Topic:- What is an Operator 01/02/25

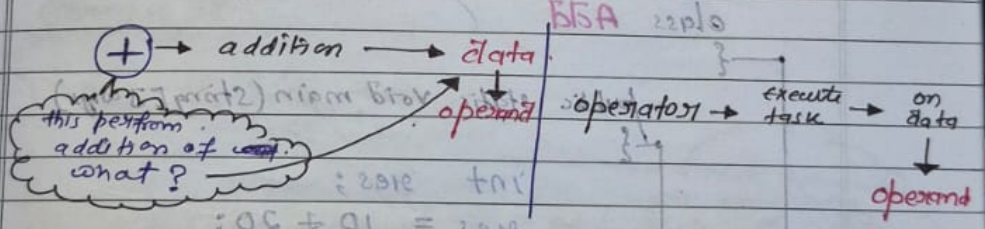
"Hey operator is not new for us we are very familiar with it" How when we are in class 1 or 2 we perform addition, subtraction, Division and many more the symbol we are using to perform '+', to perform subtraction '−' This special and pre-defined symbol is known as "operators".

→ operator:- (According to Java)

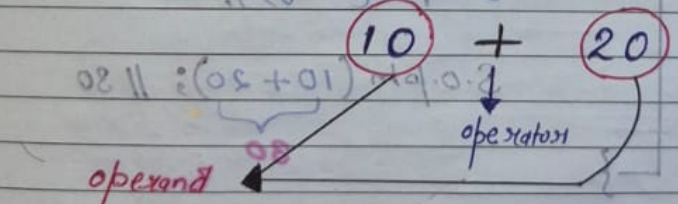
→ Operator are pre-defined symbol's in Java which is used to perform a specific task/operation into the operands and also used to execute the task is known as "operator"

Ex:- '+' → addition

'.' → access members



Example:-



PAGE :
DATE :

* Characteristic of operator :-

(11)

Example :- `10 + 20` → expression

Not an statment

(1) → After the execution of the operator is completed it returns the result.

as, as programer it our responsibilities either store the result in another variable or we can directly print the result.

Ex :-

```
class Add
{
    p "s vm()
    {
        10 + 20; // CTE
    }
}
```

this is not an statment

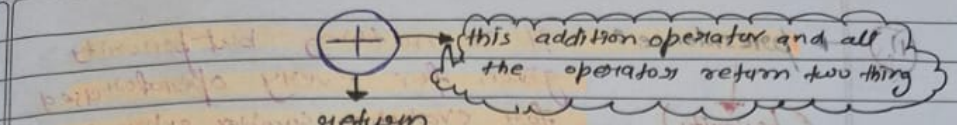
```
class Add
{
    public static void main(String[] args)
    {
        int res;
        res = 10 + 20;
                30
        S.O.pln(res); // 30
    }
}
```

OS + O1

S.O.pln(10+20); // 30

30

PAGE :
DATE : / /



return

type of data | Value.

Example 1:-

int ← 10 + 20 → int

return

type of data (int) | Value 30

Example 2:-

'a' + 'b'

97 Ascii Value | return | 98 Ascii Value

type (int) | Value 195

Ascii Value.

Example 3:-

int ← 5/2 → int

here numerator and denominator both are integer that why the original value is 2.5 but here the result is only integer part that is '2'

return

type of data (int) | Value 2

2.5

PAGE :
DATE :

(II) → presidence : It is nothing but priority given for every operator used for evaluating complex expression

(priority)

precidence : '+' < '*' ; '+' < '/'

operator 1 operator 2

10 (+) 2 (*) 3

return

type of data (int) | value 36 or 16

presidence or priority. come into picture only when we write an expression having more than one operator.

Result

what is the result if addition operation execute first the result is 36 or if multiplication occurs first than 16 is the result. ?

Here the highest property goes for '*' multiplication so, first 2*3 will execute the 6+10 then 16 is result.

(III) → Associativity: It is nothing but direction of order of execution.

type

Left to right

Right to Left

Ex:- 10 (+) 20 (+) 30

01 02

30

60

'+' operator have Left to Right associativity.

