**Operator In JAVA**

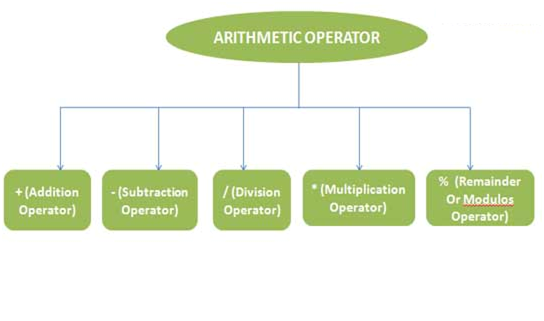
In computer programming Operator is a symbol that tells the compiler to perform specific action which can be mathematical or logical.

Types Of Operator

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

**ARITHMETIC OPERATOR:**

* There are 5 types of arithmetic operator in JAVA: addition (+), subtraction (-), division (/), multiplication (\*) and remainder (%) operator.



**Example**

**public** **class** ArithmaticOp {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** x=20,y=10;

System.***out***.println("Addition is = "+(x+y));//A

System.***out***.println("Substraction is = "+(x-y));

System.***out***.println("Multiplication is = "+(x\*y));

System.***out***.println("Divide is = "+(x/y));

System.***out***.println("Remainder is = "+(x%y));

}

}

**Output-**Addition is = 30

Substraction is = 10

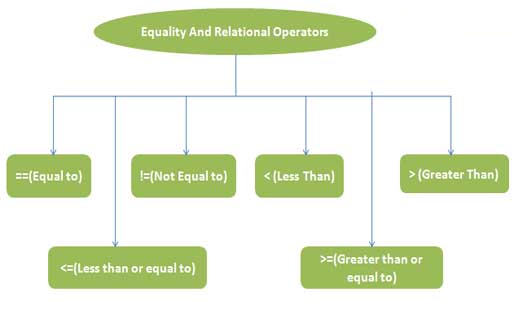
Multiplication is = 200

Divide is = 2

Module is = 0

**The Relational Operators**

Equality and relational can be considered as relationship operator because there work is to compare the first operand to the second operand testing the validity of the specified relationship between them

****

**Example**

**public** **class** RelationOp {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** x=20,y=10,z=20;

**if**(x==z)

{

System.***out***.println("X and Y are Equal");

}

**if**(x!=y)

{

System.***out***.println("X and Y are not Equal");

}

**if**(x>y)

{

System.***out***.println("X is greater than Y");

}

**if**(y<x)

{

System.***out***.println("Y is smaller than X");

}

**if**(x>=y)

{

System.***out***.println("X is greater than Y");

}

**if**(y<=x)

{

System.***out***.println("Y is smaller than X");

}

}

}

**Output**

X and Y are Equal

X and Y are not Equal

X is greater than Y

Y is smaller than X

X is greater than Y

Y is smaller than X

#### Bitwise operators

Java provides Bit wise operators to manipulate the contents of variables at the bit level.

These variables must be of numeric data type ( char, short, int, or long). Java provides seven bitwise

operators. They are AND, OR, Exclusive-OR, Complement, Left-shift, Signed Right-shift, and Unsigned Right-shift.

|  |  |
| --- | --- |
| **operator** | **description** |
| & | Bitwise AND |
| | | Bitwise OR |
| ^ | Bitwise exclusive OR |
| << | left shift |
| >> | right shift |

Now lets see truth table for bitwise

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **~A** | **A & B** | **A | B** | **A ^ B** |
| **1** | **1** | 0 | 1 | 1 | 0 |
| **1** | **0** | 0 | 0 | 1 | 1 |
| **0** | **1** | 1 | 0 | 1 | 1 |
| **0** | **0** | 1 | 0 | 0 | 0 |

**Example**

**public** **class** BitwiseOperatorsDemo {

**public** BitwiseOperatorsDemo() {

**int** x = 0xFAEF; //1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1

**int** y = 0xF8E9; //1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1

**int** z; System.***out***.println("x & y : " + (x & y));

System.***out***.println("x | y : " + (x | y));

