Arithmetic Operators:

$(+, -, *, /, \%)$

Integer Arithmetic : (integer o integer) $\rightarrow$ integer
Real   ,,   : (real o real) $\rightarrow$ real
Mixed-mode   ,,   : (integer o real) $\rightarrow$ real

Logical Operators:

$(\&\&, ||, !)$

Truth table for above three logical operators.

Assignment Operator:

$(=)$

$[V \ op= exp;] = [v = v \ op \ (exp);]$

Increment & Decrement Operators:

$(++, --)$

Conditional Operators:

$[exp1 ? exp2 : exp3]$

Bitwise Operators:

$[\&, |, \wedge, \sim, <<, >>, >>>]$

This operators may not be applied to float/double.

Relational Operators:

$[<, <=, >, >=, ==, !=]$

Precedence and order of evaluation of operators:

Operators	Associativity
( ) [ ] → .	left to right
!, ~, ++, --, +, -, *, &, (type), sizeof	Left to right
*, /, %	left to right
+, -	
<<, >>	
<, <=, >, >=	
==, !=	
^	
| (Bitwise inclusive OR)	
&&	
||	
?:	right to left
=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=	right to left
,	left to right

Input from Keyboard :

```java
import java.util. Scanner;
public class demo {
        PSVM (String arr[]){
                Scanner  sc= new  Scanner (System.in);
                int a = sc.nextInt();
                SOP(a);
            }
}
```

```
int  a= sc.nextInt();
double a = sc.nextDouble();
String  a =   sc.next();  → Reads till space (ie. only one word)
            =  sc.nextLine(); → Reads till  next line
char  a= sc.next().charAt(0);
```

Short Circuting : (Optional topic)

a	‖	b	‖	C			a	&&	b	&&	C	
T		x		x	→ T		F		x		x	→ F
F		T		x	→ T		T		F		x	→ F
F		F		T	→ T		T		T		F	→ F
F		F		F	→ F		T		T		T	→ T

Command Line Arguments:

```
psvm ( String arr[] ) {
        int  a =  Integer. parseInt (arr[0]);
        // First argument will store in 'a'.
        int  b =  Integer. parseInt (arr[1]),
        // Second argument will store in 'b'.

        SOP (a + b);   // arr[0] + arr[1]
}
```

NOTE :

Parse : It is a method which take a String (input) as an argument and Convert in other formats as like:

```
Integer. parseInt() ;
Double. parseDouble() ;
Float. parseFloat();
```