PAGE :
DATE :

Ex:- 10 + 20 / 2

Here division operator having the higher precedence.

Step1: 20/2

10

Step2: 10+10

20 → Result

Ex:- 10 + 'a' - 3 * 2

Here '*' having higher precedency.

Step1: 10 + 'a' - 3 * 2

10 + 'a' - 6

Ascii value 97

Step2: 10 + 97 - 6

107 - 6

Step3: 101 → Result

Here the precedence of '+' & '-' is same then we go for Associativity.

Imp. Question:-

* what is an operator, operand?
* Why do we need an operator?
* Explain the characteristics of operator.
* what is the need of precedence & associativity?
* who can consume the result of an operator & how?

PAGE :
DATE :

Topic:- Types of operators

(1) "we can classify the operators based on two thing"

operators

Based on **operands**
- Unary (one)
- binary (two)
- ternary (three)

Based on **task**
- Arithmetic
- Relational
- logical operators
- Assignment "
- Bitwise "
- :

Topic:- Arithmetic operators :- (binary operators)

(1) In "Arithmetic operator" we have
- '+'
- '-'
- '*'
- '/'
- '%'

All this five operators are binary operator | op1 + op2

all this five operators perform some Arithmetic operation and give result

1 → "Let us understand with one go with our first operator "+"

(1) + → Addition
+ → Concat

Merging two values

PAGE :
DATE : / /

Syntax :-

operand1 + operand2

(II)

if any one of this operands is string then it behaves like a "concat" operator & in all the remaing situation it act as a addition operator

(III)

Ex :-

(I) → 10 + 20 → Addition → 30
Integer, Integer, (integer data)

(II) → "10" + 20 → Concat → 1020
String (because 10 is written inside the double cots), Integer, (string data)

(III) → 'a' + 20 → Addition → 117
'a' → 97
Characters, Integer, (integer data)
It will take the ascii value of small a → 97

(IV) → 'a' + "b" → Concat → ab
'a' → characters, "b" → string., (string data)

PAGE :
DATE :

(V) → 10 + true → addition → CTE

10 → Integer

true → Boolean → boolean can not be converted to any other data type.

so we can not use boolean with the addition operator

error: Bad operand

(VI) → "10" + true → concat → 10true

"10" → Integer

true → boolean

10true → String data.

(VII) → 10 + 20 + "hello" → 30hello

10 + 20 → Addition → 30

30 + "hello" → Concat → 30hello

30hello → string data.

Here, there are two operator we have to use precedence or Associativity here both the operator is same so we go for Associativity so for '+' "Left to Right"

(VIII) → "hello" + 10 + 20 → hello1020

"hello" + 10 → concatanation → hello10

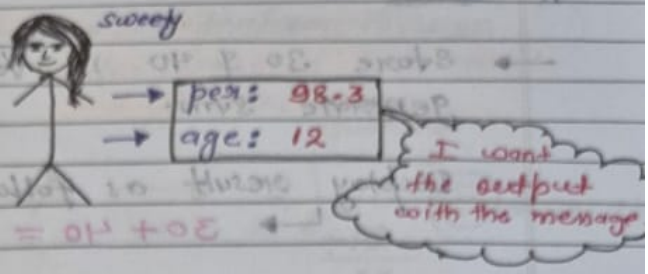
hello10 + 20 → concatanation → hello1020

hello1020 → string data.

(VI) → 'd' + 'd' → "dd"

PAGE :
DATE : / /

Example:-



(i) class Data

```
{
    public static void main (String[] args)
    {
        double per = 98.3;
        int age = 12;
        System.out.println (per);
        System.out.println (age);
    }
}
```

Console
98.3
12

→ with message

(ii) class Data

```
{
    public static void main (String[] args)
    {
        double per = 98.3;
        int age = 12;
        System.out.println ("per: " + per);
        System.out.println ("age: " + age);
    }
}
```

Console
per: 98.3
age: 12

PAGE :
DATE :

Task :-

→ store 30 & 40 in Variables and generate sum.

