Ex:7

```
class Demo7 {
public static void main (String[] args)
{
 int a = 10;
 String S = " SANTHI";
 int b = 20;
 System.out.println (a+S+b);
}
}
```

o/p → 10SANTHI20

Ex:8.

```
class Demo8 {
public static void main (String[] args){
String S = "SANTHI";
int a = 10;
int b = 20;
System.out.println( S+a+b);
}
}
```

o/p → SANTHI1020.

Ex:9

```
class Demo9 {
public static void main (String[] args)
{
 boolean F =false;
 int a = 30;
 System.out.println (F+a);
}
}
```

o/p → Compile time Error → bad operand types of binary operator '+'

* Here we cannot concatinate integer value and a boolean value.

2. Multiplication Operator (*) → This multiplication operator used to perform product of two operand

Ex: 10 * 2 = 20
2 * 3 = 6.

With the multiplication operator we cannot concatinate with the boolean value.

Ex:

```
Class Demo
{
Public static Void main(String[] args)
{
a = 10;
boolean b = true;
System.out.println(a+b);
}
}
```

o/p = Error (bad operand types of binary operator)

3. Division Operator (/) → Used to divide two numbers.
→ It is an arithmatic operator and also a binary operator.

Eg: 10/2 → 2)10(5 ← Quotient
10
(0) ← Remainder

It will give the answer as '5' (Quotient)

5 → int
2 → int
→ (Answer) o/p will be 2 as we gave both the data as integer.

5.0 → Double
2.0 → Double → Here o/p will be 2.5

4. Modulus Operator (%) → It is an arithmatic operator to perform division and it is also a binary operator (%) and return the remainder as output.

Ex: 10 % 2 = 5 → 2 ⟌ 10 (5 ← Quotient)
10
0 ← Remainder

This modulus operator return remainder as the output.

Ex: 1.
```
Class Demo 4
{
    Public static void main(String[] args)
    {
    int a = 10;
    int b = 2;
    System.out.Println(a%b);
    }
}
```
o/p → 0 (Here by using modulus as a arithmatic operator we get op as '0' (Remainder))

2. Ex: 2

3 % 10 → 10) 3 (0
0
3

o/p → 3

23/8/23

Relational Operator (or) Comparision Operator

→ == (Equality operator)
!= (Inequality operator)
> (Greater than)
< (less than)
>= (Greater than or equal to)
<= (Lesser than equal to)

* Relational or comparision Operator is also a binary operator.
* Relational or comparision operator always returs the output in the boolean value. (Either True or False)

Ex: * age1 = 23, age2 = 24

age1 == age2
23 == 24 → o/p is False (Boolean)

age1 != age2
23 != 24 → o/p is True (Bookean)

* height1 = 5.4 height2 = 6.0

height1 == height2 → 5.4 == 6.0 → o/p is False.

height1 <= height2 → 5.4 <= 6.0 → o/p is TRUE

height1 >= height2 → 5.4 >= 6.0 → o/p // False //

height1 != height2 → 5.4 != 6.0 → o/p // TRUE //
height1 < height2 → 5.4 < 6.0 → o/p // TRUE //
height > height2 → 5.4 > 6.0 → o/p // False //

Example Program : 1

```
class Relation
{
 Public static void main (String[] args)
{
  int age 1 = 22;
  int age 2 = 23;
 System.out.println (age 1 > age 2);
}
}
```

→ o/p // False // Boolean output.

Example Program 2

```
class Relation
{
 Public static void main (String[] args)
{
  int age 1 = 22;
  int age 2 = 23;
  System.out.println(age1 >= age2);
```

Logical Operator – Logical operators are used to merge more than one condition in Java.

We have 3 logical operators :

① Logical AND (&&)
② Logical OR (||)
③ Logical NOT (!)

Ex : age = 22, height = 5.0 ← Person 1
age = 23, height = 5.0 ← Person 2

Logical AND (&&) → It is a binary operator and works only on boolean data. (o/p is always boolean)

* Whenever both the conditions are satisfied then only we can used 'AND' operator. (*only when both the statements are true then only we get TRUE o/p)

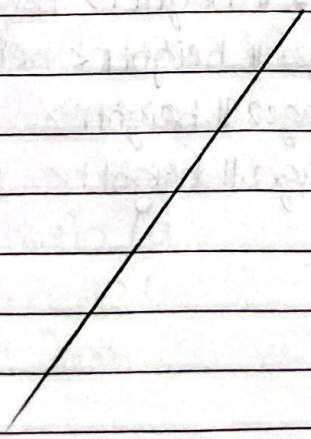
condition 1	condition 2	Result
True	False	False
False	True	False
True	True	True
False	False	False

age1 = 22, height1 = 5.0

age2 = 23, height2 = 5.0

```
class Relation
{
  Public static void main (String[] args).
  {
   int age1 = 22;
   int age2 = 23;
   double height1 = 5.0;
   double height2 = 5.0;
   System.out.println (age1 == age2 && height1 == height2);
   System.out.println (age1 != age2 && height1 != height2);
   System.out.println (age1 >= age2 && height1 >= height2);
   System.out.println (age1 <= age2 && height1 <= height2);
   System.out.println (age1 > age2 && height1 > height2);
   System.out.println (age1 < age2 && height1 < height2);
  }
}
```

O/p → false
false
false
True
false
false



Logical OR (||) → It is a binary operator, return type of 'OR' operator is boolean. This 'OR' operator will work on boolean data.

