6 native

If we want to use code of C ~~or~~ C++ in Java, then to tell Java compiler as it is method of C/C++ we use native

ex.

```
native void fun();
```

For execution, it will not use a stack of java. It go to native method stack.

The responsibility of execution of this code is of C/C++ looks. Java not involved

7 strictfp

↓

floating point

CPU

CPU never do operation on floating point in the whole world in any CPU

FPU do operation on floating point (floating point unit)

Hardware of FPU changes as per machine

ex

$$\underbrace{0.999 \quad 0.999999 \quad 0.999999999999}_{\text{different accuracy}}$$

result varies.

∴ To fix the o/p of floating point on any platform/hardware, we use strictfp. ∴ Here Java is platform independent now

Uniformness of floating point is more important than accuracy.

We do strictfp class, method.

Used in mathematical applications.

By strictfp, now floating point/decimal point value will be same on all platforms.

blocks → Static
non/static
Synchronized

we can't use transient & static at same time because no [illegible] b/w [illegible]

DATE 3 3 23

to store 3rd object then we have to tell it
If we make that data transient then it will not go to file. It permanently not stored.

⑤ Synchronized → method / → block

We do proper communication from both side by sender & receiver called synchronous communication.
It is good communication
used in multithreading only.

```
class Demo                class Test
{ int a                   {
  void fun()                Demo d = new Demo()
  { a++ }                   we make two thread

          fun               gun
     t1 | a++ |        t2 | a++ |
          10                11
          12                13
```

Value of ~~[illegible]~~ a changes for thread due to other thread.

∴ ~~[illegible]~~ we have to **lock** one fun() method for t2 and gun() for t1.

∴ we do synchronise the method.

```
synchronized void fun ()
    { a++; }
```

```
synchronise
  {
  // If we want to
  execute only one time
  then we [illegible] block
  specific method.
  }
```

⟹ Synchronised Block

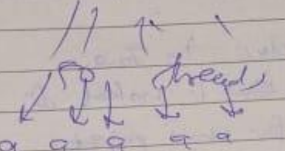
→ (If we have [illegible] line code in [illegible] we want to do only 2 lines synchronized then no need to do whole fn. synchron. we just keep the 2 line in synchronised block.)

DATE 3 3 23

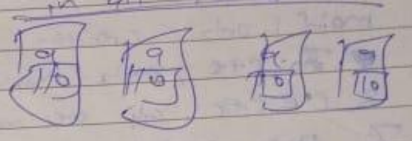
⑧ volatile ⇒
~~co~~ In c/c++, compile optimization stop due to volatile
there it is different.

static int a = 10;

50 objects



copy of a is created
in all threads.



If we do change in a in one thread then
this change is done in local memory of thread
Before do change in actual ~~a~~ thread ends and
next thread run with a=10 only. means changes not
done here.
∴ To do changes in main memory of a file
do it volatile. mean change occur to all threads.

∴ $\boxed{\text{Volatile int a = 10;}}$

DATE 2/3/23

-ect
-gged
-alculated
company.

-rtability

extends Hemo

-ged valued
-d in Hemo
child class
C-
protected

in

-ends Demo

n()

-ct override
-ern Demo{

PAGE NO
DATE 2-3-23

(IV) final class ⇒

~~Ans~~ If we want that there is no child class present of class then we make class as final.

To stop inheritance we make class final

```
final class Demo
{            // Now this class never
                be inherit.
}            // It's child class never
                be created
```

invalid ⇒ X class Test extends Demo

~~Else~~ If we do as,

```
class Demo                        It allow
{                                 ↗ inherit
}
final class Test extends Demo()
```

but Child class of Test doesn't exist.

Constructor cannot be override so final constructor concept doesn't exist.

DATE 2/3/23

In Showroom,

Company

Ex showroom price → cannot changed as it calculated by Company.

↓

insurance, accessories, commission → negotiable

```
Class Hero                              | class Showroom extends Hero
  {                                     | {
final Calculate ExShowroom Price ()     |
      {                                 |
      }                                 |
  }                                     | }
```

If we want that,
Showroom class not to be changed value in ~~Cal~~ calculate ExShowroom Price () method ~~in~~ Hero class then we make it final. Now child class cannot ~~do~~ do overriding of that fn.

