

Java Operators

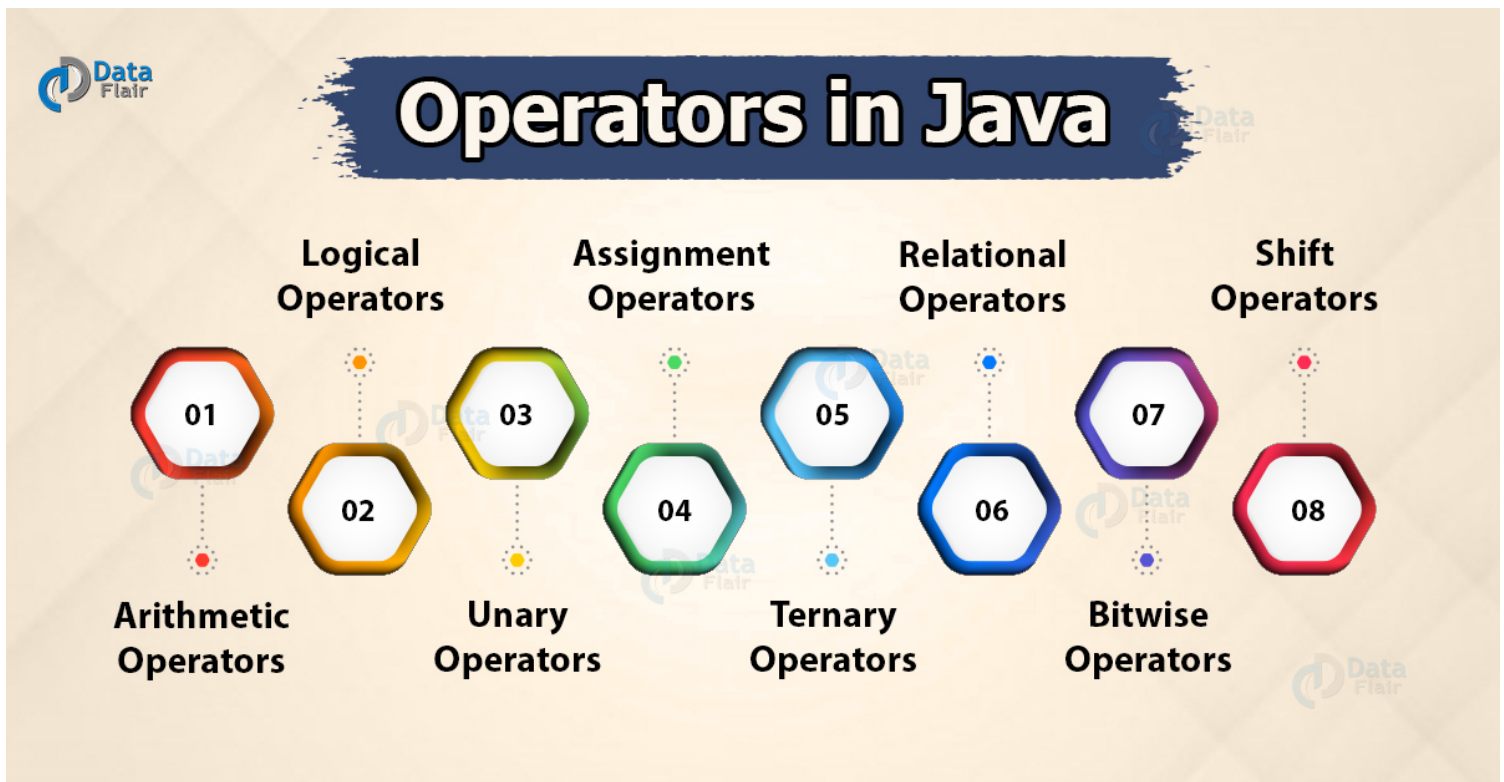
- Operators are used to perform operations on variables and values.
- In the example below, we use the + **operator** to add together two values:

```
int x = 100 + 50;
```