* We will use 'OR' operator only when any one of the condition is satisfied.

Condition1	Condition2	Result
True	False	True
False	True	True
False	False	False
True	True	True

Ex: age1 = 23, height1 = 5.0
age2 = 24, height2 = 5.0

```
class {
public static void main (String[] args)
{ int age1 = 23; int age2 = 24; double height1 = 5.0; double height2 = 5.0;
  System.out.println ( age1 == age2 || height1 == height2);
  System.out.println (age1 != age2 || height1 != height2);
  System.out.println (age1 > age2 || height1 > height2);
  System.out.println (age1 < age2 || height1 < height2);
  System.out.println (age1 >= age2 || height1 >= height2);
  System.out.println (age1 <= age2 || height1 <= height2);
}
}
```

O/P →

Logical NOT (!) – It is a unary operator which works on only boolean data. It will negate the boolean values

Syntax – ! operand
↳ Boolean data — TRUE / False.

Ex: ! true → False
! false → True

!10 ✗

Gives the opposite data as output.

Ex: (Brand == puma || Brand == nike) → o/p – True

! (True) → o/p // False //

Note: When ever we want to use exculsion (filter) we will go for Logical NOT (!)

24/08/23.

```
class Relation {
Public static void main(String[] args)
{
  int a = 10;
  int b = 20;

  System.out.Println(a == b);
  System.out.Println(a > b);
  System.out.Println(a < b);
  System.out.Println(a >= b);
  System.out.Println(a <= b);
  System.out.Println(a != b);
}
}
```

o/p → false
false
true
false
true
true

```
Class Relation
{
  Public Static Void main(String[] args)
  {
    int a = 10;
    int b = 20;
    System.out.println(a>b || a<b);
    System.out.println(a==b && a>=b);
  }
}
```

O/P →

```
Class Age
{
  Public Static Void main(String[] args)
  {
    double weight1 = 54;
    double weight2 = 56;
    int age1 = 22;
    int age2 = 22;
```

Compound Assignment Operator :

Assigning left side data to the right side Variable or data.

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

Ex : int a = 100;

a+ = 200; → a = a+200; → a = 100+200;

a = 300;

age = 23; → Already existed data.

a+ = 10;

a = a+10;

a = 23+10; → a = 33;

+= → It is going to add data to the already existed data.

-= → It is going to substract data to the already existed data

Wallet = 300; → Already existed data

Wallet -= 100;

Wallet = Wallet - 100;

Wallet = 300-100;

Wallet = 200;

*= → It is going to multiply data to the already existed data.

int mul = 10; → Already existed data

mul* = 2

mul = mul*2

mul = 10X2 = mul = 20;

/= → It is going to divide data from already existed data. and returns the Quotient as output.

Ex: int a = 10;

$a = \frac{a}{2} = a = \frac{10}{2} = 5;$

a = 5;

%= → It is going to divide the data from already data and returns the remainder as the output.

Ex: int b = 20;

b% = 2

b = b%2

b = 20%2 ⇒ $b = \frac{20}{2} = 0;$ (Here 0 is the remainde

Ex program: (+=)

```
class Assignment {
public static void main (String[] args){
  int wallet = 100;
  Wallet += 200;
 System.out.println (wallet);
}
}
```

⟶ o/p // 300//

Program (-=)

```
class Assignment1
public static void main (String [] args){
 int wallet1 = 400;
 int wallet1 -= 200;
 System.out.println (wallet1);
}
}
```

o/p = // 200 //

Program for /= , %= and *=

```
class Assignment {
public static void main (String[] args) {
  int a =10;
   a/= 2;
System.out.println(a);
 int b= 20;
   b %. = 2;
 System.out.println(b);
  int mul = 10;
     mul*=2 ;
  System.out.println (mul);
}
}
```

→ O/P -// 5, 0, 20//

Conditional Operator → It is also called as Ternary operator which means it accepts 3 or more arguments (operands)

Syntax – Op1 ? Op2 : Op3

We are applying the condition and based on that condition we are deciding which one should execute.

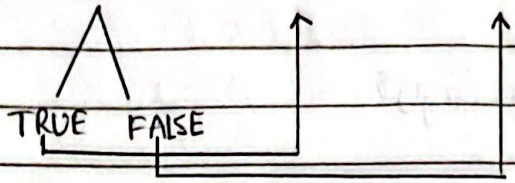
Condition ? Value / Expression / variable : Value / Expression

Condition – Any expression whose final result is boolean

Condition result will be either true or False.