To stop overriding we use final method

Now, ~~Chi~~ child class can't do change in that fn and cannot override also.

```
Class Demo                         class Test extends Demo
{                                  {
  final void fun()                   void fun()
    {                                  {
    }                                  }
}                                  }
```

o/p → error ⇒ fun() in Test cannot override ~~fi~~ fun() in Demo class Test extends Demo {
}
~~do~~ Overriden method is final

what is error to final.
If we reassign its value then we get error

```
psvm( )
{ final int a;
      a=10;
      a=10;
}
```

o/p ⇒ Error variable a might have been assigned.

~~also~~

also

```
psvm( )
{ final int a=10;
        a=10;
}
```

o/p ⇒ Error Cannot assign a value to final variable a.

Mean, we can give value to local final variable value only once.

(III) final Method ⇒

In C++, by doing method constant, we cannot ~~~~ in that constant as well as non-constant one. For using non-constant in constant constraint function. For using non-constant in constant fn we use mutable keyword.
Here in Java)

In C++ also

```
Class Employee                      | Class Manager extends Employee
{                                   | { int bonus;
virtual calculateSalary ( ) <-------|---> CalculateSalary ( ).
   {                 overriding     |    {
                                    |    }
                     by upcasting } |
   }                                |
```

DATE 3/3/23

* Non-Access Modifiers *

③ **Abstract** → class, → method

This is like pure virtual fn. in C++
virtual doesn't allow explicitly in Java.
Implicitly all fn. are virtual in Java.

```
abstract class Demo
{ abstract void fun();
}
```
→ abstract Method

// It have to compulsory give body in child class

If our class contain minimum one abstract method
then we can't create its object in Java.
To tell Java as our class contain abstract
method make class also abstract

→ abstract class

● we do class abstract if
① our class has abstract method
② OR if we don't want to create
object of class then also we can
make abstract class only, even abstract method
not present in class ex

```
abstract class Demo
{
  // no abstract method
} // can be abstract method
```

④ **transient**

Object create → memory allocated to object → Object
Stored in temporary memory (memory free when
program end) → we have to store data
permanently → Concept is Serialization →
mean we store object permanently in file/physical memory
→ to retrive object from file in program called
deserialization.

If we have 5 object and we don't want

DATE 2 3 23

```
Class Demo
{   static int x;          → make static as we are using
                              in static main fn.
        psvm ( )
        {  int a;
           sop (x);                      O/p => 0
    ✓   }
    }
```

```
2,   psvm ( )
     {
          int a;
   ✓  }
```

~~2~~ O/p => ~~0~~ NO error.

It will not give error. now.

~~It~~ only give ~~error~~ when we are using ~~it~~ it (local variable a).

~~If~~ we ~~didn't~~ use it in whole code then it will ~~not~~ give error otherwise not.

So assign it any value to make it valid.

```
        psvm ( )
        {    int a;  } or  int a=10;
local                                  O/p => 10
variable ←  ✓   a=10
             sop (a);
             a=20;
             a=30;
        }
```

~~Declare~~ Give it value at minimum one time before using it and after declaration or at time of declaration (initialization

PAGE NO.
DATE 2/3/23

```
class Demo
{
    int x;
    psvm( )
    {
                 int a;
    }
}
```

a allowed
x not allowed

allowed

(In static fn only non-static fields not allowed. But inside defined can be treated as local variables.)

and static local variable is not allowed in Java.

```
∴ psvm( )
  ✗{
        static int a;
   }
```

In static, defined non-static DM not allowed

we do as,

```
psvm(   )
{
        int a;
        sop(a);   ✗
}
```

O/p => ~~Error because~~ variable might not have been initialised.

In Java, there is no default value (garbage, NULL, 0) to local variables.

Default value is only given to fields of class which is 0, 0.1, NULL any. See code.

DATE 2/3/23

```
psvm()
{
        final int a;
        ~~sop (a);~~ sop (a);
    };
```

O/P => Error => Same error as variable might not have been initialize.
This error is not due to final. It is due to using local variable without value.

```
psvm
  {
✓     final int a;        O/P =>
  }
```

Valid Code;
as now we only declare it not using
∴ Didn't give error.

But for final Data Member we have to give compulsorily value if we use it or even not using it.

Local variable got memory only when it is given value.

If we do Local variable as final then it is called Blank final ~~Local~~ Variable.
This is only for Local not DM.
we can give it value at any time in code.

```
psvm( )
{   final int a;
        a = 10;
        sop(a);             O/P => 10
}
```

② we can give at time of declaration also.

```
✓ final int a = 10;
```

But ~~final~~ int a = 10; ✗ not valid as
~~final~~ a = 10; ⟶ we assign it again.