System.***out***.println("x ^ y : " + (x ^ y));

System.***out***.println("~x : " + (~x));

System.***out***.println("x << y : " + (x << y));

System.***out***.println("x >> y : " + (x >> y));

System.***out***.println("x >>> y : " + (x >>> y));

//There is no unsigned left shift operator

}

**public** **static** **void** main(String[] args) {

**new** BitwiseOperatorsDemo();

}

} ***Output***

x & y : 63721

x | y : 64239

x ^ y : 518

~x : -64240

x << y : 32890368

x >> y : 125

x >>> y : 125

#### Assignment Operators

Assignment operator supported by Java are as follows

|  |  |  |
| --- | --- | --- |
| **operator** | **description** | **example** |
| = | assigns values from right side operands to left side operand | a=b |
| += | adds right operand to the left operand and assign the result to left | a+=b is same as a=a+b |
| -= | subtracts right operand from the left operand and assign the result to left operand | a-=b is same as a=a-b |
| \*= | mutiply left operand with the right operand and assign the result to left operand | a\*=b is same as a=a\*b |
| /= | divides left operand with the right operand and assign the result to left operand | a/=b is same as a=a/b |
| %= | calculate modulus using two operands and assign the result to left operand | a%=b is same as a=a%b |

#### Logical operators

Logical operators return a true or false value based on the state of the Variables. There are six logical, or boolean, operators. They are AND, conditional AND, OR, conditional OR, exclusive OR, and NOT. Each argument to a logical operator must be a boolean data type, and the result is always a boolean data type

Java supports following 3 logical operator. Suppose a=1 and b=0;

|  |  |  |
| --- | --- | --- |
| **operator** | **description** | **example** |
| && | Logical AND | (a && b) is false |
| || | Logical OR | (a || b) is true |
| ! | Logical NOT | (!a) is false |

**Example**

**public** **class** LogicalOperatorsDemo {

**public** LogicalOperatorsDemo() {

**boolean** x = **true**;

**boolean** y = **false**;

System.***out***.println("x & y : " + (x & y));

System.***out***.println("x && y : " + (x && y));

System.***out***.println("x | y : " + (x | y));

System.***out***.println("x || y: " + (x || y));

System.***out***.println("x ^ y : " + (x ^ y));

System.***out***.println("!x : " + (!x));

}

**public** **static** **void** main(String args[]) {

**new** LogicalOperatorsDemo();

}

}

**Output**

x & y : false

x && y : false

x | y : true

x || y: true

x ^ y : true

!x : false

**Java Unary Operator Example: ++ and - -**

**Example**

class UnaryOp{

public static void main(String args[]){

int x=10;

System.out.println(x++);//10 (11)

System.out.println(++x);//12

System.out.println(x--);//12 (11)

System.out.println(--x);//10

}

}

**Output**

10

12

12

10

**Java Left Shift,Right Operator**

**Example**

*class OperatorExample{*

*public static void main(String args[]){*

*System.out.println(10<<2);//10\*2^2=10\*4=40*

*System.out.println(10<<3);//10\*2^3=10\*8=80*

*System.out.println(20<<2);//20\*2^2=20\*4=80*

*System.out.println(15<<4);//15\*2^4=15\*16=240*

System.out.println(20>>2);//20/2^2=20/4=5

System.out.println(20>>3);//20/2^3=20/8=2

*}}*

**Output:**

**40**

**80**

**80**

**240**

**5**

**2**

### Java Ternary Operator Example

Ternary operators in java works like if else control. In comparison to if else, ternary operator is easy to use**.** By using ternary operators you can check condition. The portion after “?” is true part portion that will execute when given condition is true and portion after “:“ will execute if condition is false.

**Syntax:**

(Condition)?True:false;

**Example**

*class OperatorExample{*

*public static void main(String args[]){*

*int a=2;*

*int b=5;*

*int min=(a<b)?a:b;*

*System.out.println(min);*

*}}*

Output:

2