- Although the + operator is often used to add together two values, like in the example above, it can also be used to add together a variable and a value, or a variable and another variable:

```
int sum1 = 100 + 50; // 150 (100 + 50)
int sum2 = sum1 + 250; // 400 (150 + 250)
int sum3 = sum2 + sum2; // 800 (400 + 400)
```

Java divides the operators into the following groups:



- *instanceOf* 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
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

```
public class ArithmeticOperators {
    public static void main(String[] args) {
        int add,
        sub,
        mul,
        div,
        mod;
        int num1 = 5,
        num2 = 8;
        add = num1 + num2;
        sub = num1 - num2;
        mul = num1 * num2;
        div = num1 / num2;
        mod = num2 % num1;
        System.out.println("Addition num1+num2 " + add);
        System.out.println("Subtraction num1-num2 " + sub);
        System.out.println("Multiplication num1*num2 " + mul);
        System.out.println("Division num1/num2 " + div);
        System.out.println("Modulus num2%num1 " + mod);
    }
}
```

```
Command Prompt
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>javac ArithmeticOperators.java

C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>java ArithmeticOperators
Addition num1+num2 13
Subtraction num1-num2 -3
Multiplication num1*num2 40
Division num1/num2 0
Modulus num2%num1 3

C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>
```

2. Logical Operators in Java

- Logical operators are used to determine the logic between variables or values:

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

```
package javaapplication2;
```

```
public class JavaApplication2 {
```

```
    public static void main(String[] args) {
```

```
        int x =5;
```

```
        int y =6;
```

```
        System.out.println("x>4 && y<8: "+(x>4 && y<8));
```

```
        System.out.println("x>7 || y<7: "+(x>7 || y<7));
```

```
        System.out.println("!(x>4 && y<8): "+(!(x>4 && y<8)));
```

Output

```
run:
x>4 && y<8: true
x>7 || y<7: true
!(x>4 && y<8): false
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

3. Unary Operator in Java

Unary operators are those which have only one operand. They are of the following types

- a. Unary plus Operator
- b. Unary minus Operator
- c. Increment Operator
- d. Decrement Operator
- e. Logical Not Operator

a. Unary plus operator in Java

The Unary plus operator converts byte short and character datatype values into integer values. However, it is redundant since an explicit conversion of character to integer can also be done. *Characters when converted to integers return their ASCII value.*

b. Unary Negative Operator in Java

Deliberately converts a positive value to a negative value.

c. Increment Operator in Java

Java increment operator is used for increasing the value of a particular value by one. However there are a few concepts linked with the increment operator.

- **Pre-increment-** In this case, the value of the variable is first increased and then its used/ or its result is computed. Example- ++a;
- **Post-increment-** In this case the value of the variable is first computed and then incremented. Example- a++;

d. Decrement Operator in Java

Java decrement operator is just as opposite to the increment operator. It decreases the value of the variable by 1.

- Pre-decrement- In this case the value of the variable is first decreased and then computed.
Example --a;
- Post-decrement- In this case the value of the variable is first computed and then decremented by 1. Example- a--;

e. Logical Not Operator in Java

Java logical Not operator flips the value of a boolean value. It is denoted by a !.

Java program to understand the concept of Unary Operators:

```
class UnaryOperators {
    public static void main(String[] args) {
        //Variable Declarations
        int num1 = 10,
            num2 = 5,
            res;
        int num3 = 1;
        boolean flag = true;
        char character = 'a';

        res = +character; //The unary + converts the character into a integer value
        System.out.println("The + operator on character transforms it to ASCII value " + res);
        num3 = -num3;
        System.out.println("The - operator on num1 positive value " + num3);

        res = num1++; //First res=num1 then num1++ executed
        System.out.println("The res=num1++ returned value of " + res);
        res = ++num1; //First num1=num1+1 then res=num1
        System.out.println("The res=++num1 returned num1 value of " + res);

        res = num2--; //First res=num2 then num2=num2-1 executed
        System.out.println("The res=num2-- returned value of " + res);
        res = --num2; //First num2=num2-1 then res=num2 executed
        System.out.println("The res=--num2 returned num1 value of " + res);

        //Learning about NOT operator
        System.out.println("The NOT operator returns num1 value of " + !flag);
    }
}
```

Output :