Display result as follows

↳ 30 + 40 = 70

(II) → * → use to find the product of two numbers

→ 10 * 2 → 20

Integer Integer Integer

→ 'a' * 2 → 194

character Integer Integer

it will take ascii value

→ "a" * 2 → CTE

string Integer (Bad operand)

→ true * 2 → CTE

boolean Integer (Bad operand)

we can not use string and boolean literal to find the product

PAGE :
DATE : / /

(iii) → / → It is used to perform division
↳ After performing division it will give the quotient

usage :-
↳ 10/2 → 2) 10 (5 → result
-10
xx

Example :-
↳ 5/2 → 2.5 → decimal
int int int
5/2 → 2 output.

→ 3/4 → 0

→ 3.0/4.0 → 0.75

→ 3.0/4 → 0.75

→ 3/4.0 → 0.75

→ 'a'/'b' → 0
97 98
take the ascii value.

→ "a"/1 → CTE

→ true/true → CTE
↳ true/1

We can not used string and boolean literal for divison purpos

PAGE :
DATE :

(iv) → % → division and return the remainder as result

Example :-

(i) 3 % 10 → (3/10) → 10) 3 (0
0
3 → result

(ii) 10 % 3 → (10/3) → 3) 10 (3
-9
1 → result

Imp Note :-

→ m % n → (m/n) → n) m (q
r → result

Case 1 :-

→ if m is < n
→ o/p : m Ex :- (1)

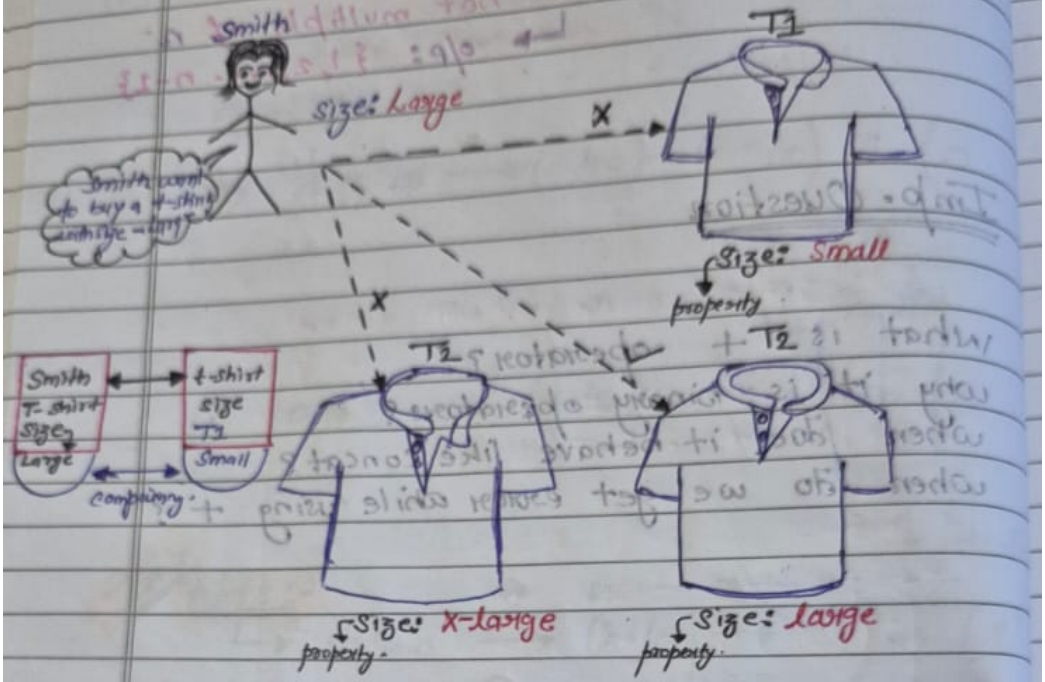
Case 2 :-

→ if m == n
→ o/p : 0

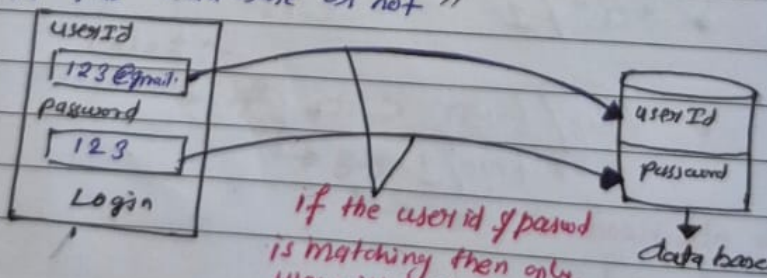
Case 3 :-

→ if m > n
→ 3.1
→ m is multiple of n
→ o/p : 0

Topic 8- Relational operators & (binary operators)



