Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:

* Arithmetic Operators
* Relational Operators
* Bitwise Operators
* Logical Operators
* Assignment Operators
* Misc Operators

The Arithmetic Operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:

Assume integer variable A holds 10 and variable B holds 20 then:

[Show Examples](http://www.tutorialspoint.com/java/java_arithmatic_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Addition - Adds values on either side of the operator | A + B will give 30 |
| - | Subtraction - Subtracts right hand operand from left hand operand | A - B will give -10 |
| \* | Multiplication - Multiplies values on either side of the operator | A \* B will give 200 |
| / | Division - Divides left hand operand by right hand operand | B / A will give 2 |
| % | Modulus - Divides left hand operand by right hand operand and returns remainder | B % A will give 0 |
| ++ | Increment - Increase the value of operand by 1 | B++ gives 21 |
| -- | Decrement - Decrease the value of operand by 1 | B-- gives 19 |

The Relational Operators:

There are following relational operators supported by Java language

Assume variable A holds 10 and variable B holds 20 then:

[Show Examples](http://www.tutorialspoint.com/java/java_relational_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Checks if the value of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the value of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

The Bitwise Operators:

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

Bitwise operator works on bits and perform bit by bit operation. Assume if a = 60; and b = 13; Now in binary format they will be as follows:

a = 0011 1100

b = 0000 1101

-----------------

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a  = 1100 0011

The following table lists the bitwise operators:

Assume integer variable A holds 60 and variable B holds 13 then:

[Show Examples](http://www.tutorialspoint.com/java/java_bitwise_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & | Binary AND Operator copies a bit to the result if it exists in both operands. | (A & B) will give 12 which is 0000 1100 |
| | | Binary OR Operator copies a bit if it exists in eather operand. | (A | B) will give 61 which is 0011 1101 |
| ^ | Binary XOR Operator copies the bit if it is set in one operand but not both. | (A ^ B) will give 49 which is 0011 0001 |
| ~ | Binary Ones Complement Operator is unary and has the effect of 'flipping' bits. | (~A ) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number. |
| << | Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand. | A << 2 will give 240 which is 1111 0000 |
| >> | Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand. | A >> 2 will give 15 which is 1111 |
| >>> | 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. | A >>>2 will give 15 which is 0000 1111 |

The Logical Operators:

The following table lists the logical operators:

Assume boolean variables A holds true and variable B holds false then:

[Show Examples](http://www.tutorialspoint.com/java/java_logical_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Called Logical AND operator. If both the operands are non zero then then condition becomes true. | (A && B) is false. |
| || | Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. | !(A && B) is true. |

The Assignment Operators:

There are following assignment operators supported by Java language:

[Show Examples](http://www.tutorialspoint.com/java/java_assignment_operators_examples.htm)

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Simple assignment operator, Assigns values from right side operands to left side operand | C = A + B will assigne value of A + B into C |
| += | Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand | C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand | C -= A is equivalent to C = C - A |
| \*= | Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand | C \*= A is equivalent to C = C \* A |
| /= | Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand | C /= A is equivalent to C = C / A |
| %= | Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand | C %= A is equivalent to C = C % A |
| <<= | Left shift AND assignment operator | C <<= 2 is same as C = C << 2 |
| >>= | Right shift AND assignment operator | C >>= 2 is same as C = C >> 2 |
| &= | Bitwise AND assignment operator | C &= 2 is same as C = C & 2 |
| ^= | bitwise exclusive OR and assignment operator | C ^= 2 is same as C = C ^ 2 |
| |= | bitwise inclusive OR and assignment operator | C |= 2 is same as C = C | 2 |

Precedence of Java Operators:

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:

For example x = 7 + 3 \* 2; Here x is assigned 13, not 20 because operator \* has higher precedence than + so it first get multiplied with 3\*2 and then adds into 7.

Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedenace operators will be evaluated first.

|  |  |  |
| --- | --- | --- |
| **Category** | **Operator** | **Associativity** |
| Postfix | () [] . (dot operator) | Left to right |
| 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 |
| Comma | , | Left to right |