Java Operators

Operators in Java are special symbols that perform operations on variables and values. There are several types of operators in Java, each serving different purposes. Here's an in-depth look at various types of Java operators with examples.

1. Arithmetic Operators

Arithmetic operators perform mathematical operations.

- **Addition (`+`):** Adds two operands.
- **Subtraction (`-`):** Subtracts the right operand from the left.
- **Multiplication (`*`):** Multiplies two operands.
- **Division (`/`):** Divides the left operand by the right.
- **Modulus (`%`):** Returns the remainder of the division.

Example:

```
public class ArithmeticOperators {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;

        System.out.println("Addition: " + (a + b)); // 15
        System.out.println("Subtraction: " + (a - b)); // 5
        System.out.println("Multiplication: " + (a * b)); // 50
        System.out.println("Division: " + (a / b)); // 2
        System.out.println("Modulus: " + (a % b)); // 0
    }
}
```

2. Relational Operators

Relational operators compare two operands and return a boolean value (`true` or `false`).

- **Equal to (`==`):** Checks if two operands are equal.
- **Not equal to (`!=`):** Checks if two operands are not equal.
- **Greater than (`>`):** Checks if the left operand is greater than the right.
- **Less than (`<`):** Checks if the left operand is less than the right.

- **Greater than or equal to (`>=`):** Checks if the left operand is greater than or equal to the right.
- **Less than or equal to (`<=`):** Checks if the left operand is less than or equal to the right.

Example:

```
public class RelationalOperators {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;

        System.out.println("a == b: " + (a == b)); // false
        System.out.println("a != b: " + (a != b)); // true
        System.out.println("a > b: " + (a > b)); // true
        System.out.println("a < b: " + (a < b)); // false
        System.out.println("a >= b: " + (a >= b)); // true
        System.out.println("a <= b: " + (a <= b)); // false
    }
}
```

3. Logical Operators

Logical operators are used to perform logical operations.

- **Logical AND (`&&`):** Returns `true` if both operands are true.
- **Logical OR (`||`):** Returns `true` if at least one operand is true.
- **Logical NOT (`!`):** Reverses the logical state of its operand.

Example:

```
public class LogicalOperators {
    public static void main(String[] args) {
        boolean x = true;
        boolean y = false;

        System.out.println("x && y: " + (x && y)); // false
        System.out.println("x || y: " + (x || y)); // true
        System.out.println("!x: " + (!x)); // false
        System.out.println("!y: " + (!y)); // true
    }
```

```
}
```

4. Assignment Operators

Assignment operators are used to assign values to variables.

- **Assignment (=):** Assigns the right operand to the left operand.
- **Addition assignment (+=):** Adds the right operand to the left operand and assigns the result to the left operand.
- **Subtraction assignment (-=):** Subtracts the right operand from the left operand and assigns the result to the left operand.
- **Multiplication assignment (*=):** Multiplies the right operand with the left operand and assigns the result to the left operand.
- **Division assignment (/=):** Divides the left operand by the right operand and assigns the result to the left operand.
- **Modulus assignment (%=):** Takes the modulus using two operands and assigns the result to the left operand.

Example:

```
public class AssignmentOperators {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;

        a += b; // a = a + b
        System.out.println("a += b: " + a); // 15

        a -= b; // a = a - b
        System.out.println("a -= b: " + a); // 10

        a *= b; // a = a * b
        System.out.println("a *= b: " + a); // 50

        a /= b; // a = a / b
        System.out.println("a /= b: " + a); // 10

        a %= b; // a = a % b
        System.out.println("a %= b: " + a); // 0
    }
```

```
}
```

5. Unary Operators

Unary operators operate on a single operand.

- **Unary plus (+):** Indicates a positive value.
- **Unary minus (-):** Negates the value.
- **Increment (++):** Increases the value by 1.
- **Decrement (--):** Decreases the value by 1.
- **Logical complement (!):** Inverts the value of a boolean.

Example:

```
public class UnaryOperators {
    public static void main(String[] args) {
        int a = 5;

        System.out.println("a: " + a); // 5
        System.out.println("++a: " + ++a); // 6 (pre-increment)
        System.out.println("a++: " + a++); // 6 (post-increment)
        System.out.println("a: " + a); // 7

        System.out.println("--a: " + --a); // 6 (pre-decrement)
        System.out.println("a--: " + a--); // 6 (post-decrement)
        System.out.println("a: " + a); // 5
    }
}
```

6. Bitwise Operators

Bitwise operators perform operations on bits.

- **AND (&):** Performs a bitwise AND.
- **OR (|):** Performs a bitwise OR.
- **XOR (^):** Performs a bitwise XOR.
- **Complement (~):** Inverts all bits.
- **Left shift (<<):** Shifts bits to the left.
- **Right shift (>>):** Shifts bits to the right.
- **Unsigned right shift (>>>):** Shifts bits to the right without sign extension.

Example:

```
public class BitwiseOperators {
    public static void main(String[] args) {
        int a = 5;  // 0101 in binary
        int b = 3;  // 0011 in binary

        System.out.println("a & b: " + (a & b)); // 1 (0001 in binary)
        System.out.println("a | b: " + (a | b)); // 7 (0111 in binary)
        System.out.println("a ^ b: " + (a ^ b)); // 6 (0110 in binary)
        System.out.println("~a: " + (~a)); // -6 (inverts all bits)
        System.out.println("a << 1: " + (a << 1)); // 10 (1010 in
binary)
        System.out.println("a >> 1: " + (a >> 1)); // 2 (0010 in
binary)
        System.out.println("a >>> 1: " + (a >>> 1)); // 2 (0010 in
binary)
    }
}
```

7. Ternary Operator

The ternary operator is a shorthand for an `if-else` statement. It takes three operands.

- **Syntax:** `condition ? expression1 : expression2`

Example:

```
public class TernaryOperator {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;

        int max = (a > b) ? a : b;
        System.out.println("Max: " + max); // 20
    }
}
```

8. Instanceof Operator

The `instanceof` operator checks if an object is an instance of a specific class or interface.

Example:

```
class Animal {}
class Dog extends Animal {}

public class InstanceofOperator {
    public static void main(String[] args) {
        Animal animal = new Dog();

        System.out.println("animal is an instance of Animal: " +
(animal instanceof Animal)); // true
        System.out.println("animal is an instance of Dog: " + (animal
instanceof Dog)); // true
        System.out.println("animal is an instance of String: " +
(animal instanceof String)); // false
    }
}
```