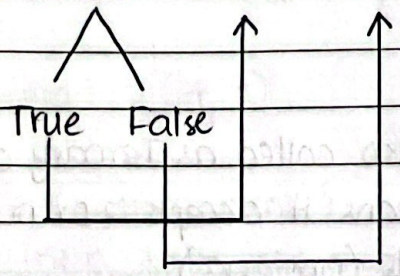
Condition

```
(cond) ? op1 : op2 ;
```



* If there is a value it returns value.
* If there is expression, it will solve and return the result of that expression.
* If there is a variable, it will return the data inside the variable and return the result of that variable.

Ex:
```
int a=10;
int b = 20;
a==b? a+b : a-b;
```



a+b = 10+20 → o/p //30//

a-b = 10-20 → o/p //-20//

Program1

```
class condition {
public static void main (String[] args) {
int a= 10;
int b= 20;
System.out.println(a==b ?a+b : a-b);
}
}
```

o/p

Program 2

```
class condition1{
public static void main(String[] args){
System.out.println(10>20?10:20);
}
}
```

o/p //

Program 3

Find the largest number of 3

int a = 10; int b = 20; int c = 30;

```
class condition2 {
public static void main (String[] args){
System.out.println(a>b?a>c?a:c:b>c?b:c);
}
}
```

o/p → 30

25/8/23 Assignment: By using Ternery operator guess whether the number is even number or odd no?

```
class Even
{
Public static void main(String[] args)
{
 int num= 10;
String res = num%2==0?"even number":"odd number";
 System.out.println(res);
}
}
```

o/p :// even number//

Increment Operator : (++) (cannot directly perform on data)

The Increment Operator will update (increase) the give value (already existing value) by 1.

Ex:1 int a = 10;
a++; → (Internally a=a+1 = a= 10+1 = a=10)
a=11;

Ex:2 int b= 20;
b++;
b= 21; (Increment operator (++) is increasing the value by 1)

Types of Increment Operator :

1. Pre- Increment Operator
2. Post- Increment Operator.

Pre- Increment Operator :

Syntax : ++VariableName;

Ex: int a=2;
++a; → Returning the data.
a= a+1; First it will increase the given value
a= 2+1; and it will return the updated value.
a=3;

x=0
x= x+1
x= x+1
x=0+1
x=1

Ex2 : int Wallet = 10;
++ Wallet;
Wallet = Wallet + 1;
Wallet = 10 + 1;
Wallet = 11;

Post Increment Operator :

Syntax : Variable++;

Ex : int a = 1;
a++; → Return the given value then increase the value by 1.
a = a+1
a = 1+1
a = 2;

Pre - Increment	Post - Increment
* First update the variable then return the data	* First it is going to return the already existed data then it is going to update the data.
* Ex : JIO Pre-Paid	* Ex : JIO - Postpaid.

* It is unary operator because these operators accepts only one value at a time.
* It can be used only with variables, it cannot be used with the literals.

Decrement Operator (--) →

This is a type of operator which is going to decrease the existed value by 1

Ex : int a = 10;
a--;
a = a-1;
a= 10-1;
a = 9;

Types of Decrement Operator

① Pre- Decrement Operator
② Post- Decrement Operator

Pre- Decrement Operator :

Syntax → --Variable Name;

Ex: int a= 10;
--a; ← First decrease the value by 1 then return the value by.
a= a-1;
a= 10-1;
a=9 O/P is the updated value.

Here o/p //9//

Post - Decrement Operator

Syntax → VariableName--;

Ex : C = 10;
C--; → First return the existing data then decrease
C = C-1; the data by 1.
C = 10-1;
C = 9; O/P will be the initial value.

Pre - Decrement	Post - Decrement
* First update the value and then return the value.	* First return the value and then update the value.

Program (Assignment)

① Class Increment {
public static void main (String[] args){
int a = 20;
System.out.println(a++ + ++a + --a + a--);
}
}
O/p // 84 //

② Class Decrement {
public static void main (String[] args) {
int a = 30;
int c = 20;
System.out.println(c-- + ++c + --a + ++a + a--);
}
}
O/p //129//

26/8/23

Assignment :

1. Assume we have 2 integer number, stored in a 2 variable named as a and b, write a program logic to obtain the largest number.

2. Assume we have 2 number stored in 2 containers, write a java logic to check whether the number is even (or) odd and print suitable message.

3. Find the largest of 3 numbers

4. Find Smallest number among 3 numbers.

5. Task 1)
```
int a = 10, b = 20;
int res = a>b ? ++a : ++b;
S.O.p (res);
S.O.p (a);
S.O.p (b);
```
Task 2)
```
int res = a>b && ++a<b? a:b;
Sop (res);
SOP (a);
SOP (b);
```

6. Assume martin opened a savings account with initial balance of 2000 Rs/-, he does following transaction
T1 = Deposit 500, T2 : Withdraw 100, T3 : Transfer 100
Display his final account balance.

7. Assme we have an integer number, check whether we have an integer number, check whether that number is divisible by 5 or not.