③

```
{
    a = 10;     ✓
}
```

It has no any name

~~It Non-Static Block.~~
Because.
⟶ (It given here thence (can't give value ~~[illegible]~~ in ~~static~~ Block, Constructor))

```
{   // Non-Static Block Or
}       Instance Block.
```

When we create object we give space to DM a.
~~[illegible]~~ we can give value to a long to Constructor.
In Java, Non-static block is executed before Constructor.
∴ we can give value to final DM in Non-Static Block also.

Non-static block executes when each object created & called before constructor

Q. Why non-static block needed?

→ In C++, we have 3 constructs default (parameterless), parameterised, and Copy.
~~If~~ In our all constructor there is same lines of code is present or it is common.
Then it increases redundancy of code.

1 3 23

sub space

e. becau

ek it

ed

be because

ual to this.

thes

ate

eated

method

eat in

n

a()

Demo();

();

n-static

Scope Java → ① local scope → inside the
② inside class but out of the scope
③ package scope (out of specific class but within program
can be outside file but within program
outside folder also)

2 3 23

* Final keyword in Java *

Access Modifier give seperate to each line. or otherwise it is default.

```
public int a;   //public
       int b;   //default
private int c;  //private
```

Java use const keyword implicitely not give explicitely. It is is reserved for native methods in C & C++ because java is made from C & C++.
So we use final keyword in Java.

final ⟹	class ✓
	fields ✓
	methods ✓
	local variable ✓

final = const
only for variable

Ⅰ final fields ⟹

```
class Demo
{
    final int a;
```

Constant variable must be initialise cannot be assigned.
In C++, Constant DM order give values in Constructor.

① In Java, Object get memory after Constructor called immediately. Means DM canbe initialise at the Most Constructor ends.
So we can give Constant variable values in Constructor.

```
    Demo()
    {
        a = 10  ✓
    }
}
```

```
Demo()
{ a=10;
}, a=20;
```

Not allowed. we can give values only once.

X ① Constructor → Static
X ② Constructor → final
✓ ③ Constructor → overloaded
X ④ Constructor → Override
⑤ Constructor public only Should. & by Constructor private, protected, default / public

Constructor is not static
Gun is is User-defined and call by implicitely.

DATE 1 8 23

(2) gun() is static fn.
Called by classname also.
∴ a is invalid becau a didnot got space without object.
a is not valid even by object called. becau it does not have this pointer.
To run non-static in static methods creat explicitely object and then use it

```
void gun()
{
        Demo d = new Demo();
        d.a
        d.b or Demo.b or b
}
```

(3)
We can call ~~non~~ static method from non-static method directly. because of this pointer is in static and is called by this.

(4) If fun() & gun() both are static. then also both are called from each other even this pointer is not present they treated as like global.

(5) we cannot call directly non-static method from ~~non~~ static method.
To call it, we have to create object in static method explicitely then call static method in non-static

```
void fun()            static void gun()
{                     {
                          Demo d=new Demo();
}                         d.fun();
                      }
```

valid now

(6) Non-static method can be called from non-static method.

a) we have to write same code in all constructor.
∴ Java removed that problem.
we write same line of code in non-static block and when we create object first non-static block called then appropriate constructor called.
So, our code being efficient. Non-static block called for every type of constructor. ~~It~~ always called when object created. ~~It~~ Non-static block is optional. not compulsory. we use it only when repeated line of code is present.

(4) Also static block executes before constructor
∴ we can give value to final DM in static block also but ~~it~~ it compulsory needed ~~final be~~ static because in static block only static DM ~~are~~ allow.

```
∴ static final int a;

    static
    {
  ✓     a=10;
    }
```

we can give final DM value any one of above.

(II)* Final Local Variable. =>

```
class Demo
{
        void fun()
        {
        final int a;
        }
}
```

PAGE NO.
DATE 1/3/23

Constructor cannot static → ① Constructor called only if object created. If we didn't create object then it will not call. Static members only called at class loading. If we don't have object then know it will be called at class loading. ∴ it is always non static.
② Constructor only called by object name always.
③ No this pointer for static. But without this constructor can't assign initial values to non-static fields. ∴ it can't/never called by class name. this will alone [illegible]

But non-static method only called by object

Q: Why main function is static?

