Email: akshithaboligila@gmail.com

Contact: +91 7396977317

BITWISE OPERATORS IN JAVA

Research Assignment

by

B. AKSHITHA

Submitted to

Technology Training Head

PUNITH KUMAR SIR

at

Kodnest Technologies Pvt.Ltd

Karnataka, Banglore

June 2023

Introduction

- Bitwise operators are fundamental tools in computer programming that allow us to manipulate individual bits of data. They operate at the binary level, dealing with the 0s and 1s that form the foundation of all digital information. In Java, bitwise operators are used to perform bit-level operations on integer types, such as int, char, long etc.
- They are often used in low-level programming, such as dealing with hardware, optimizing algorithms, or working with data compression.

There are six types of Bitwise Operators:

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.
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.
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.
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.
5. **Left Shift Operator (<<):**
   The left shift operator shifts the bits of the left operand to the left by a specified number of positions.
6. **Right Shift Operator (>>):**
   The right shift operator shifts the bits of the left operand to the right by a specified number of positions, while preserving the sign of the value.

Uses

These bitwise binary operators are particularly useful when dealing with individual bits, performing bitwise masking, or combining and extracting specific bit patterns. They find applications in areas such as bit manipulation, data compression, cryptography, and low-level hardware operations.

It's important to note that these operators work on the binary representation of integers. Therefore, understanding binary arithmetic and the binary representation of numbers is essential to effectively utilize bitwise binary operators.

Example

```
class Bitwise {
   public static void main(String[] args) {
      int a = 10; // Binary: 00001010
      int b = 6;  // Binary: 00000110

      // Bitwise AND
      int resultAnd = a & b;
      System.out.println("Bitwise AND: " + resultAnd); // Output: 2 (Binary: 00000010)

      // Bitwise OR
      int resultOr = a | b;
      System.out.println("Bitwise OR: " + resultOr); // Output: 14 (Binary: 00001110)

      // Bitwise XOR
      int resultXor = a ^ b;
      System.out.println("Bitwise XOR: " + resultXor); // Output: 12 (Binary: 00001100)

      // Bitwise Complement
      int resultComplement = ~a;
      System.out.println("Bitwise Complement: " + resultComplement); // Output: -11
     (Binary: 11110101)

      // Left Shift
      int resultLeftShift = a << 2;
      System.out.println("Left Shift: " + resultLeftShift); // Output: 40 (Binary: 00101000)

      // Right Shift
      int resultRightShift = a >> 2;
      System.out.println("Right Shift: " + resultRightShift); // Output: 2 (Binary: 00000010)
   }
}
```