Java Operators

Operators are used to perform operations on variables and values. For example: +, -, *, / etc.

Java divides the operators into the following groups:

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

---

1. Arithmetic operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y

2. Unary operators

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

- **(–) Unary minus**, used for negating the values.
- **(+) Unary plus** indicates the positive value (numbers are positive without this, however). It performs an automatic conversion to int when the type of its operand is the byte, char, or short. This is called unary numeric promotion.

- **++ : Increment operator**, used for incrementing the value by 1. There are two varieties of increment operators.
    - **Post-Increment:** Value is first used for computing the result and then incremented.
    - **Pre-Increment:** Value is incremented first, and then the result is computed.
- **– – : Decrement operator**, used for decrementing the value by 1. There are two varieties of decrement operators.
    - **Post-decrement:** Value is first used for computing the result and then decremented.
    - **Pre-Decrement: The value** is decremented first, and then the result is computed.

Operator	Name	Description	Example
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

3. Assignment operators

Assignment operators are used to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

4. Relational / Comparison operators

Comparison operators are used to compare two values (or variables).

The return value of a comparison is either `true` or `false`. These values are known as *Boolean values.*

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

5. Logical operators

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

6. Ternary operator

The ternary operator is a shorthand version of the if-else statement. It has three operands and hence the name Ternary.

```
condition ? if true : if false
```

7. Bitwise operator

These operators are used to perform the manipulation of individual bits of a number. They can be used with any of the integer types. They are used when performing update and query operations of the Binary indexed trees.

Operator	**Description**	**Example**
& (bitwise and)	returns bit by bit AND of input values.	(A & B) will give 12 which is 0000 1100
| (bitwise or)	returns bit by bit OR of input values.	(A | B) will give 61 which is 0011 1101
^ (bitwise XOR)	returns bit-by-bit XOR of input values.	(A ^ B) will give 49 which is 0011 0001
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.

8. Shift operator

These 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. General format-

```
number shift_op number_of_places_to_shift;
```

Operator	**Description**	**Example**
<< (left shift)	shifts the bits of the number to the left and fills 0 on voids left as a result. Similar effect as multiplying the number with some power of two.	A << 2 will give 240 which is 1111 0000
>> (right shift)	shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit depends on the sign of the initial number. Similar effect to dividing the number with some power of two.	A >> 2 will give 15 which is 1111
>>> (zero fill right shift)	shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit is set to 0.	A >>>2 will give 15 which is 0000 1111

9. instanceof operator

The instance of the 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. General format-

```
objectinstance of class/subclass/interface
```