OPERATOR'S IN JAVA.

1. **What is Operator.**

- This indicate that actions to perform like addition, subtraction etc.

2. **What is Operand.**

- This indicates that the items, on which action has to apply on.

3. **What is Expression.**

- It consisit of 1 or more Operand and 0 or more oprtators.

4. **Categories Of Operator**.

- **1. Arthmetic Operators**

  - / [Division]
  - - [Substraction]
  - + [Addition]
  - * [Multiplication]

- **2. Relational Operators.**

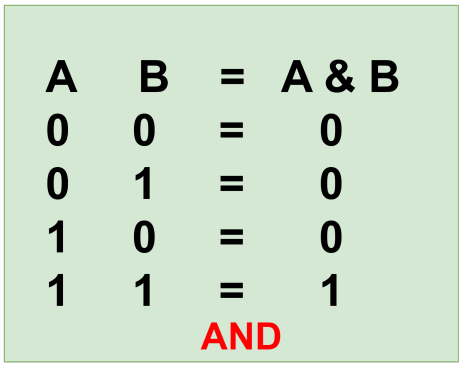
[Compress two Operand relation and return true or fasle]

  - == [Equal to]
  - != [Not Equal to]
  - > [Greater than]
  - < [Less than]
  - >=[Greater than or equal to]
  - <=[Less than or equals to]

- **3. Logical Operators.**

[Combines two or more conditions and return true or false.]

  - && [Logical And]



|| [Logical OR]

A	B	=	A || B
0	0	=	0
0	1	=	1
1	0	=	1
1	1	=	1

OR

- **4. Unary Operators.**

[Requires Only Single Operand]

- ++[Increment]
- -- [Decrement]
- - [Unary Minus]
- + [Unary plus]
- ! [Logical NOT]

- **5. Assignment Operators.**

[Used to assign value to the variable]

- =, +=, -=, *=, /=, %=

- **6. Bitwise Operators.**

[It works on bit i.e. 1 and 0 and its very fast]

- **& [Bitwise AND]**

A	B	=	A & B
0	0	=	0
0	1	=	0
1	0	=	0
1	1	=	1

AND

- **| [Bitwise OR]**

A	B	=	A | B
0	0	=	0
0	1	=	1
1	0	=	1
1	1	=	1

OR

 ^ [Bitwise XOR]

A	B	=	A ^ B
0	0	=	0
0	1	=	1
1	0	=	1
1	1	=	0

XOR

If Both operands is asme it will return 0 otherwise it will return 1

0 0 = 0

1 0 = 1

 ~ [Bitwise NOT]

 [It can be come under Unary too]

```
public class Main{
  public static void main(String[] args){
    int a=4;
    int b=5;
    //Bitwise AND
    //4 = 0 1 0 0
    //5 = 0 1 0 1
    //& = 0 1 0 0 => 4
    System.out.println(a & b);
    //Bitwise OR
    //4 = 0 1 0 0
    //5 = 0 1 0 1
    //| = 0 1 0 1 => 5
    System.out.println(a | b);
    //Bitwise ^
    //both are same -> 0
    //both are diff -> 1
    //4 = 0 1 0 0
    //5 = 0 1 0 1
    //^ = 0 0 0 1 => 14
    System.out.println(a ^ b);
```

```
      //Bitwise ~
      //Bitwise NOT.
      //bit wise complement of any integer n is -(n+1)
      // a=4 => -(a+1) => -(4+1) = (-5)
      //MSB - > MOST SIGINIFICANT BIT USED AS SIGNED BIT
      // 0  - > NEGATIVE
      // 1  - > POSITIVE
      //LSB - > LEAST SIGINIFICANT BIT
      // 128 64 32 16 8 4 2 1
      //        4 -> 0 1 0 0
      //            MSB    LSB
      //taking 1's complement
      //4 - > 0 1 0 0 => 1 0 1 1
      //convert to decimal
      //In java MSB Matter's when it is 0 then 2^(-negative power) and for 1 it is +ve
      //1 0 1 1 => 1*2^(-3) + 0*2^(2) + 1*2^(1) + 1*2^(0)
      //=>(-8)+0+2+1 = (-5)
      System.out.println(!b);
    }
  }
  output:
  4
  5
  14
  -5
```

int a = 4;
0100 [Remeber it's not eqal to 100 as int is a signed as it represent both positive and negative value and in java there is nothing like unsinged integer like in c++]
0100 ====> ~0100 = 1011
Now what is 1011?
1*2^(-3) + 0*2^(2) + 1*2^(1) + 1*2^(0) = -5
which is equivalent to **-(N+1) = -(4+1) = (-5)**
How can we confirm if **-5 is 1011** ?

