12th Aug

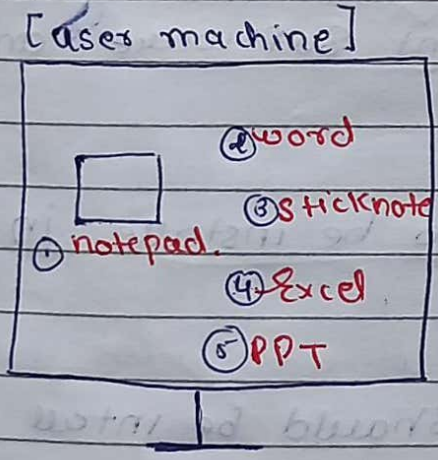
Types of Software:

Q. What are the different type of software used in the current market?

Mainly There are 4 types of software.

- Standalone / Desktop / window App.
- Client - Server App
- Distributed App
- Web App.

[1] Standalone App. / windows / Desktop.



Application which are installed on user machine and which do not interact are depend on third party application such Software or Application are called as Standalone / App / windows / Destop.

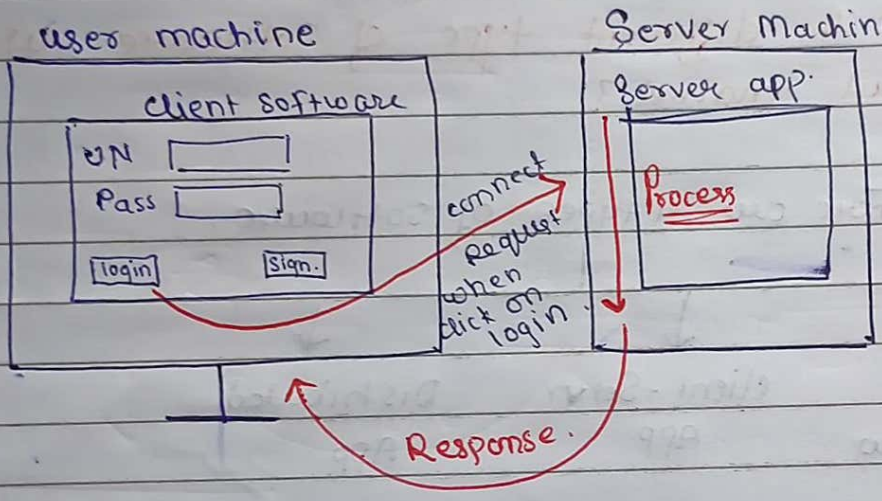
Eg: word, Stickynotes, Excel, PPT.

In one Standalone app there is only one Software.

Teacher's Signature:.....

[Q] Client - Server App. 2-3%

① ②



Server → which provides needful info,
which store info of client,

- In a client - Server app there are 2 types of softwares one is the client software and 2nd is Server software
- The client software has to be installe in user machine.
- The Server software / app should be intall on Server machine.
- Whenever we open the client software, every client software will ask (require) username & Password. and the client will send the request to the server.

Teacher's Signature:

- The Server will process the request and then send a response back to the client.

Q. In a client Server application is it possible to connect to the Server without the client Software?
Not possible.

Example : Ms Team , Google meet, Zoom,

[3] Distributed App. (0.5 – 1%)

- Applications which are distributed throughout the network such applications are called as Distributed applications.

[4] Web Applications.

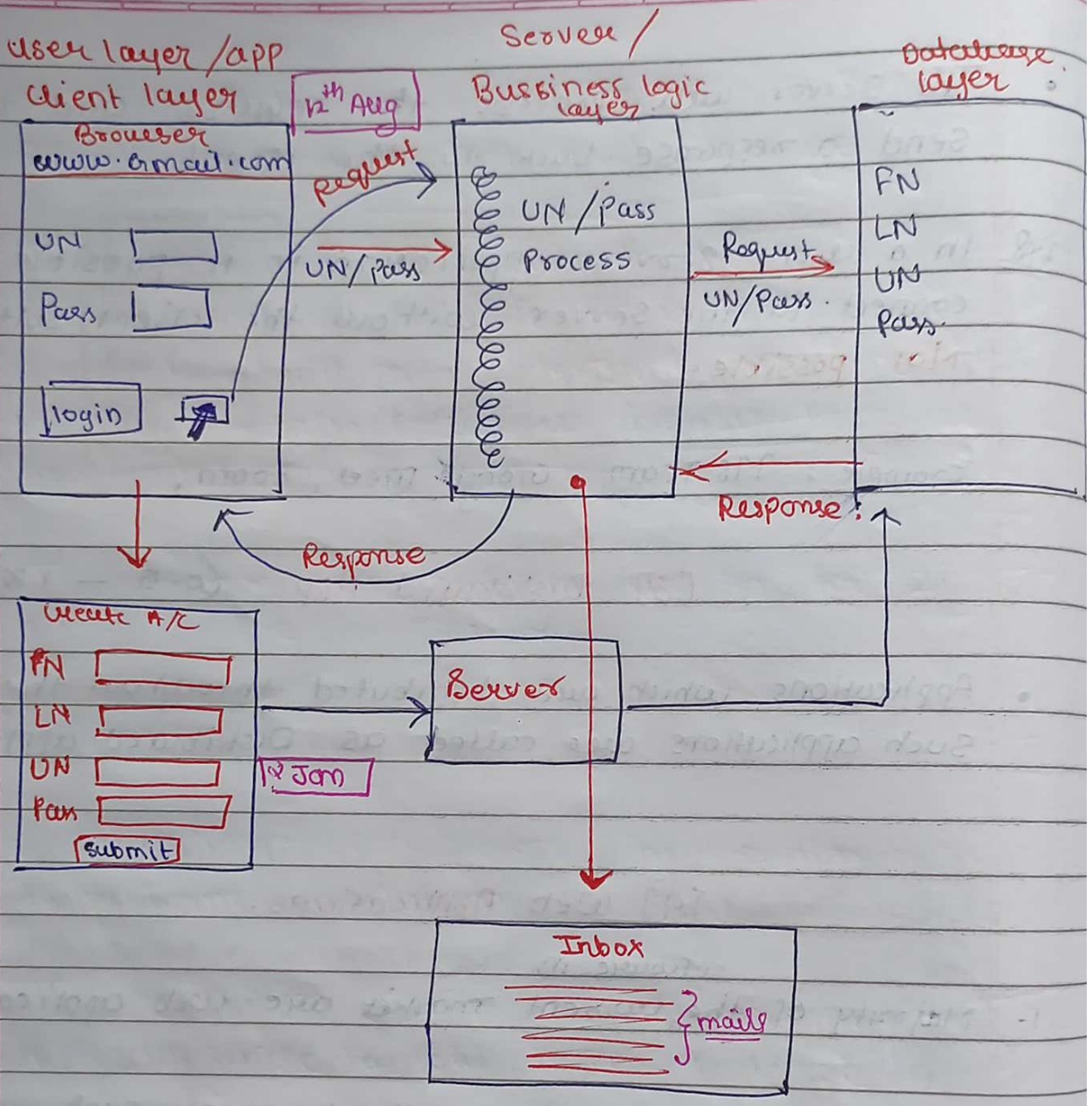
1. Majority of the current market software in are web applications
2. Any Application open through the browser such applications are called as web base application.

Web App ⇒ 3 Stages.

In a simple web base app minimum it has 3 stages.

1) User layer / Application layer / client layer.
2) Server layer / bussiness logic layer
3) Database

Teacher's Signature:.....



Teacher's Signature:.....

13th Aug

Developer ——Instructors.——→ machine.

① Java
② Html
③ C/C++

Technical languages.

Instructor → Programs
→ Scripts

- To develop any software or to test any software we have to write instructions and give to the machine
- The machine based on a instruction will either develop or test the Softwaree.
- Instructions can be given different names, Program, Script or codes.
- Majority of program languages or instruction different language will have to go through three Stages.

(i) Writing.

- In a writing Stage, we will write the instruction in java.
- After writing instruction in java we will have to Save it.
- While saving a java program, it is mandatory that the file extension should be .java.

Teacher's Signature:.....

Problem

- On Java instruction whatever is written, by default comes under high level language.
- These instructions when given to the machine, the machine will not be able to understand. because the machine will only understand, machine level language or also called as binary language or also called as low level language.
- To overcome this problem we go for 2nd stage which is the Compilation.

Compilation.

- In the compilation process, the compilar is responsible to perform compilation.
- The compilation is a process of converting high-level language to low level-language
- The compiler will accept an input and will generate an output.
- The input to the compilar is a .java file. but the output of the compiler is a ~~.text file~~ .class file.
- The .class file is given different names, low level language file, also called as binary file, also called as machine level

Teacher's Signature:.....

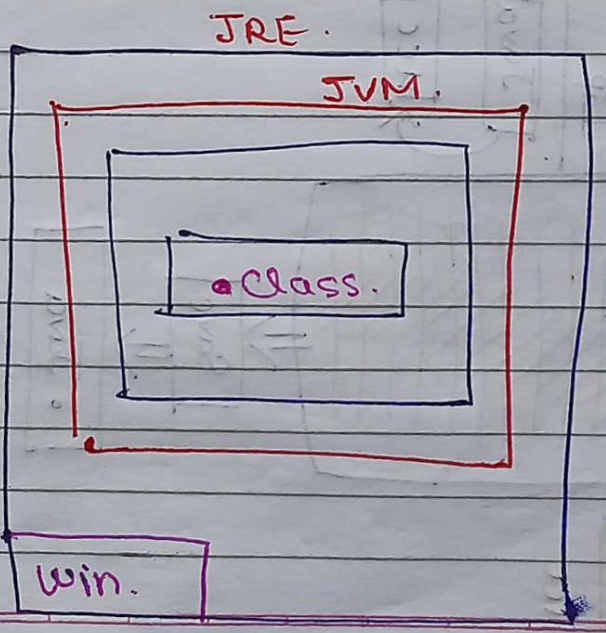
language file. but mostly it is called as white board file.

- Once the compilation is over, we then go for execution
- Execution is a process where, we make a .class has to interact with the machine, and to acheive this, java has provided a feature or a component called as JVM.

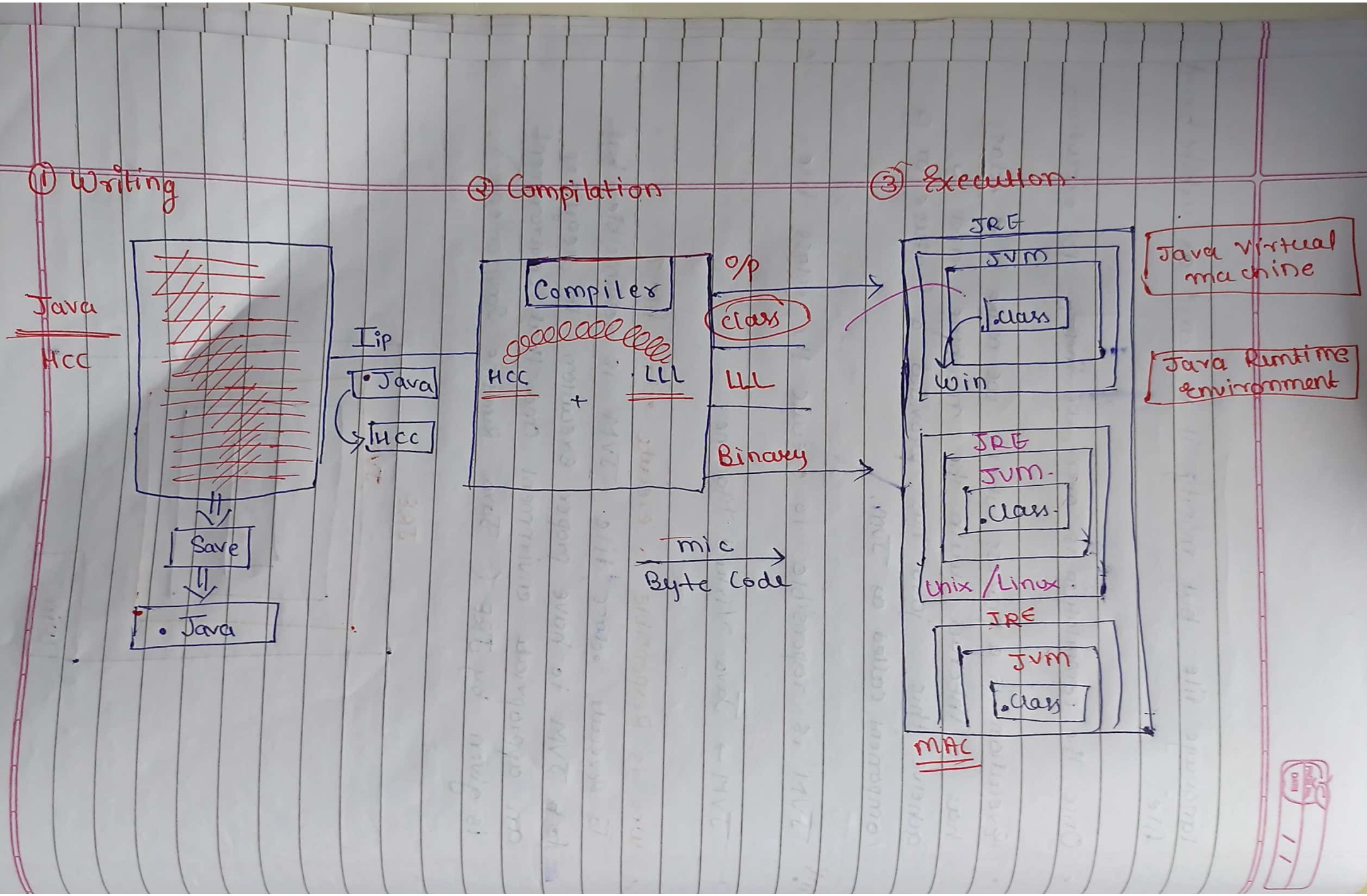
Imp

- JVM is responsible to execute the .class file.
- JVM → Java Virtual Machine.