→ public static void main ( )
Compulsory
valid both
Can be interchanged but compulsory.
∴ static public void main ( )

We run program by using tool java in cmd.
or >java Demo
We can ∞ no. of method, in Demo class.

We run Demo class means we run main method
As java tool is not part of class. It is part of out ~~of~~ class. and Java tool is accessing main fn in class Demo
∴ This main ~~[illegible]~~ method need to be public.

We can't make main() as private

We got object after main ~~[illegible]~~ called.
we didn't got object befor main() called
∴ by doing >java Demo java ~~tool didn't~~ get object to call main() method.
So to call main without object we make it Static. Now it called by Classname.
and it get classname by >java Demo
here.
∴ So it appropriately go to main() of only given class.

Call of Java tool to main() in Demo as
Demo. main (null);
→ We send nothing from main() ∴ ~~[illegible]~~ ~~or~~ purposely null.

DATE 1 3 23

we saw Static ⟹ field
block
method

Local variable Cannot be static. Why?
Local variable is present in method.
So it get memory when function called. only.
And Function called when program executes.
~~So~~ And if it is static then ~~ext~~ how to load class again? So Local variable cannot be static

```
void fun( )          void fun( )
{ int a;             { static int a;
}                    }
✓                    ✗
```

```
Class Demo
{   int a;
    static int b;
    void fun( );          static void gun( )
    { a                   { a
    } b                   } b

    psvm (   )
    {
       a   b
    }
```

were a and b allowed ~~not~~ allowed. ~~[illegible]~~.

~~a b~~
(1) fun() is non static. ∴ only called by object.
∴ a and b both allowed.
∴ static & not static DM are allowed in non static method.

DATE 1/3/23

∴ ~~Now~~.

```
class Test
{ pub st v m(String args[]) throws Exception)
  {

  Class.forName ("Demo");
                   ↑
                Name of
                class to be
                Load.
   }          only name given
              ∴ in " "
}
```

to demonstrate loading of class & demonstrate of static are executes when class loads.

demonstrate of loading static & run → main method of classes

As forName is not an method. It is of predefined class. 90% methods of predefine class are static because we not to give space to non-static DM in that predefine class.
∴ forName ( ) is also static Method
So we have to call it by class name.
and its Classname is Class
↳ It is class name ∴ by Pascal Case
C is capital

∴ we use it as

Class.forName ("Demo");

forName only load class. ∴ Main of Demo will not execute. From it & since we can see, it only Load not run class. ∴ load & Run are different things.

Now Demo also loads.
To prove it loads, we also do
in class Demo
```
{ psvm(...)
   { sop("Demo static main method");
```
To tell throw exception for if Demo class not found from Test then we do in Test class as

```
static void gun()
{ this.a; X   this will not
              use in static
              direction
}
```

DATE 1 3 23

```
psvm (String args[]) throw Exception
{
    ~~sop~~
    Class.forName ("Demo");
}
```

→ Only loads Demo class not execute

When loads, Demo static main method in main method in Demo will not print

It will be printed when class executed.

O/p ⇒ Demo static block
        Demo static fun method

(Run it by ~~Test~~ name)

In psvm also only Demo loads not executes

If we do

`int b = gun();`

then gun() should be non-static because b is also non-static.

and both executed when object created.

Static Method :→

member fun()

~~[illegible]~~ (to use only static field in fun we take it as static method) access by classname without create object, as well as object name also).

Same reason as C++.

JVM ~~only~~ executes main() fun.

Non-static DM fields are not allowed in static methods. Because ~~this~~ it didnot have this pointer. Because static DM no need object creation so we can use static methods directly without object so the Non-static DM not allow in static method.

as well we call static method by class name. as object name also

Eg. Demo.fun(); — (we use . operator)

obj.fun();

* Static Part 2 *

work of static is same in C++ & Java.

Program compiles without main function. But cannot run without run. Because java do dynamic binding for main function.

To prove as static block and static variable executes in sequence at time of class loading see below program.

```
class Demo
 {
   static
   {       SOP ("Demo Static Block"); }

   static   int   a = fun( );

     static  int  fun( )
        {   SOP (" static  fun");
               return  10;
         }
 }
 Class  Test
 {
         psvm (S a[])
      {

      }
 }
```

we have to run it by Test name not by Demo becau main is present in Test class

we run Test class means only Test class load. Demo will not load.
To load Demo, it need to be used.
So how to use it to load;

will apply)
this will not save for Static
DATE 28 2 23
go bcoz gm's static function

Space got to Non-static fields after object creation
Static → get at time of class loading.

