Command line Argument:-

- There are 3 command line argument.
    1. Single line command
    2. Multi line command
    3. Java DOC command

1:SINGLE LINE COMMAND:-

- It can be achived by "//"symbol.
- It is used to provide description about the code in a single line.

EX:- //class declaration

```
Class comp {
                }
```

2: Multi line command :-

- To provide description about the cod more than one line be can use multi line command .

```
EX :-/* function party in our collage of P N S college */
Class function{
        Public static void main ()
{
System . out. PrintIn(welcome to party);
}
}
```

3.JAVA DOC COMMAND:-

- It is used to provides document details about the method or class.
- EX :-/* FUNCTION PATY IN OUR COLLAGE OF P N S SCHOOL OF ENG.&TECH.
- @AUTHOR
- @CLASS TYPE
- @EXEEPTION
- @METHOD
- @//LIBRARY*/

OPERATOR:-

1. Arithmatic operator
2. Conditional operator
3. Logical operator
4. Unary operator
5. Assignment operator
6. Ternary operator
7. Bit wise operator

- 1:- Arithmetic operator :-
  - It is used to perform arithmetic operation .
  - The operation are (+, -,*,/,%) add ,sub multiplication division ,modulus.
    - Duotient (“/”) ->division
    - Reminder (%) →modulus

```
Class Arithmatic {
        Public static void main (String args [])
            {
            Int a =5;
            Int b=6;
        System . out . print ln(a+b);
        System . out . print ln(“a+b is equal to);
}
}
```

EX:-

```
        Class Arithmatic {
                Public static void main(String args[])
                {
                        Int a=5;
                        Int b=6;
                        C =a-b;
                        System .out .print ln (a-b);
                        System .out .print ln("a-b is equal to" +c);
                }
}
```

EX:-

```
Class Arithmatic {
        Public static void main ( String arss[])
        {
        Int a=5;
        Int b=6;
        C=a*b;
System .out .print ln(a*b);
System .out .print ln("a*b is equal to"+c);
}
}
```

EX:→ class arithmetic {

Public static void main(string args[])

{

Int a=5;

Int b=6;

C =a/b;

System .out .print ln (a/b);

System .out .print ln("a/b is equal to"+c);

}

}

EX :-Class Arithmatic {

Public static void main (String args[])

{

Int a=5;

Int b=6;

C =a%b;

System .out .print ln (a%b);

System .out .print ln("a%b is equal to" +c)

}

}

EX:-

```
Class Arithmatic {
    Int a=6;
    Int b=7;
Public static void main (String artgs [])
    {
    Int c =a+b;
System .out .print ln (" The sum of two no's is"+c); //13
Int d =a-b;
System .out .print ln("The sub of 2 no's :"+d); //-1
Int e=a*b
System .out .print ln("The multiplication of 2 no's:"+e );//42
Int x=1;        int y=a/X;
Int y=a%X;
System .out .print ln("The division of 2no's :");
System .out print ln (" The modulus of 2 no's :");//0
}
}
```

OPERATOR OVER LODEING:-

- Java does not support operator over loading. The only aspect of java ,which comes to operator over loading is using '+' operator.
- In terms of number , it proforma addition operation ,where ever a string comes into a picture . it will act as a concatenation operation .

EX:→

```
Class operator {
        Public static void main ( String args[] )
        {
        Int a =10;
        Int b=20;
    System .out .print ln (a+b);
        String S1= "P N S" ;
        String S2= "college";
    System .out. print ln ( String S1+ " "+ String S2);
        Int  sum = a+b;
System .out . print ln ("sum:"+sum);
}
}
```

EX→

```
    a= 50;
int String S1=computer
  int b=10
int b=10
String S2= scinence
System . out .print ln (a + b+ String S1)
a+ String  S1+ String S2
b+a +string s1+a+b
String s1+ string S2+(a+b)
```