Q. Who is Responsible to execute .class file?
To execute .class file JVM is responsible, but for JVM to have proper execution, it requires an appropriate envirnment and that environment is given by JRE (Java Runtime Environment).



Teacher's Signature:.....


① Writing
② Compilation
③ Execution
Java
HCC
Save
• Java
Iip
• Java
HCC
Compiler
HCC
+
LLL
o/p
.class
LLL
Binary
mic
Byte Code
JRE
JVM
.class
Win
JRE
JVM
.class
Unix / Linux
JRE
JVM
.class
MAC
Java virtual machine
Java Runtime environment

- In a java Program, minimum we can write one statement or we can write multiple statements

1]

```
class Sample1
{
   Public static void main (String args[])
   {
        System.out.Print ("first Statement");
        System.out.Print ("Second Statement");
        System.out.Print ("Third Statement");
   }
}
```

Sample1.java.

2] using Println to move on to the second line.

Q. what is a difference Between Print and Println Statement?

→ • The print statement will print the msg and will move the control at the end.

• But the Println Statement will not only Print the msg but will also move the control to the next line

```
System.ow.Println ("first Statement");
System.out.Println ("Second Statement");
```

Sample2.java.

Teacher's Signature:.....

3] It is not mandatory that the file name should be same as class name. we can give any name to .java file.

```
class Demo2
{
    public static void main (String[] arg);
    {
        System.out.println ("Epsion Demo2");
    }
}
```

Epsion.java

8 what is the output of following code?

```
class sample 3
{

}
```

output: Javac sample3.java.
Java Sample3
error.
main method not declare.

In the above code can only be ~~comple~~ compile but cannot be execute

Teacher's Signature:.....

The above code, it is observed that the compiler doesn't depend on main. but JVM depends on main.

IQ. Can we execute the java class without main?

→ No, we cannot execute. (why)
Because JVM depend on main.

4] Program [4]

- In a java program, minimum we can create one class or we can create multiple classes.

- Depending on Total number of classes that many no. of ~~java files~~ .class files will be created.

```
class A
{
}
class B
{
}
class C
{
}
```

B.java

```
javac B.java
java B
```

output :— A.java
B.java
C.java

Teacher's Signature:

[5] Program.

- The compiler whenever performs compilation, will compile, all classes at one time.

- When it comes for execution, the JVM will only execute one class at a time.

```
class Cyber
{
}
class Success
 {
   public static void main (String[], arg);
  {
   System.out.println ("Success");
  }
}
class Training :
 {
 }
```

Save : Training.java

Javac Training.java.

Execution: Java Success

Output : Success.

Ques: Is it possible to have multiple mains in multiple classes

yes. it is possible.

Program:

```
class Demo11
{
  Public static void main (String[],arg)
  {
    System.out.println("Demo11");
  }
}
class Sample11
{
  Public static void main (string[],arg)
  {
   System.out.Println ("Sample11");
  }
}
class Tech11
{
  Public static void main (string[], arg)
  {
    System.out.println ("Tech11");
  }
}
```

Save : Demo11.java.

Compile: Javac Demo11.java.

Output; i) Demo11.java
ii) Sample11.java
iii) Tech11.java.

Teacher's Signature:.....

Ques: Can you tell me the situation where it is mandatory that file name should be same as class name?

→ whenever the class is made public, then it is mandatory that the file name should be same as class name.

* Program: find the error!

```
~~Public~~ public class Cyber1
{
    Public static void main (String[] arg)
    {
        Sys'.out.println ("Cyber");
    }
}
```

Success1.java ✗	cyber1.java ✓

javac cyber1.java.

Teacher's Signature:.....

* Program:

- In a java program, we cannot create more than one public class.
- In a java program, we can create multiple classes but there should be only one public class.

```
Public class Dell;
{
}
Public class Lenovo.
{
}
```

It will throw an error because.

Save: Lenovo.java. (it will throw error)

There should be only one public class.

Teacher's Signature:.....

19 Aug

* fundamentals of Java.

Q What does a java class contain?
A java class can contain the following

1) Variables
2) Methods
3) Constructors
4) Non-static / Instance Blocks
5) Statics Blocks.

VARIABLES:

- A variable can be defined as a container which is meant to store data and whatever is stored can go and change. Such a container is called as a variable.

- As variables are ment to store data, data are of two types which are supported by java.
  ① Primitive
  ② Non Primitive / Reference.

Primitive Datatypes :

- Java supports 8 primitive Datatypes.
  They are : ~~Byte~~. byte, short, int, long, float, double, char, boolean.

Teacher's Signature:.....

life Human Stomach = food
Generic Type variable = data / infor
Java: byte x = 10 ;

- byte datatype :

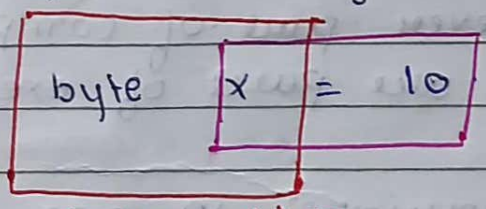
ex.1
```
class Demo1
{
    Public static void main(String[] args)
    {
        byte x = 10;
        System.out.Println(x);
    }
}
```

compile : javac Demo1.java
execution: java Demo1
(10)

- The rule to create and implement the variable is
  Type variable = data / infor.

- while Printing a variable we donot use double {" "} we directly use variable name.

- we use (" ") only when, when we want to display a message or a sentance or words.



- Every variable will have to go through two stages: (1) Declaration. (2) Initiation.

Teacher's Signature:.....

- It is possible to delare and initialize in same line

IQ Is it mandatory to declare the variable before initializing?

→ Yes it is mandatory

ex-2

```
class Demo2
{
    psvm(Str[] args)
    { x=10;
      S.O.p(x);
    }
}
```

output : cannot find symbol.
2.(error)

- Comments are used to inform the developer and tester, the reason of writing the codes
- Comments are never part of compilation and neither they are part of execution.
- We can declare variable in one line and initialize in the next line.
- A variable can be declared once but can be initialize multiple number of times.

Teacher's Signature:

ex.3

```
class Demo 3
{
    P.S.V.M (String[] args)
    {
        byte x ; // declaration
        x = 10; // Initialization
        x = 20; // Assignment
        x = 30; // Assignment
        S.O.P (x);
    }
}
```

(comments)

* In a java program, we can create minimum one variable, or we can create multiple variable

ex.4.

```
class Demo 4;
{
    P.S.V.M ( String[] args)
    {
        byte x = 10;
        byte y = 20;
        byte z = 30;
        S.O.P (x);
        S.O.P (y);
        S.O.P (z);
    }
}
```

Teacher's Signature:.....

- We cannot declare 2 variables having the same name in the same area but we can declare 2 or more variables in the same area but with different names.

ex:-

```
class Demo5
{
    P.S.V.M (String[] args)
    {
        byte x = 10;
        byte x = 20;
        S.O.P. (x);
    }
}
```

Cannot declare 2 time for same variable name.

Output: It will throw an error.

Teacher's Signature:.....

20/AUG

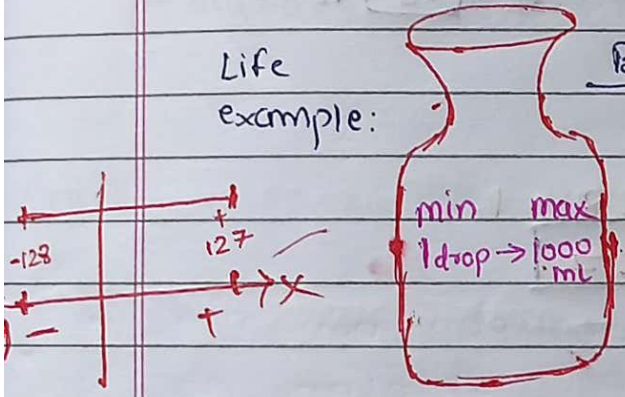
machine ⟹ Memory → RAM, ROM, HDD. (code store in RAM)

PAGE NO.
DATE / /

* Important Note:
Whenever we execute a program or create a variable both program execution and variable creation happens within a RAM memory.

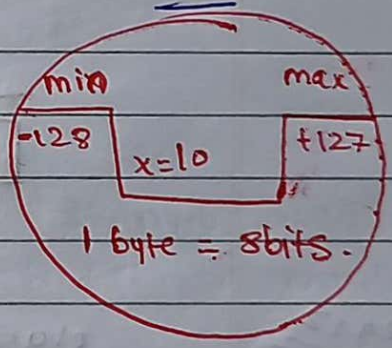
- A byte variable in java will consume one byte of memory.

Life example:



Bottle → variable
water → Data.
Capacity ⇒ 1L
1L → 1000 ml.
1 byte → 8 bits.

RAM



-2^n to $+2^n-1$
-2^7 to $+2^7-1$
-128 to +128-1

+127 to -128.
max min

1 byte = 8 bits.
1 — 8
0 — 7
(n)

eg:1

```
class Demo 8.
{
    Public Static void main (String [] args)
    {
        byte var1 = +127 ;
        S.O.P. (var1)
        byte var2 = +128 ; → compile error
        S.O.P. (var2)
        byte var3 = -128 ;
        sop (var3)
```

Teacher's Signature:.....
→ compile error

Short Data Type :

- A Short Datatype contain 2 bytes of memory
  2 bytes = 16 bits.

2 bytes = 8 x 2 bits = 16 bits

1 - 16
0 - 15 n.

∴ -2^n to $+2^n - 1$
-2^{15} to $+2^{15} - 1$
-32768 to $32768 - 1$
-32768 to 32767.

eg:2.

```
class Demo9
{
  Public static void main (String[] args)
  {
    Short s1 = 32767;
    System.out.println (s1);
  // Short s2 = 32768; <- error.
    SOP(s2)
    Short s3 = -32768;
    SOP (s3)
  // Short s4 = -32769; error
    SOP(s4).
  }
}
```

Integer Datatypes.

int ⇒ 4 bytes = 8×4 = 32 bits.

1 - 32

0 - (31) n.

-2^n to $2^n - 1$

-2^{31} to $2^{31} - 1$

-2147483648 to +2147483647

eg:3

```
Class Demo1a
{
    Public Static void main (String[] args)
    {
        int var1 = 2147483647;
        SOP (var1);
        //int var2 = 2147483648;   compilation error.
        SOP (var2);
        int var3 = -2147483648;
        SOP (var3);
        //int var4 = -2147483649;   compilation error.
        SOP (var4);
    }
}
```

Beyond range

Long Datatype.

long = 8 bytes = 8×8 = 64 bits.

@ 1 - 64

0 - 63 n.

$\therefore -2^n$ to $+2^n - 1$

-2^{63} to $+2^{63} - 1$

```
class Demo11
 {
   Public static void main (String [] args)
   {
     long var1 = 1000;  → within range
     S.O.P (var1);
   }
 }
```

String.

- Anything within a pair { " " } java represents it as a String.
- If we have to combine a String along with the value this process is called as concatination
- To implement concatination we need to use '+' operator.

Teacher's Signature:....

eg: 5]

```
Class Demo12
{
    Public static void main (String [] args)
    {
        long var1 = 1000;
        SOP ("The value of var1 is" + 1000);
    }
}
```

(+) → concatination operator

imp.

* Byte, short, int, long, they all are meant to store integral values.

- Hence byte, short, int, long are also called as integral Datatypes.

IDENTIFIERS.

- An identifiers can be defined as a name given to a class, method, variable. Such names can be called as identifiers.

21/Aug

IDENTIFIERS Naming Rules:

Rule 1] A valid identifier can start with a letter, a currency symbol ($), and connecting character (_).

Teacher's Signature:.....

Ex.1

```
class Demo 13
{
    Public static void main (String [] args)
    {
        int x=10;
        int $=20;
        int _ =30;
        SOP (x);
        SOP ($);
        SOP (_);
    }
}
```

It will throw warning during compilation and then execute.

Rule: 2] A valid Identifier cannot Start with the number.

```
class Demo14
{
    P.S.V.M (String [] args):
    {
        int 1 = 10;
    }
}
```

Teacher's Signature:.....

Rule:3] We can use numbers only after a letter, or currency Symbol ($) and connecting character (_).

```
class Demo15
{
Public static void main (String [] args)
{
    int x1 = 100;
    int $2 = 200;
    int _3 = 300;
    SOP (x1);
    SOP ($2);
    SOP (_3);
}
}
```

Rule 4: A valid identifier does not have any particular limit, however, as per global standards. Or reccommanded standards, a valid identifier should be between the length 10-15, anything beyond that is not considered as good coding practice.

```
class Demo16
{
Public static void main (String [] args)
{
    int xxx.....111 = 1000;
    SOP (xxx.....111);
}
}
```

Teacher's Signature:.....

Rule 5] Keywords or reserved words cannot be use as a valid Identifier.

```
class Demo {
  Public static void main (String [] args)
  {
    int byte = 10;
    SOP (byte);
  }
}
```

error :

Q. can you tell me total number of keyword or reserved words in java.

→ There are 53 keywords

Teacher's Signature:

Rule 6] Class names and method names can be used as a valid identifier, it is supported by java but is not considered as a good coding practice

```
class Demo18
{
  Public Static void main (String [] args)
  {
    int Demo18 = 10;
    int main = 20;
    SOP (Demo18);
    SOP (main);
  }
}
```

output : 10
20

Rule 7] A valid identifier can be declared more than once with the same name, same area but different case.

