Simple Java Maths

In this lesson, an introduction of the basic operators used in Java. For example, subtraction, addition, division, and multiplication are explained.

We'll cover the following

Introduction

Math in Java is very simple. Keep in mind that Java mathematical operations follow a particular order much the same as high school math.

For example, multiplication and division take precedence over addition and subtraction. The *order* in which these operations are evaluated can be changed using *parentheses*.

Operations in Java

The arithmetic operators in Java are listed below.

Symbols	Arithmetic operators
+	add
-	subtract
/	divide
*	multiply

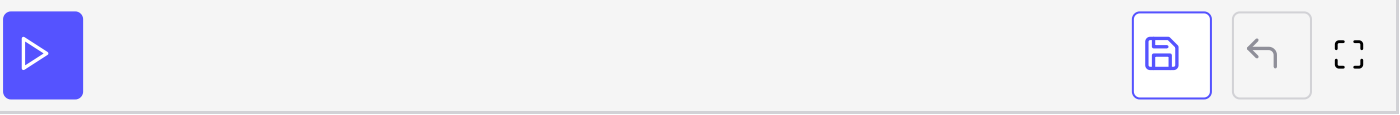
	multiply
%	modulus division
++	post and pre-increment
--	post and pre-decrement

Example

Let’s take a look at how to use these operations while coding in Java.

```
public class Operators {
    public static void main(String[] args) {
        int answer;

        System.out.println("ADDITION");
        int add = 20;

        System.out.println("Initial value: " + add);
        answer = add + 2;
        System.out.println("add + 2 = " + answer);
        answer = add;
        System.out.println("add = " + answer);
        System.out.println();

        System.out.println("SUBTRACTION");
        int sub = 15;

        System.out.println("Initial value: " + sub);
        System.out.println("sub - 4 = " + (sub - 4));
        System.out.println("sub = " + sub);
        System.out.println();

        System.out.println("MULTIPLICATION");
        int mult = 25;

        System.out.println("Initial value: " + mult);
        answer = mult * 3;
        System.out.println("mult * 3 = " + answer);
        answer = mult;
        System.out.println("mult = " + mult);
        System.out.println();

        System.out.println("DIVISION (INT)");
        int div_int = 15;

        System.out.println("Initial value: " + div_int);
        System.out.println("div_int / 2 = " + (div_int / 2));
        System.out.println("div_int = " + div_int);
        System.out.println();

        System.out.println("MODULO (REMINDER)");
```

```
        int rem = 5;

        System.out.println("Initial value: " + rem);

        answer = rem % 2;
        System.out.println("rem % 2 = " + answer);
        answer = rem;
        System.out.println("rem = " + answer);
        System.out.println();

        System.out.println("PREINCREMENT BY ONE");
        int pre_inc = 5;

        System.out.println("Initial value: " + pre_inc);
        System.out.println("++pre_inc   = " + (++pre_inc));
        System.out.println("pre_inc = " + pre_inc);
        System.out.println();

        System.out.println("PREDECREMENT BY ONE");
        int pre_dec = 5;

        System.out.println("Initial value: " + pre_dec);
        answer = --pre_dec;
        System.out.println("--pre_dec   = " + answer);
        answer = pre_dec;
        System.out.println("pre_dec = " + answer);
        System.out.println();

        System.out.println("POST INCREMENT BY ONE");
        int post_inc = 5;

        System.out.println("Initial value: " + post_inc);
        System.out.println("post_inc++   = " + (post_inc++));
        System.out.println("post_inc = " + post_inc);
        System.out.println();

        System.out.println("POSTDECREMENT BY ONE");
        int post_dec = 5;

        System.out.println("Initial value: " + post_dec);
        answer = post_dec--;
        System.out.println("post_dec--   = " + answer);
        answer = post_dec;
        System.out.println("post_dec = " + answer);
        System.out.println();
    }
}
```



Operator precedence

Operator precedence specifies the order in which operations will execute provided that the expression contains more than one operator. With mathematical operations, the precedence follows the rule of BODMAS, which says; Brackets, Order, Division, Multiplication, Addition, and then Subtraction. Let's look at an example below to check this!

example below to check this!

For the purpose of these examples, we will set the following variables; **x=5**, **y=10**, **z=7**, and **w=6**. Now let's evaluate the expressions below using these variables.

- `(x - z) + y * y`
    - The part that will be evaluated first is `(x - z)` as it is in the brackets and will come out to be *-2*
    - The second part that will be calculated is `y * y` as BODMAS ensures that multiplication is done before addition, giving us the answer *100*
    - The final step will be `-2 + 100` giving us the answer *98*
- `w / y + (x * z)`
    - The first step to be calculated is `x * z`, giving us the answer, *35*
    - The second step is the division, `w / y`. Keep in mind that since the variables are assumed to be integers, the result will also be an integer. Hence the answer will be *0* (an int data type rounds to the lower number)
    - The last step will be the addition of the two answers `0 + 35` giving us the result *35*

Now, let's look at the code below to see if this actually works!

```
public class Oper_Prec {
    public static void main(String[] args) {
        int x = 5;
        int y = 10;
        int z = 7;
        int w = 6;
        int answer;

        System.out.println("x: " + x);
        System.out.println("y: " + y);
        System.out.println("z: " + z);
        System.out.println("w: " + w);

        System.out.println("Calculating Expressions with Multiple Operators");

        System.out.println("(x-z) + y*y = " + ((x - z) + y * y));
        answer = (w / y + (x * z));

        System.out.println("w/y + (x*z) = " + answer);
    }
}
```

Operator associativity

Operator associativity determines whether, in an expression, if there are multiple operators like (`1` + `2` - `5`), how will they be evaluated if they are of the same precedence. The addition `+` and subtraction `-` have left associativity.

- `x + y - z`
  - The part `x + y` will be evaluated first, which will give us *15*
  - Then it will calculate `15 - 7` and hence give the final answer *8*

RUN the code given below and see the output!

```
public class Oper_Prec {
    public static void main(String[] args) {
        int x = 5;
        int y = 10;
        int z = 7;
        int w = 6;
        int answer;

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

        System.out.println("Calculating Expression containing operators the with same precedence")

        answer = x+y-z;

        System.out.println("x+y-z = " + answer);
    }
}
```

The following table shows the associativity of **Java** operators (from highest to lowest precedence).

Operators	Description	Associativity
+ , -	Unary Plus and minus	Right to left
! , ~	Logical NOT and bitwise NOT	Right to left

=	Direct assignment	Right to left
+= , -=	Assignment by sum and difference	Right to left
*= , /= , %=	Assignment by product, quotient, and remainder	Right to left
<<= , >>=	Assignment by bitwise left shift and right shift	Right to left
&= , ^= , =	Assignment by bitwise AND, XOR, and OR	Right to left
++ , –	Suffix/postfix increment and decrement	Right to left
* , / , %	Multiplication, division, and remainder	Left to Right
+ , -	Addition and subtraction	Left to Right
<< , >>	Bitwise left shift and right shift	Left to Right
< , <=	For relational operators	Left to Right
> , >=	For relational operators	Left to Right
== , !=	For relational	Left to Right
&	Bitwise AND	Left to Right

^	Bitwise XOR (exclusive or)	Left to Right
&&	Logical AND	Left to Right
||	Logical OR	Left to Right
|	Bitwise OR (inclusive or)	Left to Right

Going good so far? Then let's move on to the next lesson for more interesting operations that you can play around within Java!