DATE 28 2 23

O/p =) Static block
Function
Hello

} see changes. how.
b/, Static variable called ~~fast~~ later, ∴ static variable executes later after static block.

we have to create its Reference — is is like malloc

Student Sanket ~~=~~ = new Student( );

Student — type is same
Sanket — Reference variable
new Student — object creation
→ how much memory take. is given by class name.

In C++, → called Object
space → on stack

type of Sanket & this is same
Same as malloc.

In Java → only dynamic MA
∴ called Reference variable
Object is on Heap
we cant give it name we cant
∴ give name at compiletime/~~Runtime~~
In C we use pointer to give name
In Java, pointer is not exists
∴ Reference variable is like pointer in Java
It refers to that object
~~Stack~~ Reference variable is on Stack.
Reference variable is only of Class.
Reference variable is Created by class only.

Fields got space when object is created by new keyword

( ) → to pass values to fields using Constructor.
Constructor called immediately after object creation and it is called automatically.
Constructor & classname is same.
Used to give values to field immediately after object creation.

Static block executes at time of class loading.
Before main() fn. also.
Main executes after class ~~run~~.

① In Java everything out of class is invalid. ∴ No concept of global variable exist in Java. No global, No static global, No Global ✗
② C & C++ allow static variable.
③ In Java, Nothing is allowed static locals ∴ No static local variable → because if fn not called & it is static variable so Java gives space. If fn not called then it is wastage of memory.
④ static fields allowed.

① ~~cla~~ inherit Test class by Demo as
class Test extends Demo

OR

② Make object of Demo in Test class

~~Then~~ Demo class load, now because it is in use now.

But, it is bad due to
① is back ~~is~~ because useless space is increased due to inheritance while creating object of ~~test~~ class.
② is bad due to useless space is given to non-static DM in class Demo. by object creation.

So, to ~~ab~~ Avoid above situation and also to load class Demo without using above two methods we use forName("…") ~~Met~~ Function

→ Name of class to be load.

forName( )
Camelcase writing.
If we have more than two words then keep first letter smallcase and remaining first letter of each word is uppercase.

Java all methods are ~~ete~~ in Camelcase writing.

In Java Interface class, they use PascalCase writing
e.g. ArrayIndexOutOfBoundsException

All keywords are in smallcase. e.g. class, static.

In C++, we have to compulsory give value to static data member ~~out~~ out of class a/ ~~Student :: College = 5; using this method.~~

DATE 28 2 23

In Java we have block to give value.

we have to give value to static fields.
we give values to that static fields in **Static Block.**

Static Block called before main( ).
we use ~~Static Block as~~

```
static
 {
    a = 10;
 }
```

by default,
static int a;
has 0 value.
for string it is by default null.

Static fields got memory at time of class loading
also static block executed at time of class loading.

Memory allocation

Static Block execute

Ex College name need not to be repeated every time in object.
∴ we make it static
then college name got common space for all objects,
& it is created before object creation.

Q. when data/field got memory?
→ Our answer is what type of data → static / non static

Non static got space only after object creation,

```
class Demo
{
      int a;
      static int b;
      public static
        {
             sop("Hello");
        }
      static
       {
          sop("static block");
       }
}
```

o/p => Static block
Hello

Means first static block executes then main function executes.
(block at time of class loading.
block → class loading
main → class running
[illegible] algorithm

we can take Multiple static block and they execute in sequence. Recommend → take single static block

d. fun() → d implicitly

void fun() { a }
void fun(Demo this)
implicitly. { this.a; → (implicitly) }

If static void gun()
d. gun() this will not come. for static
d ✗
d will not go becau gun is static function

DATE 28 2 23

Static Variable and Static block executes at time of class loading.

Static variable and blocks execute in sequence in which they are defined.

```
    int a;
static int    b = fun();
    Static  {  sop("static block");}
    Static  int fun()
          {  sop (" Function");
             return 10;
          }
        {main( )
        }
}
```

as fun also have to execute at time of static b.

O/P ⇒ ~~static~~ Function
Static block
Hello

As, fun, block are called as we ~~defined~~ call in sequence in which they called.
As fun() called before of static block.
∴ it execute first. then block executes.

