Introduction to Java Programming :-

Java is programming Language that was introduced in the year 1995 by James Gosling & co who wire working for "Sun Microsystems".

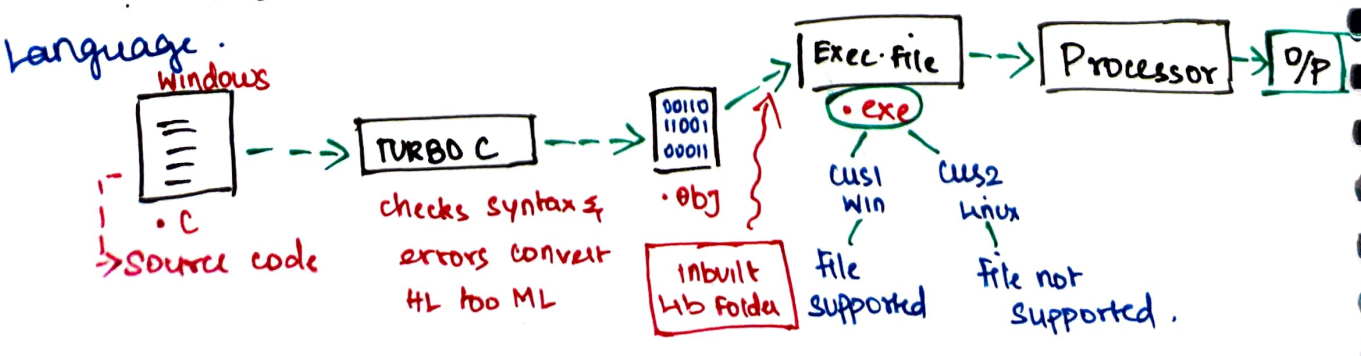
Later in the year 2010 entire "Sun microsystems" was overtaken by "oracle" from then Java is the product of oracle company.

NOTE : The initial name of JAVA is "OAK".

C	JAVA
* platform dependent	* Platform independent
* less libraries	* More libraries.
* less Secure	* More secure
* Hard to study	* Easy to study.
Contains pointer	* dos not contain pointer.

Properties of JAVA :-

Property 1 :- Java is a platform dependent programming Language.

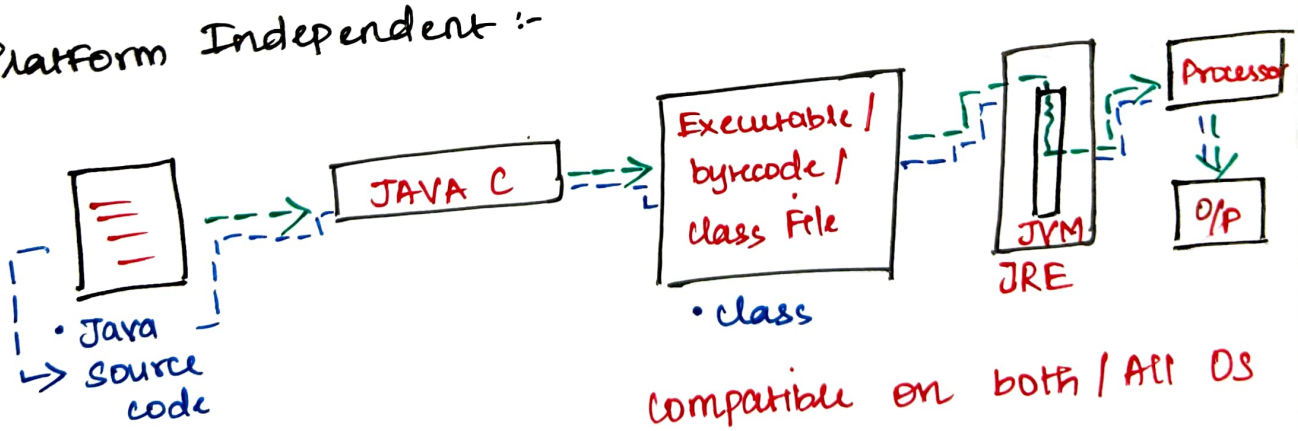


* C is a platform dependent programming language that means code written on one platform / OS can't be executed on different platforms / OS. Just by copy paste of the executable file.

* If the programmer wants to execute the same piece of code on different platforms or OS then they should rewrite & recompile & re execute the code.