- WAP FIND AREA OF CIRCLE ?

```
Class Circle {
        Public static void main ( String args [])
        {
        Float radious =3.0f;
        Float pie =3.141f;
        Float result =pie *r*r;
System .out .print ln ("The area of circle is :"+result);
}
}
```

- Wap find the area of rectangle ?

```
Class rectangle {
        Public static void main (String ares[])
        {
                Int l =6;
                Int b=5;
                Int r =l*b;
System .out .print ln ("the area of rectangle is:"+r);
        }
}
```

CONDITIONAL OPERATOR:-

- Conditional operator return bullean value .
- It is used for comparession or variatation purpose . (<,>,<=,>=,== ,!=)
- Less than ,greator than ,less than equal to ,greater than equal to, not equal to).

1) WRP ADDING 2 NO ‘S IN JAVA PROGRAM

```
Class Arithmatic {
        Public static void main (string args [])
{
        Int x=2;
        Int y=3;
        Int z=x+y;
System .out .print ln(“the sum of x&y is :”+z);
}
}
```

o/p:→the sum of x&y is :5

2) WAP SUB 2NO’S IN JAVA PROGRAM .

```
Class Arithmatic {
        Public static void main ( string args[])
{
        Int x=3;
        Int y=2;
        Int z=x-y;
s.o.p(“the sub of x&y is :”+z);
        }
}
```

3) WAP MULTI 2 NO’S IN JAVA PROGRATM.

```
Class Arithmatic {

Public static void main (string args [])

{

        Int x=2;

        Int y=3;

        Int z=x*y;

System .out .print ln (“the multi of x&y is :”+z);

}

        }
```

Primitive data type & declaration :-

- One primitive data type can declare in java.
- Class & main method(public static void main())
- It can also be declared with in main method i. e (public static void main ())
- It can also declared in with in a used defined or pre-declared function .

Local variable for primitive data type :-

- If a variable & a primitive data type is declared in main method then this data type or variable called as local variable .
- Similarly global variable .
- If a variable is declared with in class &main method is called as Global variable .

Ex:→

```
Class A{
Int a;
Char c;                    global variable
String s;
Public static void main ( string args [])
{
        Int a1;
        Char c1;                    local variable
        String s2;
}
}
```

String :-

Combination of char .is called as string .

- A string is compiled of an 8byte object header (4-byte sync block &a 4 -byte type decriptor).
- The maximum length of string in java in 0 to 214783647.
- Char :-
  Byte -2 byte
  Rang-0 to 65535

- How to address memory location ?
  - The memory location are addressed from 0 to2^k-1 I.e a memory has 2k addressable locations . and this the address space of the computer have 2k addresses.

-:CASTING & TYPE CASTING :-

UP CASTING :-

- The process of creating obj for the sub -class with the help of super class reference variable is said to be up-casting .
- The up-casting object refer to the method & variable of super class . if the method is over idied in the sub class then it will takes sub class .
- If it is not overided than it will take super class implementation .

Casting :-

- Casting is a method & process that converts a particular data type in both ways space manually & automatically .
- The automatic convert perform by a program.

Narrowing type casting

DOUBLE	FLOAT	LONG	INT	SHORT	BYTE

WIDENING

Types of casting :-

I. Narrowing type casting
II. Widening type casting

Type casting :-

- Convert a value from one data type to another data type is called as type casting .

Narrowing type of casting :-

- Converting a higher data type into lower data type one data type is called as narrowing type of casting.
- It is also known as casting up or explicity convert .

Doble	Float	Long	Int	Short	Byte

WIDENING TYPE CASTING :-

- Converting a lower data type into a higher one is called as widening type casting .
- It is also known as casting down or implicity conversion .

Byte	Short	Int	Long	Float	Double

Ex:-

```
Public class casting {
        {
Public static void main( string args [])
{
Int x=7;
//conversion of integer to long
Int y=x;
//convert long to float;
Int z=y;
s.o.p (“Before conversion int value :”+x);
s. o.p(“After conversion int to long value :”+y);
s.o.p(“after conversion of long to float value :”+z);
}
}
}
```

O/P:-

Before conversion ,int value =7

After conversion in to long value =7

After conversion of long to float value =7

Control flow :-

What is statement ?

