

## Operators

- In java we have few symbols and pre-defined keywords which behave like operators,
- operator is used to perform a specific task on the given data(literals)
- The data provided to the operator is known as **operand**
- Every operator returns data back. ( result of the operator, therefore every operator will have a return type.)

Note :

- Operators in java can be classified based on the number of operands.

1. unary operator
2. binary operator
3. ternary operator

### 1. unary operator :

An operator which can accept only **one** operand as input is known as unary operator.

### 2. Binary operator :

An operator which can accept **two** operands is known as binary operator.

### 3. Ternary Operator :

An operator which can accept **three** operands is known as ternary operator.

- We can also classify operators based on the task that they perform

1. Arithmetic Operators
2. Assignment operators

3. Increment & decrement operators
4. Relational operators
5. logical operators
6. Miscellaneous



## 1. Arithmetic Operators :

➤ Arithmetic operators is binary operator.

➤ Operators :

1. + ( addition & concat)
2. - ( subtraction )
3. \* (multiplication)
4. / (division)
5. % (modulus)

1. + :

It is a binary operator.

LHS	RHS	Action	Result
int	int	addition	int
double	double	addition	double
char	char	addition	int
boolean	boolean	CTE	-
String	String	Concat	String

$$\underbrace{10}_{\text{int}} + \underbrace{10}_{\text{int}} \Rightarrow \underbrace{20}_{\text{int}}$$

$$\underbrace{10.0}_{\text{double}} + \underbrace{11.1}_{\text{double}} \Rightarrow \underbrace{21.1}_{\text{double}}$$

$$\underbrace{'a'}_{\text{char}} + \underbrace{'b'}_{\text{char}} \Rightarrow \underbrace{97}_{\text{int}}$$

$$\underbrace{\text{true}}_{\text{boolean}} + \underbrace{\text{true}}_{\text{boolean}} \Rightarrow$$

$$\underbrace{\text{"Hello"}}_{\text{String}} + \underbrace{\text{"baby"}}_{\text{String}} = \underline{\hspace{2cm}}$$

Note:

> If The LHS & RHS is not of same type, then compiler tries to perform widening into the larger type. The result of the expression will be same as the larger datatype.

LHS	RHS	Action	Result
int	char	addition	int

$$\underbrace{10}_{\text{int}} + \underbrace{'A'}_{\text{char}} = \underbrace{75}_{\text{int}}$$

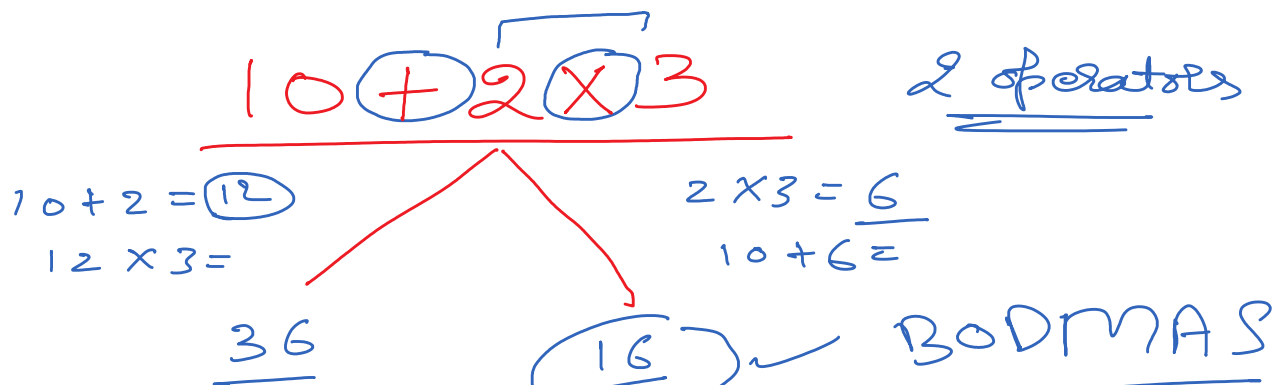
int	double	addition	double
char	double	addition	double
int	boolean	CTE	-
String	int	Concat	String
double	String	Concat	String
String	boolean	Concat	String

$$\underbrace{10}_{\text{int}} + \underbrace{20.1}_{\text{double}} = \underbrace{30.1}_{\text{double}}$$

Note:

,mbhn

```
class A11
{
    public static void main(String[] args)
    {
        int a = 10 ;
        double b = 20.01 ;
        int result = a + b ;
        System.out.println( result ) ;
    }
}
```