If we develop an application on windows OS then it will run only on windows OS, not only other OS like MACOS, LINUX, UBUNTU.

b) Platform Independent :-



compatible on both / All OS

Works on : Linux, windows or any other file. as it is Platform Independent

* Java is platform independent programming language that means the class file which is generated after compilation can be executed on any OS with the help of JAVA.

JVM [JAVA VIRTUAL MACHINE]

* It will perform the task like a machine without having any physical existence.

* The task of JVM is to process the executable file to the processor.

JRE [JAVA RUNTIME ENVIRONMENT]

* It is executable responsible for providing all the necessary environment & libraries to the JVM.

JAVAC [JAVA COMPILER]

* It will take the JAVA source code as input & it will check the syntax & rule.

* If there is any mistake then it will display the error message.

* If there is no errors, the source code is then generated to executable class file.

* It is highly optimized version of the Java source code.
* It is generated by the JAVA C compiler with class extension.
* It cannot be read by humans it can understood only by the machine.

JIT (JAVA INTERPRETOR)

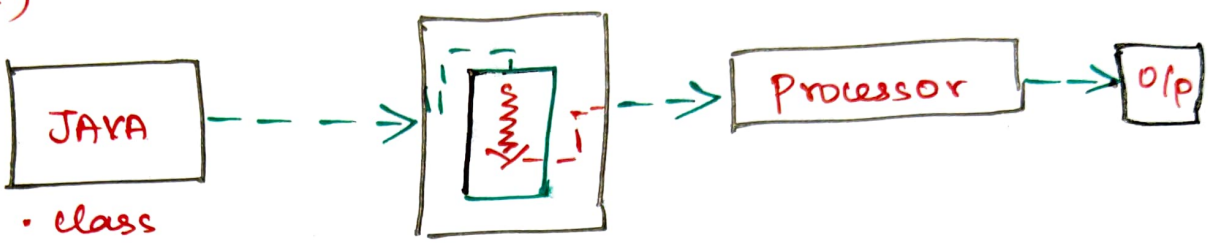
* Along with JAVAC there two are additional compilers in JAVA which is responsible for line by line compilation of the byte code file & along with this it also change the configuration of the byte code file.

Property 2:- No pointer concept.

* In JAVA there is no pointer concept
* In JAVA the pointer concept was eliminated from the programmer perspective but its task was done by JVM.
* By using pointer concept we can directly communicate with the processor that is why the performance of C is better than compare to any other prog. Language.

• exec
(C+/C++)

(Faster as there is no intermediary)



• class
(JAVA)

(compairtively slower than C)

Property 3 :- OBJECT ORIENTED PROGRAMING

* JAVA is Object oriented programming language any programming language that supports the concept of Object, class, Encapsulation, Abstraction Inheritence & Polymorphism is called as OOPS.

* By using these concept the application will get easy. C is procedure oriented.

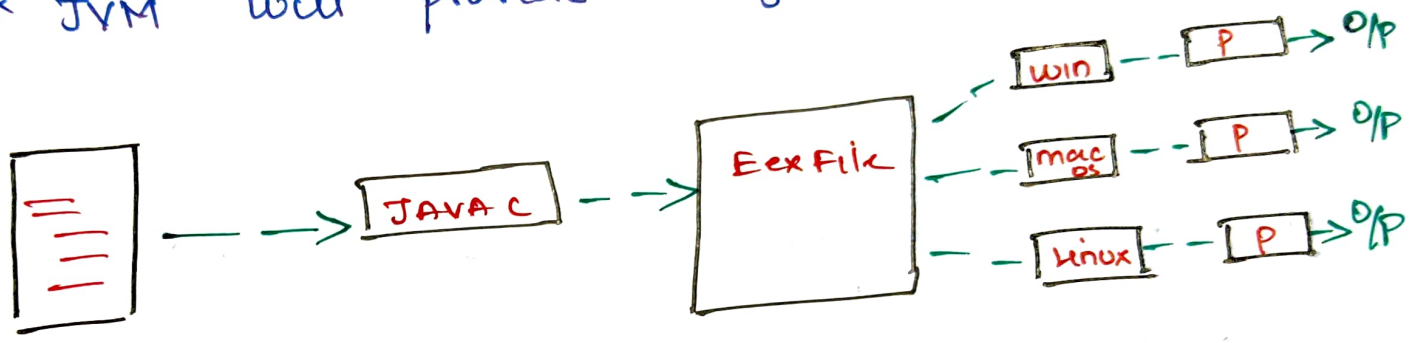
Property 4 : Simple programming Language.

