BIG INTEGER

AND

BITWISE COMPLEMENT OPERATOR

Submitted by

Ragendu V S

E-mail:ragendhuvs02@gmail.com

BigInteger

In Java, you can handle large integers using the BigInteger class from the java.math package. The BigInteger class provides operations for mathematical operations on integers of arbitrary size.

```
import java.math.BigInteger;

public class BigIntegerExample {
    public static void main(String[] args) {
        BigInteger num1 = new BigInteger("123456789012345678901");
        BigInteger num2 = new BigInteger("987654321098765432109");

        // Addition
        BigInteger sum = num1.add(num2);
        System.out.println("Sum: " + sum);

        // Subtraction
        BigInteger difference = num2.subtract(num1);
        System.out.println("Difference: " + difference);

    }
}
```

Output:

Sum: 111111111011111111100

Difference: 86419753208641975320

Bitwise Complement Operator

In Java, the bitwise NOT operator is represented by the exclamation mark (!) symbol. However, it is important to note that the exclamation mark is not specifically a bitwise operator; it is primarily used as the logical negation operator. The logical negation operator reverses the logical state of its operand.

If you want to perform a bitwise NOT operation in Java, you should use the tilde (~) symbol. The tilde operator flips the bits of the operand, changing each 0 bit to 1 and each 1 bit to 0.

Here's an example:

int x = 10; // binary: 00000000000000000000000000001010

int y = ~x; // binary: 11111111111111111111111111110101

System.out.println(y); // Output: -11

In the example above, the bitwise NOT operator ~ is applied to the variable x, which is initially assigned the value 10. The resulting value of y is -11 after the bitwise NOT operation.

The result of a bitwise NOT operation depends on the data type being used. In the example above, the data type int is used, which is a 32-bit signed integer. The result is a two's complement representation of the binary value after flipping the bits.