36

16

BODMAS

`int result = (int) a + b ;`

→ Precedence

int + double  
double


`int result = (int) (a + b)`

`= (int) (int + double)`

`= (int) (double)`

`= (int)`

Assignment 1 :

 → Name : Rifa  
→ id : 101  
→ salary : 2500

Employee

Wajp to store the above given data of an employee and then display the details in the given format :

Employee Details :

EID : 101  
Employee Name : Rita  
Employee Salary: 2500

Solution:

```
class Employee
{
    public static void main(String[] args )
    {
        String name = "rita" ;
        int eid = 101 ;
        double salary = 2500 ;
        System.out.println("Employee Details : ") ;
        System.out.println("Employee Name : " + name ) ;
        System.out.println("Employee ID : " + eid ) ;
        System.out.println("Employee salary : " + salary) ;
    }
}
```

## 2. - ( subtraction )

LHS	RHS	Action	Result
int	int	subtraction	int
double	double	subtraction	double
float	float	subtraction	float
char	char	subtraction	int
boolean	boolean	CTE	-
String	String	CTE	-

$$\begin{array}{r} \text{'a'} - \text{'b'} \\ 97 - 98 \\ \hline -1 \end{array} = \text{int}$$

Note:

- if the LHS or RHS is either a boolean or a String type we get CTE.  
for example, refer app4/arithmetic/S2.java

## 3. \* (multiplication) :

LHS	RHS	Action	Result
int	int	product	int
double	double	product	double
float	float	product	float
char	char	product	int
boolean	boolean	CTE	-
String	String	CTE	-

$$\begin{array}{r} \text{'a'} * \text{'b'} \\ 97 * 98 \\ \hline 9506 \end{array} = \text{int}$$

Note:

- if the LHS or the RHS is either a String or a boolean we get CTE.  
for examples refer, **app4/arithmetic/M3.java, M4.java**

## 4. / (division) :



LHS	RHS	Action	Result
int	int	divide	int
double	double	divide	double
float	float	divide	float
char	char	divide	int
boolean	boolean	CTE	-
String	String	CTE	-

$$5 / 2 = \underline{\underline{2.5}} \checkmark$$

$$5 / 2 = \boxed{2} \checkmark$$

$$'a' / 'b' = \text{int}$$

$$97 / 98 = 0 \checkmark$$

Note :

- if the LHS or RHS is either String or boolean we get CTE.  
for example refer, app4/arithmetic/D3.java, D4.java

## 5. % (modulus) :

it will divide LHS and RHS, gives the remainder.

syntax:

$$m \% n$$

LHS	RHS	Action	Result
int	int	mod	int
double	double	mod	double
float	float	mod	float
char	char	mod	int
boolean	boolean	CTE	-
String	String	CTE	-

$$3 / 10$$

$$\underline{3 \% 10} \Rightarrow \boxed{3}$$

$$10 \overline{) 3} \{ \underline{0} \checkmark$$

$$0$$

$$\boxed{3} \checkmark$$

$$m \% n$$

$$n \overline{) m} \{ q$$

$$\underline{\quad}$$

$$r$$

Note :

1. if  $m < n$  : then the result is **m**      **ex: 10 % 25 ---> 10**
2. if  $m == n$ : then the result is **0**      **ex: 25 % 25 ---> 0**
3. if  $m > n$  :

case1: if m is a multiple of n, then the result is **0**

**ex: 50 % 25 ---> 0**

case2: if m is not a multiple of n, then result could be anything between  $\{ 1, \dots, n-1 \}$

**ex: 55 % 25 ---> 5**

$$= \begin{array}{r} 25 \overline{) 55} \quad (2 \\ \underline{50} \\ 5 \end{array}$$

Note :

➤ if the LHS or RHS is either boolean or String we get CTE.

Task1 :

Assume a frog is in loc A, it wants to go to location B. The distance between A and B is 1.5kms. The distance covered by the frog for 1 hop is 2.2 m, WAJP to calculate the number of hops required for the frog to reach B from A. (Store the given data and then perform the required calculations ).

Assignment1 :

Assume a jug of capacity 1.2 liters is 3/4th filled with water. the task given is, we need to empty the jug by pouring the water into the glasses of capacity 250ml. WAJP to calculate how many glasses is required to empty the jug.

Assignment2:

WAJP, to convert 24hr time format into 12hr time format.

test case1:

i/p : hh = 22

o/p : hh = 10

test case2:

i/p: hh = 7

o/p : 7

RR % 12

test case3:

i/p: hh = 24

o/p : 0

## Assignment Operators :

it is a binary operator, it is used to store a value into the variable.