" when ever we have to make any decision comparing data is very very important, In the same way when we are writing code or developing an application it is mandatory in many of cases we have to compare data for example suppose user want to login into facebook you have to give user id and password the user id and password typed by you or user should be validated with the user id and password which is saved in the data base or not "



if the user id & passwd is matching then only user will allow to go there home page.

PAGE :
DATE : / /

→ It is used to compare two values.

(I) → == (equality operator) → true / false
(II) → != (Not equal operator)
(III) → > (Greater than operator)
(IV) → < (Less than operator)
(V) → >= (Greater than equal to operator)
(VI) → <= (Less than equal to operator)

Note:- • They can compare two value's at a time
• They are binary operator.
• The return type of Relational operator is boolean.

1) ==	10==10 → true	'a'=='a' → true	10==20 → false	true==true → true
2) !=	10!=10 → false	'a'!='a' → false	10!=20 → true	true!=true → false
3) >	10>20 → false	20>10 → true	'a'>'b' → false	
4) <	10<20 → true	20<10 → false	'a'<'b' → true	
5) >=				

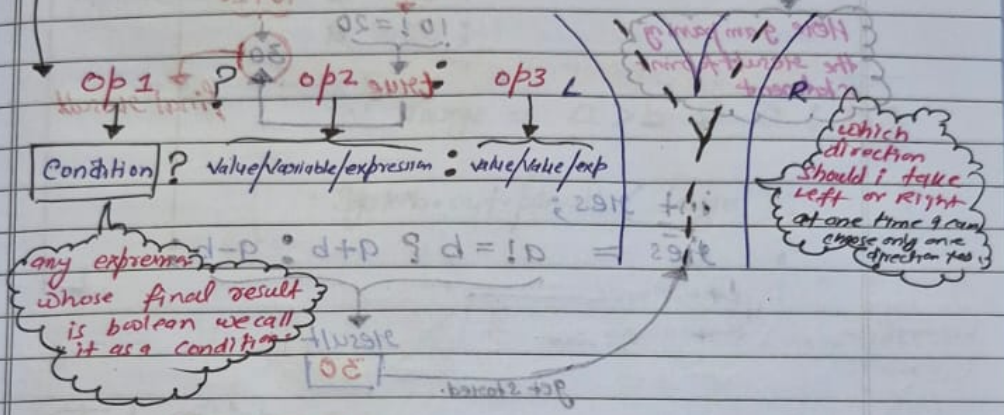
PAGE :
DATE : / /

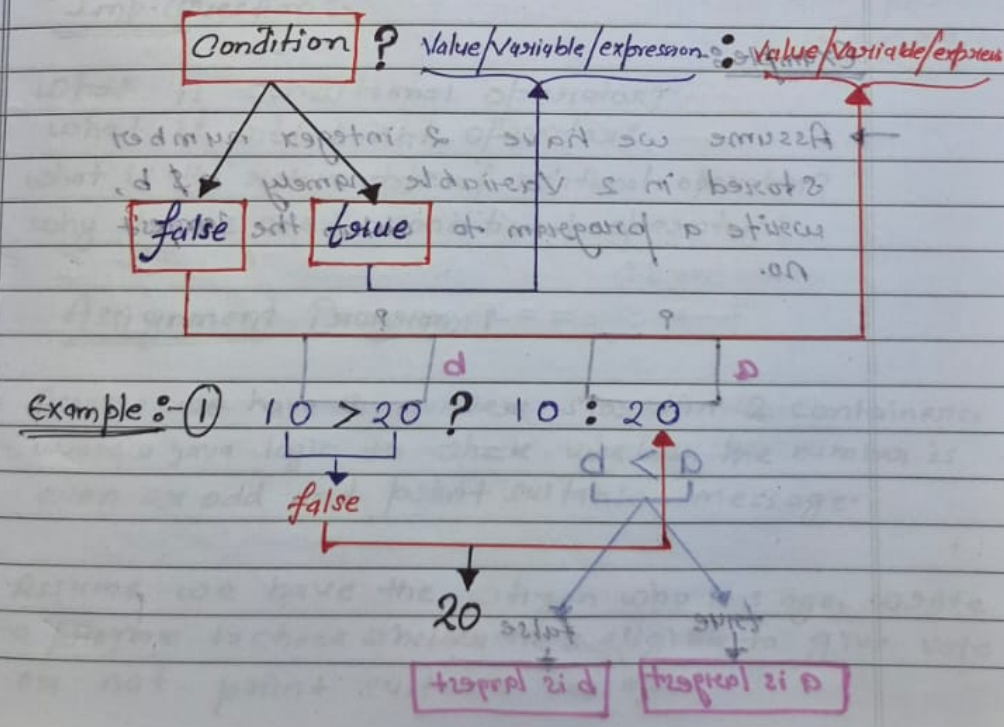
