Java Operators

Arithmetic Operations

Operator	Name	Example
`+`	Addition	`a + b`
`-`	Subtraction	`a - b`
`*`	Multiplication	`a * b`
`/`	Division	`a / b`
`%`	Modulo	`a % b`
`++`	Increment	`a++` or `++a`
`--`	Decrement	`a--` or `--a`

The `++` and `--` operators can be used in both postfix (e.g., `a++`) and prefix (e.g., `++a`) forms, and their behavior can differ based on their usage in expressions. The difference between these two forms lies in the order of operations when they are used within larger expressions.

Prefix Form (`++a` and `--a`)

In the prefix form, the variable is incremented or decremented **before** its value is used in the expression.

```
public class Main {
    public static void main(String[] args) {
        int a = 5;
        int b = ++a;  // First, 'a' is incremented to 6. Then, this new value is assign
        System.out.println(a);  // Outputs: 6
        System.out.println(b);  // Outputs: 6
    }
}
```

Postfix Form (`a++` and `a--`)

In the postfix form, the variable's current value is used in the expression **before** it is incremented or decremented.

```
public class Main {
    public static void main(String[] args) {
        int a = 5;
        int b = a++;  // First, the current value of 'a' (which is 5) is assigned to 'b
        System.out.println(a);  // Outputs: 6
        System.out.println(b);  // Outputs: 5
    }
}
```

Bitwise Operations

Operator	Name	Example
`&`	Bitwise AND	`int result = a & b;`
`|`	Bitwise OR	`result = a | b;`
`^`	Bitwise XOR	`result = a ^ b;`
`~`	Bitwise complement	`result = ~a;`
`<<`	Left shift	`result = a << 1;`
`>>`	Right shift	`result = a >> 1;`
`>>>`	Unsigned right shift	`result = a >>> 1;`

Relational Operations

Operator	Name	Example
`==`	Equal to	`boolean isEqual = (a == b);`
`!=`	Not equal to	`boolean isNotEqual = (a != b);`
`>`	Greater than	`boolean isGreater = (a > b);`
`<`	Less than	`boolean isLess = (a < b);`

Operator	Name	Example
`>=`	Greater than or equal to	`boolean isGE = (a >= b);`
`<=`	Less than or equal to	`boolean isLE = (a <= b);`

Using `==` to compare objects checks for reference equality, not content equality. To compare the content of objects, such as strings, use the `.equals()` method.

```
String a = new String("example");
String b = new String("example");
boolean isSameContent = a.equals(b); // true, compares content
boolean isSameObject = (a == b); // false, because they are different objects in memory
```

Assignment Operations

Operator	Name	Example
`=`	Assignment	`int c = a;`
`+=`	Addition assignment	`c += a;`
`-=`	Subtraction assignment	`c -= a;`
`*=`	Multiplication assignment	`c *= a;`
`/=`	Division assignment	`c /= a;`
`%=`	Modulus assignment	`c %= a;`
`<<=`	Left shift assignment	`c <<= 2;`
`>>=`	Right shift assignment	`c >>= 2;`
`>>>=`	Unsigned right shift assignment	`c >>>= 2;`
`&=`	Bitwise AND assignment	`c &= a;`
`|=`	Bitwise OR assignment	`c |= a;`
`^=`	Bitwise XOR assignment	`c ^= a;`

Implicit Type Casting in Assignment Operators

> The `+=`, `-=`, `*=`, `/=`, and `%=` operators do implicit type casting when the right-hand side operand is of a different type than the left-hand side operand. For example, if `a` is an `int` and `b` is a `double`, then `a += b;` is equivalent to `a = (type of a) (a + b);` or `a = (int) (a + b);`.