary( )
```

Expression Evaluation

```
a = 9;
b = 12;
c = 3;
x = a - b / 3 + c * 2 - 1;
x = 9 - 12 / 3 + 3 * 2 - 1;
  = 9 - 4 + 3 * 2 - 1;
  = 9 - 4 + 6 - 1;
  = 5 + 6 - 1;
  = 11 - 1;
  = 10
```

Expression Evaluation

```
y = 9 - 12 / (3 + 3) * (2 - 1);
  = 9 - 12/ 6 * (2 - 1);
  = 9 - 12/ 6 * 1;
  = 9 - 2 * 1;
  = 9 - 2;
  = 7
```

Type Conversion

- **Automatic**

  If expression contains different type of operands, lower type is converted to higher type automatically.

  Result is converted to the type of operand available in LHS. But,

  - float to int truncates the fractional parts
  - double to float rounds digits
  - long to int drops the excess higher order bits

- **Typecasting**

  (type) Expression;

Operator Precedence

Operator	Associativity	Rank
. ( ) [ ]	Left to Right	1
- ++ -- ! ~ (type)	Right to Left	2

Operator Precedence

*** / %**	**Left to Right**	**3**
+ -	Left to Right	4
<< >> >>>	Left to Right	5
< <= > >= instanceof	Left to Right	6

Operator Precedence

== !=	**Left to Right**	**7**
&	Left to Right	8
^	Left to Right	9
|	Left to Right	10
&&	Left to Right	11
||	Left to Right	12
?:	Right to Left	13
= Op=	Right to Left	14

Mathematical Functions

sin( )
cos( )
tan( )
	asin( )
	acos( )
	atan( )
		pow(x,y)
		exp(x)
		log( )

sqrt()
ceil( )
floor( )
	round( )
	abs( )
		max(a,b)
		min(a,b)

Thank you