Topic :- Conditional operator (ternary operator)

↳ syntax :-

`op1 ? op2 : op3`

purpose :- It behaves like a decision making. person ed It helps us for decision making.





Example :- (1) `10 > 20 ? 10 : 20`

false

20

PAGE :
DATE :

Example:- (11)

```
int a = 10;
int b = 20;
System.out.println(a != b ? a+b : a-b);
```

a → 10
b → 20

Here comparing the result to print statement

10 != 20 → true

10+20 → 30

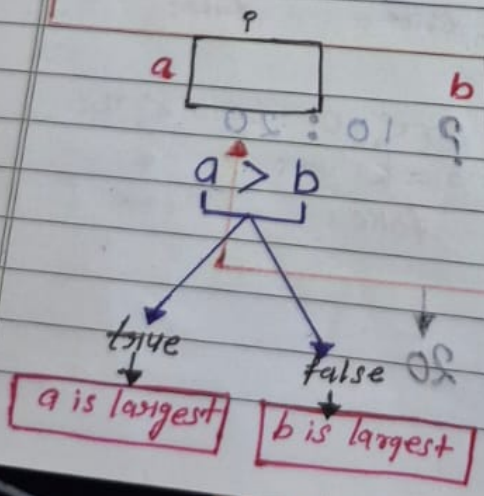
30 → final result

```
int res;
res = a != b ? a+b : a-b;
```

result → 30 → get stored.

Example:-

→ Assume we have 2 integer number stored in 2 variable namely a & b, write a program to obtain the largest no.



PAGE:
DATE:

Code:-

```
class Program3
{
    public static void main (String args[])
    {
        int a, b;
        a = 10;
        b = 20;
        int large = a>b ? a : b;
        System.out.println(large);
    }
}
```

Imp-Question:-

* what is conditional operator?
* what is relational operator?
* what is the return type of relational operator?
* why do we need conditional operator?

Assignment Programs:-

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

2) Assume we have the citizen who has age, write a program to check whether he is eligible to give vote or not. print suitable message.

PAGE :
DATE :