* Java is easy to study when compared to C programming language.

Property 5 :- SECURE PROGRAMING LANGUAGE:

* Java is more secure when compared to C because in every program will be executed through JVM instead of direct communication with processor.

* JVM will provide layer of security.



WORA stands for Write Once Run Anywhere.

* In JAVA class file which is generated after compilation can be executed on any OS with the help of JVM.

* The property class file is to be generated only once any to be executed on any OS. For any Number of time is known as WORA principal.

Operators :-

CHECK THE MISSING NOTES OF OPERATORS AFTER VARIABLES

* Operators are the symbols which are used to perform some operations on the values.

* The values on which we perform the operations are technically called as operands.

eg :- 100 + 200 in the above expression '+' is operator and 100, 200 is operands.

In Java operators are classified into,

a) Arithmetic operator (+ , - , * , % , /)

b) Relational operator (< , > , >= , <= , == , !=)

c) Logical operator (&& , || , !)

d) unary operator (++ , --)

e) Bitwise operator (& , |)

f) Shift operator.. (>> , <<)

Arithmetic operator :-

* It used to perform mathematical operation on the values. Whenever multiple operators are present in single expression the JVM will follow BODMAS rule.

BODMAS

{}, (), ! , * , + , -

%

Types :-

* Addition - + * Subraction - - * Multiplication - *

* Division (Quotient) - / * Modulus (Remainder) - %.

Example :-

1) 10+10 = 20 2) 2+2 /2 = 3 3) 2*2 %2 = 2 4) 2%.2 = 0

5) 4-4 /4 = 3 6) 4+2-6 = 0 7) 9 %. 5 /2 = 2 8) 10 - 10 = 0

9) 10 x 10 = 100 10) 10%.2 = 0 11) 10 /2 = 5 12) 10*(2+10) = 120

13) 10*10+2 = 102 14) 2/2 = 1 15) "Hello" +10+20 = Hello1020

16) "Hello" + (10+20) = Hello30 17) 4*4+4 = 20 18) 8/2*10+2 = 42.

Program Example :-

```
class Prog8 {
    P.S.V.M ( ) {
        int a = 100, b = 50;
        S.O.P (a+b);   SOP (a*b);   SOP (a-b);
        S.O.P (a%.b);  S.O.P (a/b);  S.O.P (a+a);  SOP (a%.b+a);
        S.O.P (a/b+a*b);  S.O.P (a+'a');  S.O.P (a+"a");
        S.O.P ('a'+'a');  S.O.P ("a"+"a");
    }
}
```

O/P : 150 5000 50 0 2 200 100 5002 197

100a 194 aa

Interview Questions :-

1) Write a program to swap 2 numbers using a temporary variable.

```
class prog9 {
    P.S.V.M () {
int a = 100, b = 200;
S.O.P ("before swapping");
S.O.P ("a =", a);
S.O.P ("b =", b);

//Logic
    temp = a;
    a = b;
    b = a;

S.O.P("after swapping");
S.O.P ("a =", a);
S.O.P ("b =", b);
```

Output :-
Before swapping.
a = 100
b = 200
after swapping
a = 200
b = 100

// Logic, using a temp var we give the value of a to temp then b's value to a and a's value again to b.

// Hence swapped with temp var

2) Write a program to swap 2 no.s without using a temporary variable.

// Logic

a = 100 a + 200 b ; ⇒ a [300] ✗ gets erased
b = 300 a - 200 b ; → b [100]
a = 300 a - 100 b ; ⇒ a [200] ← new a = 200, b = 100

// Hence swapped without temp var.