Imp Java is a case Sensitive language.

```
class Demo19
{
  Public Static void main (String [] args)
  {
    int cyber = 10;
    int CYBER = 20;
    Sop (cyber);
    Sop (CYBER);
  }
}
```

output : 10
20

Teacher's Signature:.....

Summary

R # A valid identifier will contain the following :-

① a - z
② A - Z
③ 0 - 9
④ $
⑤ _

Assignment

1) int Cyber2Success = 10; ✓
2) int 4Cyber = 10; ✗
3) int all@cyber = 10; ✗
4) int cyber_Success_training = 10; ✓
5) int _$_ = 10; ✓
6) int cyber# = 10; ✗
7) int int = 10; ✗
8) int integer = 10; ✓
9) int 0123_10 = 10; ✗
10) int cyber$ = 10; ✓
11) int 1981$_10 = 10; ✗

Teacher's Signature:

Long Datatype Concept

- Any integral value, given to integral Datatype the compiler treats thoes values, as integer type.

- To treat a value as a long type we need to ~~Suffix~~ the letter L.

- Java by nature is a case sensitive language but when it comes to value it behave as case insensitive.

```
class Demo20
{
  Public Static void main (String [] args)
  {
  // byte var1 = 128;  int to byte
  //Short var2 = 32768; → int to short.
  // int var3 = 2147483648; integer num to large
    long var4 = 2147483648L;
    SOP (var4);
    long var5 = 2147483649L;
    S.O.P (var5);
  }
}
```

output: 2147483648
2147483649

Teacher's Signature:.....

Float & Double Datatype & variable Store.

- The data stored in float & double are in decimal format.

float	Double.
① In the decimal values are in the range of 5-7, we go for float.	① If the decimal values are in range of 12-14 we go for double.
② float can also be called as single accuracy or Single Precesion.	② whereas, Double can be defined as Double accuracy or Double Precesion.
③ float consumes 4 bytes of memory.	③ Double consumes 8 bytes of memory.

Imp Any Decimal value given to float or double by default internally is considered by the compiler as a type of double.

Teacher's Signature:.....

```
class Demo21
{
    Public static void main (String[] args)
    {
        float var1 = 10.5f;
        System.out.Println (var1);
    }
}
```

output : 10.5

22/Aug # Double.

```
class Demo22
{
    Public static void main (String [] args)
    {
        double d1 = 150.123;
        S.O.P (d1);
        double d2 = 250.321D;
        S.O.P (d2);
        double d3 = 350.321d;
        S.O.P (d3);
    }
}
```

output : 150.123
250.321
350.321

Teacher's Signature:.....

Typecasting.

- Typecasting is a process of converting ,(int to float) one form to another form.

- we can convert an int to float but we cannot convert float to int.

```
class Demo23
{
 Public static void main (String [] args)
 {
   float f1 =100;  → int → float
   S.O.P (f1);
   int i1 = 100.5;
   S.O.P (i2);
 }
}
```

Character Datatype :

- A character can be represented by a single letter, number, special character within a pair of {' ' } Such a type of data is called as character data.

Teacher's Signature:.....

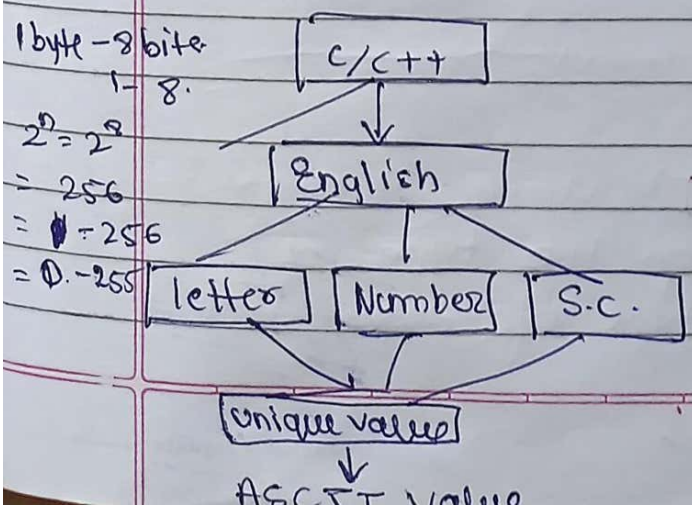
~~American Standard Code Information~~

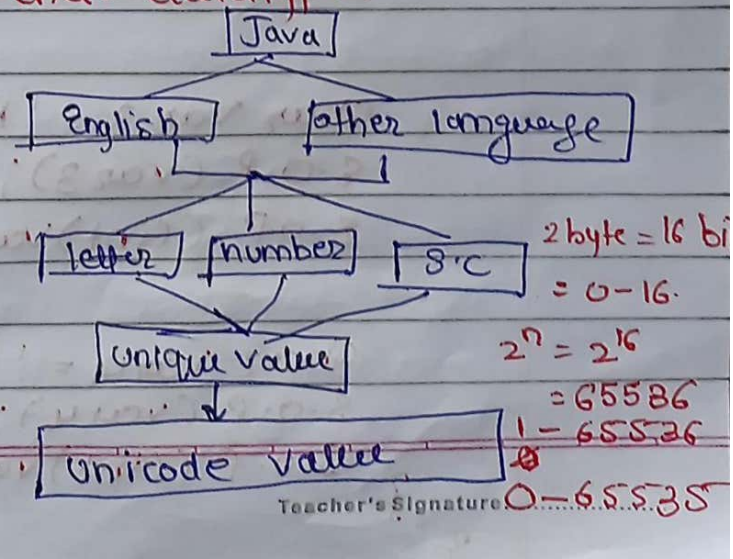
```
class Demo 24
{
   Public Static void main (String[] args)
   {
     char var1 = 'a';
     char var2 = '1';
     char var3 = '#';
     S.O.P (var1);
     S.O.P (var2);
     S.O.P (var3);
   }
}
```

Output :- a
1
#

- A character datatype in java consumes 2 bytes of memory.

IQ A character datatype consumes one byte of memory in c/c++ technology but why does java consume 2 bytes for a char datatype.





Teacher's Signature

- C/C++ technology by default supports english language but when it comes to java, it not only supports english language but also supports other language. Hence java tech. consume more memory compare to C/C++

- every language will give ~~the~~ a unique value to every letter, number as well as a special character, these unique values in C/C++ is called as ASCII values (American standard code for information interchange ~~interface~~) as well as in java it is called unicode value

```
class Demo25
{ Public static void main (String [] args
  char var1 = 'a';
  System.out.println (var1);
  S.O.P ((int) var1);

  Char var2 = 'b';
  S.O.P (var2);
  S.O.P ((int) var2);

  Char var3 = 'A';
  S.O.P (var3);
  S.O.P ((int)var3);

  Char var4 = 'B';
  S.O.P (var4);
  S.O.P ((int)var4);
}
```

Output : a - 97 A - 65
b - 98 B - 66.

- Unicode concept of int and character :

```
class Demo26
{
  Public Static void main (String[] args)
  {
    int x = 'a';
    S.O.P(x);
    char y=98;
    S.O.P(y);
  }
}
```

Output : 97
'b'.

#

Teacher's Signature:.....

Boolean Datatype.

- A Boolean Datatype can only store 2 types of values. True & False.
- A Boolean Datatype will consume one bit of memory. but its 'size' isn't Precisely defined.

```
class Demo27
{
  Public Static void main (String [] args)
  {
    boolean Status1 = true;
    S.O.P (Status1);
    boolean status2 = false;
    S.O.P (status2);
  }
}
```

output : true
false.

- In c/c++ technology '0' represents false and any other number represents true
- But Whereas in java technology, it is strickly typed language, based on the type of variable, only that type of data will be stored.

Teacher's Signature:

```
class Demo28
{
    Public static void main (String [] args)
    {
        boolean status1 = 0;
        S.O.P (status1);
    }
}
```

Output : error.

23/Aug

OPERATORS.

Operators	Precedence :
• PostFix	expr++ expr--
• unary	++expr --expr +expr -expr ~
• multiplicative	* / %
• additive	+ -
• shift	<< >> >>>
• relational	< > <= >= instanceof
• equality	== !=
• bitwise AND	&
• bitwise exclusive OR	^
• bitwise inclusive OR	|
• logical AND	&&
• logical OR	||
• ternary	? :
• assignment	=, +=, -=, *=, /=, %=, &=, ^=, |=
	<<= >>= >>>=

Teacher's Signature:.....

```
class operator1 {
  Public Static void main (String [] args)
  {
    int result;
    // Addition.
    result = 10+20;
    S.O.P (result)
    // Subtraction.
    result = 20-10;
    S.O.P (result);
    // multiplication.
    result = 10*20;
    S.O.P (result);
    // Division.
    result = 10/2;
    S.O.P (result);
    // Modulus;
    result = 10%2;
    S.O.P (result);
  }
}
```

output : 30
10
200
5
0

Control flow Statement :

Control flow statements can be catagorize mainly in 3 type.

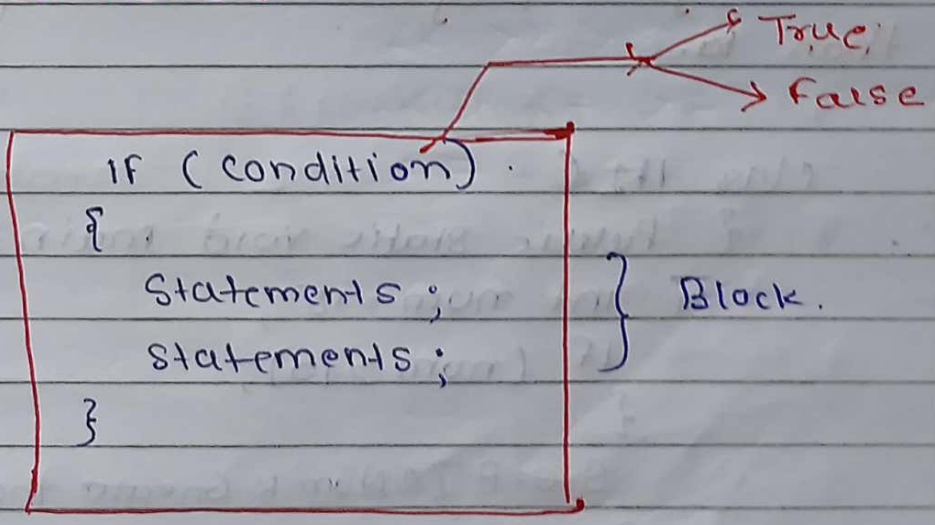
① Conditional Statements.
② Looping statements
③ Transfer Statements.

1] Conditional Statements :

- Conditional statements comprises of the following.
  1. If Statements
  2. If-else statements
  3. If-else-if Statements
  4. Nested if-else Statements
  5. Switch statements.

- **If Statements :**

Syntax :



- As seen in the syntax if the condition is true it will execute the statements within the if block.

- But if the condition is false it will not execute the statements within the if block.

- Conditions are in the format of True and false which are boolean values. But to generate these Boolean values we need to use comparison operators.

- Comparison operators are combination of ~~arithematic operator~~ equality and relational operators.

- It include `>  <  >=  <=  ==  !=`

Q. Write a java program to check if a no. is greater than 10.

```
class If1 {
  { Public static void main (String[] args)
    { int num = 15;
      if (num > 10);
      {
      S.O.P ("Num is Greater than 10");
      }
}  }
```

Teacher's Signature:.....

Q. Write a java program to check if a person is eligible for voting.

```
class IF2
{
  Public static void main (String[] args)
  {
    int age = 19;
    If (age >= 18)
    {
      S.o.p (" Eligible for voting ");
    }
} }
```

2) If - else Statements :

Statements

↓

If - else

Syntax :

```
If (condition)
{
  Statements;
  Statements;
}
else
{ Statements;
  Statements;
}
```

Teacher's Signature:.....

- As seen above in the syntax, if the condition is true, then it will execute the statements within the if block.
- But if the condition is false it will then execute the statement within the else block.

IQ Write a java program to check out of 2 no. which one is greater.

```
class IfElse1
{
  Public Static void main (String[] args)
  {
    int num1=20, num2=30;
    IF (num1 > num2)
    {
      S.O.P("Num1 is greater");
    }
    else
    {
      S.O.P("Num2 is greater");
    }
  }
}
```

Teacher's Signature:.....

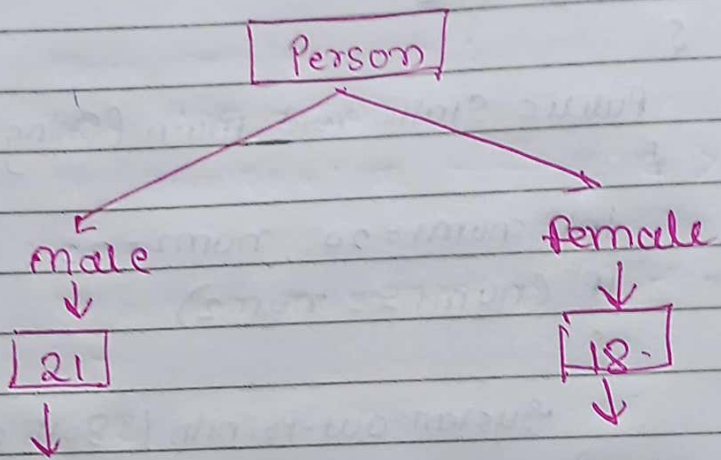
Q. write a java program to check if the age of two people are same or not.

```
class IfElse2.
{
  Public Static void main (String [7 args)
  {
    int num1 = 20 , num2 = 20 ;
    IF (num1 == num2)
    {
      System.out.Println ("Both age are same");
    }
    else
    {
      System.out.Println ("Both age are not same");
    }
  }
}
```

output:

Both age are same.

