Operator in Java

Java operators are symbols that are used to perform operations on variables and manipulate the values of the operands. Each operator performs specific operations.

Let us consider an expression 5 + 1 = 6; here, 5 and 1 are **operands,** and the symbol + (plus) is called the **operator**. We will also learn about operator precedence and operator associativity.

Types of Java Operator

- **Arithmetic Operators**
- **Assignment Operators**
- **Logical Operators**
- **Bitwise Operators**
- **Ternary Operators**
- **Relational Operators**

➔ Arithmetic Operators :

Arithmetic operators are used to performing addition, subtraction, multiplication, division, and modulus. It acts as a mathematical operations.

Operator	Description
+ (Addition)	This operator is used to add the value of the operands.
– (Subtraction)	This operator is used to subtract the right-hand operator with the left hand operator.
*** (Multiplication)**	This operator is used to multiply the value of the operands.
/ (Division)	This operator is used to divide the left hand operator with right hand operator.
% (Modulus)	This operator is used to divide the left hand operator with right hand operator and returns remainder.

Example

Let us look at a simple code that in which all the **arithmetic operators** are used.

```
public class ArithmeticOperatorsExample {
   public static void main(String[] args) {
      int num1 = 10, num2 = 5, result;

      // Addition Operator
      result = num1 + num2;
      System.out.println("Addition: " + result);

      // Subtraction Operator
      result = num1 - num2;
      System.out.println("Subtraction: " + result);

      // Multiplication Operator
      result = num1 * num2;
      System.out.println("Multiplication: " + result);

      // Division Operator
      result = num1 / num2;
      System.out.println("Division: " + result);

      // Modulus Operator
      result = num1 % num2;
      System.out.println("Modulus: " + result);

      // Increment Operator
      num1++;
      System.out.println("Increment: " + num1);

      // Decrement Operator
      num2--;
      System.out.println("Decrement: " + num2);
   }
}
```

OUTPUT

```
 Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2
Modulus: 0
Increment: 11
Decrement: 4
```

➔ Assignment Operators :

Assignment operator are used to assign new value to a variable. The left side operand of the assignment operator is called variable and the right side operand of the assignment operator is called value.

Operator	Description
=	This operator is used to assign the value on the right to the operand on the left.

+=	This operator is used to add right operand to the left operand and assigns the result to the left operand.
-=	This operator subtracts right operand from the left operand and assigns the result to the left operand.
*=	This operator multiplies right operand with the left operand and assigns the result to the left operand.
/=	This operator divides left operand with the right operand and assigns the result to the left operand.

Example

Let us look at a simple code in which all the **assignment operators** are used.

```
public class Topperworld {
public static void main ( String[] args ) {
int a = 10;
int b = 20;
int c;
System.out.println ( c = a );
System.out.println ( b += a );
System.out.println ( b -= a);
System.out.println ( b *= a );
System.out.println ( b /= a );
System.out.println ( b ^= a );
System.out.println ( b %= a );
}
}
```

OUTPUT

```
10
30
```

```
10
200
2
0
```

➔ Logical Operators :

Logical operators are used to combining two or more conditions or complement the evaluation of the original condition under consideration.

Operator	Description
&& (Logical AND)	This operator returns True if both the operands are true, otherwise, it returns False.
|| (Logical OR)	This operator returns True if either the operands are true, otherwise it returns False.
! (Logical AND)	This operator returns True if an operand is False. It reverses the logical state of an operand.
&& (Logical AND)	This operator returns True if both the operands are true, otherwise, it returns False.
|| (Logical OR)	This operator returns True if either the operands are true, otherwise it returns False.

Example

Let us look at a simple code that in which all the **logical operators** are used.

```
public class Topperworld {
public static void main ( String args[] ) {
        int a = 5;
        System.out.println ( a<5  &&  a<20 );
        System.out.println ( a<5 || a<20 );
        System.out.println ( ! ( a<5  &&  a<20 ));
}
}
```

OUTPUT

```
False
True
True
```

➔ Bitwise Operators :

The bitwise operator operates on bit string, binary number, or bit array. It is fast and simple and directly supported by the processor. The bitwise operation is also known as bit-level programming.