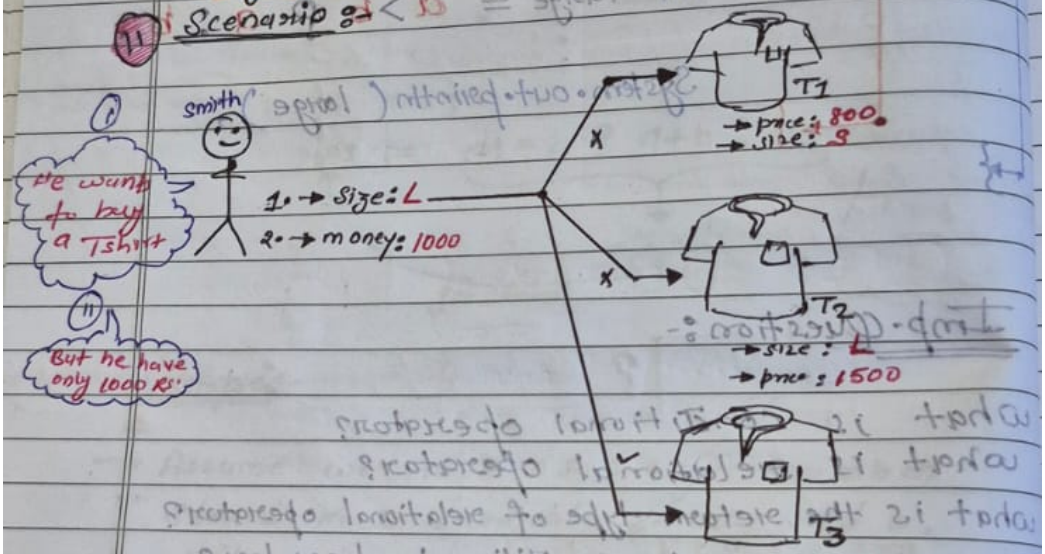
Topic :- Logical operator

→ logical and "&&" → double (ampersand)
→ logical or "||" → (pipe)
→ logical not "!" → (Exclamation) mark

(1) "Obviously you have some question like.
what is logical operator,
why do we need logical operator
let us discuse this things ok"

→ **Logical and "&&" operator :-**

(11) **Scenario :-**



condition 1 :
↳ Size == Tsize

Condition 2 :
↳ money >= price

PAGE :
DATE : / /

when we go for "&&" logical operator
↳ whenever we want to compare two condition and also want that all the condition must be satisfied. then we can go for '&&' logical operator.

how to implement logic :-

Size == Tsize && money >= price.

true true

final result (true)

if in the both condition any one is 'false' then the final result will be 'false'

→ check for Tshirt 1 :-

L == S && 1000 >= 800

① false

false

"Don't buy"

if the first condition is getting false then it won't check the second condition and the final result will be false.

→ check for Tshirt 2 :-

L == L && 1000 >= 1500

① true

② false.

false.

"Don't Buy"

→ check for Tshirt 3 :-

↳ L == L && 1000 >= 900

① true ② true

true

↓

"Buy the Tshirt"

"T3 is eligible for buy"

(IV)

→ Code :-

```
class Program1
{
    public static void main (String[] args)
    {
        // smith properties
        char Size = 'L';
        int money = 1000;

        // properties of Tshirt size
        char Tsize = 'S' / 'L' / 'L';
        int price = 800 / 1500 / 900;

        String result = Size == Tsize && money >=
            ? "Buy" : "Don't Buy";

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

PAGE :
DATE : / /

Conclusion :-

- It is binary operator.
- It works only on boolean data.
- return type is always boolean
- we use only if both the condition to be satisfied.

Condition 1	Condition 2	Result
false	not execute	false
true	false	false
true	true	true

Task :-

(1)

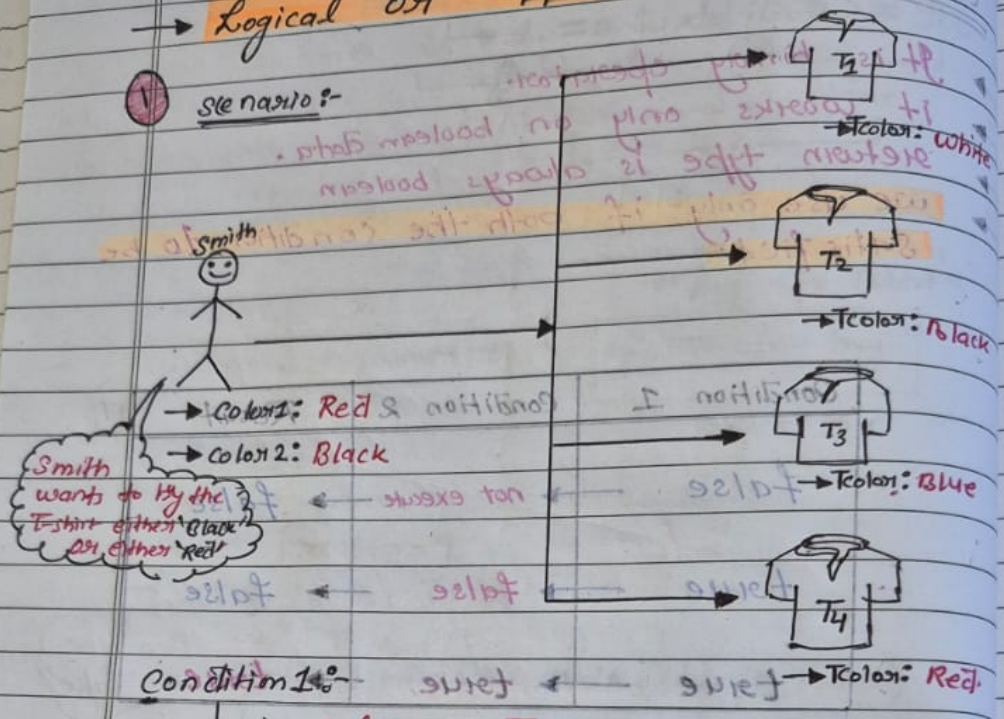
```
int a=10, b = 20;
int res = a>b?++a:++b;
S.o.pln(res);
S.o.pln(a);
S.o.pln(b);
```

(11)