8. write a java program to check if person is eligible for marriage or not ?



char gender = male
int age = 21

char gender = female
int age = 18

If Gender == 'male' And (age >= 21)

f.f

Condition ①		Condition ②	Total condition
	f.f		
F (0)	multiplication	F (0)	F (0)
F (1)		F (0)	F (0)
F (0)		T (1)	F (0)
T (1)		T (1)	T (1)

Teacher's Signature:

```
class IfElse3
{
  Public Static Void main (String [] args)
  {
    int age = 18;
    char gender = 'f';
    {
     If (age >= 18) && (gender == 'F')
     {
      S.O.P ("Eligible for marrige");
     }
     else
     {
      S.O.P ("Not Eligible for marrige");
     }
  }
}
```

output:

Eligible for marrige.

24/Aug

```
Class IFElse
{
  Public Static void main (String[] args)
  {
    Char gender = 'm';
    int age = 20;
    IF (gender = 'm' && age > 21)
    {
      System.out.Println ("Eligible");
    }
    else
    {
      S.O.P ("not Eligible");
    }
  }
}
```

Ques: write a java program to check person is eligible for IIT if the score in Suience OR in maths above 90.

Teacher's Signature:.....

```
class IfElse
{
  Public static void main (String[] args)
  {
    System.out.Println ("Start");
    int Science = 90 , maths = 90 .
    If (Science >= 80 || maths >=90
    {
      S.O.P (" eligible for IIT ");
    }
    else
    {
      S.O.P ("Not eligible for IIT");
    }
  }
```

Q. Write a java program to check if a female is eligible for marriage or not.

```
class IfElse.
{
  P.S.V.M (String[] args)
  { char gender ='f';
    int age =20;
    If (Gender = 'f' && age >= 18)
    { S.O.P (" eligible");
    }
    else,
    { S.O.P (" not eligible");
    }
  }
```

Teacher's Signature:.....

If - else - if.

Syntax.

```
IF (condition)
{
 //Statement
}
else if (condition)
{
  //Statement.
}
else if (condition)
{
  Statement
}
else
 {
   Statement
  }
```

Teacher's Signature:.....

Q. Write a java program to check out of which one is greater and also check both numbers are same.

```
class ifelseif.
{
  Public static void main(String[] args)
  {
    int num1 = 10, num2 = 20;
    if (num1 > num2)
    {
      S.O.P ("no. 1 is greater");
    }
    else if (num2 > num1)
    {
      S.O.P ("no.2 is greater");
    }
    else if (num1 == num2);
    {
      S.O.P ("Invalid");
    }
  }
}
```

Teacher's Signature:.....

Ques: write a java program to simulating traffic signal

```
Class Ifelseif
{
Public Static void main (String[] args)
{
   Char Sign = 'r'
   If (sign = 'r')
   {
    S.O.P ("stop");
   }
   else if (Sign = g);
   {
    S.O.P ("Go");
   }
   else if (Sign == 'y');
   {
    S.O.P ("Slow Down")
   }
   else.
   {
    S.O.P ("not valid");
   }
  }
}
```

Teacher's Signature:

* Nested - if - else.

Syntax:

```
if (condition)
{
  if (condition)
    { Statement
    }
  else
    {
    Statement;
    }
}
else if (condition)
{
  Statement;
}
}
```

Q. write a java program to purchase product online if it is the laptop & quantity is more to then purchase it else donot Purchase it the product is mobile & Quatity is above 35 then purchase if else Do not ~~Purchase~~. Purchase.

```
class Nested
{
  Public static void main (String [] args)
  {
    char prodtype = 'M';
    int custQty = 25;
    if (prodtype == 'L')
    {
      if (custQty > 20)
      {
        S.O.P ("Purchase");
      }
      else
        {
          S.O.P ("not purchase");
        }
    }
    else
    {
      if (custQty > 35)
      {
        S.O.P ("Purchase mobile");
      }
      else
        {S.O.P ("mobile Qunatity is low")}
      }
  }
```

Teacher's Signature:

Switch Case

Syntax:

```
Switch (variable / argument / Parameters)
{
  case caselalue : statements ;

          [ break ]

  case caselalue : statement ;

          [break]
  default : statements ;

          [break]
  }
}
```

- In the switch statement there is no condition match.
- In the switch statements the way it works the aregurments within the switch is matched, or mapped with the case label.
- If the matching happens then the respective statement within the case will get executed. and after which, due to the break statement, the control will be broken within the switch which means the control will be taken outside the switch block.

Teacher's Signature:.....

```
class Switch1
{
  Public static void main (String[] args)
  {
   System.out.Println ("start");
   int a=2;
   Switch (a)
   {
    case1 : S.O.P("inside case 1");
            break;
    case2 : S.O.P("Inside case 2");
            break;
    case 3 : S.O.P("Inside case 3");
            break;
    default : S.O.P ("Inside Default");
            break;
    }
    S.O.P ("stop");
  }
}
```

- If the switch argument does not match any of the cases, then by default it will match default case.

Teacher's Signature:.....

```
int a=5;
Switch (a)
{
 case 1: S.O.P ("inside case 1");
         break;
 case 2: S.O.P ("inside case 2");
         break;
 case 3: S.O.P ("inside case 3");
         break;
 default : S.O.P ("Inside Default");
         break;
}
```

10 Are cases mandatory within the switch statement?
Cases are not mandatory.

```
int a=1;
Switch (a)
{
 Default :
        S. O. P ("Inside Default");
              break;
   }
```

Teacher's Signature:.....

IQ Is Default case mandatory within Switch?

It is not mandatory

```
int a = 3;
switch (a)
{
  case 1 : S.O.P ("Inside case 1");
           break;
  case 2 : S.O.P ("Inside case 2");
           break;
}
```

Q. What is the output of following code?

```
int a=3;
Switch (a)
{

}
```

- Above program will compile and execute.

- Statements are allowed within the Switch but independent Statement have to be either within the case or should be eithere within the Default, without that it will throw an error.

Teacher's Signature:

```
int a=3;
switch (a);
{
   S.O.P("Inside the statement");
}
```

Q. Which are the valid and ~~non~~ invalid argument within the switch statement?

→ The valid argument supported by switch are int, byte, short, char.

- The invalid argument supported by switch are long, float, boolean, double.

Q.

```
Char a = 'red';
//Char b = 'Green';
//Char c = 'yellow';
```
} Take single

```
switch (a)
 {
   case 'red': S.O.P ("Stop");
               break;
   case 'Green': S.O.P ("Go");
                 break;
   case 'yellow': S.O.P ("Go Slow");
                  break;
   default: S.O.P ("Invalid Signal");
            break;
 }.
```

Teacher's Signature:.....

- As break statements are not mandatory without breakalso, the program will be compiled and executed.

- Without break statement, whichever state case is goes and match from that case onwards. it will execute till the end of the switch statement:

```
int x=2;
Switch (x)
{
 case 1: S.O.P ("one");
 case 2: SOP ("Two");
 Case 3: SOP ("Three");
 default : S.O.P ("Invalid signal");
}
```

It will show output from case : 2 onwards till the end (default).

- We can use default case anywhere in the switch statement.

```
int x = 3;
Switch (x)
{
 case 1 : S.O.P("Inside case one");
          break;
default : S.O.P("Inside default");
          break;
Case 2 : S.OP("Inside case two");
          break;
}
```

* from java 1.5 version onwards, it was allowed to use String as a valid argument to switch.

```
String *Str = "Cyber";
Switch (Str)
{
 case 'Cyber" : S.O.P("Inside Cyber");
                break;
 case "Success" : S.O.P("Inside Success");
                  break;
 default : S.O.P("Inside default");
           break
}
```

Teacher's Signature:.....

* Based on the switch argument the case labels should also be of same type.

```
String str = "Cyber";
Switch (str)
{
 case 10 : S.O.P ("Inside Cyber");
           break;
 case 20 : S.O.P ("Inside Success");
           break;
 Default : S.O.P ("Inside Default");
           break;
}
```

It will throw an error.

28/Aug

Practice codes

#. Based on: If Statement, If-else-If, nested-If and Switch case.

Q. Check if a person can apply for a driving license.

```
class DrivingEligibility
{
    Public static void main (String[] args)
    {
        int age = 20; // change age to test.
        If (age >= 18)
        {
         S.o.P("you are eligible to apply for
                a driving license");
        }
        else
         {
          S.o.P("you are not eligible to
                 apply for driving license");
         }
    }
}
```

Output: You are eligible to apply for a driving license.

Teacher's Signature:.....

Q. Check if a student was passed or failed:

```
class PassorFail
{
  Public static void main (String[] args)
  {
    int marks = 28;
    if (marks >= 40)
    {
      S.O.P ("You have passed the exam")
    }
    else
    {
      S.O.P (" You have failed the exam")
    }
  }
}
```

Ouput: You have failed the exam.

Teacher's Signature:

Qw: Check if a person is eligible for a senior citizen discount.

```
class SeniorCitizenDiscount
{
  Public Static void main (String [] args)
  {
   int age = 65;
   IF (age >= 60)
   {
    S.O.P ("you are eligible for Senior
            citizen discount")
   }
   else
   {
    S.O.P ("you are not eligible for
            Senior citizen discount")
   }
  }
}
```

Output : you are eligible for Senior citizen discount.

Teacher's Signature:.....

Qus: If - else - if Example.
Grade classification.

```
class Gradecheck.
{
  Public static void main (String[] args)
  {
    int marks = 75 ;

    if (marks >= 90)
    {
     S.o.p ("Grade A");
    }
    else if (marks >=75);
    {
     S.o.p ("Grade B");
    }
    else if (marks >=60);
    {
     S.o.p ("Grade c");
    }
    else if (marks >=40);
    {
      S.o.p ("Grade D");
    }
    else.
    { S.o.p ("fail");
     }
  }
}
```

Output : ~~Grade~~ Grade B

Teacher's Signatu

Q. If-else-if example:
Electricity Bill slab.

```
class ElectricityBill
{
 Public static void main(String[] args)
 {
  Int units = 250; electricity units used by customer
  double bill; total amount to pay;

  if (units <=100)  Each unit cost ₹5.
  { bill = units * 5;
  }
  else if (units <=200)  units between 101 & 200
  {
   bill = (100*5) + (units-100) * 7;
  }
  else if (units <=300)
  {
   bill = (100*5) + (100*7) + (units-200) * 10
  }
  else
  {
  bill = (100*5) + (100*7) + (100*10) + (units-300) *
                                        * 15;
  }
  S.O.P ("Total Electricity Bill : ₹" + bill);
 }
}
```

0-100 units → ₹5/unit
101-200 U → ₹7/U
201-300 U → ₹10/U
301+ units → ₹15/U

Output → for 250 units the bill is.

100*5 + 100*7 + 50*10 = 1700

Total Electricity Bill : ₹ 1700.0

Teacher's Signature:.....

Qus: Nested if example → ATM withdrawal check

```
Class ATMwithdrawal
{
  Public static void main (String[] args)
  {
    int balance = 5000;
    int withdraw = 2000;
    boolean atmworking = true;  is atm working or not

    IF (atmworking)
    {
      IF (withdraw <= balance)
      {
        S.O.P ("Transaction Succesfull!
                Remaining Balance: " + (balance-
      }
    else {
        S.O.P ("Insufficient Balance");
      }
    else
    {
      S.O.P ("ATM is out of Service.");
    }
  }
}
```

Teacher's Signature:

Looping Statements

- A loop can be defined as [illegible] or an instruction [illegible] is called as loop or an iteration.
- There are three major looping statements or iteration statements supported by java.
  - For loop
  - While loop
  - Do-while loop

For loop

- For loop can be categorize mainly in three types
  (i) For i
  (ii) Nested for
  (iii) For each (Arrays / collections)

Syntax for i

```
for (Initialization ; condition ; Increment/Decrement)
{
    Statements;
    Statements;
}
```

Looping Statements.

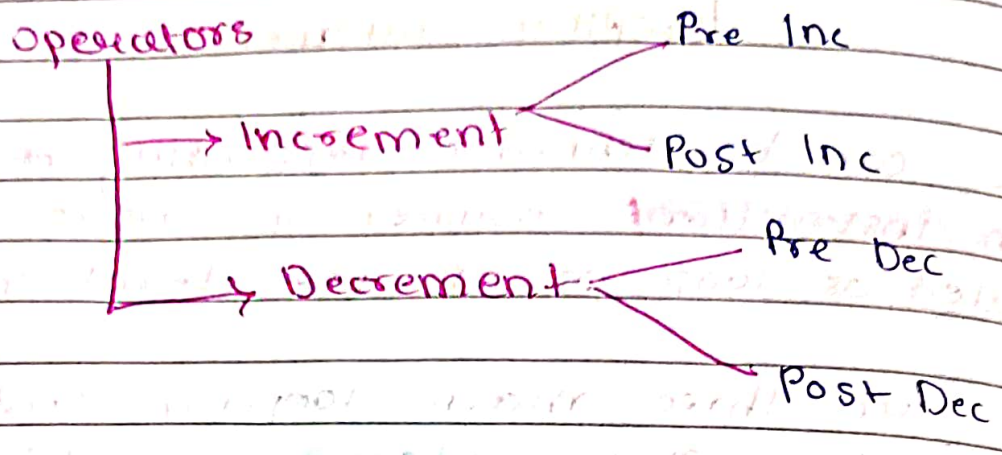
- A loop can be defined as Repeating a statement or an instructions number of times. itself is called as loop or an ~~ire~~ iteration.
- There are three main looping statements or iteration statements. supported by java.
  - For loop
  - While loop
  - Do-while loop

For loop.

