

Lecture2

Java Operators

Java Operators are the symbol which tells the interpreter or compiler to execute specific task like – arithmetic, relational, logical, etc. to get desired results. Java is a very popular language. There are many features and operators in java which are used to perform different operations.

Today, we come up with the different types of operators with their examples to learn and master the concept quickly.

Types of Java Operators

Java Operators are used for performing many operations. Following are the types of operators available in Java.



1. Arithmetic Operators in Java

Java arithmetic operator takes numerical values as operands, operates on them, and returns a single numerical value. Java Arithmetic operators are used for simple mathematic operations, they are

Addition (+)

Subtraction (-)

Multiplication (*)

Division (/)

Modulus (%)

```
/*
 * Read Me: Arithmetic Operators in Java
 */
package javaapplication15;

public class JavaApplication15 {

    public static void main(String[] args) {
        int num1 = 20, num2 = 10;
        // using +
        int ans1 = num1 + num2;
        System.out.println("Adding(+)= " + ans1);
        System.out.println("Adding(+)= " + (num1 + num2));

        // using -
        int ans2 = num1 - num2;
        System.out.println("Subtracting(-)= " + ans2);
        System.out.println("Subtracting(-)= " + (num1 - num2));

        // using *
        int ans3 = num1 * num2;
        System.out.println("Multiplying(*)= " + ans3);
        System.out.println("Multiplying(*)= " + (num1 * num2));
        // using /
        int ans4 = num1 / num2;
        System.out.println("Dividing(/)= " + ans4);
        System.out.println("Dividing(/)= " + (num1 / num2));
    }
}
```

```

        // modulus operator gives remainder on dividing first operand with second
        System.out.println("Modulus(%)= " + (num1 % num2));
    }
}

```

Output

```

run:
Adding(+)= 30
Adding(+)= 30
Subtracting(-)= 10
Subtracting(-)= 10
Multiplying(*)= 200
Multiplying(*)= 200
Dividing(/)= 2
Dividing(/)= 2
Modulus(%)= 0
BUILD SUCCESSFUL (total time: 0 seconds)

```

2.Unary Operators in Java

Unary Operator operates on a single value. Following are the unary operators:

Operator Name	Description
Unary Minus (-)	For decreasing the value
Unary Plus (+)	For converting the negative values into positive
Increment operator (++)	Used for increasing value of the operand by 1
Post-Increment	The value is incremented first then the result is computed
Pre-Increment	The value is incremented later first the result is computed
Decrement Operator	Used for decreasing the value of operand by 1
Post-Decrement	The value is decremented first then the result is computed
Pre-Decrement	The value is decremented later first the result is computed
Logical not Operator (!)	Used for inverting the values of boolean

```

/*
 * Read Me: Unary Operators in Java
 */

```






```

package javaapplication15;

public class JavaApplication15 {

    public static void main(String[] args) {
        int n1 = 20, n2 = 10, n3 = 0, n4 = 20, n5 = 40;
        boolean condition = true;
        // pre-increment operator
        n3 = ++n1;// operand1 = operand1 + 1
        System.out.println("Value of operand3 (++operand1) = " +
n3);//operand3 = operand1;
        // post increment operator
        n3 = n2++;// operand3 = b
        System.out.println("Value of operand3 (operand2++) = " +
n3);//operand2 = operand2 + 1
        // pre-decrement operator
        n3 = --n4;// operand4 = operand4-1
        System.out.println("Value of operand3 (--operand4) = " +
n3);//operand3=operand4
        // post-decrement operator
        n3 = --n5;// operand3 = operand5
        System.out.println("Value of operand3 (--operand5) = " +
n3);// operand5 = operand5 - 1
        // Logical not operator
        System.out.println("Value of !condition =" + !condition);
    }
}

```

Output - JavaApplication15 (run) ×	HTTP Server Monitor
 run:  Value of operand3 (++operand1) = 21  Value of operand3 (operand2++) = 10  Value of operand3 (--operand4) = 19  Value of operand3 (--operand5) = 39 Value of !condition =false BUILD SUCCESSFUL (total time: 0 seconds)	

3.Assignment Operators in Java

An assignment operator assigns a value to the left operand based on the value of its right operand with the help of equals = sign. Thus, x = y assigns the value of y into x. Its types are,