- A single line of code is known as statement &its end with semicolumn(;).

How do use control flow ?

- Executing a single line of code within a minimum space of time is called as control flow .

How to use loop?

- Executing a same line of code repeatly is known as loop .

Control flow :-

- Control flow describe the order is which the statement will be executing at run time .

Flow control

Selection statement	interative statement	transfers statement
1.if	1.while	1.Break
2.if -else	2.do while	2.continue
3.switch	3.for loop	3.return
	4.for eachloop	4.try catch finally
		5.as set

I. If statement :-

```
Class loop{
Public static void main (string arss[])
{
        Int x=17;
        If (<18)
        System.out .print ln ("I am not a adult");
            If (x>18)
        System .out .print ln ("I am a adult");
}
}
```

1) EXAMPLE OF IF ()

```
Class mimic {
Public static void main (String args [])
    {
      Int x=6;
      Int (x>10)
System .out .print ln (" your now going to become a student");
      If (x<10)
System .out .print ln ("your now a child");
}}
```

2. Example of if ()_else

```
Class flow
{
        Int x=18;
        If  (x>18);
{
System .out .print ln("you are now a Indian citizen")  ;
}
Else
{
System .out .print ln("you are not a Indian citizen")
}
}
```

Example no.3:-

```
Class flow {
Public static void main (String args[])
{
Boolean b=true ;
If (b=true );
{
System .out .print ln ("your right man");
}
Else
{
System .out. print ln ("your wrong man");
}
}
}
```

Example no .4:-

```
Class flow
{
Public static void main (String args [])
{
Boolean b=false ;
If (b==false );
{
System .out .print ln(“ your right man”);
}
Else
{
System . out .print ln (“your wrong man”);
}
}
}
```

Ex :-5

```
class flow {
        Public static void main (String args []){
        Int b=10;
        If (b==20){
    System .out .print ln(“the no is 20”);
        }
        Else
        }
    System .out .print ln (“the no is 10”);
        }
    }
}
```

Ex 6:-

```
class flow {
        public static void main (String args [])
{
String s1= “p n s”;
If (s1=p n s )
        {
System .out .print ln (“This is college name”);
}
Else
{
System .out .print ln (“This is not a college name”);
}
}
}
```

Ex :-6

```
Class flow {
        Public static void main (String args )
        {
                Char = “p n s”;
                If (ch ==p n s){
System .out .print ln (“This is college name”);
        }
        Else
        }
System .out .print ln (“this is college name”);
}
}}
```

-:Even odd number's EX :-

```
Class even
{
Public static void main (String args [])
{
        Int x=18;
        If (x%2==0)
{
System .out .print ln ("x is even no");
}
Else
{
System .out .print ln ("x is a odd no");
}
}
}
```

```
Class odd {
        Public static void main (string args [])
{
        Int x=18;
        If (x%2!=0)
{
System .out .print ln ("x is a even no");
}
Else
{
System .out .print ln (" x is a odd no");
}}}
```

-:Switch case statement :-

Syntax :-

```
Switch (x)
{
    Case 1:
    Action 1:
    Break ;
Case 2;
    Action 2;
    Break ;
Case 3;
    Action 3;
    Break ;
    :
    :
Case n:
    Action n;
    Break ;
Default :
    Default action ;
}
```

→ if several option are available then it is never recommended to use if else statement ,

In that case we should use switch case statement .

→ The advantages of this approach is readability will be improved.

→The valid argument types for switch statement are bite ,short ,integer ,char but this rule is applicable unit 1.4 version .

→But from 1.5 version onwards corresponding wrapper classes & enum types allowed .

