Java™

Operators in Java

Operators

Java operators are special symbols that perform operations on variables or values.

different types of operators in Java

1. Arithmetic Operators
2. Unary Operators
3. Assignment Operator
4. Relational Operators
5. Logical Operators
6. Ternary Operator
7. Bitwise Operators
8. Shift Operators
9. instance of operator

Arithmetic Operators

Arithmetic Operators are used to perform arithmetic operations mainly on primitive numeric data types.. This Are:

1. * : Multiplication
2. / : Division
3. % : Modulo
4. + : Addition
5. - : Subtraction

ex. a + b

- Division (/) truncates decimal points for integers.
- The final data type of an arithmetic expression is promoted to the largest type among operands, following this order: byte → short → int → long → float → double.

Unary Operators

Unary Operators need only one operand. They are used to increment, decrement, or negate a value.

1. **-** ,Negates the value.
2. **+** , Indicates a positive value (automatically converts byte, char, or short to int. Unary + does nothing practically.
3. **++** , Increments by 1.
   a. Post-Increment: Uses value first, then increments.
   b. Pre-Increment: Increments first, then uses value.
4. **--** , Decrements by 1.
   a. Post-Decrement: Uses value first, then decrements.
   b. Pre-Decrement: Decrements first, then uses value.
5. **!** , Inverts a boolean value.

Assignment Operator

'=' Assignment operator is used to assign a value to any variable. It has right-to-left associativity, i.e. value given on the right-hand side of the operator is assigned to the variable on the left, and therefore right-hand side value must be declared before using it or should be a constant.

1. = , Assignments.
2. += , Add and assignment.
3. -= , Subtract and assignment.
4. *= , Multiply and assignment.
5. /= , Divide and assignment.
6. %= , Modulo and assignment.

Relational Operators

Relational Operators are used to check for relations like equality, greater than, and less than. They return boolean results after the comparison and are extensively used in looping statements as well as conditional if-else statements. This are:

1. == , Equal to.
2. != , Not equal to.
3. < , Less than.
4. <= , Less than or equal to.
5. > , Greater than.
6. >= , Greater than or equal to.

Logical Operators

Logical Operators are used to perform "logical AND" and "logical OR" operations, similar to AND gate and OR gate in digital electronics. They have a short-circuiting effect, meaning the second condition is not evaluated if the first is false.

1. **&&** →Logical AND: returns true when both conditions are true.
2. **||** → Logical OR: returns true if at least one condition is true.
3. **!** →Logical NOT: returns true when a condition is false and vice-versa
4.

Ternary operator

The Ternary Operator is a shorthand version of the if-else statement. It has three operands and hence the name Ternary. The general format is,

condition ? if true : if false

```
int x = 10>9 ? 15 : 20;
```


Source: Internet

Bitwise Operators

Bitwise Operators are used to perform the manipulation of individual bits of a number and with any of the integer types.

1. & (Bitwise AND): returns bit-by-bit AND of input values.
2. | (Bitwise OR): returns bit-by-bit OR of input values.
3. ^ (Bitwise XOR): returns bit-by-bit XOR of input values.
4. ~ (Bitwise Complement): inverts all bits (one's complement).

Shift Operators

Shift Operators are used to shift the bits of a number left or right, thereby multiplying or dividing the number by two, respectively. They can be used when we have to multiply or divide a number by two. The general format ,

```
number shift_op number_of_places_to_shift;
```

1. << (Left shift): Shifts bits left, filling 0s (multiplies by a power of two).
2. >> (right shift): Shifts bits right, filling 0s (divides by a power of two), with the leftmost bit depending on the sign.

instanceof Operator

The instanceof operator is used for type checking. It can be used to test if an object is an instance of a class, a subclass, or an interface. The general format,

object instance of class/subclass/interface

Precedence and Associativity of Operators

Operators	Associativity	Type
++ --	Right to left	Unary postfix
++ -- + - ~ ! (type)	Right to left	Unary prefix
* / %	Left to right	Multiplicative
+ -	Left to right	Additive
<< >> >>>	Left to right	Shift
< <= > >=	Left to right	Relational
== !==	Left to right	Equality
&	Left to right	Boolean Logical AND
^	Left to right	Boolean Logical Exclusive OR
|	Left to right	Boolean Logical Inclusive OR
&&	Left to right	Conditional AND
||	Left to right	Conditional OR
?:	Right to left	Conditional
= += -= *= /= %=	Right to left	Assignment

Precedence and Associativity of Operators

Rule:

If the operators have different precedence, solve the higher precedence first. If they have the same precedence, solve according to associativity, that is, either from right to left or from left to right.

Precedence and Associativity of Operators

1. a + b * c - b / c

2. a=b+++c;

3. a=b+++++c;

4. int result = a + (b * (c - b)) / b;

5. int a = 5, b = 10, c = 3;
   result = a + b * c - ++a / b % c + (b - c) * a;

Common Mistakes to Avoid

The common mistakes that can occur when working with Java Operators are listed below:

1. Confusing == with =: Using == for assignment instead of = for equality check leads to logical errors.
2. Incorrect Use of Floating Point Comparison: Comparing floating point numbers using == can lead to unexpected results due to precision issues.
3. Integer Division Confusion: Dividing two integers will result in integer division (truncating the result).
4. Overusing + for String Concatenation in Loops: Using + for concatenating strings in loops leads to performance issues because it creates new string objects on each iteration.

THANK YOU!!

C-DAC Patna