```
Command Prompt
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>java UnaryOperators
The + operator on character transforms it to ASCII value 97
The - operator on num1 positive value -1
The res=num1++ returned value of 10
The res=++num1 returned num1 value of 12
The res=num2-- returned value of 5
The res=--num2 returned num1 value of 3
The NOT operator returns num1 value of false

C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>
```

4. Assignment Operators in Java

Java Assignment operators are used to assign values to the variables on the left side of the equals sign. The associativity is from right to left. Meaning that the values on the right are assigned to the values on the left. The right hand side has to be a constant or a defined variable.

There are the following types of assignment operators

- a. += This returns left=left+right
- b. -= This returns left=left-right
- c. *= This returns left=left*right
- d. /= This returns left=left/right
- e. %= This returns left=left%right

An example of java assignment operator:

```
class AssignmentOperator {
    public static void main(String[] args) {
        int num1 = 10;
        num1 += 5; //action of num1=num1+ 5
        System.out.println("The output of num1+=5 is " + num1);
        num1 -= 5; //action of num1=num1- 5
        System.out.println("The output of num1-=5 is " + num1);
        num1 *= 5; //action of num1=num1* 5
        System.out.println("The output of num1*=5 is " + num1);
        num1 /= 5; //action of num1=num1/ 5
        System.out.println("The output of num1/=5 is " + num1);
        num1 %= 5; //action of num1=num1% 5
        System.out.println("The output of num1%=5 is " + num1);
    }
}
```

5. Ternary Operators in Java

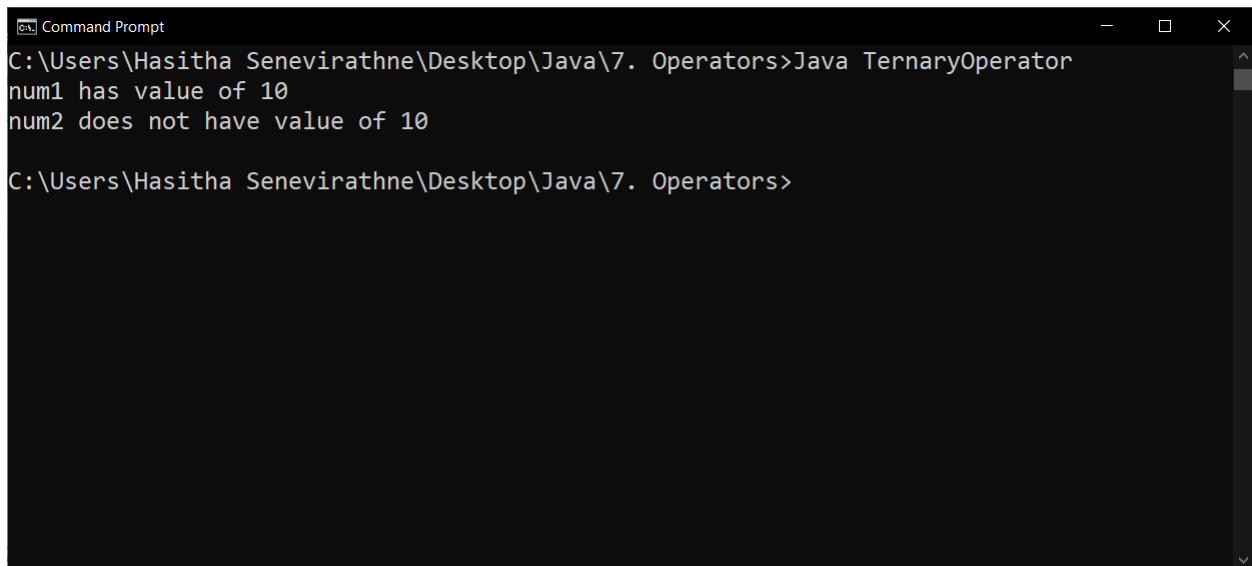
Java ternary operator minimizes and mimics the if else statement. It consists of a condition followed by a question mark(?). It contains two expressions separated by a colon(:). If the condition evaluates to true, then the first expression is executed, else the second expression is executed.

Syntax :

(Condition)?(expression 1):(expression 2);