1.4 v	1.5v	1.7v
Byte	byte	
Short	short	
Int	char	String
Char	Int	
	+ ,enum	

- ➔ Colibresses are mandatory.
- ➔ Switch case is the only place where colibresses are mandatory .
- ➔ Within the switch both case & default are optical.
- ➔ Ex :-

```
Class enum {
        Int x =10;
        Switch (x){
        Print x;
   }
}
```

- ➔ Every statement inside switch should be under some case or default . i.e we can't right independent statement inside the switch .
- ➔ Every case level should be compile time constant if we are taking one variables as case level than we will get compile time error .

-:Switch ()EXAMPLE :-

```
Public class college {
                    Int day =6;
Public static void main (String args[])
{
Switch ( day ){
Case 1:
System .out .print ln (“today is Monday” );
Break ;
Case2:
System .out .print ln (“today is Tuesday”);
Break ;
Case 3:
System .out .print ln (“today is Wednesday”);
Break ;
Case 4:
System .out .print ln (“today is Thursday”);
Break ;
Case5:
System .out .print ln (“today is Friday”);
Break ;
Case6:
System .out .print ln (“today is Saturday”);
Break ;
Case 7;
System .out .print ln (“today is Sunday”);
Break ;
Default :
System .out print ln (“today is on day now”);    }    }        }
```

Do while ():-

Syntax :-

```
Do
{
    Statement
}
While (condition )
```

Properties :-

→ it is an exit check loop .

→ do while execution the statement at less one even the condition is returning false.

Ex :-

W A P to print the no from 1 to 100 !!

```
    Class loop {
Public static void main (String args[])
    {
        Int i=1;
Do
{
System .out .print ln (“print I”);
I++;
}
While (i<=100)
System .out .print ln (“Throw an error”);
}
}
```

While :-

```
Syntax :-        While condition{
                        Statement ;
                        }
```

- ➔ It is an entry check loop , while loop will always execpect the condition is return loop.
- ➔ It will exicuit all the statement inside the loop until the condition become false .

Ex :-

```
Class loop 1
        {
Public static void main (String args [])
{
        Int i=1;
        While ( i<=100)
        {
System .out .print ln ( “print I”);
        I++;
        }
System . out .print ln (“loop execution end”);
}
}
```

Ex :-

```
Class loop
{
Public static void main (String args [])
        {
Int i=100;
While ( i<=100)
{
System .out .print ln (“print I”);
I++;
}
System .out .print ln (“loop execution end”);
}       }
```

for loop:-

syntax :-

```
for (intilization ;condition ;increment /decrement );
{
Statement ;
}
```

Ex :-

```
Class loop3
{
Public static void main ( String args [])
{
        Int i=1;
        For (int I;i<=10;i++);
        {
System .out .print ln (“print I”);
}
        System .out .print ln (“loop executing end”);
}
}
```

What is class ?

→ class is the blue print of program .

→ class is use to defind state & behaviour of the program.

→ in other-words class contain method ,variable inside it .

Ex :- student ; logic

Syntax of class :-

Class class name

{

Variable & method name ;

}

Object :-

- ➔ Object are real time entity provided for the class .
- ➔ It is used to access method &variable inside the method .

Syntax for the object :-

Class name .object name =new class name ();

Eg :-

Class name object name = new class name ();

Student st=new student (); keyword construction if cla

- ➔ We can create 'n' no of object for a single class.

Method :-

- ➔ Methods are the collection of the statement which is used to perform some specific operation.
- ➔ Method allows as to reuse the code the inside of refrying or retyping .
- ➔ The main ablvantages of method is reusability & code optimization .
- ➔ Main method is the building block of the program .
- ➔ Hear executing always start from main method .
- ➔ If the main method is not defined in the class then during execution we will get compilation error .
- ➔ To defined a method we need to use 6 components:-
    1. Access specifier
    2. Return type
    3. Method name
    4. Parameter
    5. Paraenthesis /method block (())
       (A,b)
    6. Return keyword

Syntax:-

Access specifier return keyword method name (parameter)

{

Return value

}

Access specifier :-

→ it is used to define the sequirity & visibility for both methods & variables .