Note :

The LHS should always be a variable

ex: 10 = 20 ; // CTE

1. =
2. +=
3. -=
4. \*=
5. /=
6. %=

operator =

case 1:

'='

int a ;

a 10 20

a = 10 ;

a = 20 ;

case 2:

" += "

int a ;

a 10 30

a = 10 ;

a += 20 ; =>

a = a + 20  

$$\begin{array}{r} 10 + 20 \\ \hline 30 \end{array}$$

case 3:

\*=

int a = 3 ;

a 3 -6

int a = 3 ;     a ~~3~~ - 6

$$a * = -2 \circ \Rightarrow a = \frac{a * - 2}{3 * - 2}$$

### Case 4 :

```
int a;    a 
```

$$\underline{a} \quad t = 10^\circ \Rightarrow a = \underline{a} + 10 // \underline{\underline{CTE}}$$

In General :

```
variable operator= literal/expression ;
```

expand it as :

```
variable = variable operator ( literal/expression)
```

Case 6:

```
int a; a 3
```

$$a = 3g$$
$$\underline{a} \% = \underline{4 * 3 + a} \%$$
$$a = \frac{a \% (4 * 3 + \underline{a})}{3 \% (4 * 3 + \underline{3})}$$

$$3 \% (12 + 3)$$

$$3 \% (15)$$

$$\begin{aligned}3 \% (12 + 3) \\3 \% (15) \\ \underline{\underline{3 \% 15}} &= \boxed{13}\end{aligned}$$

Task1 :

Assume Smith had 100 Rs in his wallet, he bought 5 chocolates worth rupees 3 each, write a program to find out the money left in smith's wallet, display suitable message.

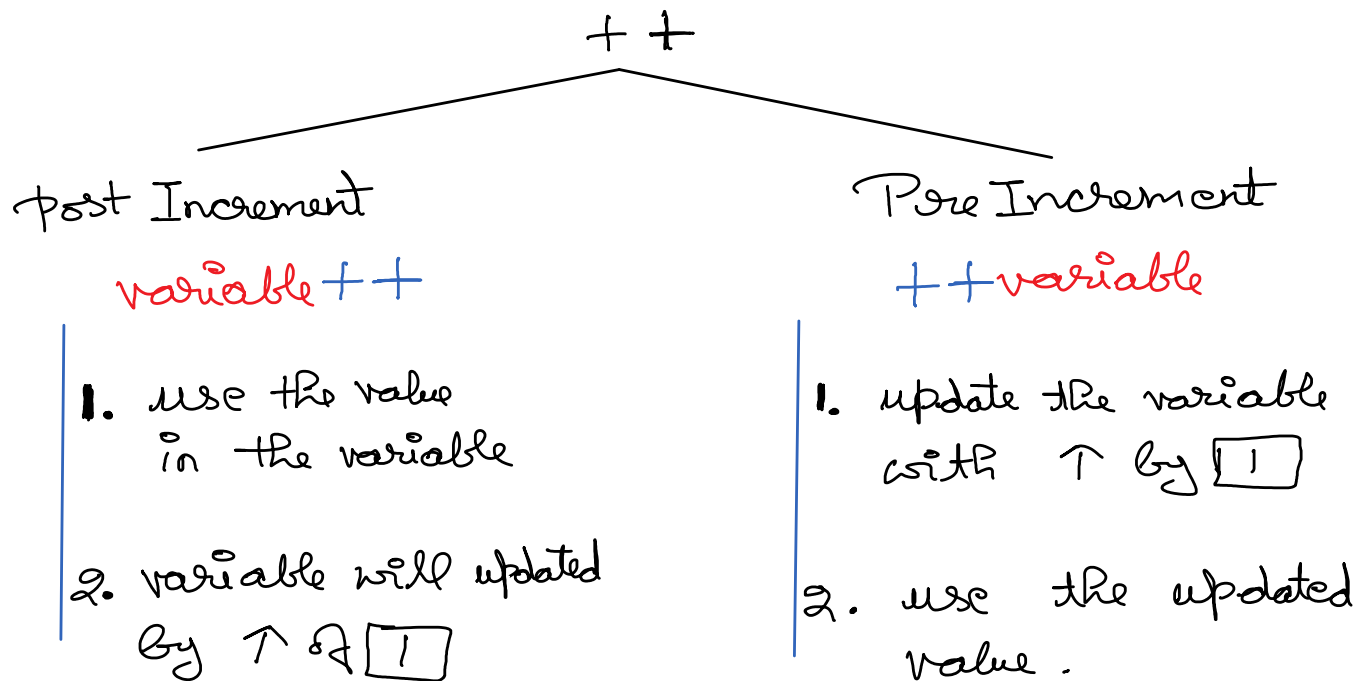
(Store the data & then do the task )

## Increment & Decrement Operators :

### 1. Increment Operator ++ :

it is a unary operator. It is used to update the variable with the increment of **1**.