Relational operator :-

* It is used to compare 2 values. every relational operator will return either true or false after comparing the values.

types

>	greater than	<	less than	>=	greater than equal to
<=	less than equal to	==	equal to equal to	!=	not equal to

Example :-

```
class prog10 {
    P.S.V.M ( ) {
        int x = 100 , y = 200 , z = 300 ;
        S.O.P (x>y);      S.O.P (y<z);
        S.O.P (x>=z);     S.O.P (y<=x);     S.O.P (x==3);
        S.O.P (x==z);     S.O.P (x!=y);     S.O.P (x>5);
        S.O.P (y!=200);   S.O.P (z<=100);   S.O.P (y>=200);
        S.O.P (y>200);    S.O.P (z!=x);
```

Output :-

false	false
true	false
false	true
false	false
false	true .
false	
true	
true	

difference b/w "=" & "=="

=	==
* It is assignment operator	* It is relational operator
* it is used to assign values to the var.	* it is used compare 2 values.
* it will assign Rhs to Lhs variable.	* it will compare Rhs to Lhs variable.
eg: a = 10	eg: 10 == 10 11 == 12

Shift operator :-

* it is used to perform operation on individual bits of the given values.

>> → right shift << ← left shift.

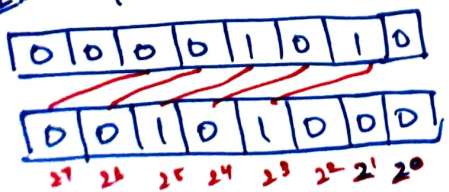
1) byte a = 10;
S.O.P (a << 2); // 40

Step 1 :-

10 → 1010

2 | 10
2 | 5 - 0
2 | 2 - 1
1 - 0

Step 2:



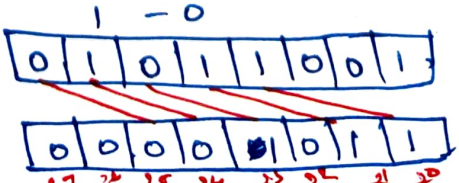
32 + 8 = 40

2) byte a = 89;
SOP (a >> 3); // 11

Step 1

89 — 1011001

2 | 89
2 | 44 - 1
2 | 22 - 0
2 | 11 - 0
2 | 5 - 1
2 | 2 - 1
1 - 0



8 + 2 + 1 = 11

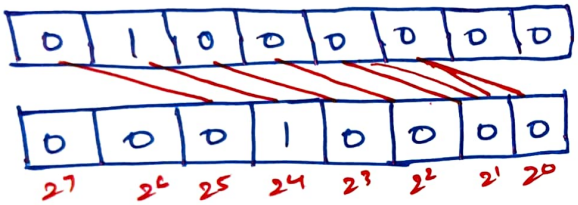
task

1) byte a=64;
SOP (a >> 2); // 16

2) byte b=7 SOP (b<<3);
// 88

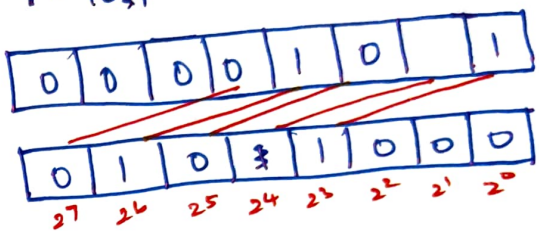
2 | 64
2 | 32 - 0
2 | 16 - 0
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
1 - 0

64 - 1000000



2 | 7
2 | 3 - 1
2 | 1 - 1
1 - 0

7 - 101



UNARY OPERATOR :-

* It will increase / decrease the value of A variable by 1

Types :- ++ (Increment) -- (decrement)

Pre ++ Pre --

post ++ post --

Examples :-

(1) int a=10;
int b = a++;
S.O.P (a); 11
SOP (b); 10

(2) int a = 10;
int b = ++a;
S.O.P (a); 11
S.O.P (b); 11

(3) int a=10;
int b = a--;
S.O.P (a); 9
S.O.P (b); 10

(4) int a = 10;
int b = --a;
S.O.P (a); 9
S.O.P (b); 9

Interview Question :-

1) int a = 10 [10 11 12 11 10] // 0

SOP (a++ + ++a - a-- - --a);

10 + 12 - 12 - 10

2) int a = 10 [10 11 10 9 10 11] // 11

int b = ++a - a-- - --a + a++ + ++a ;

++ - ++ - 9 + 9 + 11

S.O.P (a); // 11

S.O.P (b); // 11

3) int a = 10 [1 2 3 4]

int b = 2 [2 1 2 1]

int c = (--b + a++ + ++b - a++ + b-- - ++a ;

1 + 1 + 2 - 2 + 2 - 4

S.O.P (a); // 4

S.O.P (b); // 1

S.O.P (c); // 0

Task :-

1) int a = 10 [10 9 10 11 10] // 11 ?