Operator	Description
& (Bitwise AND)	This operator takes two numbers as operands and does AND on every bit of two numbers.
| (Bitwise OR)	This operator takes two numbers as operands and does OR on every bit of two numbers.
^ (Bitwise XOR)	This operator takes two numbers as operands and does XOR on every bit of two numbers.
~ (Bitwise NOT)	This operator takes one number as an operand and does invert all bits of that number.

& (Bitwise AND)	This operator takes two numbers as operands and does AND on every bit of two numbers.

Example

Let us look at a simple code that in which all the b**itwise operators** are used.

```
public class Topperworld {
   public static void main(String[] args) {
      int num1 = 10, num2 = 5;

      // Bitwise AND Operator
      int result = num1 & num2;
      System.out.println("Bitwise AND: " + result);

      // Bitwise OR Operator
      result = num1 | num2;
      System.out.println("Bitwise OR: " + result);

      // Bitwise XOR Operator
      result = num1 ^ num2;
      System.out.println("Bitwise XOR: " + result);

      // Bitwise Complement Operator
      result = ~num1;
      System.out.println("Bitwise Complement of num1: " + result);

      // Left Shift Operator
      result = num1 << 2;
      System.out.println("Left Shift of num1: " + result);

      // Right Shift Operator
      result = num1 >> 2;
      System.out.println("Right Shift of num1: " + result);

      // Unsigned Right Shift Operator
      result = num1 >>> 2;
      System.out.println("Unsigned Right Shift of num1: " + result);
   }
}
```

OUTPUT

```
Bitwise AND: 0
Bitwise OR: 15
Bitwise XOR: 15
Bitwise Complement of num1: -11
Left Shift of num1: 40
Right Shift of num1: 2
Unsigned Right Shift of num1: 2
```

➔ Ternary Operators :

Ternary operator is an conditional operator, it reduces the line of code while performing the conditional or comparisons. It is the replacement of if-else or nested if-else statements. It is also referred to as inline if, conditional operator, or ternary if..

Example

Let us look at a simple code of **Ternary operator**.

```
public class Topperworld {
public static void main ( String args[] ) {
int a = 4;
int b = 9;
int min = ( a<b ) ? a : b;
System.out.println ( min );
}
}
```

OUTPUT

```
4
```

➔ Relational Operators :

Relational operator compares two numbers and returns a boolean value. This operator is used to define a relation or test between two operands.

Operator	Description
< (Less than)	This operator returns True, if the value of the left operand is less than the value of the right operand, else it returns False.
> (Greater than)	This operator returns True, if the value of the left operand is greater than the value of the right operand, else it returns False.
<= (Less than or equal to)	This operator returns True, if the value of the left operand is less than or equal to the value of the right operand, else it returns False.
>= (Greater than or equal to)	This operator returns True, if the value of the left operand is greater than or equal to the value of the right operand, else it returns False.
== (Equal to)	This operator returns True, if two operands are equal, else it returns False.

Example

Let us look at a simple code that in which all the **realational operators** are used.

```
public class Topperworld {
public static void main ( String args[] ) {
             int  a = 10;
             int  b = 20;
System.out.println ( a < b );
System.out.println(  a > b );
System.out.println ( a <= b );
System.out.println (a >= b );
System.out.println ( a == b );
System.out.println ( a != b );
    }
}
```

OUTPUT

```
true
false
true
false
false
true
```

Java Operator Precedence Table

Category	Operator	Associativity
Postfix	>() [] . (dot operator)	Left toright
Unary	>++ - - ! ~	Right to left
Multiplicative	>* /	Left to right
Additive	>+ -	Left to right
Shift	>>> >>> <<	Left to right
Relational	>> >= < <=	Left to right
Equality	>== !=	Left to right
Bitwise AND	>&	Left to right
Bitwise XOR	>^	Left to right
Bitwise OR	>|	Left to right
Logical AND	>&&	Left to right
Logical OR	>||	Left to right
Conditional	?:	Right to left
Assignment	>= += -= *= /= %= >>= <<= &= ^= |=	Right to left