

Operators in Java

[→ prev](#)[next →](#)

Operator in **Java** is a symbol which is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in Java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Java Operator Precedence

Operator Type	Category	Precedence
Unary	postfix	<code>expr++ expr--</code>
	prefix	<code>++expr --expr +expr -expr ~ !</code>
Arithmetic	multiplicative	<code>* / %</code>
	additive	<code>+ -</code>
Shift	shift	<code><< >> >>></code>
Relational	comparison	<code>< > <= >= instanceof</code>
	equality	<code>== !=</code>
Bitwise	bitwise AND	<code>&</code>
	bitwise exclusive OR	<code>^</code>
	bitwise inclusive OR	<code> </code>
Logical	logical AND	<code>&&</code>
	logical OR	<code> </code>
Ternary	ternary	<code>? :</code>
Assignment	assignment	<code>= += -= *= /= %= &= ^= = <=>= >>>=</code>

Java Unary Operator

The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a boolean

Java Unary Operator Example: ++ and --

```
class OperatorExample{
public static void main(String args[]){
int x=10;
System.out.println(x++);//10 (11)
System.out.println(++x);//12
System.out.println(x--);//12 (11)
System.out.println(--x);//10
}}
```

Output:

```
10
12
12
10
```

Java Unary Operator Example 2: ++ and --

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=10;
System.out.println(a++ + ++a);//10+12=22
System.out.println(b++ + b++);//10+11=21
}}
```

Output:

```
22
21
```

Java Unary Operator Example: ~ and !

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=-10;
boolean c=true;
boolean d=false;
System.out.println(~a);//-11 (minus of total positive value which starts from 0)
System.out.println(~b);//9 (positive of total minus, positive starts from 0)
System.out.println(!c);//false (opposite of boolean value)
System.out.println(!d);//true
}}
```

Output:

```
-11
9
false
true
```

Java Arithmetic Operators

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

Java Arithmetic Operator Example

```
class OperatorExample{  
    public static void main(String args[]){  
        int a=10;  
        int b=5;  
        System.out.println(a+b);//15  
        System.out.println(a-b);//5  
        System.out.println(a*b);//50  
        System.out.println(a/b);//2  
        System.out.println(a%b);//0  
    }  
}
```

Output:

```
15  
5  
50  
2  
0
```

Java Arithmetic Operator Example: Expression

```
class OperatorExample{  
    public static void main(String args[]){  
        System.out.println(10*10/5+3-1*4/2);  
    }  
}
```

Output:

```
21
```

Java Left Shift Operator

The Java left shift operator `<<` is used to shift all of the bits in a value to the left side of a specified number of times.

Java Left Shift Operator Example

```
class OperatorExample{  
    public static void main(String args[]){  
        System.out.println(10<<2);//10*2^2=10*4=40  
        System.out.println(10<<3);//10*2^3=10*8=80  
        System.out.println(20<<2);//20*2^2=20*4=80  
        System.out.println(15<<4);//15*2^4=15*16=240  
    }  
}
```

Output:

```
40  
80  
80  
240
```

Java Right Shift Operator

The Java right shift operator `>>` is used to move left operand's value to right by the number of bits specified by the right operand.

Java Right Shift Operator Example

```
class OperatorExample{
public static void main(String args[]){
System.out.println(10>>2); //10/2^2=10/4=2
System.out.println(20>>2); //20/2^2=20/4=5
System.out.println(20>>3); //20/2^3=20/8=2
}}
```

Output:

```
2
5
2
```

Java Shift Operator Example: `>>` vs `>>>`

```
class OperatorExample{
public static void main(String args[]){
//For positive number, >> and >>> works same
System.out.println(20>>2);
System.out.println(20>>>2);
//For negative number, >>> changes parity bit (MSB) to 0
System.out.println(-20>>2);
System.out.println(-20>>>2);
}}
```

Output:

```
5
5
-5
1073741819
```

Java AND Operator Example: Logical `&&` and Bitwise `&`

The logical `&&` operator doesn't check second condition if first condition is false. It checks second condition only if first one is true.

The bitwise `&` operator always checks both conditions whether first condition is true or false.

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a<b&&a<c);//false && true = false
System.out.println(a<b&a<c);//false & true = false
}}
```

Output:

```
false
false
```

Java AND Operator Example: Logical `&&` vs Bitwise `&`

```

class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a<b&&a++<c);//false && true = false
System.out.println(a);//10 because second condition is not checked
System.out.println(a<b&a++<c);//false && true = false
System.out.println(a);//11 because second condition is checked
}}

```

Output:

```

false
10
false
11

```

Java OR Operator Example: Logical || and Bitwise |

The logical || operator doesn't check second condition if first condition is true. It checks second condition only if first one is false.

The bitwise | operator always checks both conditions whether first condition is true or false.

```

class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int c=20;
System.out.println(a>b||a<c);//true || true = true
System.out.println(a>b|a<c);//true | true = true
//|| vs |
System.out.println(a>b||a++<c);//true || true = true
System.out.println(a);//10 because second condition is not checked
System.out.println(a>b|a++<c);//true | true = true
System.out.println(a);//11 because second condition is checked
}}

```

Output:

```

true
true
true
10
true
11

```

Java Ternary Operator

Java Ternary operator is used as one liner replacement for if-then-else statement and used a lot in Java programming. it is the only conditional operator which takes three operands.

Java Ternary Operator Example

```

class OperatorExample{
public static void main(String args[]){
int a=2;
int b=5;
int min=(a<b)?a:b;
}

```

```
System.out.println(min);
}}
```

Output:

```
2
```

Another Example:

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}}
```

Output:

```
5
```

Java Assignment Operator

Java assignment operator is one of the most common operator. It is used to assign the value on its right to the operand on its left.

Java Assignment Operator Example

```
class OperatorExample{
public static void main(String args[]){
int a=10;
int b=20;
a+=4;//a=a+4 (a=10+4)
b-=4;//b=b-4 (b=20-4)
System.out.println(a);
System.out.println(b);
}}
```

Output:

```
14
16
```

Java Assignment Operator Example

```
class OperatorExample{
public static void main(String[] args){
int a=10;
a+=3;//10+3
System.out.println(a);
a-=4;//13-4
System.out.println(a);
a*=2;//9*2
System.out.println(a);
a/=2;//18/2
System.out.println(a);
}}
```

Output:

```
13  
9  
18  
9
```

Java Assignment Operator Example: Adding short

```
class OperatorExample{  
    public static void main(String args[]){  
        short a=10;  
        short b=10;  
        //a+=b;//a=a+b internally so fine  
        a=a+b;//Compile time error because 10+10=20 now int  
        System.out.println(a);  
    }  
}
```

Output:

```
Compile time error
```

After type cast:

```
class OperatorExample{  
    public static void main(String args[]){  
        short a=10;  
        short b=10;  
        a=(short)(a+b);//20 which is int now converted to short  
        System.out.println(a);  
    }  
}
```

Output:

```
20
```