- For loop can be categorize mainly in three type
  (i) For i
  (ii) Nested for
  (iii) For each. (Arrays / collections)

Syntax → for i

```
for (Declaration / Initialization ; condition ; Increament / Decreament)
{
    Statements;
    Statements;
}
```



Increment & Decrement Operators.

- Operators can be categorize mainly into 2
  (i) Increment operator
  (ii) Decrement operator.

Increment Operator / Decrement Operator.

Inc & Dec. operator can futher be categorize mainly into 2 types.

(i) Pre Inc. (ii) Post Inc.
(i) Pre Dec. (ii) Pos Dec.

Pre & Post Increment operators.

The Pre & Post Increment operators, will increment the value of i or any other variable by 1

Pre ⇒ ++i
Post ⇒ i++

Teacher's Signature:

```
class IncPostPre1
{
  Public Static void main(String[] args)
  {
    int i=1;
    S.o.p(i); //1
  //PreIncrement → // First Increment the i value
  by 1 & then Print the value of i.

    S.o.p(++i); //1 Increment, 2 → Print 2
    S.o.p(i); //2

  // PostIncrement → // First Print the value of i
  & then Increment the i value

    S.o.p(i++); //2 Print → 2- Increment 3
    S.o.p(i); //3
  }
```

Output: 1
2
2
2
3

Teacher's Signature:.....

Ques: Write a java program to print numbers from 1-3 using the for loop.

```
Class For1
{
  Public static void main (String[] args)
  {
    for (int i=1 ; i>=3 ; i++)
    {
      S.O.P.("The value of i is" + i);
    }
  }
}
```

Declaration, Initialization, condition, Increment

Output; The value of i is 1
The value of i is 2
The value of i is 3

① Dec / Initi ⇒ i=1.
② Condition → F ⇒ ~~Stop~~ Stop
→ T 1 >=3
③ ↳ Statement
④ Inc / Dec → i ⇒ 2
⑤ Condition → F = Stop.
→ T 2 >=3
Statement.
⑥ Inc / Dec → i ⇒ 3
⑦ Condition → F
→ T 3 >=3
⑧ ~~Inc / Dec~~
⑨ Statement.
⑩ Inc / Dec → i ⇒ 4
⑪ Condition → F
→ T 4 >= 3
⑫ Stop

∴ It will stop iteration

Teacher's Signature:.....

Q - Within the for loop execution, how many times will the Declaration & Initiation happens.?

→ Only 1 time.

Q. within the for loop how many times. will the Condition will check.?

→ Based on total number of Iteration + 1

Q. Within the for loop how many times will Incr & Decrement will happen.?

→ Based on total numbers of Iterations.

Q. Write a java program to print values from 1 to 3 in Decending order.

```
for (int i=3 ; i >=1 ; i--)
{
  S.O.P ("The Decending order is "+i)
}
```

Teacher's Signature:

Q. Write a java program to check if number 5 exists from 1 to 5 iterations?

```
for (int i=1 ; i<=5 ; i++)
{
  S.o.p(i);
   if (i==3)
   {
     S.o.p("The value of i is" +i);
     break;
   }
}
```

Initialization & Declaration :

- Declaration and initialization are not mendetory within for statement.

```
int i=1;                          → Declaration & Initialization are outside the loop
for (i<=5 ; i++)
{
  S.o.p(i);
}
```

Output :
1
2
3
4
5

Teacher's Signature:.....

- We can use minimum one or multiple Statements Seperated by comma (,) in the declaration as well as Initialization.

```
int i = 1;
for (S.O.P("one"), S.O.P("Two"); i<=5; i++)
{
  S.O.P(i);
}
```

output : one
Two
1
2
3
4
5

Condition Stage in for loop :

- Condition is not mandatory
- If we do not set any condition, then the compiler will by default set the condition as true.

```
for (int i=1 ;  ; i++)
{
  S.O.P (i);
}
```

Output ; infinite.

Teacher's Signature:

- Statements are not allowed within the condition.

```
for (int i=1; S.O.P("one"); i++)
```

It is invalid, will throw an error.

Increment & Decrement :

- Increment & Decrement is not mandatory
- In the below code, as i value is not incremented or Decremented, hence the i value will never change. and the condition will always remain True and hence it will always enter into infinite loop.

```
for (int i = 1 ; i <= 5 ; )
{
  S.O.P (i);
}
```

It will print 1 infinite times.

Teacher's Signature:.....

- Single statement or multiple statements in increment and decrement state.

```
for(int i=1; i<=5; S.O.P("one"), S.O.P(
{
  S.O.P(i); i++
}
```

Q. What is the output of the following code

```
class For10
{
  Public static void main (String[] a
  {
  for(;;)
  {
    S.O.P("Hello");
  }
```

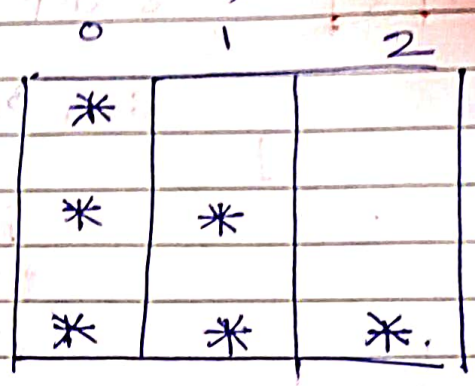
Above code will enter into infinite loop

- None of the stages are mandatory, If we donot set any condition, then comp will itself enter into infinite loop by default.

Teacher's Signature:

For Nested.

- A for loop within another for loop is called as Nested for loop.




```
i = 0 , j = 0
0       0
1       0 1
2       0 1 2
```

```
for (int i=0 ; i<=2 ; i++)
{
  for (int j=0 ; j<=i ; j++)
  {
    S.O.P ("*")
  }
    S.O.Println ();
}
}
```

Teacher's Signature:.....

eg

1		
1	2	
1	2	3

```
for (int i = 1 ; i <= 3 ; i++)
{
  for (int j = 1 ; j <= i ; j++)
  {
    S.O.P ("j");
  }
    S.O.Pln ();
}
```

i j

1

```
int temp = 1;
for (int i = 1 ; i <= 3 ; i++)
{
  for (int j = 1 ; j <= i ; j++)
  {
    S.O.P (temp);
    temp++;
  }
  S.O.Pln();
}
```

1		
2	3	
4	5	6

3
3

```
*  *  *        1  2  3
*  *           1  2
*               1
```

While loop :

Syntax :

```
while (condition)
{
    statements ;
    Statements ;
}
```

- If the while condition is true then the statements within the while will get executed.

If the condition is false then the statement within the while will not get executed.

Q. Write a java program to display a number from 1 to 5 using while loop.

```
int i = 1 ;
while ( i <= 5)
{
    S.O.P (i) ;
    i++ ;
}
```

Teacher's Signature:.....

8. Write a java program to check if the number 5 is present while iterating in decending order from 1 to 10.

```
int i = 10;
while (i >= 10)
{
  S.O.P (i);
  i++;
}
if (i == 5);
{ S.O.P ("The value of i is" + i);
  break;
}
  i--;
}
}
```

Teacher's Signature:

- In the while loop, it is mandatory to set the condition, If we do not set the condition then the compiler will also never set the condition.

```
{
 int i=1;
 {
  S.O.P (i);
  i ++;
 }
}
```

It will throw an error :

- We cannot use statement in while condition. It will throw an error.

Q. Write a java program to print even and odd numbers. from 1 to 10

Q. write a java program to find if a no. is prime or not.

Q. write a java program to find the factorial of a given number.

```
class fact1
{
 Public static void main(String[] args)
{
 int a=5;
 :
```

Teacher's Signature:.....

Transfer Statement.

- To Transfer the control from one area to another area, we go for transfer statement
- Transfer Statement involve :
  (i) Go to
  (ii) Return
  (iii) Try-catch
  (iv) break.
  (v) continue.

(i) Goto

- goto statements are considered as bad coding practice, because they bring lot of complications, Hence java had given alternative mechanism

(ii) Return statements:

Return statement will be teaching during method, and constructor

(iii) Try-catch :

being tought during exception handli

(iv) Break.

* Break statements are only meant to break the control either from the switch statement or from the loop

- Break statements only and only used in the conditions.

```
int i=3;
if (i==3)
{
  S.O.P (i);
  break;
}
```

It will throw an error.

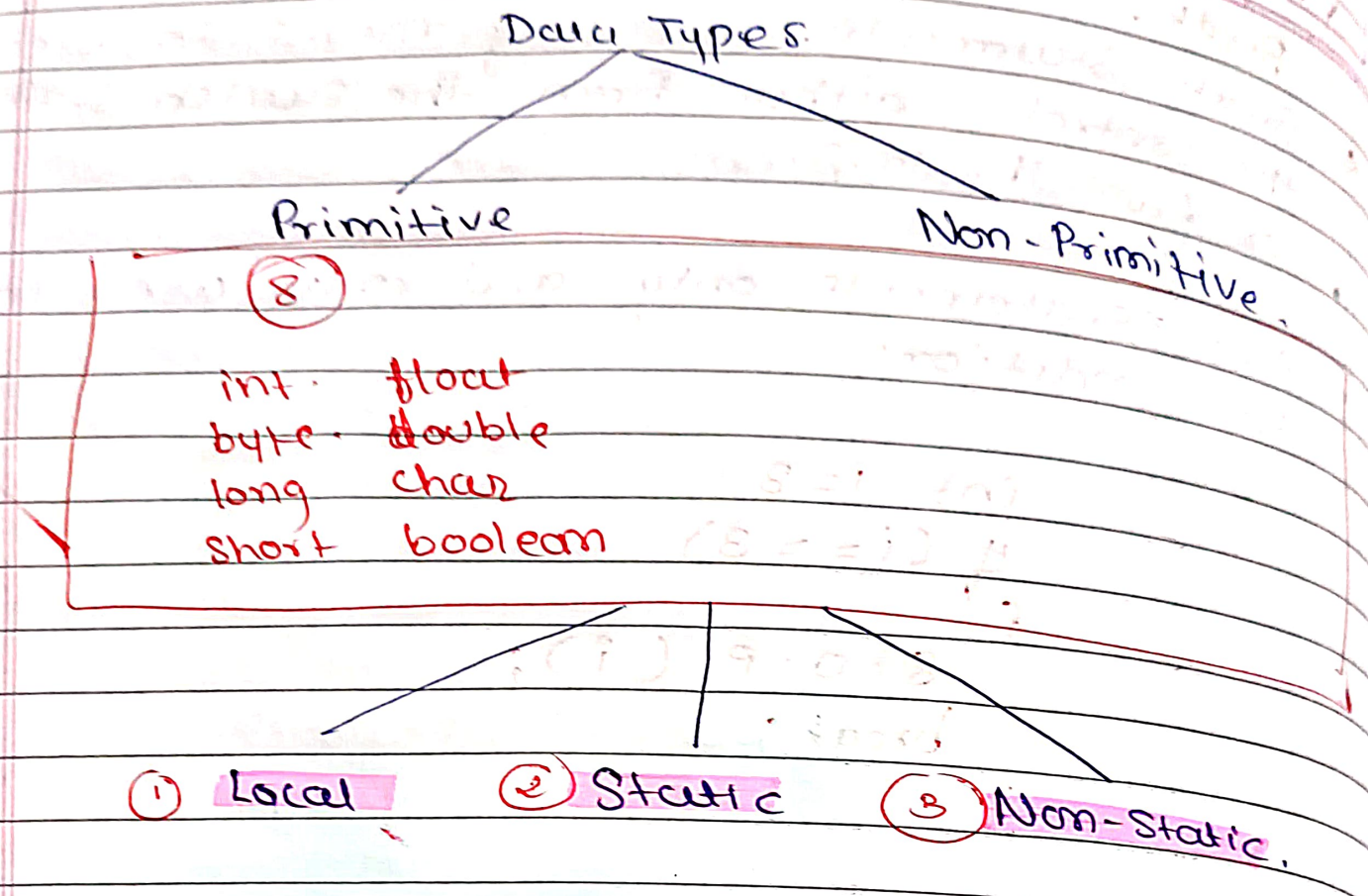
(v) Continue statement.

- The continue statements act like a skipping statements.
- The continue statements will be responsible to skip all the statements and move the java control at the end of the for loop. (not outside the for loop)

```
for (int i=1; i<=5; i++).
{ if (i==3)
{ S.O.P (i);
  continue; // skip.
} S.O.P (i);
}
```

Teacher's Signature:.....

3/09/25



(1) Local Variable:

Variable which are declared within a metho
or constructor or blocks, such variables are
called as local variables.

```
class Localvar1 {
    public static void main (String[] args)
{   //Local var
    int a=100;
    // Directly
    S.O.P (a);
}
}
```

Teacher's Signature:

Q. Is it mandatory to print the initialize local variable before printing?

→ Yes, it is mandatory.

```
Class Localvar2.
{
  P.S.V.m(String[] args)
  {
    int a;
    S.O.P(a);
  }
}
```

It will throw an error.

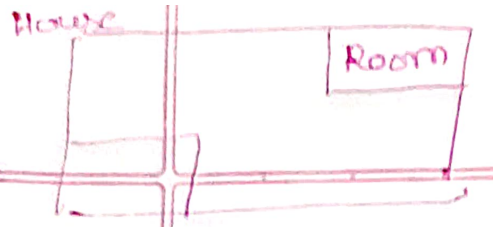
[2] Static Variable

- Variables which are declared outside the methods, constructor or block but within a class and is prefixed with a keyword called static such variables are called as static variable.