5=0101
to get the **-5** , we know that we have to find it **2nd complement.**

0101 =>
1st complement => 1010
2nd complement => **1010 + 1 = 1011 = (-5)**

- **7. Bitwise Shift Operators.**

[Used to shift the bits of a number left or right]

- << [Signed Left Shift]
- >> [Signed Right Shift]
- ^ [Bitwise XOR]
- >>> [Unsigned Right Shift]
  - MSB IS ALWAYS 0 AFTER SHIFTING

int a = -5;
int t = a>>>1;

-5 = > 1 0 1 1
=> 1 0 1 1
1 0 1
=> 0 1 0 1=5;

[There is no <<< (Unsigned Left shift as {**<< and <<< are equal**}]

```
public class Main{
   public static void main(String[] args){
       int a=4;
       int b=-5;//1011
      //left shift
      //After left shifting MSB 0 WILL be discarded and left over LSB will filled with 0
      // 16 8 4 2 1
      // 0  0 1 0 0 => 4
      //left shift by 1 bit
      // 32 16 8 4 2 1
      // 0  0  1 0 0 0 <- LSB Always filed with 0
      // 0  0  1 0 0 0 => 8
      System.out.println(a<<1);

      // 16 8 4 2 1
      // 0  0 1 0 0 => 4
      //left shift by 2 bit
      // 32 16 8 4 2 1
      // 0  0  1 0 0 0 <- LSB Always filed with 0
      // 0  0  1 0 0 0 => 8
      //Again Shift 1 bit
      //64 32 16 8 4 2 1
      //0  0  1  0 0 0 0 <- LSB Always filed with 0
      //0  0  1  0 0 0 0 =>16
      System.out.println(a<<2);

      //Signed right shift
      //right shift by 1

      // 1 0 1 1 => -5
      //  1 0 1
      //HERE LSB will be discarded
```

```
        //Here MSB will vaccant but in it always Signed
        //MSB WILL COME Always from above number MSB Like for -5 MSB WILL Always 1
        //So, after right shift MSB always replaced by previous MSB OF NUMBER.
        //Here it will be like => 1 1 0 1
        // 1*2^(-3) + 1*2^(2) + 0*2^(1) + 1*2^(0)
        // -8 + 4 + 0 + 1 = (-3)
        System.out.println(b>>1);

       //  System.out.println(a>>2);

    }
}
```

- **8. Ternary Operators.**

[It mimics the if else conditions]
[condition] ? (expression 1) : (expression2)

```
public class Main{
  public static void main(String[] args){
    int a= 4;
    int b=9;
    String str = a>b ? "lola" : "pola"
    System.out.println(str)
  }
}
output:
pola
```

- **9. Type comparision Operator.**

[It is used to do the type check, whether particular object is of certan class or not]
InstanceOf (Sometimes also shown under elational Operator list)

```
public class Parent{
}
```

```
public class Child1 extends Parent{
}
```

```
public class Child2 extends Parent{
}
```

```
public class Main{
  public static void main(String[] args){
     Parent obj = new Child2();
     System.out.println(obj instanceOf Child2)
     System.out.println(obj instanceOf Child1)
  }
}
output:
true
false
```

Operators	Precedence	Associativity
Parentheses	{},[],()	Left to right
Unary:Postfix	expr++,expr--	Left to right
Unary:Prefix	++exp, --exp	Righ to Left
Multiplicative	*, /, %	Left to right
Additive	+, -,	Left to right
Bitwise 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
LLogial OR	||	Left to right
Ternary	?:	Right to Left
Assignment	=, +=, -=, *=, /=, %=, &=, ^=, !=, <<=, >>=, >>>=	Right to Left

a=b=c

- c value assign to b first.

- after that b value assign to a

int a = 4, 5,6,5,4
int d = a + a++ + ++a * --a + a--;
4 + 4 +6 * 5 + 5
8 + 30 +5 = 43
a=4;

int x =2; ,1 ,2
z = (++x / (x++ - 1))
(1 / (1-1)) = 1/0 =infinity