Based on usage we can classify into 2 types :



Case 1 :

```

int a = 10;           a 10 //
S.o.pln(a++); // 10
S.o.pln(a); // 11
  
```

Case 2 :

```

int a = 10;           a 10 //
S.o.pln(++a); // 11
S.o.pln(a); // 11
  
```

Assignment1 :

Create notes for decrement operator. 1 example each

Assignment2 :

Try increment and decrement operator on different datatypes.



```
int a = 10 ;
```

```
int b = 20 ;
```

```
int c = a++ + ++b ;
```

Handwritten diagram for the expression `a++ + ++b`:  
 - A box contains `10 + 21` with a checkmark.  
 - An arrow points from the `10` to the `31` in the next box.  
 - The next box contains `31` with a checkmark.

```
System.out.println(++a) ; // 12
```

```
System.out.println(b++) ;
```

```
System.out.println( ++c ) ;
```

```
System.out.println( a ) ;
```

```
System.out.println( b ) ;
```

```
System.out.println( c ) ;
```

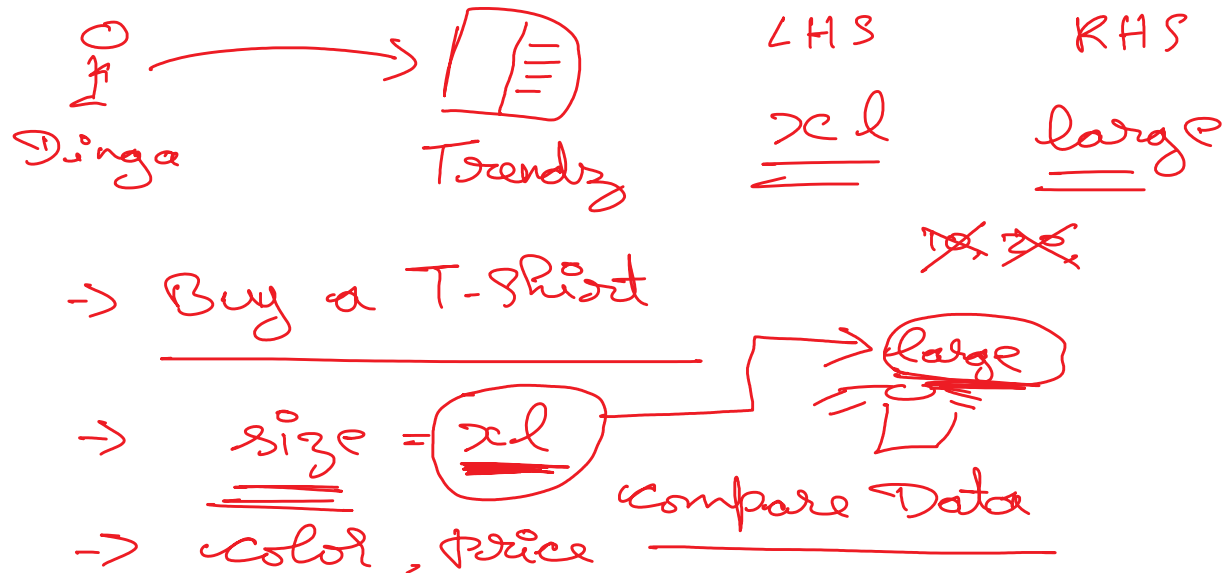
a ~~10~~ 11 12

b 20 21

c 31

Handwritten diagram for the expression `++c`:  
 - A box contains `++` with a checkmark.  
 - Below the box, the word "Sub" is written with a checkmark.  
 - Below the box, the word "Update" is written with a checkmark.

