Java Operators

Topics : JAVA
Written on December 15, 2022

Java Operators are used to perform operations on variables and values.

There are many Java Operators available as below.

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Shift Operator
- Unary Operator
- Ternary Operator

Arithmetic Operators

Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

Operator	Description	Example (Where P=40 and Q=20)
+ (Addition)	Adds values on either side of the operator.	P + Q will give 60
- (Subtraction)	Subtracts right-hand operand from left-hand operand.	P - Q will give -20
* (Multiplication)	Multiplies values on either side of the operator.	P * Q will give 800
/ (Division)	Divides left-hand operand by right-hand operand.	P / Q will give 2
% (Modulus)	Divides left-hand operand by right-hand operand and returns remainder.	Q % P will give 0
++ (Increment)	Increases the value of operand by 1.	Q++ gives 21
-- (Decrement)	Decreases the value of operand by 1.	Q-- gives 19

Relational Operators

Relational operators are used to compare two values or variables.

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Bitwise Operators

Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.

Operator	Description	Example (where P=80 and Q=15)
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.	(P & Q) will give 0
| (bitwise or)	Binary OR Operator copies a bit if it exists in either operand.	(P | Q) will give 95
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.	(P ^ Q) will give 95
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~P) will give -81
<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	P << 2 will give 320
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	P >> 2 will give 20
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	P >>>2 will give 20

Logical Operators

Java logical operators is used to check true or false value.

Operator	Name	Description
&&	Logical and	Returns true if both statements are true
||	Logical or	Returns true if one of the statements is true
!	Logical not	Reverse the result, returns false if the result is true

Assignment Operators

Assignment Operators is used to assign value in different way.

```
=, +=, -=, *=, /=, %=, &=, ^=, |, =, <, <=, >, >=, >>>=
```

Shift Operator (>> or <<)

The Java shift operator is used to shift all of the bits in a value to the left side or right side of a specified number of times.

The Java right shift operator >> is used to move the value of the left operand to right by the number of bits specified by the right operand.

The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.

Unary Operator (++ , --, +, - , ~ and !)

The Java unary operators is used to perform various operations like incrementing/decrementing a value by one, negating an expression and inverting the value of a boolean.

++expression, --expression, +expression, -expression, ~variable, !variable

Ternary Operator(? :)

Ternary operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide, which value should be assigned to the variable.

```
variable p = (expression) ? value if true : value if false
```

Example :

```
public class Aryatechno {

public static void main(String args[]) {
int P = 80;
int Q = 15;
int c = 0;

c = P & Q;
```

```
System.out.println("P & Q = " + c );

c = P | Q;
System.out.println("P | Q = " + c );

c = P ^ Q;
System.out.println("P ^ Q = " + c );

c = ~P;
System.out.println("~P = " + c );

c = P << 2;
System.out.println("P << 2 = " + c );

c = P >> 2;
System.out.println("P >> 2 = " + c );

c = P >>> 2;
System.out.println("P >>> 2 = " + c );
}
}
```

Output :

```
//Bitwise Operators
P & Q = 0
P | Q = 95
P ^ Q = 95
~P = -81
P << 2 = 320
P >> 2 = 20
P >>> 2 = 20
```