S.O.P (--a - a++ - ++a + a-- + ++a);

9 - 9 - 11 + 11 + 11

2) int a = 10 [10 9 8 9 8 9 10 9] // 27

int b = a-- + --a * ++a - a-- + ++a + a++ + -a ;

10 + 8 - 9 - 9 + 9 + 9 + 9

S.O.P (a); 9 //

S.O.P (b); 27 //

3) int a = 1

int b = 2

int c = 3

int d = ++a + --b - c++ + a-- - b++ + ++c - ++b + --c

- b++ + c-- ;

③ int a = 1; 1 2 2

int b = 2; 2 3 3 4

int c = 3; 3 4 3 4 3 2

int d = ++a + --b - c++ + a-- - b++ + ++c - ++b +
2 + 1 - 3 + 2 - 1 + 5 - 3 + 4 - 1 + 3 - 3 + 3
--c - a++ + --c - b++ + c--;

3 3 + 1 + 2 + 3 + 3 + 3 = 9

S.OP (a); 2

S.O.P (b); 4

S.O.P (c); 2

S.O.P (d); 9

Arsh note :- CHECK THE MISSING NOTES AFTER VARIABLES:

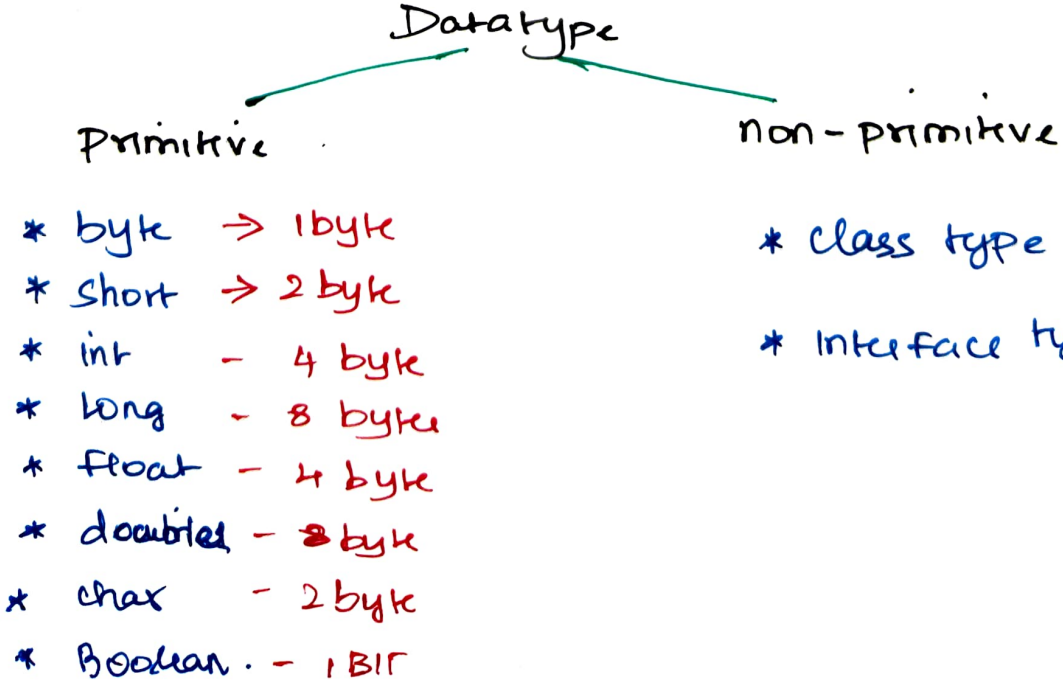
VARIABLES

* A variable is a container which is used to store some value. The type and size of value will be decided by the datatype.
* A datatype will be used while creating a variable.
* In Java datatype are of 2 types.

a) primitive datatype b) non-primitive datatype.

note :-

* In Java every datatype is a keyword.
* byte, short, int & long are used to store non-decimal numbers.
* char is used to store single character in '' single quotes.
* boolean is used to store true / false.
* float & double are used to store decimal numbers.