## Relational operators / Comparison :



- relational operators are used to compare two literals(data).
- relational operators always return boolean data as a result. i.e. (true/false)

1. == ( equality )
2. != (non-equality operator)
3. >
4. <
5. >=
6. <=

### 1. ==

- if both LHS and RHS literal is having same value it returns **true**
- if both LHS and RHS literal is different value it returns **false**

### 2. !=

- if both LHS and RHS literals is different it return **true**
- if both LHS and RHS literal is having same value it returns **false**

### 3. >

- if LHS is greater than RHS it returns **true**

4. <

5. >=

➤ if the LHS is either greater or same as RHS it returns **true**

**10 >= 10 // true**

**10 >= 9 // true**

**10 >= 11 // false**

6. <=

Note :

Never compare double with float, most of the cases it results in false.  
for example refer, **app4/rel\_op/P8.java**

**Conditional Operator ? : :-**

- ◆ it is a ternary operator. ( 3 operands )
- ◆ it is used for decision making

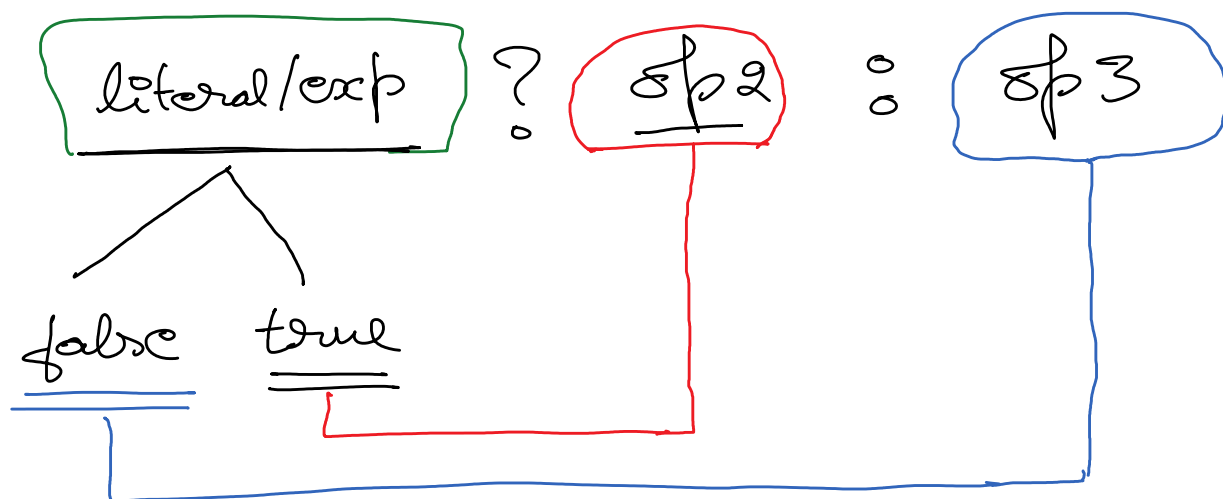
Syntax :

**op1 ? op2 : op3**

Note:

1. op1 should always be a boolean type.

literal / expression ? op2 : op3  
*boolean type*



Case 1:

true ? 10 : 20 // o/p -> 10

true ? 10 : 20 // o/p → 10

Case 2:

