Bitwise Operators in Java

Difficulty Level : Easy Last Updated : 06 Dec, 2021

Operators constitute the basic building block to any programming language. Java too provides many types of operators which can be used according to the need to perform various calculations and functions, be it logical, arithmetic, relational, etc. They are classified based on the functionality they provide. Here are a few types:

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

This article explains all that one needs to know regarding Bitwise Operators.

Bitwise Operators

Bitwise operators are used to perform the manipulation of individual bits of a number. They can be used with any integral type (char, short, int, etc.). They are used when performing update and query operations of the Binary indexed trees.

Now let's look at each one of the bitwise operators in Java:

1. Bitwise OR (|)

This operator is a binary operator, denoted by '|'. It returns bit by bit OR of input values, i.e., if either of the bits is 1, it gives 1, else it shows 0.

Example:

```
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise OR Operation of 5 and 7
  0101
```

```
| 0111
 ________
  0111  = 7 (In decimal)
```

2. Bitwise AND (&)

This operator is a binary operator, denoted by '&.' It returns bit by bit AND of input values, i.e., if both bits are 1, it gives 1, else it shows 0.

Example:

```
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise AND Operation of 5 and 7
  0101
& 0111
 ________
  0101  = 5 (In decimal)
```

3. Bitwise XOR (^)

This operator is a binary operator, denoted by '^.' It returns bit by bit XOR of input values, i.e., if corresponding bits are different, it gives 1, else it shows 0.

Example:

```
a = 5 = 0101 (In Binary)
b = 7 = 0111 (In Binary)

Bitwise XOR Operation of 5 and 7
  0101
^ 0111
 ________
  0010  = 2 (In decimal)
```

4. Bitwise Complement (~)

This operator is a unary operator, denoted by '~.' It returns the one's complement representation of the input value, i.e., with all bits inverted, which means it makes every 0 to 1, and every 1 to 0.

Example:

```
a = 5 = 0101 (In Binary)

Bitwise Complement Operation of 5

~ 0101
 ________
 1010  = 10 (In decimal)
```

***Note:** Compiler will give 2's complement of that number, i.e., 2's complement of 10 will be - 6.*

```java
// Java program to illustrate
// bitwise operators

public class operators {
    public static void main(String[] args)
    {
        // Initial values
        int a = 5;
        int b = 7;

        // bitwise and
        // 0101 & 0111=0101 = 5
        System.out.println("a&b = " + (a & b));

        // bitwise or
        // 0101 | 0111=0111 = 7
        System.out.println("a|b = " + (a | b));

        // bitwise xor
        // 0101 ^ 0111=0010 = 2
        System.out.println("a^b = " + (a ^ b));

        // bitwise not
        // ~0101=1010
        // will give 2's complement of 1010 = -6
        System.out.println("~a = " + ~a);

        // can also be combined with
        // assignment operator to provide shorthand
        // assignment
```

```
            // a=a&b
            a &= b;
            System.out.println("a= " + a);
        }
    }
```

Output

```
a&b = 5
a|b = 7
a^b = 2
~a = -6
a= 5
```

Bit-Shift Operators (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.

Syntax:

```
 number shift_op number_of_places_to_shift;
```

Types of Shift Operators:

Shift Operators are further divided into 4 types. These are:

1. Signed Right shift operator (>>)
2. Unsigned Right shift operator (>>>)
3. Left shift operator
4. Unsigned Left shift operator (<<<)

> ***Note:*** *For more detail about the Shift Operators in Java, refer Shift Operator in Java.*