NAME- RAGHAVENDRA KOWTAL

EMAIL-raghavendrakowtal@gmail.com

BATCH- A2 (2 – 4pm)

ASSIGNMENT TOPIC- BITWISE UNARY OPERATORS IN JAVA

BITWISE UNARY OPERATORS

Bitwise unary operators in Java are operators that perform operations on individual bits of binary representations of numbers. These operators work on a single operand and are used to manipulate and modify the bits within a binary number.

The main bitwise unary operator in Java is the bitwise complement operator, which is represented by the tilde (~) symbol. This operator performs the negation of each individual bit in the binary representation of a number. It flips each bit, changing 0s to 1s and 1s to 0s. The bitwise complement operator is also known as the "one's complement" operator.

By applying the bitwise complement operator to a binary number, you can obtain the complement of each bit. This operator is useful for performing logical negation on individual bits or creating the one's complement of a number.

It's important to note that the bitwise complement operator can be applied to integer types (byte, short, int, long) as well as char. When using this operator, it's crucial to consider the number of bits used to represent the data type, as it affects the range of values and the behaviour of the operator.

BITWISE NOT OPERATOR

The bitwise NOT operator, denoted by the tilde (~) symbol in Java, is a unary operator that performs the negation of each individual bit in a binary number. When applied to an operand, it flips each bit in its binary representation, effectively changing 0s to 1s and 1s to 0s. This operator is also known as the "one's complement" operator. It can be used with integer types (byte, short, int, long) and char. The bitwise NOT operator is particularly useful for manipulating individual bits or performing bitwise operations on binary data. By negating each bit, it allows for logical operations such as masking or toggling specific bits within a binary number. It's important to consider the number of bits used to represent the data type to ensure the desired behaviour and avoid unexpected results.

Example:

```
int x = 5;        // Binary: 00000000 00000000 00000000 00000101

int result = ~x;            // Binary: 11111111 11111111 11111111 11111010
```

BITWISE COMPLEMENT OPERATOR

The bitwise complement operator in Java, represented by the tilde (~) symbol, is a unary operator that performs the complement of each individual bit in a binary number. When applied to an operand, it flips each bit, changing 0s to 1s and 1s to 0s. This operator is also known as the "one's complement" operator.

The bitwise complement operator is particularly useful for performing logical negation on individual bits or creating the one's complement of a number. By applying this operator to a binary number, you can obtain the complement of each bit, effectively inverting its value.

It's important to note that the bitwise complement operator can be applied to integer types (byte, short, int, long) as well as char. However, the result of the operation depends on the number of bits used to represent the data type. It's crucial to consider the data type's bit size to ensure the desired behaviour and avoid unexpected results.

In practical applications, the bitwise complement operator is commonly used in bitwise operations, data manipulation, and low-level programming. It allows for fine-grained control over individual bits within binary numbers, enabling operations such as masking, toggling specific bits, or checking parity.

Example:

```
int x = 5;        // Binary: 00000000 00000000 00000000 00000101

int result = ~x;            // Binary: 11111111 11111111 11111111 11111010
```

BIG INTEGERS

Refers to a class in Java's standard library that provides support for arbitrary-precision integers. Unlike primitive integer types (e.g., int, long) that have fixed ranges, Big Integer allows you to work with integers of any size, limited only by the available memory.

The Big Integer class is part of the java. Math package and provides various methods to perform arithmetic operations, comparisons, and other operations on large integers. Some key features and characteristics of Big Integer are:

Arbitrary Precision: Big Integer stores integers using an array of digits and allows for numbers of arbitrary length. This makes it suitable for dealing with extremely large numbers that would exceed the range of primitive integer types.

Immutable: Big Integer objects are immutable, meaning their values cannot be changed once created. Instead, operations on BigInteger objects return new BigInteger instances representing the results of those operations.

Operations and Methods: BigInteger provides methods for basic arithmetic operations such as addition, subtraction, multiplication, and division. It also offers methods for comparison, exponentiation, modular arithmetic, and more.

Support for Special Operations: The BigInteger class includes methods for handling prime numbers, greatest common divisor, least common multiple, and other specialized operations commonly needed in number theory or cryptography.

Example:

```
import java.math.BigInteger;

public class BigIntegerExample {

    public static void main(String[] args) {

        BigInteger num1 = new BigInteger ("12345678901234567890");

        BigInteger num2 = new BigInteger ("98765432109876543210");

        BigInteger sum = num1.add(num2);

        System.out.println("Sum: " + sum);

    }

}
```