false ? 10 : 20 // 20

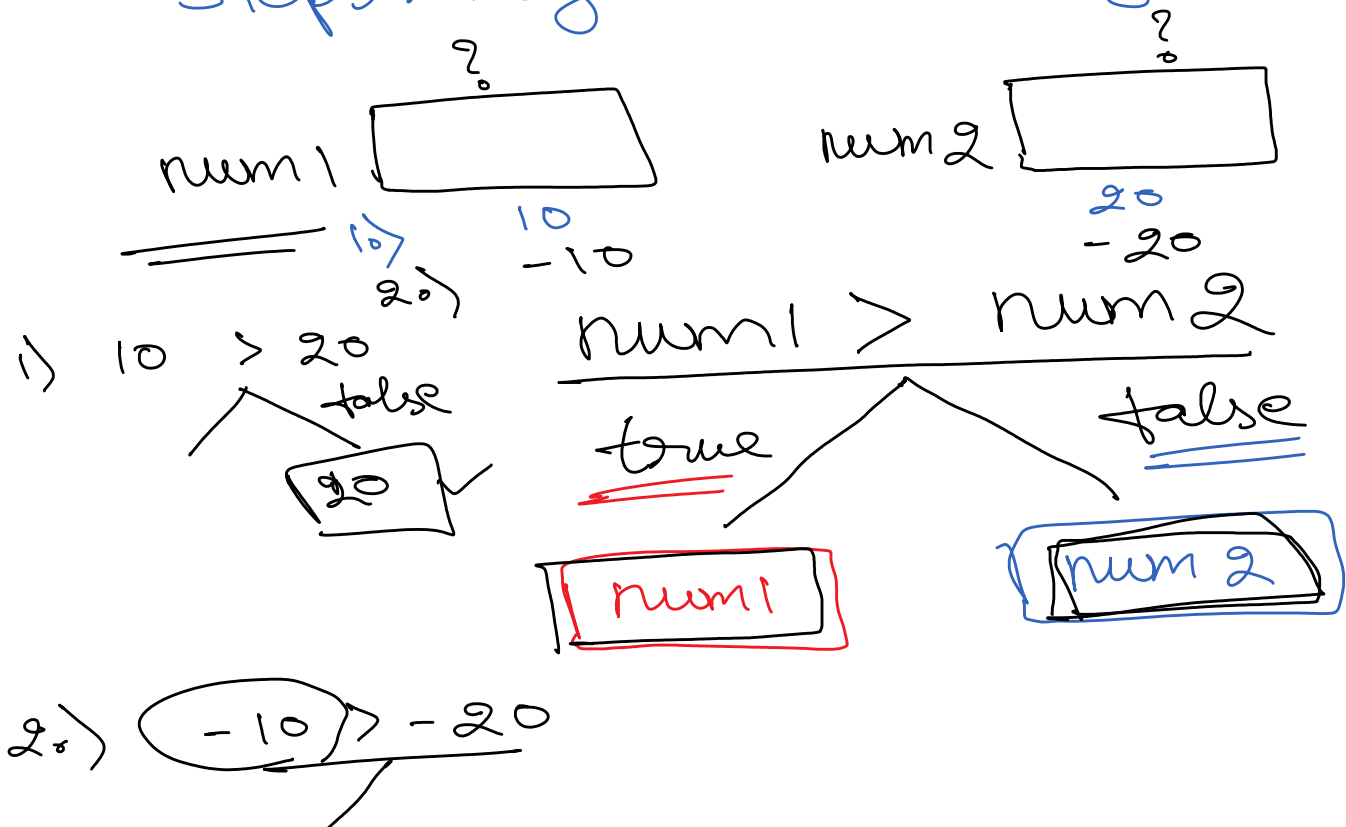
Task1 :

Write a java program to find largest of 2 numbers.

Step1: create 2 containers to store numbers

Step2: input (Read) 2 numbers  
(test data)

Step3: logic to find largest of 2





$num1 > num2 ? num1 : num2$

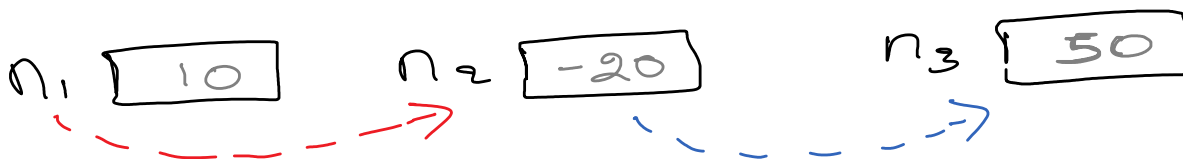
---

Assignment1 :

WAP to find out smallest of two numbers.

Task2 :

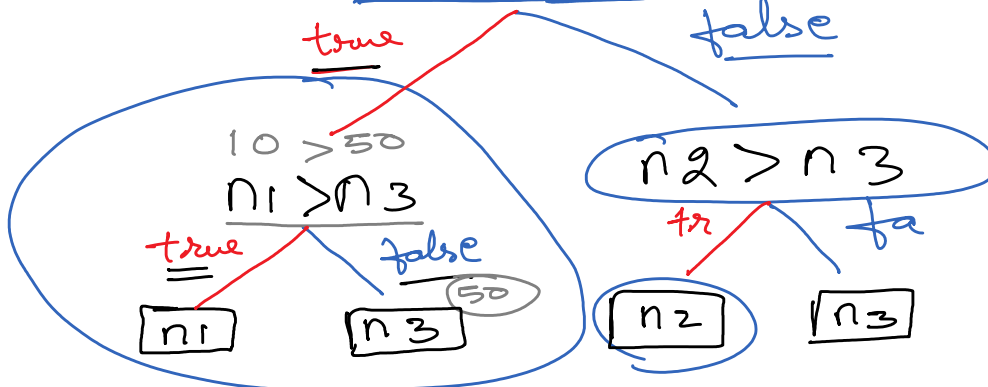
WAP to find out largest of three numbers :



$10 > -20$   
 $n_1 > n_2$

true

false



$(n_1 > n_2) ? (n_1 > n_3 ? n_1 : n_3) : (n_2 > n_3 ? n_2 : n_3)$

true      false

Assignment2 :