* As a programmer we can create a variable in 2 approaches.

approach 1 :-

Syntax :- datatype variablename ; (var declaration)
variablename = value ; (var initialization)

example :-

```
int a ;
a = 100 ;
```

a [100] 4byts

```
char c ;
c = 'A' ;
```

c [A] 2byts

```
double b;
b = 100.12;
```

b [100.2] 8 byts

```
boolean d ;
d = false ;
```

d [False] 1 bit.

approach 2 :-

Syntax :- datatype variablename = value ;
(variable declaration / initialization)

example :-

```
int a = 100 ;
```

a [100] 4byts

```
char c = 'A' ;
```

c [A] 2byts

```
double b = 100.12 ;
```

b [100.12] 8 byts

```
boolean d = false ;
```

d [False] 1 bit.

Program Example 1 :-

```
class A
{
  P.s.v main (String [] args)
  {
    int a = 100
    S.O.P (a);
    S.O.P (a+10);
    " " (a+'a');
    S.O.P (a+'a');
    " " (a+"a");
    " " ('a'+'a');
    " " ('a'+"a");
    " " ("a"+"a");
  }
}
```

O/P :
100
110
200
197
100a
194
aa
aa

Program Example 2 :-

```
class program2
{
    P.S.V.$main (String []args).
    {
        int var1;
        double var2;
        char var3;
        boolean var4;

        var1 = 145
        var2 = 145.063
        var3 = 'Q';
        var4 = False;

        S.OP ("var1=" + var1);
        S.O.P ("var2 =" + var2);
        S.O.P ("var3 = " + var3);
        S.O.P ("var4 = " + var4);
    }
}
```

O/P :- Javac Prog6.Java;

Java Prog6

var1 = 145

var2 = 145.063

var3 = Q

var4 = False

Important Conclusions on variables

Conclusion 1:- As a programmer we cannot use a variable without assigning values then we will end up getting CTE.

examples :-

```
class program7
{
    P.S.V.M ( )
    {
        int a;
        S.O.P ("a");
    }
}
```

O/P :-

CTE (variable a might not have been initialized)

Conclusion 2 :- In a single program, we can create any number of variable to store data.

Conclusion 3 :- As a programmer we cannot create multiple variables with same name, if we try we will cause CTE.

conclusion4 :- As a programmer we can create multiple variables with same type / datatype.

Conclusion 5 :- As a programmer we can reinitialize a value of an variable.

Eg :-
```
class program {
    P.S.V.M ( ) {
        int num = 100;
        S.O.P (num);
                 |--> 100
    int num = 200
        S.O.P (num);
                 |--> 200
```
The value is reassigned, existing will get deleted.

Conclusion 6 :- As a programmer we cannot change / modify / reinitialize the final variable.

syntax :-

final int = 100; // n is the final var

-> we cannot change if changed causes CTE.

Conclusion 7 :- Variables contain single copy in memory, means variables can hold only one value at a time.

int n = 10 → valid

int n = 10, 20, 30 → ✗ valid

└ causes CTE

Logical Operators :-

* It is used to compare 2 conditions, every logical operator will return true or false based on the comparison.

Types :- && Logical and
|| logical OR
! NOT

WORKING OF && :-

cond1 && cond2	O/P
true true	true
true false	False
False true	False
false false	False

NOTE :- The output will be true only if all the conditions are true if any one condition is false then the output will be False.

WORKING OF (OR) ||

cond1 || cond2	O/P
true true	true
true False	true
False true	true
False False	False

NOTE :- The output will be true only if any one of the condition is true, if all the conditions are false then the ouptput will be false.

working of ! :-

input	output
!(true)	false
!(false)	true.

```
Class Prog11 {
    P.S.V.M ( ) {
    int x=100 , y=200 , z=100
    S.O.P (x==100 && x<y);          True
    S.O.P (x>y || x<=y);            true
    S.O.P (x>=z && z<=99);          false
    S.O.P (z>x || y!=x);            True
    S.O.P (z!=-100 && x>0);         True
    S.O.P (z!=z || y==x);           False
        S.O.P ( ! (10==10));        false
        S.O.P ( ! (10>10));         true
        S.O.P ( ! (x==100) && ! (x<y) );   False
        S.O.P ( ! (z>x) || !(y!=x));       true
    }
}
```

BITWISE OPERATOR :-

* It is used to perform operation on individual bits of the given number.