```
int a=10, b=20;
int res= a>b && ++a<b?a:
S.o.pln(res);
S.o.pln(a);
S.o.pln(b);
```

Logical OR " || " :-

1 Scenario :-



Condition 1 :-

→ color1 = = Tcolor

Condition 2 :-

→ color2 = = Tcolor

When we go for Logical OR " || " operator.

→ whenever we want to compare two condition and also want that if any one condition is satisfied, then the whole result is (true).

PAGE :
DATE : / /

11

* How to implement Logic :-

Color1 == Tcolor || Color2 == Tcolor

true — won't check

final result (true)

If any one condition is true then final result is also true.

→ check for Tshirt 1 :-

Red == White || Black == White

false — false

false

If first condition is false then it will check for second condition.

→ check for Tshirt 2 :-

Red == Black || Black == Black

false — true

true

→ check for Tshirt 3 :-

Red == Blue || Black == Blue

false — false

false

→ check for the Tshirt Uniform of cell

→ Red == Red || Black == Red

true

X

true

if the first condition will get true it won't check the next condition hence the whole result will be true

(iii)

Conclusion:-

→ it is binary operator
→ it works only on boolean
→ result is always boolean
→ we use || operator only when any one of the condition to be satisfied.

Condition 1	Condition 2	Result
true	not execute	true
false	true	true
false	false	false

Imp Question:-

* How many logical operators are available?
* Explain && operator?
* Explain || operator?

PAGE :
DATE : / /

Logical Not operator :- (Unary operator)

- It is a unary operator.
- It can work only on boolean data
- It will Negate (opposite) the boolean value.

Syntax-

! operand

example : (i) !true (ii) !false

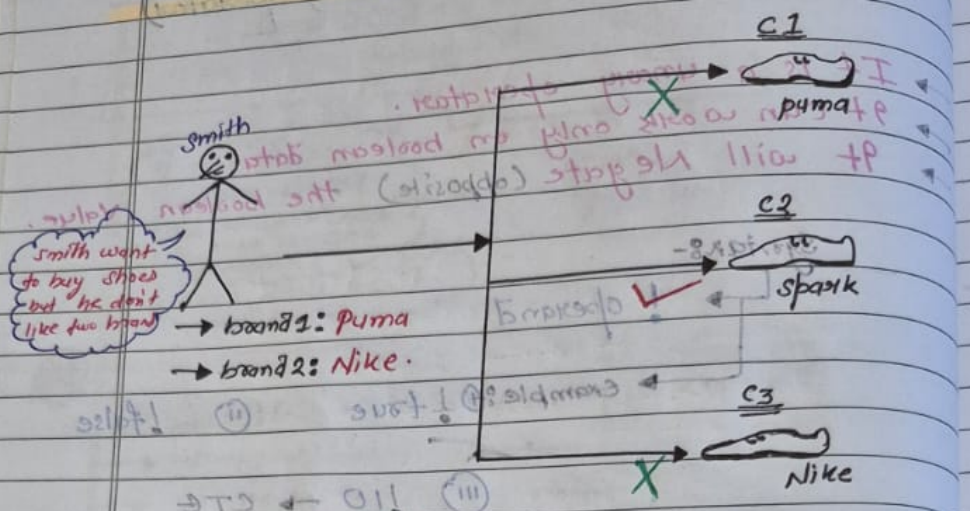
(iii) !10 → CTE

(iv) !10==10 → CTE

not operator having higher priority as compare to equality operator so !10 will execute first and give you CTE

(v) !(10==10) → Now equality operator will execute first.

true



→ Logic:-

!(brand == "puma" || brand == "nike")

C1 :- !(puma == puma || puma == nike.)

true

true → False

False

Final result is false so it will not select the puma

C2 :- !(spark == puma || spark == nike)

false

false

false

True

Hence the final result is true so spark will be selected.

PAGE :
DATE :

Code :-