* The dot (.) operator in java is also called as access operator.

* The access operator is used to access a member within a specific area (area can be a classname)

Teacher's Signature:.....



House → Room
classname → Variable
classname.variable.

Q How many ways are there to access static members? How to access static members with respect to single class.

There are totally 3 ways to access static members.

(i) Directly
(ii) classname
(iii) Object creation.

```
Class Staticvar1
{
 Static int a=100;
 P.S.V.M (String[] args);
{
 S.O.P ("Start");
// Directly.
 S.O.P(a);
// classname.
 S.O.P(Staticvar1.a);
// Object creation.
 S.O.P ("Stop");
}
}
```

will teach during the concept of non-static & non-primitive.

Teacher's Signature:

Q. Add 2 numbers.

```
class StaticVar2.
{
  Static int a=100;
  Static float b=200.5f;
  P.S.V.m(String[] args);
  {
    float result;
    result = a+ StaticVar2.b; // int+float=float
    S.O.P("The result is"+result);
  }
}
```

output : 300.5

Q. Example : 3

```
Class StaticVar3..
{
  Static int a=100;
  Static float result;
  P.S.V.m(String[] args)
  { float b=200.5f;
    StaticVar3.result = a+b;
    S.O.P(result);
  }
}
```

result can be access by 2 way directly and classname.

result will be float.

dot access operator.

output : 300.5

Teacher's Signature:.....

8. How do we access static members from another class?

→ There are only 2 uproches they are
(i) classname
(ii) Object creation.

```
class Demo
{
   static int x = 100;
}
class Static Var4.
{
  P.s.v.m(String[] args)
  { // Directly
    System.out.println(x); → invalid.
    // classname
    System.out.println(Demo.x); → valid.
    // object → valid.
  }
}
```

Teacher's Signature:

staticVar5

```
class Cyber
{
  static float x = 100.5f;
}
class Success
{
  static int y=200;
}
class StaticVar5
{
  static float result;
  P.S.V.m(String[] args)
  {
    int z=300;
    result = Cyber.x + Success.y + z;
    S.O.P(result);
  }
}
```

① classname.var

② classname.var

another class

another class

directly

staticVar6

* Is it mandatory to initialize static variable before printing ?

→ No, it is not mandatory

- If we do not initialize static variable then the compiler will initialize the static variable with default values.

Teacher's Signature:.....

```
class StaticVar6
{
  static int x;
  static float y;
  Public static void main (String[] args)
  {
    S.O.P (x);
    S.O.P (y);
  }
}
```

compiler will directly print default values

what are default values of // byte // short // long // double // char // boolean

output : 0
0.0

StaticVar7

* We cannot make local var as static var.

```
class staticVar7
{
  P.S.V.m (String[] args)
  { static int x = 10;
    S.O.P (x);
  }
}
```

Output: error.

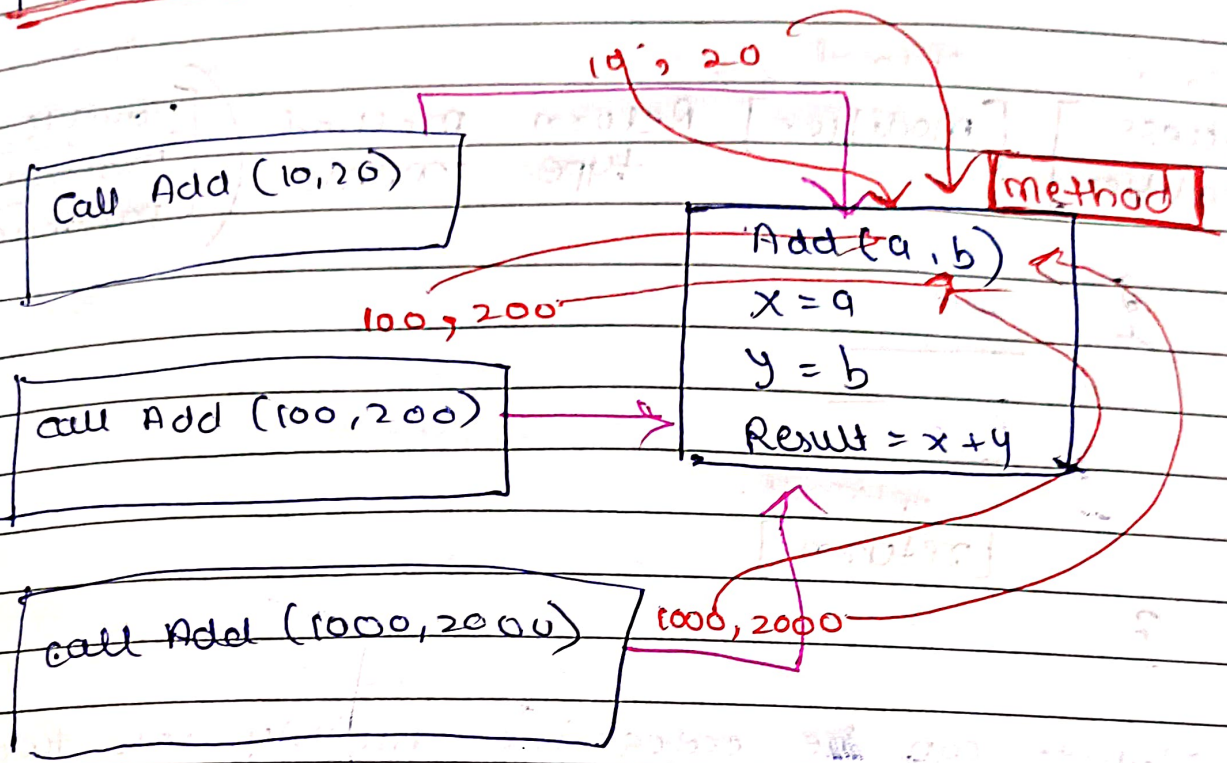
* Static variable cannot use inside main method.

Teacher's Signature:

To avoid the need methods.

less ① Duplicate code
less ② Length of code will be more
less ③ more memory
avoid ④ more time for execution.
⑤ Application will be slow

METHODS.



- A method can be defined as ~~Block~~ Set of statement which are meant to perform certain operation.

- Benefits of methods.
  (i) Due to methods we can reuse the code.
  (ii) Because of Reusability of code we can avoid duplicate code.
  (iii) The code length will be less
  (iv) Will consume less memory.
  (v) Will take less time for execution.
  (vi) It will help in increase the speed of the application.

Teacher's Signature:.....

4/09/25

{ [] } → optional.

Syntax of Method.

optional optional optional

[Access modifier] [modifier] Return type method name ([variable Argument parameter])

{

_____

_____

_____

optional.

[return]

}

- Methods can be categorize mainly into two ty
  (i) Static
  (ii) Non-static.

* Return statement;

The return statement is meant to return the control back to the caller method

- To access static method within a Sing class, there are 3 approches.
  i) Directly
  ii) classname
  iii) Objed creation.

Teacher's Signature:.....

```
class Example1
{ // called
  p.s.v.m (String[] args)
  {
    s.o.p ("Start");
  // Directly.
    m1();
  // classname.
    Example1.m1();
  // Object.
    s.o.p ("stop");
  }
  // called
  static void m1(){
  {
    s.o.p ("Inside the m1");
    return;
  }

}
```

Output : Start
Inside the m1
Inside the m1
Stop.

Teacher's Signature:.....

- The return statement should always be the last statement within the method.

```
static void test1()
{
  return;
  S.O.P ("Inside the test1 method");
}
```

Output: It will throw an error.

- There should be one return statement at the end of the method.

```
static void test1()
{
  S.O.P ("Inside the test1 method");
  return;
  return; ×
}
```

It will throw an error.
(unreachable statement)

- In a java program, we can create minimum one method or we can create multiple methods.
- If we do not write the return statement then the compiler will set the return statement
- The main method after the execution will also have to return the control back to the JVM.

void

- A method will use void as a return type whenever it only needs to return the control without any data or information.

```
class Example4.
{
  P.S.V.m (String[] args)
  {
    S.O.P ("Start");
    test1 ();
    S.O.P ("Stop");
    // return;
  }
  static void test1 ()
  {
    S.O.P (" Inside the test1 method");
    // return.
  }
```

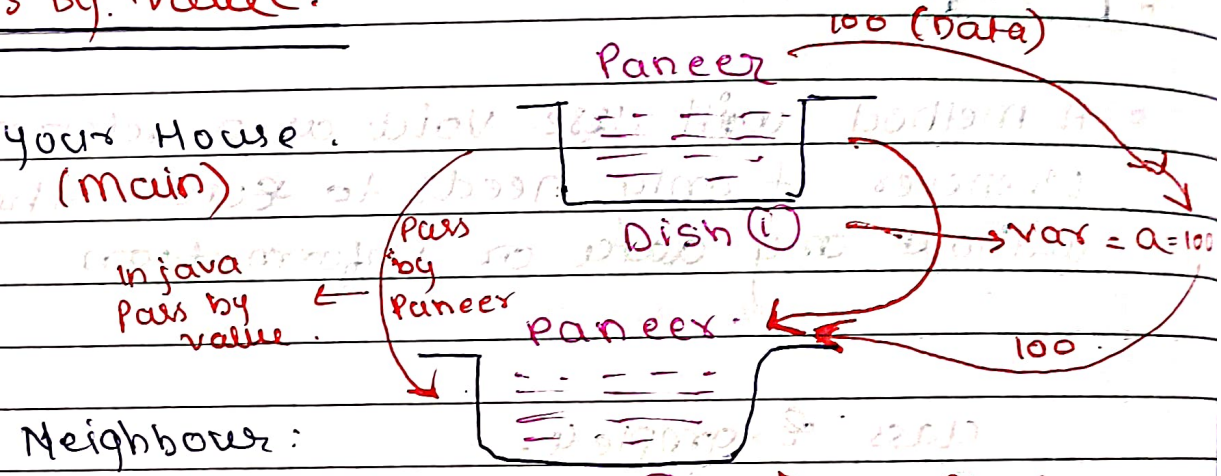
Teacher's Signature:.....

* Imp Note:

Whenever we declared local variable within a method, constructor, or block that variable can only and only be access directly within the same method, same constructor area, and same block area.

Q How do you access local var from one area to another?

Ans# Pass by value:



Q. When will you in real time go for pass by value

→ To access local var from one area to another

o Another defination of local variable: (Final Variables declared within a method or part of the method, constructor, blocks such variables are called as local variables.

Teacher's Signature:

Default Para.

```
class Example
{
    p.s.v.m(String[] args)
    {
        int a = 100;
        s.o.p("inside main" + a);
        test1(a);
    }
    static void test1(int a)
    {
        s.o.p("inside test1 method" + a);
    }
}
```

(1) (2) (3) (4) (5) 100 Part of a local variable. return (by default)

Final Keyword

Q. How do you make a variable behave as a constant?

→ We can make a variable behave as a constant variable by Prefixing the keyword / modifier called as Final.

We can ensure that a variable can behave as a constant.

```
{
    final int a = 100;
    a = 200;
    s.o.p(a);
}
```

without using final the output can be 200 but coz of using final keyword the output will be

Teacher's Signature:

Q. Write a java program to Calculate an area of rectangle ?

```
class Example 6
{
  P.S.V.m (String[] args)
  {
    int length = 100, breath = 200;
    test1 (length, breath);
    S.O.P ("Calculate");
  }
  Static void test1 (int length, int breath)
  {
    S.O.P ("Area of rec" + (length * breath))
  }
}
```

Q. Area of Triangle :

```
{ int breath = 10;
  int height = 20;
  test1 (breath, height);
}
Static void test1 (int a, int b)
{
  S.O.P ("Area of triangle" + (0.5 * a * b));
}
}
```

Teacher's Signature:.....

Q. Can a method in Java have default parameters. like in Python.

Q. Area of circle :

```
{
  int r = 10;
  final float pi = 3.14f ;
  test1 (r, pi);
  }
   Static void test1 (int r, float pi)
   {
    S.O.P ("area of circle" + (pi * r * r));
   }
}
```

Q. Area of Square :

```
{
   int side = 5;
   test1 (side);
}
   Static void test1 (int side)
   {
     S.O.P (" Area of Square" + (4 x side));
   }
}
```

Teacher's Signature:.....