Types :-

& → bitwise and | bitwise OR

Ex:- int a = 100 , b = 7 ;

S.O.P (a & b) ; S.O.P (a | b) ;

Step 1:-

a - 2|100
2|50 - 0
2|25 - 0
2|12 - 1
2|6 - 0
2|3 - 0
1 - 1

1100100

b - 2|7
2|3 - 1
1 - 1

111

Step :- 2

1100100
& 0000111
0000100

1100100
| 0000111
1100111

Step:-3

0000100
$2^6 2^5 2^4 2^3 2^2 2^1 2^0$
0+0+0+4+0+0

4 ||

1100111
$2^6 2^5 2^4 2^3 2^2 2^1 2^0$
64+32+0+0+4+2+1

103 ||

TASK ON BITWISE OPERATOR:-

1) int a = 21 ; int b = 56 ;

S.O.P (a & b) ; SOP (a | b) ;
↳ 16 ↳ 61

STEP 1 :-

2|21
2|10 - 1
2|5 - 0
2|2 - 1
1 - 0

a - 10101

2|56
2|28 - 0
2|14 - 0
2|7 - 0
2|3 - 1
1 - 1

b - 111000

STEP 2 :-

010101
& 111000
010000

010101
| 111000
111101

STEP 3 :-

010000
$2^5 2^4 2^3 2^2 2^1 2^0$
0+16+0+0+0+0

16 ||

111101
$2^5 2^4 2^3 2^2 2^1 2^0$
32+16+8+0+0+1

61 ||

2) int a = 15 , b = 68 ;

S.O.P (a & b) ; SOP (a | B) ;
↳ 4 ↳ 79

2|15
2|7 - 1
2|3 - 1
1 - 1

a - 1111

2|68
2|34 - 0
2|17 - 0
2|8 - 1
2|4 - 0
2|2 - 0
1 - 0

b - 1000100

STEP - 2 :-

0001111
& 1000100
0000100

0001111
| 1000100
1001111

STEP - 3 :-

0000100
$2^6 2^5 2^4 2^3 2^2 2^1 2^0$
0+0+0+0+4+0+0

4 ||

1001111
$2^6 2^5 2^4 2^3 2^2 2^1 2^0$
64+0+0+8+4+2+1

79 ||

int a = 121 b = 99
↳97 ↳123

2 | 121
2 | 60 - 1
2 | 30 - 0
2 | 15 - 0
2 | 7 - 1
2 | 3 - 1
1 - 1

1111001

2 | 99
2 | 49 - 1
2 | 24 - 1
2 | 12 - 0
2 | 6 - 0
2 | 3 - 0
1 - 1

1100011

1111001
& 1100011
1100001

1100001
$2^6 2^5 2^4 2^3 2^2 2^1 2^0$
64 32 0 0 0 0 1

97 ||

1111001
| 1100011
1111011

1111011
$2^6 2^5 2^4 2^3 2^2 2^1 2^0$
64 32 16 8 0 2 1

123 ||

AISH NOTE: CHECK THE MISSING NOTES BEFORE VARIABLES

TASK ON ~~UNARY~~ OPERATORS :-

1) int a = 100

(++a - a-- + --a + a++ + --a - a-- + a++ + --a

SOP (a) ~~101~~ ~~100~~ ~~99~~ 98 98

SOP (b) ~~-101~~ + 99 + 99 + ~~99~~ ~~-99~~ + 98 + 98 = 395 396

2) int x = 1 ; ~~x 3 2 x 0~~
int y = 2 ; ~~2 3 4 5 6~~
int z = ++x + ++y - x++ - y++ + --x + ++y - x-- + --x

SOP x 0 ~~2~~ + ~~3~~ ~~-2~~ ~~-3~~ + ~~2~~ + 5 ~~-2~~ + 0 + 5 + y++ ;

SOP y 6

SOP z 10

3) int a = 51 ;
int b = 66 ;
SOP (a & B) ; 2
SOP (a|B) ; 115

2 | 51
2 | 25 - 1
2 | 12 - 1
2 | 6 - 0
2 | 3 - 0
1 - 1

2 | 66
2 | 33 - 0
2 | 16 - 1
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
1 - 0