```
class Program
{
    public static void main(String[] args)
    {
        // brand smith dont like....
        String brand1 = "nike";
        String brand2 = "puma";

        // shoe brand
        String brand = "puma" / "sparxx" / "nike";

        // logic
        String res = !(brand == brand1 || brand == brand2)
                     ? "can buy" : "can't buy";

        System.out.println("smith" + res);
    }
}
```

PAGE :
DATE :

Topic 8- Increment operator

It is denoted by (++)

It already increase the current variable value by 1

```
int a = 10;
a++;
```

It is used as em statment

// a = a+1;
10 + 1 → 11

Increment operator (++)

- post-increment (Variable ++)
  - first use the older data and then increment by 1
- pre-increment (++ Variable)
  - first increment the data by 1 and than use for execution

To difference b/w them let us take two cases

Case 1

```
int a = 10;
int b = a++;
sop (a); // 11
sop (b); // 10
```

a: 10 → 11

b: 10

first it store 10 in b, then it will increment 10+1 = 11

Case 2

```
int a = 10;
int b = ++a;
sop (a); // 11
sop (b); // 11
```

a: 10 → 11

10+1 = 11

It will store the value after execution.

PAGE :
DATE : / /

→ post increment operator :- (a++)

→ (i) 1st use existing value in the variable

→ (ii) then update the variable by 1

→ pre-increment operator :- (++a)

→ (i) 1st update variable by 1

→ (ii) use updated variable

Important point :-

→ ++ is unary operator.

→ ++ can be used only with a variable

Example :-

```
int a = 10;
```

a | 10 | 11 | 10

```
a = a++;
```

Code :-

```
class Program {
    ps vm (--){
        int a = 10;
        a = a++;
        Sopln (a);
    }
}
```

Output :- 10

PAGE :
DATE : / /

Topic:- Compound Assignment operators:-

Types of Compound Assignment operators:-

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

Scenario:-

int wallet = 500; → wallet 500

// I want to deposite 300 more

wallet = wallet + 300; → wallet 500 800

500 + 300

800

It is a inefficient way of writing because same variable we are writing again

Instead of writing variable again and again we came up with an interesting operator that is nothing but compound Assignment operator

wallet += 300;

Here we are adding the 300 to the existing value of wallet i.e 500 + 300 = 800

Short hand notation

(wallet = wallet + 300;)

PAGE :
DATE :

`+=` :→ It will add new data to the existing data present in the variable.

`-=` :→ It will subtract the new data from the existing data inside the variable.

`*=` :→ It will multiply the new data from the existing data present inside the variable.

`/=` :→ In this the new data will divide the existing data present inside the variable and quitent is the result

`%=` :→ In this the new data will divide the existing data and give reminder as the result.

Example :-

```
int a = 10;
```

a | 10

```
a %= 2;
```

a = a%2 → 0 | 10%2=0

a | 0

```
sop(a);
```

→ // 0

```
int a = 20;
```

a | 20

```
int b = 30;
```

b | 30

```
a /= b-10;
```

(i) This will execute first
30 − 10 → 20
(ii) a = a/20
20/20 → 1

```
sop(a);
```