5/09/25

- If a method returns a data, based on the type of the data. return, the return type with should also be of the same time

if Data = int. return = int type. } Data = float return = float type }

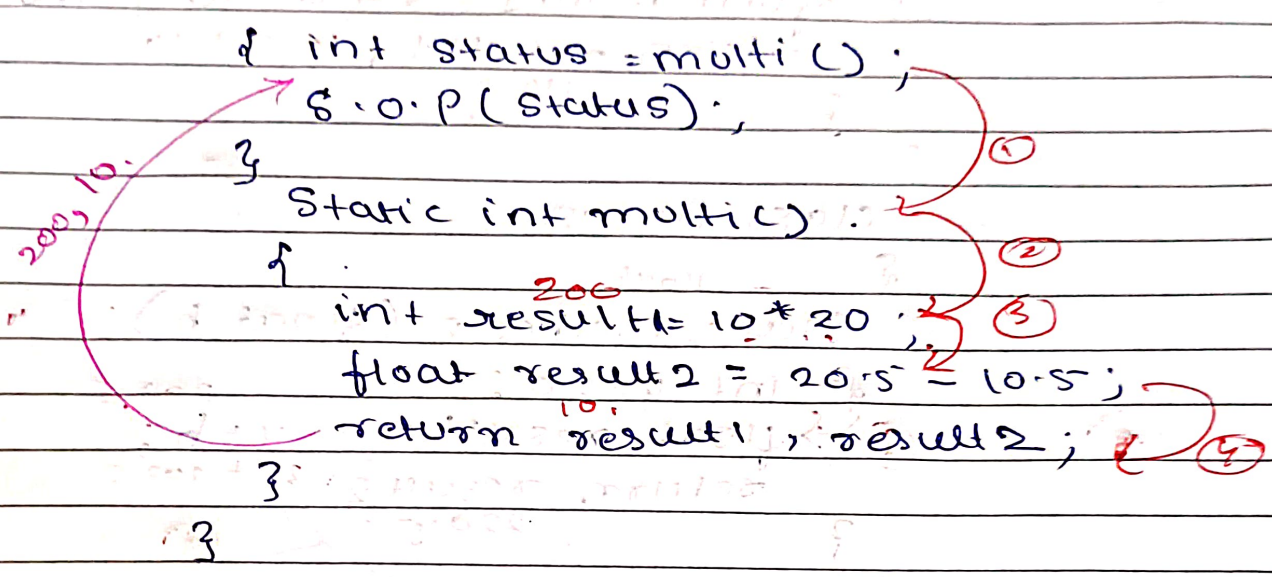
```
class Example7
{
    P.S.V.m(String[] args)
    int num1=10, num2=20;
    int result = add(num1, num2);
    S.O.P("Result is " + result);
}
Static int add(int num1, int num2)
{
    int result = num1 + num2;
    return result;
}
}
```

Output: Result is 30.

Teacher's Signature:

Return type.

- A method can accept multiple arguements, but cannot return multiple arguments.
- A method at any point of time can return only a single argument.



How to access method outside the main method.

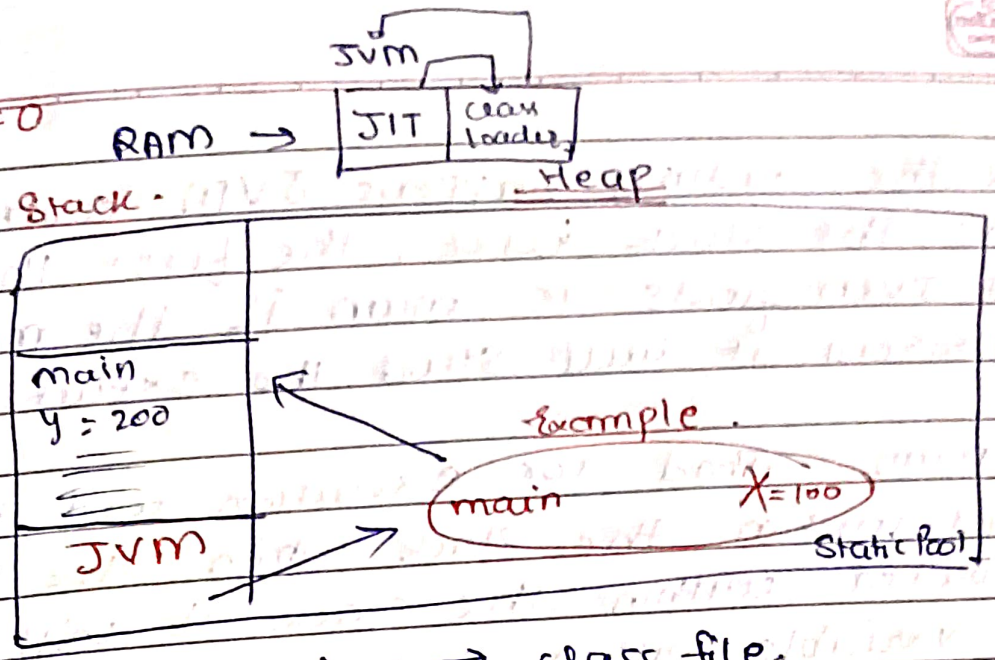
```
class Sample
{ static void m1()
  { S.O.P("Inside the M1");
  }
}
  class Example9
  { P.S.V.m(String[] args)
    { S.O.P("Start");
      // classname
      Sample.m1();
      // object
      S.O.P("Stop");
    }
}
```

Teacher's Signature:.....

```
class Cyber
{
    static float method1(int a)
    {
        float x = 10.5f;
        float result1 = x + a;
        return result1;
    }
}

class Success
{
    static float method2(float b)
    {
        int y = 20;
        float result2 = y + b;
        return result2;
    }
}

class Example10
{
    P.S.V.M(String[] args)
    {
        S.O.P("Start");
        int a = 100;
        float b = 200.5f;
        float result1 = Cyber.method1(a);
        float result2 = Success.method2(b);
        float finalResult = result1 + result2;
        S.O.P("final result" + finalResult);
        S.O.P("Stop");
    }
}
```

// calling m(1)
// call m(2)

Teacher's Signature:



LIFO

Javac → Compiler → class file.

1) Whenever JVM comes for execution the first thing it performs is it split the RAM into 2 parts are stack and one is heap.

2) After splitting the stack and heap the control is given to class loaders along with other components like JIT (just in time) compiler. The other components including JIT and class loaders are part of the JVM.

3) The class loaders are responsible to load the static member within the Static Pool

4) During the class loading the JIT compiler will not only monitor but will also initialize during the particular period.

5) The static members ~~before~~ whenever loaded is called as static pool and is given the same name as class name

Teacher's Signature:.....

6) Once the loading happens JVM control enters the stack area, the first thing what JVM finds is main if the main is present it will start the execution

7) The main method for execution will be loaded within the stack when the method is loaded within the stack it will create local variable within stack area. The stack always follows LIFO fashion (last in first out)

8) Once the main method performs the operation the control will be given to the JVM and the main method will come out of stack area and based on execution the JVM will shut down by coming out of stack area

10) Why is the main made as static.

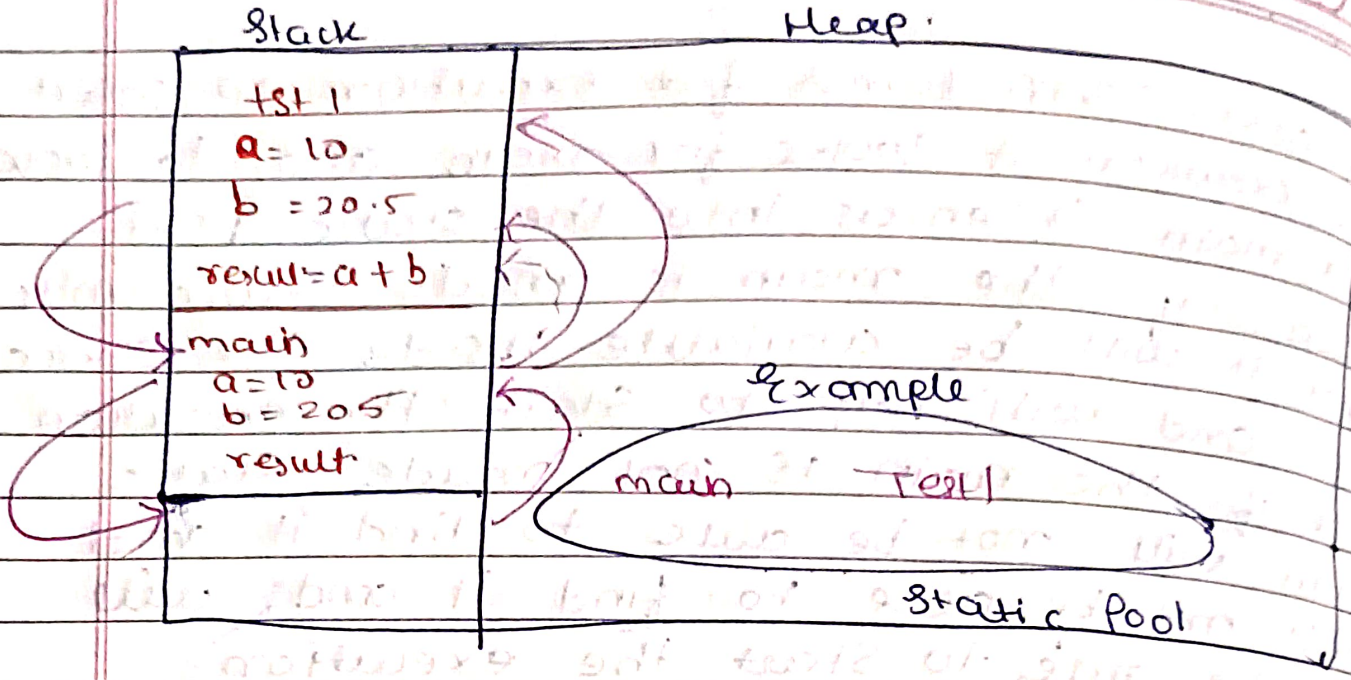
→ For the JVM to execute it will first search main in the static Pool and for main to be in static Pool it require to be static thats why we with public static void main.

Teacher's Signature

Whenever JVM comes for execution to start the execution it looks for main and to look for main it enters into the static pool area, if the main is made static only then it will be available inside the static pool and will help to start the execution but if the main is not made static, JVM will not be able to find it and will not be able to find it and will not be able to start the execution. Hence it is mandatory the main method should be static.

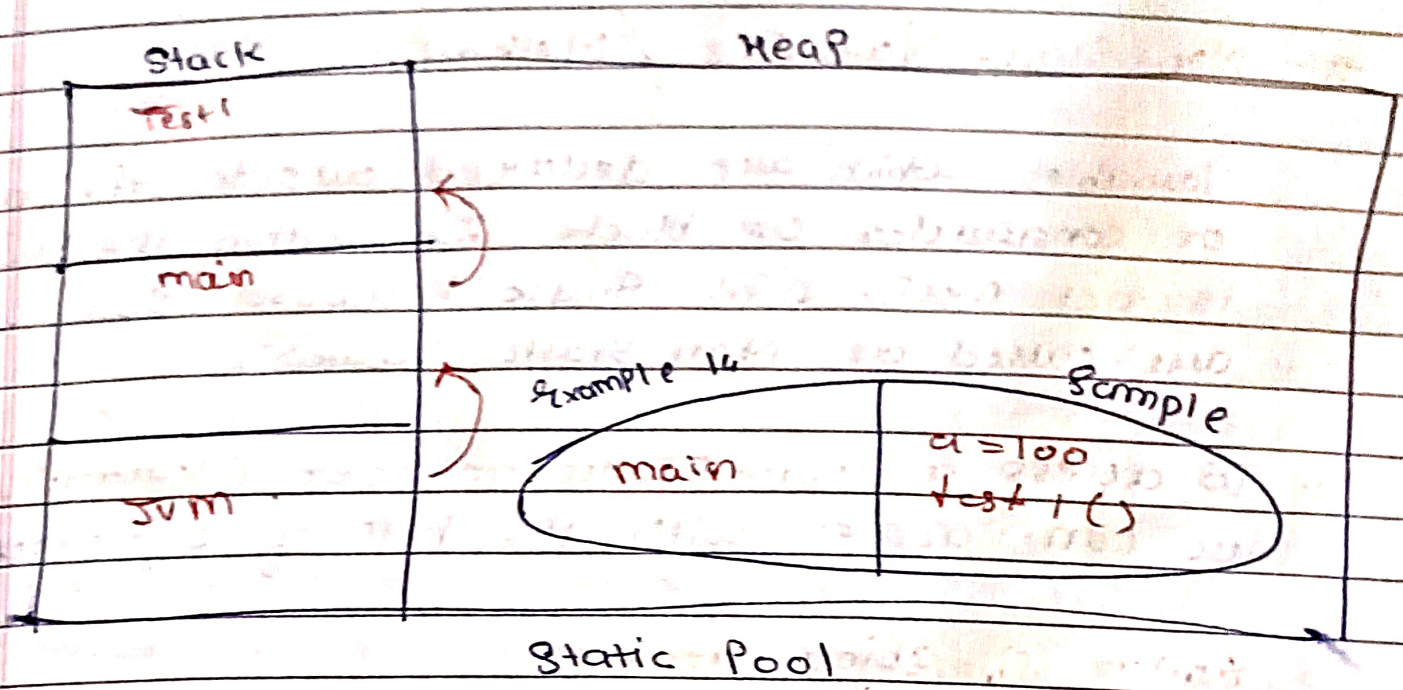
```
class Example 13
Public static
{ int a = 10;
  float b = 20.5f;
  float result2 = test1 (a, b);
  S.O.P ("Result is " + result2);
}
  Static float test1 (int a, float b)
{
  S.O.P ("inside test1 method");
  float result1 = a + b;
  float result1;
}
```

Teacher's Signature:.....




```
class sample.
{
   Static int a=100;
   Static void test1()
{
   S.O.P ("Inside Test1 method");
class Example1
{
   Public
{  S.O.P (" Start ").
   S.O.P (a), invalid.
   S.O.P (sample.a);
   Sample.. test1();
   S.O.P ("Stop");
}
```

Teacher's Signature:



Teacher's Signature:.....

4/sep/25

Non-Static Variable / Method.

Variables which are declared outside the method or constructor or block But within the class is not prefix with Static keyword such variable are called as Non Static Variable.

1. To access a non-static member (variable/method) we can access with the help of object creation.

Syntax of object creation:

```
classname.variable = new.classname();
                     ↓
                   keyword.
```

New Keyword.

The New keyword will perform 3 operations:

i) It is meant to allocate the memory into heap area. (Object)

ii) It will then load all the non-static members of the respective class. within the allocated memory or object area.

iii) It is responsible to generate a address with the help of JVM.

IQ When exactly are non-static member are loaded inside the memory?

→ During object Creation.