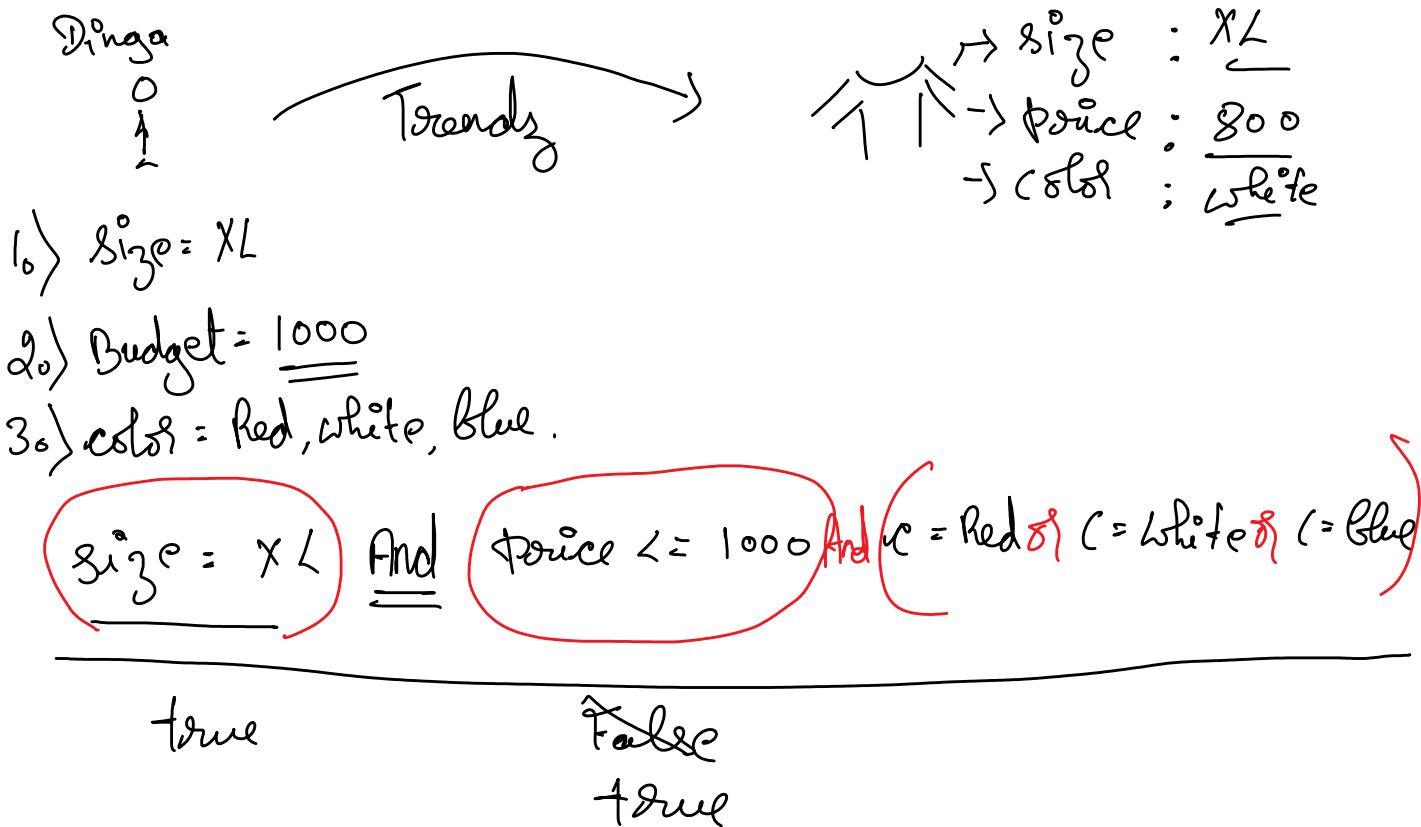
WAP to find out smallest of 3 numbers

Assignment3 :

WAP to find largest of 4 numbers.

**Logical Operators :**

➤ operands of logical operator should always be boolean type.