- Types of access specifier :-

i. Privet → it is define within class .
ii. Default → within same package with folder .
iii. Protected → out side package with inheritance .
iv. Public → can be used any where .

❖ Return type :-

- It is mandatory to define return type for method .
- Return type is used to defined what type of value will be return as out from the metho .
- We can define return type as primitive data type as return type then that method should written variable or value of the same return type .

```
Ex :-
        M2()
                {
                        Return 10 ;
                }
```

```
Ex :-
    Int m1 ()
                {
                        Int a=2;
                        Return 2/a;
                }
```

- If a method is define with primitive data type as return type then the method should return object of the same class which is define in the return type .
- When we wand to get a bunch of data as out put we can define the method with non -primitive data type as return type .

```
Ex :-
        Test m1 ( )
                        {
                        test t=new test ();
                        return t;
                        }
```

Ex 2:-

```
String s1()
        {
        String s= “college”;
        Return s;
        }
```

- If a method is defined as void return type then the method don’t return anything .
- Return keyword is not allowed for void type method .

❖ Method name :-

- Method name should be uniqucinside the class .
- Duplicate method name are not allowed in java .
- Method name should not start with lower case letter .
- Camel case letter us are allowed .
- Alpha numeric value are allowed .
- Spaces are not allowed in java .

  Ex :-

  m1( )X

  M1( )correct

  $set ()correct

  Void set prior( )×

  Voidset-prior ( )correct

  1-college( )correct

  1-college staff ( )×

❖ Return keyword :-

- Return key word is mandatory for all method depend with return type other than void
- It should be the last statements inside the method .
- Once return key word has been executed the control will given back to the place where the method is called .
- Return key word should return type to value which is matching with return type .
- Types of method .
- There are 2 types of method
  i. Static method
  ii. Non static method

1. Static method :-
   - Any method define with static keyword is said to static method .
   - We will get memory first .
   - It will get memory in static pool .
   - Static method can be access by there ways.
     i. Within the same class we can access it directly .
     ii. We can access it with the help of class name .
     iii. We can access it using object .
2. Non -static method :-
   - Any method which is defined without static keyword is said to be non static method .
   - It will get memory after object creation .
   - It will get memory in heap area .
   - It has multiple copies of memory based on no. of object create .

Ex :- class test

```
{
//static void main m1( )
{
System .out .print ln (" static method");
}
Non  //void method (  )
{
system .out .print ln ("non static method");
}
Public static void main (string args [])
{
//m1( );
Test m1( );
//test t1=new test ( );
{
T1 .m1(  );
T2 . m1( );
}
}
```

Static method →

4. Parameter :-
   - Parameter are local variable which as the access within that method only .
   - It should be defined inside the method or signature .
5. Argument :-
   - Argument of the data given to parameter .
   - It should be match with total number of parameter ,data type parameter ,& sequence of parameter .

   Ex :-

```
Class method -2
                {
        Int a ;
        Int b;
// int add (int a,int b)
                {
    Int sum =a+b;
    Return sum;
                }
//int sum ( int a, int b)
                {
    Int sub =a-b;
    Return sub ;
                }
Public static void main ( string args [])
    {
Method -2 m2=new method -2( );
Int return 1=m2 add (10,20);
System .out .print ln ("add ="+ result 1);
  Int result 2=m2.sub(10,20);
System .out .print ln ("sub="+result 2);
                }
    }
```

Ex :-class method -3

```
{
Int a;
Int b;
Int multiplication (int a, int b)
{
Int multiplication =a*b;
Return multiplication ;
}
Int division ( int a ,int b)
{
Int division =a/b;
Return division ;
}
Public static void main ( String args [])
{
Method _3 m3=new method -3( );
Int return 1= m3.multiplication (10,20);
System .out .print ln (“multiplication =”+result 1);
Int result 2=m3.devision (10,20);
System .out .print ln (“division =”+result 2);
}
}
```

Constructor :-

- Constructor are special type of method which is used to initialize the object .
- The main use construction is to assign the value to the global variable .

