BITWISE OPERATORS:

In Java, bitwise operators are used to perform operations on individual bits of integer values. Java provides several bitwise operators that can be used to manipulate and analyze bits at a low level. The types of bitwise operators are as follows:

1. **Bitwise AND (&):**

The bitwise AND operator compares the corresponding bits of two operands and produces a result where each bit is set to 1 if both corresponding bits are 1; otherwise, it sets the bit to 0.

Example:

```
int a = 5;      // binary: 0101
int b = 3;      // binary: 0011
int result = a & b;   // binary: 0001, decimal: 1
System.out.println(result);
```

Output: 1(0001)

2. **Bitwise OR (|):**

The bitwise OR operator compares the corresponding bits of two operands and produces a result where each bit is set to 1 if either of the corresponding bits is 1; otherwise, it sets the bit to 0.

Example:

```
int a = 5;      // binary: 0101
int b = 3;      // binary: 0011
int result = a | b;   // binary: 0111, decimal: 7
System.out.println(result);
```

Output: 7(0111)

3. **Bitwise XOR (^):**

The bitwise XOR (exclusive OR) operator compares the corresponding bits of two operands and produces a result where each bit is set to 1 if only one of the corresponding bits is 1; otherwise, it sets the bit to 0.

Example:

```
int a = 5;      // binary: 0101
int b = 3;      // binary: 0011
int result = a ^ b;   // binary: 0110, decimal: 6
System.out.println(result);
```

Output: 6(0110)

4. **Bitwise NOT (~):**

The bitwise NOT operator is a unary operator that flips the bits of its operand. It sets each bit to the opposite of its current value, resulting in a one's complement of the operand.

Example:

```
int a = 5;      // binary: 0101
int result = ~a;     // binary: 1010, decimal: -6 (due to two's complement representation)
System.out.println(result);
```

Output: -6 (1010)

5. **Left Shift (<<):**

The left shift operator shifts the bits of its left operand to the left by a specified number of positions. It discards the shifted bits and fills the vacated positions with zeros.

Example:

```
int a = 5;      // binary: 0101
int result = a << 2; // binary: 010100, decimal: 20
System.out.println(result);
```

Output: 20 (10100)

6. **Right Shift (>>):**

The right shift operator shifts the bits of its left operand to the right by a specified number of positions. It discards the shifted bits and fills the vacated positions with the sign bit (the leftmost bit for signed data types).

Example:

```
int a = -10;    // binary: 11111111111111111111111111110100
int result = a >> 2; // binary: 11111111111111111111111111111101, decimal: -3
System.out.println(result);
```

Output: -3 (11111111111111111111111111110)

7. **Unsigned Right Shift (>>>):**

The unsigned right shift operator shifts the bits of its left operand to the right by a specified number of positions. It discards the shifted bits and fills the vacated positions with zeros.

Example:

```
int a = -10;    // binary: 11111111111111111111111111110100
int result = a >>> 2; // binary: 00111111111111111111111111111101, decimal: 1073741821
System.out.println(result);
```

Output: 1073741822(00111111111111111111)