Java Operators and Input/Output

This guide explains the various **operators** in Java and how to use **input/output** operators effectively. Operators are symbols that perform operations on variables and values. Understanding operators is a crucial part of mastering Java.

Table of Contents

---

8. Operators

8.1 Arithmetic Operators

These operators perform basic arithmetic operations.

8.1.1 *+, -, , /, %

1. `+` **(Addition)**: Adds two operands.

```
int result = 5 + 3;  // result is 8
```

2. `-` **(Subtraction)**: Subtracts the second operand from the first.

```
int result = 5 - 3;  // result is 2
```

3. `*` **(Multiplication)**: Multiplies two operands.

```
int result = 5 * 3;  // result is 15
```

4. `/` **(Division)**: Divides the first operand by the second. Be cautious with integer division; it truncates the result.

```
int result = 10 / 3;  // result is 3 (not 3.333)
```

5. `%` **(Modulus)**: Returns the remainder when one operand is divided by another.

```
int result = 10 % 3;  // result is 1 (remainder)
```

8.2 Relational Operators

These operators compare two values and return a boolean result (`true` or `false`).

8.2.1 ==, !=, >, <, etc.

1. `==` **(Equal to)**: Checks if two values are equal.

```
boolean result = (5 == 5);  // result is true
```

2. `!=` **(Not equal to)**: Checks if two values are not equal.

```
boolean result = (5 != 3);  // result is true
```

3. `>` **(Greater than)**: Checks if the first value is greater than the second.

```
boolean result = (5 > 3);  // result is true
```

4. `<` **(Less than)**: Checks if the first value is less than the second.

```
boolean result = (3 < 5);  // result is true
```

5. `>=` **(Greater than or equal to)**: Checks if the first value is greater than or equal to the second.

```
boolean result = (5 >= 3);  // result is true
```

6. `<=` **(Less than or equal to)**: Checks if the first value is less than or equal to the second.

```
boolean result = (3 <= 5);  // result is true
```

8.3 Logical Operators

These operators are used to perform logical operations, often used with boolean values.

8.3.1 &&, ||, !

1. `&&` **(Logical AND)**: Returns `true` if both operands are true.

```
boolean result = (5 > 3) && (3 > 1);  // result is true
```

2. `||` **(Logical OR)**: Returns `true` if at least one operand is true.

```
boolean result = (5 > 3) || (3 < 1);  // result is true
```

3. `!` **(Logical NOT)**: Reverses the boolean value (true becomes false, false becomes true).

```
boolean result = !(5 > 3);  // result is false
```

8.4 Unary Operators

These operators work with only one operand.

8.4.1 ++, --

1. `++` **(Increment)**: Increases the value of the operand by 1.
   - **Pre-increment**: `++x`
   - **Post-increment**: `x++`

```
int x = 5;
++x;  // x becomes 6 (pre-increment)
x++;  // x becomes 7 (post-increment)
```

2. `--` **(Decrement)**: Decreases the value of the operand by 1.

- **Pre-decrement**: `--x`
- **Post-decrement**: `x--`

```
int x = 5;
--x;  // x becomes 4 (pre-decrement)
x--;  // x becomes 3 (post-decrement)
```

8.5 Assignment Operators

These operators are used to assign values to variables.

8.5.1 =, +=, -=, etc.

1. `=` **(Assignment)**: Assigns a value to a variable.

```
int x = 5;  // x is assigned 5
```

2. `+=` **(Addition assignment)**: Adds the right operand to the left operand and assigns the result to the left operand.

```
x += 3;  // x becomes 8 (x = x + 3)
```

3. `-=` **(Subtraction assignment)**: Subtracts the right operand from the left operand and assigns the result to the left operand.

```
x -= 2;  // x becomes 6 (x = x - 2)
```

4. `*=` **(Multiplication assignment)**: Multiplies the left operand by the right operand and assigns the result to the left operand.

```
x *= 2;  // x becomes 12 (x = x * 2)
```

5. `/=` **(Division assignment)**: Divides the left operand by the right operand and assigns the result to the left operand.

```
x /= 3;  // x becomes 4 (x = x / 3)
```

8.6 Bitwise Operators

Bitwise operators perform operations on bits and return an integer result.

8.6.1 &, |, ^, <<, >>, ~

1. `&` **(AND)**: Performs a bitwise AND operation.

```
int x = 5 & 3;  // result is 1 (0101 & 0011 = 0001)
```

2. `|` **(OR)**: Performs a bitwise OR operation.

```
int x = 5 | 3;  // result is 7 (0101 | 0011 = 0111)
```

3. `^` **(XOR)**: Performs a bitwise XOR operation.

```
int x = 5 ^ 3;  // result is 6 (0101 ^ 0011 = 0110)
```

4. `<<` **(Left shift)**: Shifts the bits to the left by a specified number of positions.

```
int x = 5 << 1;  // result is 10 (0101 << 1 = 1010)
```

5. `>>` **(Right shift)**: Shifts the bits to the right by a specified number of positions.

```
int x = 5 >> 1;  // result is 2 (0101 >> 1 = 0010)
```

6. `~` **(NOT)**: Inverts the bits of the operand.

```
int x = ~5;  // result is -6 (inverts the bits of 0101)
```

8.7 Ternary Operator

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

Syntax: `condition ? value_if_true : value_if_false;`

Example:

```
int x = 10;
int result = (x > 5) ? 100 : 200;  // result is 100 because x > 5
```

---

9. Input and Output Operators

Java uses `System.in` for input and `System.out` for output.

Using `System.in` and `System.out`

1. **Output (Printing to the console)**: Use `System.out.println()` to print data to the console.

```
System.out.println("Hello, Java!");  // Prints "Hello, Java!" to the console
```

2. **Input (Reading from the user)**: Use `Scanner` class for reading input from the console.

```
import java.util.Scanner;

Scanner scanner = new Scanner(System.in);
System.out.print("Enter your name: ");
String name = scanner.nextLine();  // Reads a line of text from the user
System.out.println("Hello, " + name + "!");
```

Summary

This guide introduced the key **operators** in Java:

- **Arithmetic Operators** for basic math.
- **Relational Operators** for comparing values.
- **Logical Operators** for working with boolean values.
- **Unary Operators** for incrementing/decrementing values.
- **Assignment Operators** for assigning and updating variables.
- **Bitwise Operators** for performing operations on bits.
- **Ternary Operator** for concise if-else logic.

It also covered **Input and Output operators** like `System.out` and `Scanner` to interact with the user.

These operators and tools are essential for performing operations, managing data, and interacting with the user in Java.