Rules of defining construction :-

- Construction name should be same as class name .
- Construction doesn't have any return type .
- We should not write any business logic inside construction .

  How to call constructor :-
- Construction will be invoked during the object creation .

  Type of constructor :-

  There are two types of constructor .
  1. Default constructor
  2. User defined construction

Default constructor :-

- It is system generated constructor . which is created by the compiler during compile type.
- Default constructor will be created only where there is no. constructor is used to assign default values .

Compilation time :-

```
Class Test
        {
        Test ( )
        {
System .out .print ln ("default constructor");
        }
Public static void main ( )
        {
Test    t =new test ( );
        }
}
```

Ex :-2

```
Class Test {
        Int x;
Public static void main (String args [])        {
System .out .print ln ( "print x=20");
        }       }
```

User defined constructor :-

- The constructor which as created by the developer during coding time is side to be user define constructor .
- It is also called as parameterise constructor .
- It used to assign user defined value to global variable .

Ex :-

```
        Public class Demo
                        {
                Demo (   )
                        {
System .out .print ln ("Default constructor");
                        }
        Demo (int a )
                        {
System .out .print ln ("one -argument constructor");
                        }
        Demo (int a ,int b)
                        {
System .out .print ln ("two argument constructor");
        Demo ( char  c)
                        {
System .out print ln ("character type argument");
                        }
        Demo (String  s)
                        {
System .out .print ln (" string type constructor");
                        }
Public static void main (String args [])
                        {
Demo d=new demo (    );//default constructor .
Demo d1 =new demo (10);//1-arg constructor .
Demo d2 =new demo(10,20);//2-arg constructor .
Demo d3=demo ('a');//character type args constructor .
Demo d4=new demo (" my college");//string constructor type constructer .
                        }
                }
```

Ex :-

```
        Public class student
                        {
                Student (   )
                        {
System .out .print ln ( “default constructor”);
                        }
                Student ( int a )
                        {
System .out .print ln (“one argument constructor”);
                        }
                Student (int a ,int b )
                        {
System .out .print ln (“two argument constructor”);
                        }
                Student ( Char c)
                        {
System .out print ln (“character type argument”);
                        }
                 Student ( string s)
                        {
System .out print ln (“ string type constructor”);
                        }
Public static void main (String args [])
                        {
        Student s =new student ();
        Student s1=new student (  10 );
        Student s2 =new student
```

-:OOPS CONCEPTS & TERMINOLOGY :-

What is oop?

- Java coding is completely dependent on object so ,it is said to be object oriented programming language .
- Oop is catagorieg into 4 parts .
  1. Inheritance
  2. Polynerphism
  3. Encapsalation
  4. Abstraction

1. Inheritance:-
   - The process of acquiring the properties from one class to another class is said to be inheritance .
   - To archive inheritance in java we need to use extends keyword .
   - The class which given properties to another class is said to be super class .
   - It is also called as base class /parent class .
   - The class which gates properties from super class in side to be sub class .it is also called as child class or /derived class.
   - The main advantage of inheritance is code reusability .
   - Inheritance is also known as relationship .
   - When we create object for super class it can access only the super class method & variable .
   - When we create object for sub-class it can access both super class & sub-class .

EX :-with inheritance

```
Class a{
        }
Void add (  ){
        }
Void sub (   ){
        }
Class b extends a{
Void mul (  ){
        }
Class c extends b{
Void div (  ){
        }
```

```
Void mod (   ){
          }
    }
}
Class run {
Public static void main (  ){
A         a1=new A(   );
          a1 =add (    );
          a1=sub (    );
B         b1 =new B(    );
          b1= mul(    );
c          c1=new c(   );
          c1= div (    );
          c1=mod (   );
          c1.mul (     );
          c1.sub (    );
          c1.add(    );
          }
       }
}
```