Means, static variable executes first then block

Now, if

```
Static {  sop ("block")
Static int b = fun();
~~static~~ int fun ()
    {  sop("return")
       return 10;
    }
```

PAGE NO.
DATE 28 2 23

Program.
in.
file
by java tool.
classpath
ath
Have

Object is Real Time Entity => we can create object of only real things in world. ex. person, student.
~~&~~ Bharat is not entity. ∴ Object is real time entity.
Class is use to create datatype.
Class is a Blueprint. (~~[illegible]~~ model/prototype). Class contain data & operation.
Class is collection of data & operation → In all OO lang. class have only 2 things → data, operation.

28 2 23

Modifiers in Java

In Java only Modifiers are called to all.

(type)

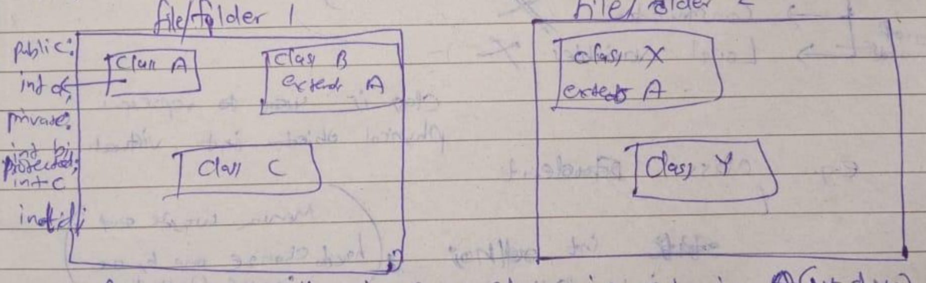
Access Modifiers
- private
- public
- protected
- default

Non-Access Specifiers
- Static
- final
- abstract
- transient
- synchronized
- native
- strictfp
- Volatile

data member → C++
local data → ~~[illegible]~~ & global data only in C++; No java
member of class: data member
In Java → field called to data member

C → function
C++ → inclass ⇒ member function
out of class → standalone/naked fun.
Java → not the part of class → methods

As in C++, default specifier is private.
Similarly in java, it is default we didn't write it.
As in C++, we use specifier to only limited for classes and child classes.
But in java we can use this Access modifier ~~its~~ access by folderwise also.
(Seperately need to tell to all field/method in Java is it private/protected/public.



* Private only allow in same class i.e. only in A (not childs)
* public is used everywhere in A, B, C and even in folder 2 X and Y.
* Protected is used in A, B, C, ~~X but~~ but not in ~~[illegible]~~ Y. Protected are ~~[illegible]~~ only ~~[illegible]~~ allowed in all Classes of same folder. and only in child classes of classes in ~~[illegible]~~ folder 2.

used in all classes of same files & childs in other files

Local variable don't have any access modifier. because it is use for temporary purpose & it come with & for short time then destroyed.

	C	C++		Java
Data	data	Data member		field
fn.	fn.	member fn	~~Standalone~~ naked fn	method

C++ → are data member & member fn
Java ⇒ are field, method, block

PAGE NO.
DATE 28/2/23

* default is allowed into ~~to~~ the classes within same folder. i.e. A, B, C not in X & Y.

Scope ⇒ (only allow all classes of same files. not outside even child not outside) we dont have permission to write manually default into it; it is internally put. Don't write default

public > protected > default > private

Java doesn't allow Naked fn. because Java is fully OO Language. ∴ classes are Compulsory.

* Non — Access Modifiers :→

(I) Static :→

we can do inner class as static. but outer

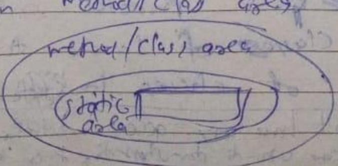
- → Outer class ✗
- → Inner class ✓
- → fields / Data Members ✓
- → Methods / Member function ✓
- → ~~static~~ Block ✓ static { }
- → Main method ✓
- → Constructor ✗
- e.g. of facebook → Local variables ✗

class is used to represent physical object into virtual.

e.g.
```
class Student
{
    ~~static~~ int rollno;
        int Collegename;
}
```

(Memory waste and have to change one by one ∴ we use static to frequently used data)

static data member have to give space before object creation at time of Class loading.
This memory get in ~~method area or static~~ area in method/class area.


method/class area
static area