```
class TernaryOperator {  
    public static void main(String[] args){  
        int num1 = 10,  
            num2 = 5;  
        String result = (num1 == 10) ? "num1 has value of 10": "num1 does not have value of 10";  
        System.out.println(result); //the value of result is printed  
        result = (num2 == 10) ? "num2 has value of 10": "num2 does not have value of 10";  
        System.out.println(result); //the value of result is printed  
    }  
}
```

Output:



```
Command Prompt  
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>Java TernaryOperator  
num1 has value of 10  
num2 does not have value of 10  
  
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>
```

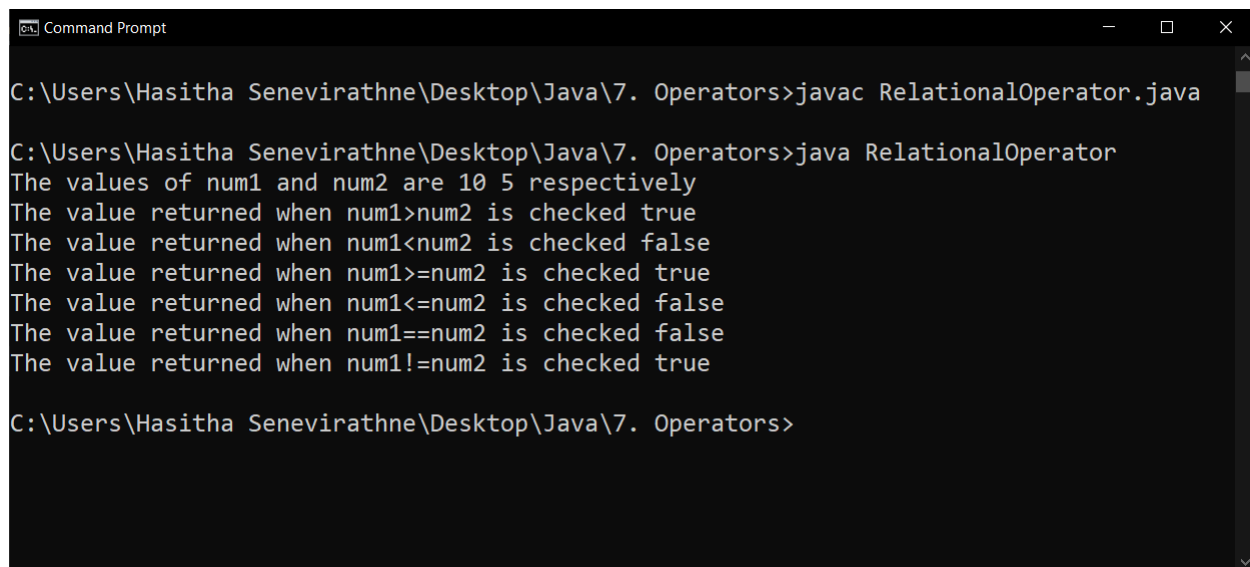
6. Relational Operators in Java

Java Relational Operators are used to check the relation between values or variables. The output of relational operators is always a boolean value. Relational operators are used by if conditions and for loop constraints.

Some relational operators are:

- a. <(Less than) -returns true if left entity lesser than right entity
- b. >(Greater than)- returns true if left entity greater than right entity.
- c. <=(Less than or equal to)- returns true if the left entity is smaller than or equal to right entity.
- d. >=(Greater than or equal to)- returns true if left variable is greater than or equal to right entity
- e. ==(equals to) – returns true if the left and the right entities are equal
- f. !=(not equals to) – returns true if left and right entities are not equal.

Output :