1. logical And ( && )
2. logical Or ( || )
3. logical Not ( ! )

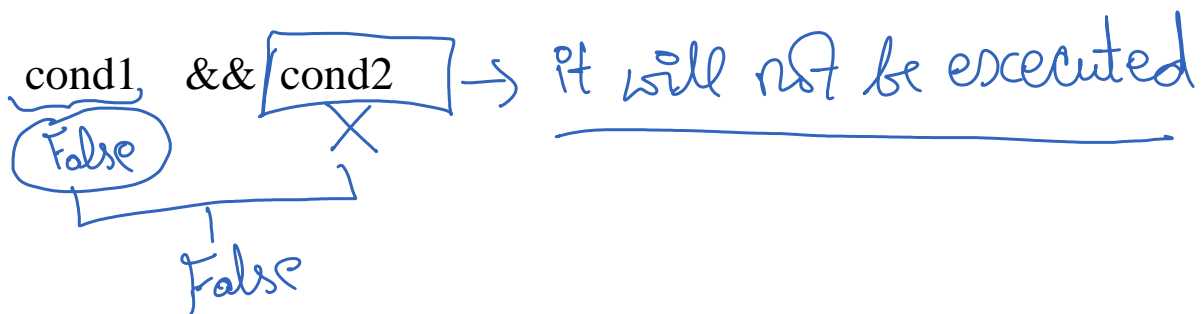


## 1. Logical And (&&) :

- it is a binary operator
- both the operands should be of boolean type
- it is used to combine one or more logical conditions.

Truth Table :

LHS(cond1)	RHS (cond2)	Result
false	true	false
false	false	false
true	false	false
true	true	true



```
char ch1 = 'a';
```

```
char ch2 = 'c';
```

```
boolean b1 = ch1 < ch2 ;
```

```
boolean b2 = ch2 > ch1 ;
```

```
boolean b3 = b1 && b2 && ch2++ < ch1 && ++ch1 < ch2
```

```
System.out.println(b1);
```

= true

```
System.out.println(b2);
```

= true

```
System.out.println(b3);
```

= false

```
System.out.println(ch1);
```

= a

```
System.out.println(ch2);
```

```
System.out.println(ch1 );  
System.out.println(ch2 );
```

= a  
= d

## 2. Logical Or (||):

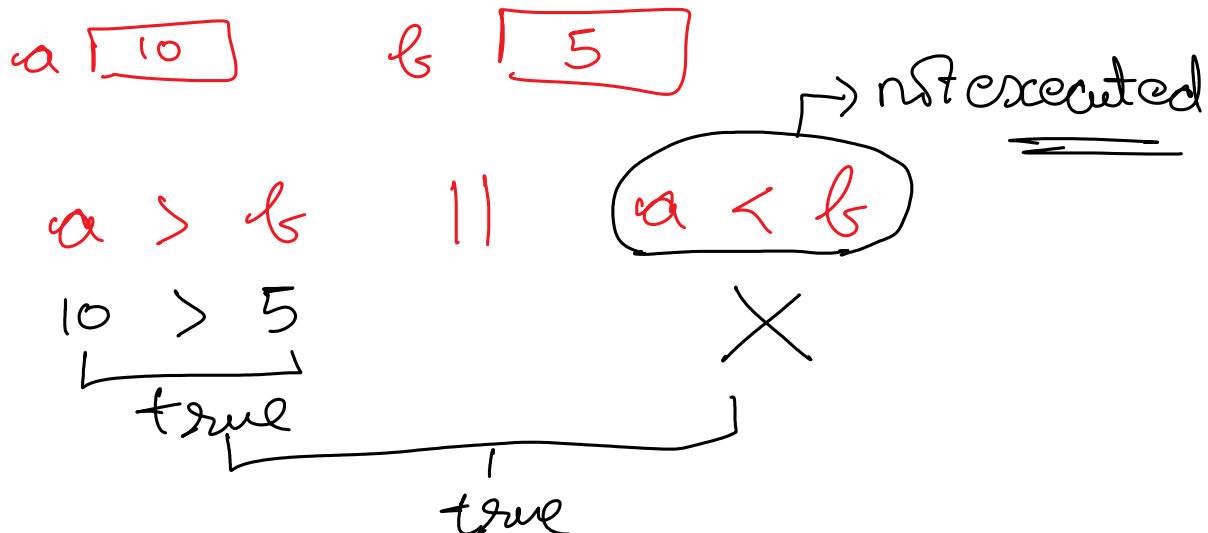
- it is a binary operator
- both the operands must be boolean type

### Truth Table :

LHS(Cond1)	RHS (cond2)	Result
true	false	true
true	true	true
false	true	true
false	false	false

### Note:

- if LHS is true, RHS instruction is not executed.





### 3. Logical Not ( ! ) :

- it is a unary operator ( only one operand )
- operand should be of boolean type.
- it negates the logical value

#### Truth Table

RHS(condition)	Result
true	false
false	true

syntax:

**! literal/ exp**

**ex:**

1. ! true ---> false
2. !false ---> true
3. !10 ---> CTE
4. ! 10 < 5 ----> CTE
5. !( 10 < 5 ) ---> true