**Operators**

**Arithmetic operators**

Arithmetic operators are used in mathematical expressions, which have the same functions in math. All arithmetic operators are listed in below table. In the instance of the table, the integer variable of A is assumed as 10, and the variable B as 20:

| **Operators** | **Descriptions** | **Instances** |
| --- | --- | --- |
| + | addition - the values on two sides of the operator are added | A + B equals 30 |
| - | subtraction - the operating value on the left is subtracted by the operatin value on the right | A – B equals -10 |
| \* | multiplication - the values on the two sides of the operator are multiplied | A \* B equals 00 |
| / | Division - The left operating value is divided by the right operating value | B / A equals 2 |
| ％ | remainder - the remainder left after the left operating value is divided by the right operating value | B%A equals 0 |
| ++ | auto increment: the operating value is added by 1 | B++ or ++B equals 21 |
| -- | auto decrement: the operating value is subtracted by 1 | B-- or --B equals 19 |

**Operators of auto increment and auto decrement**

b++ has executed auto inrement operation to the variable of b. As a result, the variable value is added by 1, the value of b++ expression remains the same, ++b has executed auto inrement operation to the variable of b. As a result, the variable value is added by 1, and the value of b++ expression is also added by 1.

**Instance**

**public** **class** **TestArithmeticOperator** {

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

**int** a = 10;

**int** b = 30;

System.**out**.println("a++ = " + (a++) );

System.**out**.println("a =" + (a));

System.**out**.println("a-- = " + (a--) );

System.**out**.println("a =" + (a));

*// compare the difference in b++ and ++b*

System.**out**.println("b++ = " + (b++) );

System.**out**.println("b =" + (b));

System.**out**.println("++b = " + (++b) );

System.**out**.println("b =" + (b));

}

}

The operatin results for above instance is as follows:

a++ = 10

a =11

a-- = 11

a =10

b++ = 30

b =31

++b = 32

b =32

**Relational operators**

Below table lists the relational operators that Java supports

For the instance in the table, the value of the integer variable A is 10, and variable B is 20:

| **Operators** | **Description** | **Instance** |
| --- | --- | --- |
| == | check if the values of two operands are same. If same, the condition is true. | (A == B）is false (untrue). |
| != | check if the values of two operands are same. If no thesame, the condition is true. | (A != B) is true. |
| > | check if the value of the left operand is bigger than that of the right operand. If yes, the condition is true. | (A> B）is untrue. |
| < | check if the value of the left operand is smaller than that of the right operand. If yes, the condition is true. | (A <B）is true. |
| >= | check if the value of the left operand is bigger than or the same with that of the right operand. If yes, the condition is true. | (A> = B）is false. |
| <= | check if the value of the left operand is smaller than or the same with that of the right operand. If yes, the condition is true. | (A <= B）is true. |

**Logic operators**

Below is the basic operations with logic operators. Suppose Boolean variable A is true and variable B is false

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| && (logical and) | Called Logical AND operator. If both the operands are non-zero, then the condition becomes true. | (A && B) is false |
| || (logical or) | Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true. | (A || B) is true |
| ! (logical not) | 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 |

**Assignment operators**

Below are the assignment operators that supported by Java:

| **Operators** | **Description** | **Instance** |
| --- | --- | --- |
| = | Simple assignment operators, to assign the value of right operand to the left operand | C = A + B will assign the result of A + B to C |
| += | Addition assignment operators, which assign the addition result of the left and right operands to the left operand | C += A equals C = C + A |
| -= | Subtraction assignment operators, which assign the subtraction result of the left and right operands to the left operand | C -= A equals C = C - A |
| \*= | Multiplication assignment operators, which assign the multiplication result of the left and right operands to the left operand | C *= A equals C = C*A |
| /= | Division assignment operators, which assign the division result of the left and right operands to the left operand | C /= A equals C = C / A |
| (％)= | Module assignment operators, which assign the module operation result of the left and right operands to the left operand | C ％= A equals C = C ％ A |

**Priority of operatorrs**

When there are multiple operators in a single expression, which one should be processed first? To solve this problem, we should deal with the priority of operators. In an expression with multiple operators, the change of priorities may bring totally different results.

In below table the operators are listed in descending order from top down, with the highest priority level listed on the top.

| **Category** | **Operator** | **Associativity** |
| --- | --- | --- |
| Postfix | expression++ expression-- | Left to right |
| Unary | ++expression –-expression +expression –expression ~ ! | Right to left |
| Multiplicative | \* / % | Left to right |
| Additive | + - | Left to right |
| Shift | << >> >>> | Left to right |
| Relational | < > <= >= instanceof | 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 |

**Reference material**

[JAVA - Basic operators](https://www.tutorialspoint.com/java/java_basic_operators.htm)