Operator Name	Description
+=	To add the right and left operator and then assigning the result to the left operator
-=	To subtract the two operands on left and right and then assign the value to the left operand
*=	To multiply the two operands on left and right and then assign the value to the left operand
/=	To divide the two operands on left and right and then assign the value to the left operand
^=	To raise the value of left operand to the power of right operator
%=	To apply modulus operator

```
/*
 * Read Me: simple assignment operator
 */
package javaapplication15;

public class JavaApplication15 {

    public static void main(String[] args) {
        int operand1 = 20, operand2 = 10, operand3 = 10, operand4
= 4;
        // simple assignment operator
        operand3 = operand2;
        System.out.println("Value of operand3 = " + operand3);
        operand1 = operand1 + 1;
        operand2 = operand2 - 1;
        operand3 = operand3 * 2;
        operand4 = operand4 / 2;
        System.out.println("operand1,operand2,operand3,operand4 =
"
+ operand1 + "," + operand2 + "," + operand3 + "," + oper
and4);
        operand1 = operand1 - 1;
```

```

        operand2 = operand2 + 1;
        operand3 = operand3 / 2;
        operand4 = operand4 * 2;
        // shorthand assignment operator
        operand1 += 1;
        operand2 -= 1;
        operand3 *= 2;
        operand4 /= 2;
        System.out.println("operand1,operand2,operand3,operand4 (
using shorthand operators)= "
        + operand1 + "," + operand2 + "," + operand3 + "," + operand4);
    }
}

```

```

Output - JavaApplication15 (run) × HTTP Server Monitor
run:
Value of operand3 = 10
operand1,operand2,operand3,operand4 = 21,9,20,2
operand1,operand2,operand3,operand4 (using shorthand operators)= 21,9,20,2
BUILD SUCCESSFUL (total time: 0 seconds)

```

4. Relational Operators in Java

Relational Java operators are used to check the equality of operands. We can also use them for comparison of two or more values.

Operator Name	Example	Description
== (equals to)	$x == y$	True if x equals y, otherwise false
!= (not equal to)	$x != y$	True if x is not equal to y, otherwise false
< (less than)	$x < y$	True if x is less than y, otherwise false
> (greater than)	$x > y$	True if x is greater than y, otherwise false
>= (greater than or equal to)	$x \geq y$	True if x is greater than or equal to y, otherwise false
<= (less than or equal to)	$x \leq y$	True if x is less than or equal to y, otherwise false

```

/*
 * Read Me: relational operator
 */
package javaapplication15;

public class JavaApplication15 {

    public static void main(String[] args) {
        int num1 = 20, num2 = 10;
        int ar = 5;
        int br = 5;
        boolean condition = true;
        //various conditional operators
        System.out.println("operand1 == operand2 :" + (num1 == num2));
        System.out.println("operand1 < operand2 :" + (num1 < num2));
        System.out.println("operand1 <= operand2 :" + (num1 <= num2));
        System.out.println("operand1 > operand2 :" + (num1 > num2));
        System.out.println("operand1 >= operand2 :" + (num1 >= num2));
        System.out.println("operand1 != operand2 :" + (num1 != num2));
        System.out.println("ar == br : " + (ar == br));
        System.out.println("condition==true :" + (condition == true));
    }
}

```

```
Output - JavaApplication15 (run) X HTTP Server Monitor
operand1 < operand2 :false
operand1 <= operand2 :false
operand1 > operand2 :true
operand1 >= operand2 :true
operand1 != operand2 :true
ar == br : true
condition==true :true
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Logical operators in Java

Operator Name	Description
&& (Logical AND)	Returns the value if both the conditions are true otherwise returns zero.
(Logical OR)	Returns the value even if one condition is true




```
/*
 * Read Me: logical operator
 */
package javaapplication15;

public class JavaApplication15 {

    public static void main(String[] args) {
        int num1 = 20, num2 = 10;
        if (num1 < 50 || num2 > 11) {
            System.out.println("condition 1 true");
        }

        if (num1 < 50 && num2 > 11) {
            System.out.println("condition 2 true");
        } else {
            System.out.println("condition 2 false");
        }
    }
}
```


Output - JavaApplication15 (run) × HTTP Server Monitor

 run:
 condition 1 true
condition 2 false
 BUILD SUCCESSFUL (total time: 0 seconds)