```
Command Prompt
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>javac RelationalOperator.java
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>java RelationalOperator
The values of num1 and num2 are 10 5 respectively
The value returned when num1>num2 is checked true
The value returned when num1<num2 is checked false
The value returned when num1>=num2 is checked true
The value returned when num1<=num2 is checked false
The value returned when num1==num2 is checked false
The value returned when num1!=num2 is checked true
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>
```

7. Bitwise Operators in Java

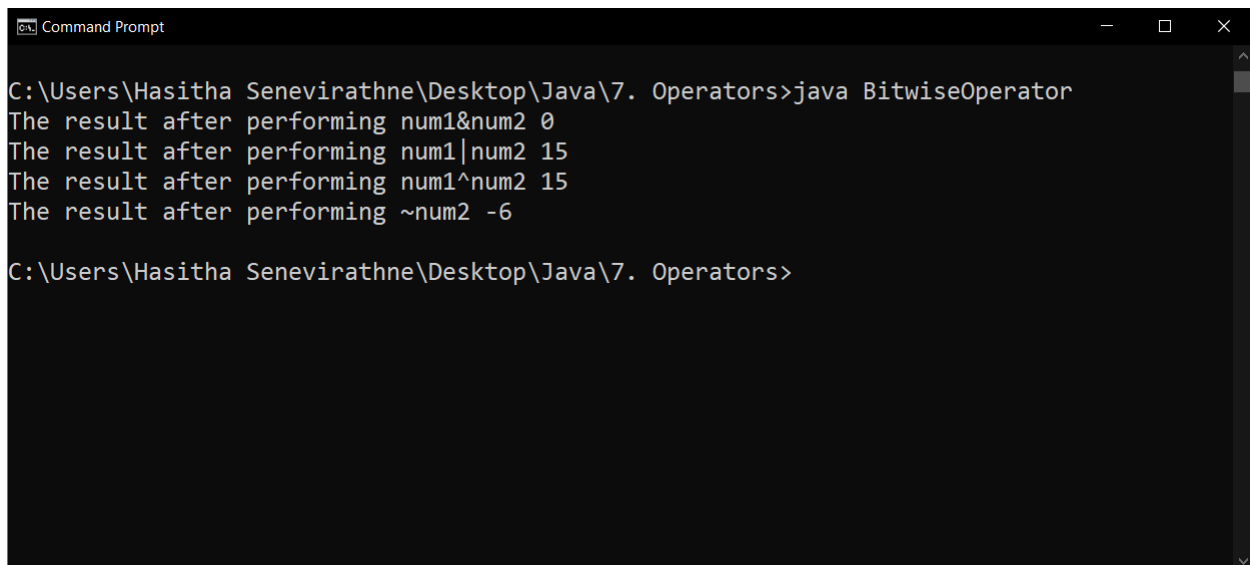
Java Bitwise operators are generally used to perform operations on bits of data. The individual bits of a number are considered in calculation and not the entire number itself. There are the following types of bitwise operators:

- a. AND(&)
- b. OR(|)
- c. XOR(^)
- d. COMPLEMENT(~)

Java program to understand the Bitwise Operators:

```
class BitwiseOperator {  
    public static void main(String[] args) {  
        int num1 = 10,  
            num2 = 5;  
        //num1=10 1010  
        //num2=5 0101  
        System.out.println("The result after performing num1&num2 " + (num1 & num2));  
        System.out.println("The result after performing num1|num2 " + (num1 | num2));  
        System.out.println("The result after performing num1^num2 " + (num1 ^ num2));  
        System.out.println("The result after performing ~num2 " + (~num2));  
    }  
}
```

Output :