0110011
& 1000010
0000010

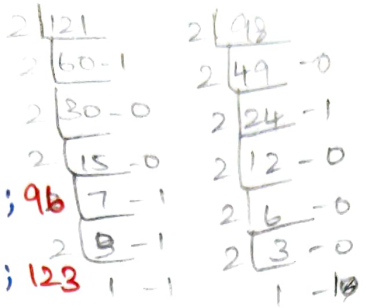
2

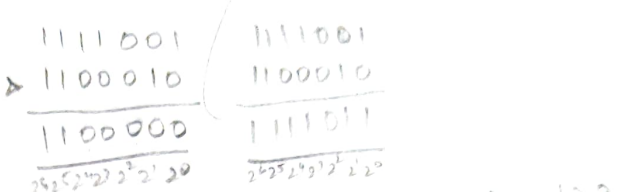
0110011
| 1000010
1110011

64 + 32 + 16 + 2 + 1

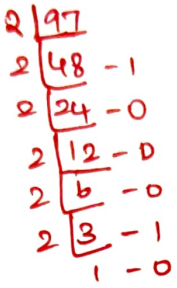
115

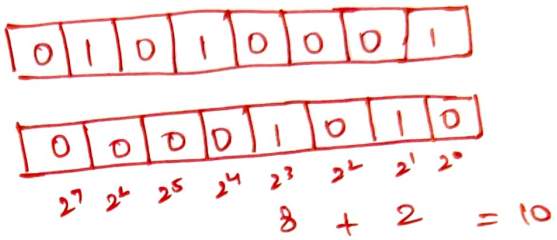
4) int a = 121;
int b = 98;
SOP (a&b); 96
SOP (a|b); 123





5) byte a = 97
SOP(a >> 3);





ASSIGNMENT OPERATOR:-

* It is used to assign or reassign values to a variable.

Operator	Example	Same as Ex.
=	n=5	n=5
+=	n+=3	n=n+3
-=	n-=3	n=n-3
*=	n*=3	n=n*3
/=	n/=3	n=n/3
%=	n%=3	n=n%3

Example

① int a = 10; 10 20
a=a+10 or a+=10;
SOP (a); 20

② int b = 10; 10 5
b= b-5; b-=5
SOP (b); 5

③ int c = 50 50 200
c = c*4 or c*=4;
SOP (c); 200

④ int d = 100; 100 20
d= d/5; or d/=5;
SOP (d) 20

⑤ int e = 10; 10 4
e= e%6 or e%=6;
SOP (e); 4

Control Statement :-

* Control statements are used to control the flow of execution of a program.

* Control statement is classified into 5 types,

a) decision making statement (IF, IF-else, else-IF ladder, nested if)

b) selection statement (switch)

c) looping statement (for, while, do-while)

Decision Making statement :-

1) IF :- syntax :-

```
IF (condition)
{
- - - -
- - - -
}
```

IF block executes when the given condition is true.

example :

```
class Demo {
P.S.V.M( ) {
S.O.P ("hi Main");
IF (10==11){
   S.O.P ("hi hello");
}
S.O.P (Bye main);
```

O/P :-
hi main
bye main

2) IF-else :- syntax :-

```
IF (condition) {
- - - -
- - - -
- - - -
}
```

IF block executes when cond is true.

```
Else {
- - - - -
- - - - -
}
```

else block executes when cond is False.

Example :-

```
class demo {
P.S.V.M ( ) {
S.O.P("Hi Main");
IF (10>10) {        checks cond 10>10 False
   S.O.P ("Hi Dinga");
}
else {
   S.O.P ("Bye Dinga");
}
S.O.P("Bye main");
}
}
```

O/P :-
Hi main
Bye Dinga

Example 3 :-

WAP to read a num from user and preform the below operation if the num is greater than 100 increase by 1 else decrease by 1.

```
class Prog13 {
    P.S.V.M() {
    int num = 90;
     S.O.P("num before:" + num);
    IF (num > 100) {
        num = num + 1 (or) num++;
    }
    else {
       num-- or num = num - 1;
     }
     S.O.P ("num after:" + num);
  }
}
```

O/P :-

num before : 90

num after : 89

Example 4

WAP read a num from user & perform the below operation, if the num is greater or equal to 100, then increase the num by 30%. else decrease by 40%.

```
class prog {
    P.S.V.M() {
    int num = 100
    S.O.P ("num before=" + num);
  IF(num >= 100) {
        num = num * 30/100 + num
  }
  else {
      num = num * 40/100 + num
   }
   S.O.P ("num after=" + num);
}}
```

100 130

O/P :-

num before = 100

num after = 130