```
Command Prompt  
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>java BitwiseOperator  
The result after performing num1&num2 0  
The result after performing num1|num2 15  
The result after performing num1^num2 15  
The result after performing ~num2 -6  
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>
```

[Read more...](#)

8. Shift Operators in Java

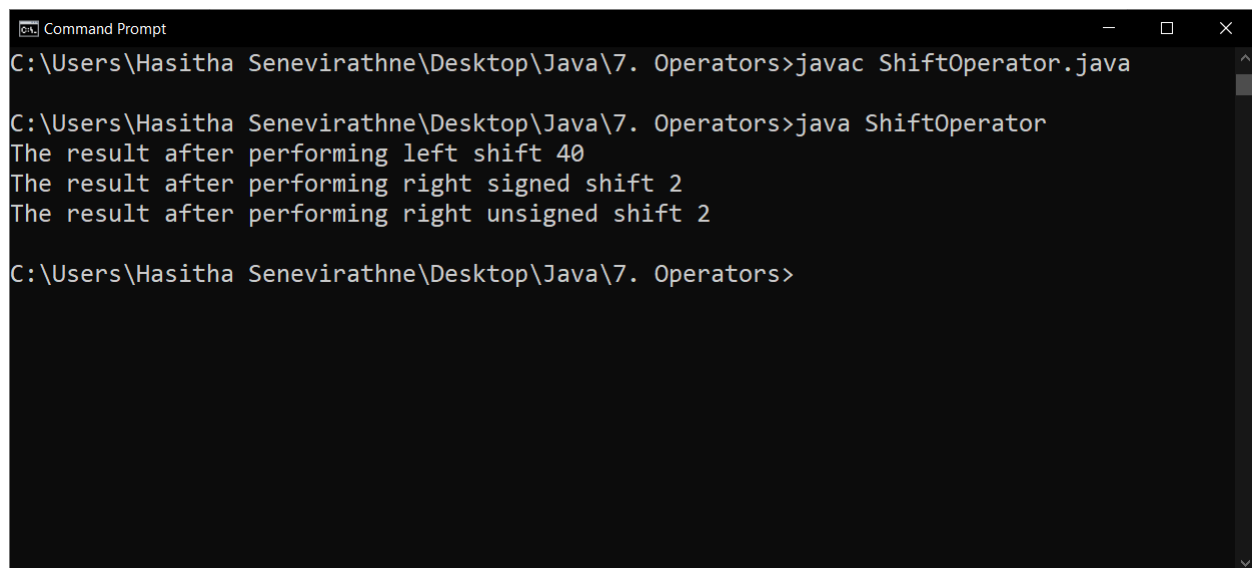
Java Shift Operators are also a part of bitwise operators. There are the following types of shift operators.

- a. Left Shift-** Shifts the bits of the number two places to the left and fills the voids with 0's.
- b. Right Shift-** Shifts the bits of the number two places to the right and fills the voids with 0's. The sign of the number decides the value of the left bit.
- c. Unsigned Right Shift-** It is also the same as the right shift however it changes the leftmost digit's value to 0.

Example program in Java to understand shift operators:

```
class ShiftOperator {  
    public static void main(String[] args) {  
        int num1 = 10;  
        //num1=10 1010  
        //num2=5 0101  
        System.out.println("The result after performing left shift " + (num1  
<< 2));  
        System.out.println("The result after performing right signed shift "  
+ (num1 >> 2));  
        System.out.println("The result after performing right unsigned shift  
" + (num1 >>> 2));  
    }  
}
```

Output:



```
Command Prompt  
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>javac ShiftOperator.java  
  
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>java ShiftOperator  
The result after performing left shift 40  
The result after performing right signed shift 2  
The result after performing right unsigned shift 2  
  
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>
```

9. instanceof Operator in Java

This is a type-check operator. It checks whether a particular object is the instance of a certain class or not.

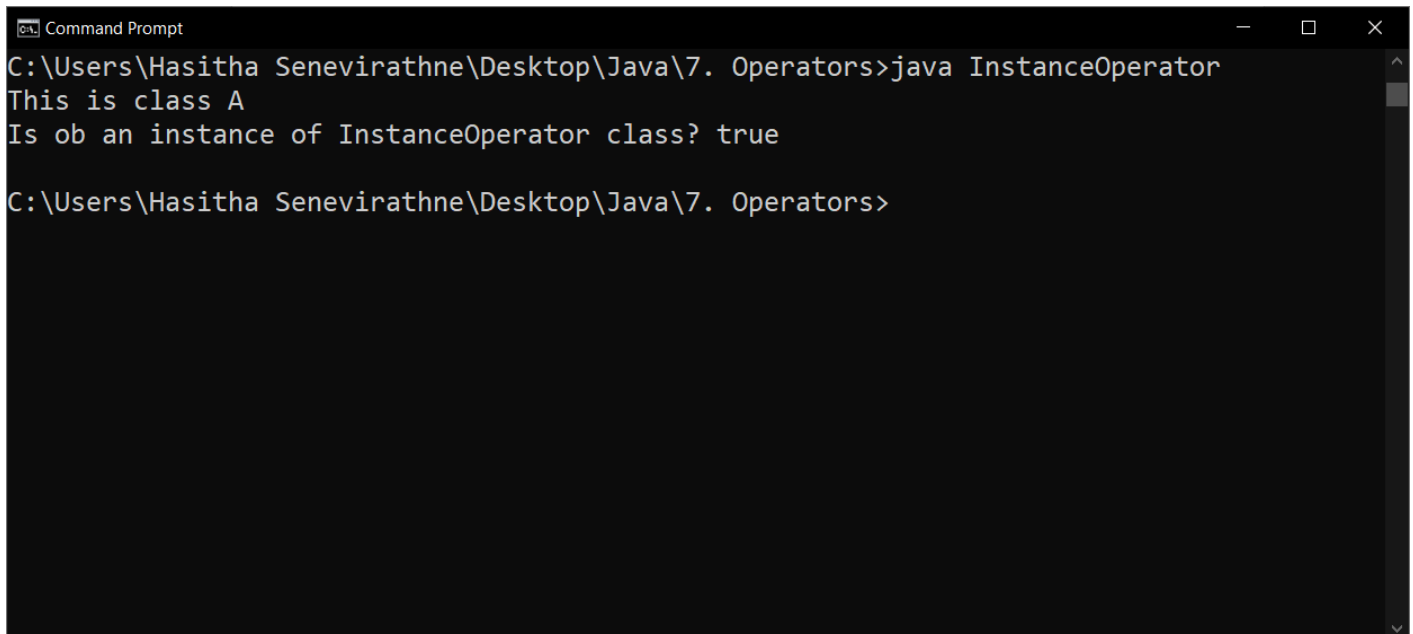
It returns true if the object is a member of the class and false if not.

Java program to test the instanceof Operator:

```
class InstanceOperator extends A{
    public static void main(String[] args){
        InstanceOperator ob = new InstanceOperator();
        System.out.println("Is ob an instance of InstanceOperator class? " +
        (ob instanceof A));
    }
}

class A{
    A(){
        System.out.println("This is class A");
    }
}
```

Output:



```
Command Prompt
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>java InstanceOperator
This is class A
Is ob an instance of InstanceOperator class? true
C:\Users\Hasitha Senevirathne\Desktop\Java\7. Operators>
```

Operators Precedence and Association:

Precedence of operators defines which operator will be executed first if both operators are mentioned in a single expression.

Example in the expression – $a+b*c$. Here the compiler will read it as $(a+(b*c))$.

Association is the method by which the compiler decides which expression to evaluate first if the operators are of the same precedence.

For example, the expression $a=b=c=100$ will be evaluated by the compiler as $(a=(b=(c=100)))$

Association groups the expressions together. However some operators do not have an order of association. These are called non-associative operators. Associative is of two types namely left to right and right to left.

A table with the operator precedences in Java is mentioned below:

LEVEL	OPERATOR	DESCRIPTION	ASSOCIATIVITY
1	$=, +=, -$ $=, *=, /=, \%, \&=, \wedge=, =, <=<, >=>, >>=$	Basic assignment Operators	Right to left
2	$?:$	ternary	Right to left
3	$ $	Logical OR	Left to right
4	$\&\&$	Logical AND	Left to Right
5	$ $	bitwise OR	left to right
6	\wedge	bitwise XOR	left to right
7	$\&$	bitwise AND	left to right
8	$==, !=$	equality operator	left to right
9	$<, <=, >, >=, \text{instanceof}$	relational	not associative
10	$<<, >>, >>>$	shift	left to right
11	$+, -$	additive	left to right
12	$*, /, \%$	multiplicative	left to right
13	$()$, new	object creation	right to left
14	$++, -, +, -, !, \sim$	pre-increment, pre-decrement, plus, minus, NOT, bitwise NOT	right to left
15	$++, -$	unary post increment, unary post decrement	not associative
16	$[], ., ()$	access array element, object member, parentheses	left to right