Teacher's Signature:.....

```
class NonStaticVar1
{
    int a=10;
    Public Static void main (String[] args)
    { NonStaticVar1 var = new NonStaticVar1() }
    S.O.P (var.a);
}
```

object creation

- Based on total number of new keywords that many number of object will created inside the memory.

- ~~Whenever ... of times get created~~

- If we create multiple objects for every object creation, non-static members will get loaded.

- Whenever changes are made to one object it will never had a inpact on other object.

Teacher's Signature:..................

```
class NonStaticVar2
{
  int a=10;
  P.S.V.m (String[] args)
  {
    NonStaticVar2 var1 = new NonStaticVar2();
    NonStaticVar2 var2 = new NonStaticVar2();
    S.O.P (var1.a);
    S.O.P (var2.a);
    S.O.P (var1);
    S.O.P (var2);
    var1.a = 100;
    var2.a = 1000;
    S.O.P (var1.a);
    S.O.P (var2.a);
  }

}
```

2 objects created

2 variables.

Q. Where exactly are static member created in the memory?

→ Inside the static pool.

Q. Where exactly are non-static member created in the memory?

→ Within the object area.

Q. Where exactly are local variables created in the memory?

→ Inside the static pool

Teacher's Signature:

```
class NonStaticVar3.
{
  int a=10; // Non static
  Static float b=20.5f;
  P.S.V.m(String[] args)
  {
    int c=20; // Local variable
    NonStaticVar3 ref = new NonStaticVar3();
    float result = ref.a+b+c;
    S.O.P("Result is"+result);
  }
}
```

* To access non-static member of another class we have to create an object.

```
class Demo.
{ int a=10;
  float b=20.5f;
}
class NonStaticVar3
{
  P.S.V.m(String[] arg)
  {
    Demo d=new Demo();
    S.O.P(d.a+d.b);
  }
}
```

Teacher's Signature:.....

```
class Demo
{
    int a=10;
}
class Sample
{
  float b= 20.5F;
}
class Tech {
  float result ;
}
class. Nonstaticvarx.
 {
  P.S.V.m (String[] arg)
  {
     Demo d = new Demo();
     Sample s = new Sample();
     Tech t = new Tech();

     t.result = d.a + s.b;

     S.O.P ("Result is" + result);
  }
}
```

Non-static Methods.

- If a method is not Prefixed with static Keyword. then by default. the method becomes non-static.

- Regardless wheather a method is static or non-static both the methods will always be executed in the stack.

```
class Nonstaticmethod1
{
  P.S.V.m(String[] args);
  {
  Nonstaticmethod1 ref = new Nonstaticmetho
                                        ();
  ref.m1();
  }
  void m1()
  {
   S.O.P ("inside m1 method");
  }
}
```

Teacher's Signature:.....

Multiclass

- classname
- object

2

4

object creation.

Which class
which area
which members.

Single class

- Directly
- classname
- object.

1

3

object creation

Static members.

Static Variable
Static method

Static Area.
Non-static Area

Non-static Variable
Non-static variable

Non-static members.

Single class

Directly
classname
object creation

5

7

Directly
object creation.

Multiclass

classname
object creation

6

8

object Creation.

Single class → Static area → Static members

(1) (Block 1)

```
Class Combo 1                    →  Single class
Static int a = 100;
Static void m1()
{
   S.O.P ("      ");
}
Public Static void main (String [] args)
{
   // Directly
   S.O.P (a); m1();
   // Classname
   S.O.P (combo1.a); combo1.m1();
   // Object
   Combo1 ref = new combo1();
   S.O.P (ref.a); ref.m1();
}
}
```

Static member

Static area

Which class :- multiclass → class 1, class 2

Which area - Static area →

Which member → Static member →

Teacher's Signature:.....

(2)

Multiclass
Static Area
Static Member

```
class Sample
{
  static int a = 100;
  static void m1()
  {
    S.o.p ("Inside m1");
  }
}
class combo2
{
  public static void main (String[] args)
  {
    // classname
    S.o.p ("Sample.a) ; Sample.m1();
    // object
    Sample s = new Sample ();
    s.o.p (s.a ; s.m1 ();
  }
}
```

multi class

Static members

Static area

{ which class = Single class
which area = Static area
which member = Non-Static }

(3) Which class - single class
which area - static area
which member - Non-static member.

```
class Combo3                          → Single class
{
    char c = 'x';                     } Non static
    void test1()                      } member.
    {
        S.o.p ("Inside test 1");      }
    }
    Public static void main().
    combo3 ref = new combo3();        } static area.
    S.o.P (ref.c);   // object Creation.
    ref.test1();
}
}
```

(4) which class - Multiclass.
which area = Static Area
which member - Non-static member

```
class cyber.                          } multi class
{
    boolean status = true;            } Non-static
    void test1()                      } member
    { S.o.P ("Inside test1");
}}
class combo4
{
    Public static void main (string[] args).
    {
        cyber c = new cyber()   // object creation
        SoP (c.status);
        c.test1();
    }
```

Static Area

Teacher's Signature:.....

block : 5

which class → Single class.
which area → Non-static Area.
which member → Static Member.

Single class

```
class combos
{
  Static int a = 100;
  Static void m1()
{
  S.o.p ("inside m1");
{
  Public Static void main();
{
  combos ref = new combos();
  ref.test1();
  S.o.p ("Stop");
}
  void test1()
{
  S.o.p ("inside test1");
//Directly  S.o.p(a); m1();
//Object    combos ref = new combos();
  S.o.p (ref.a); ref.m1();
}
}
```

Static member

Non-Static mem

Teacher's Signature:

[block : 6]

which class → multiclass
which area → Non- Static area
which member → Static member.

```
class Success
{
    Static int a = 100;
    Static void m1 ();
{
    S.o.p (" ");
}
class combo 6.
{
    Public Static void main( )
    combo6 ref = new combo6 ();
    ref test1 ();
}
    void test1 ()
    S.o.p (" Inside Test1)
// classname.
    S.o.p (Succes .a) ; Success .m1();
// object.

    Success s = new Success ();
    S.o.p (s.a) ; s.m1 ();
}
}
```

Static member.

Multi class

Non-Static area

Teacher's Signature:.....

[block 7]

- which class → Single class
  which Area → Non-Static Area
  which Member → Non-Static member

```
class Combo7
{
  int a = 100;
  void m1()
  {
    S.O.P ("Inside m1 method");
  }
  Public Static void main (String[] args)
  {
    Combo7 c1 = new combo7 ();
    c1.method1()
  }
  void method1()
  // Directly
  S.O.P (a);
  S.O.P (m1);

  // Object creation
  Combo7 c2 = new Combo7();
  S.O.P (c2.a);
  S.O.P (c2.m1);
  }
}
```

Teacher's Signature:

[block 8]

which class → Multi class
which Area → non-Static Area.
which member → Non-Static member.

Ex :-

```
class Demo
{
  int a = 100;
  void m1 ()
  {
    S.O.P ("m1 method");
  }
}
class Combo1
{
Public Static void main ();
{
  Combo1 C = new Combo1 ();
  C.method1 ();
}
  void method1 ()
  { // object Creation
  Demo D = new Demo ();
   S.O.P (D.a);
   S.O.P (D.m1);
  }
}
```

Multi class

non-Static member.

non-Static Area.

Teacher's Signature:.....

Non-Primitive or Reference or class type.

* Primitive type

↓

```
int a = 100;
```

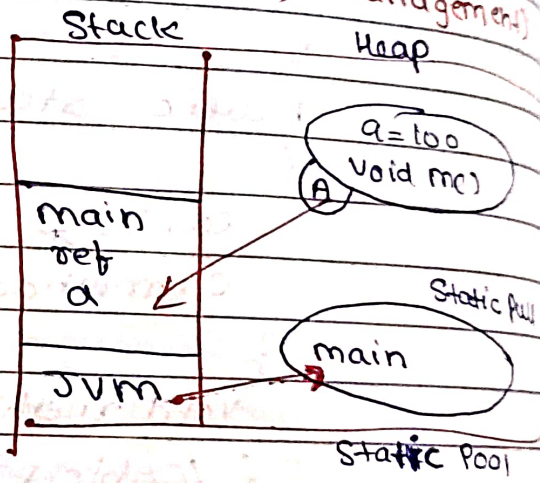
* Non-Primitive type / Reference.

```
Demo d = New Demo();
```

* In a non-Primitive variable it involves 2
  i) Declaration
  ii) Instantiation (Initialization)

```
class NonPrim1
{
  int a = 100;
  void m1()
  {
    S.o.p("Inside m1");
  }
  P.S.V.m()
  {
    NonPrim1 ref1 = new NonPrim1();
    ref.m1();
  }
```

memory (management)



Output : a = 100
inside m1.

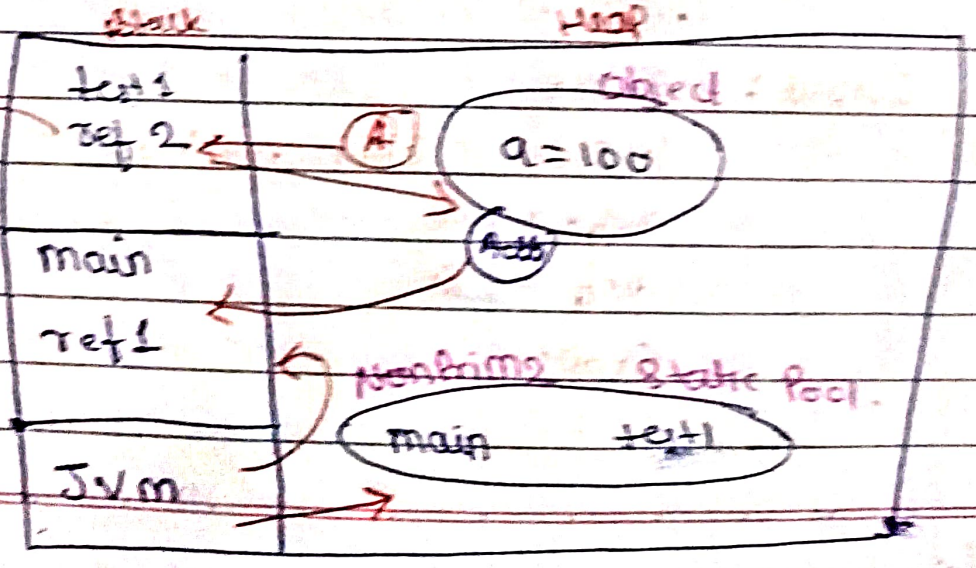
Teacher's Signature:

Q. What will you go for Pass by reference?

→ To access local variable from one area to another area we use pass by reference.

* Pass by Reference / Address

```
class NonPrim2
{
  int a = 1000;
  Public static void main (String[] args)
  {
    System.out.println ("Start");
    NonPrim2 ref1 = new NonPrim2();
    S.o.p (ref1.a);
    int a=100;
    test1 (ref1);
    S.o.p ("Stop");
  }
  Static void test1 (NonPrim2 . ref2)
  {
    S.o.p ("Inside test1");
    S.o.p (ref2.a);
  }
}
```



Output: Start
ref1.a = 1000
Inside test1
ref2.a = 100
Stop

Teacher's Signature:

Sample3

```
class NonPrim3
{
  char x = 'a';
  P.S.V.m (String [] args)
{
  S.O.P ("Start")
  NonPrim3 ref = new test1();
  S.O.P (ref.x)
  S.O.P ("Stop");
}
Static NonPrim3 test1 ()
{
  S.O.P ("Inside Test1 method");
  NonPrim3 ref = new NonPrim3();
  S.O.P (ref.x);
  return ref;
}
}
```

Note :- Return by Reference one area to meth in a same class.

Output : Start
Inside Test1 method.
ref.x = a
ref.x = a
Stop.

Teacher's Signature:

Example 3

```
Class Nonprim4
{
  char x = 'a';
  Public Static void main (String[] args)
  {
    char x = m1 ();
    S.o.p ("Inside the main method");
    S.o.p (x);
  }
  Static char m1 ()
  {
    S.o.p ("Inside the m1 method");
    S.o.p (x);
    Nonprim4 ref = new Nonprim4 ();
    S.o.p (ref.x);
    m2 (ref);
    char y = ref.x;
    return y;
  }
  Static void m2 (Nonprim4 ref)
  {
    S.o.p ("Inside m2 method");
    S.o.p (ref.x);
    return;
  }
}
```

(return ref.x)

output: Inside m1 method
a
Inside m2 method
a
Inside m3 method.
a.

Teacher's Signature:

Assignment

```
class Nonprims
{
  boolean status = true;
  Public static void main (String [] args)
  {
   Nonprim5 ref = new Nonprims();
   m1 (ref);
  }
   Static void m1 (Nonprim5 ref);
   {
     S.O.P ("Inside m1 method");
     S.O.P (ref.status);
     Nonprims ref1 = new Nonprims();
     ref1.m2 (ref);
     return;
   }
   void m2 (Nonprims ref);
   {
     S.O.P ("Inside m2 method");
     S.O.P (ref.status);
     return;
   }
}
```

O/P :- Inside m1 method
true
Inside m2 method
true.

Teacher's Signature:

Example

```
class cyber {
int a=100;

Static void m1 (cyber cc) {
 S.O.P ("Inside m1 method" + cc.a);
 }
}

class Success {
float b = 200.5f;

Static void m2 (Success ss) {
 S.O.P ("Inside m2 method" + ss.b);
 }
}

class Nonprime {
 public Static void main (String[] args) {
  S.O.P ("Start");
  cyber c = new cyber();
  Success s = new success();

  Cyber.m1 (c);
  Success.m2 (s);

  S.O.P ("Stop");
 }
}
```

O/P = Start
Inside m1 method 100
Inside m2 method 200.5
Stop

Teacher's Signature:

//Example7

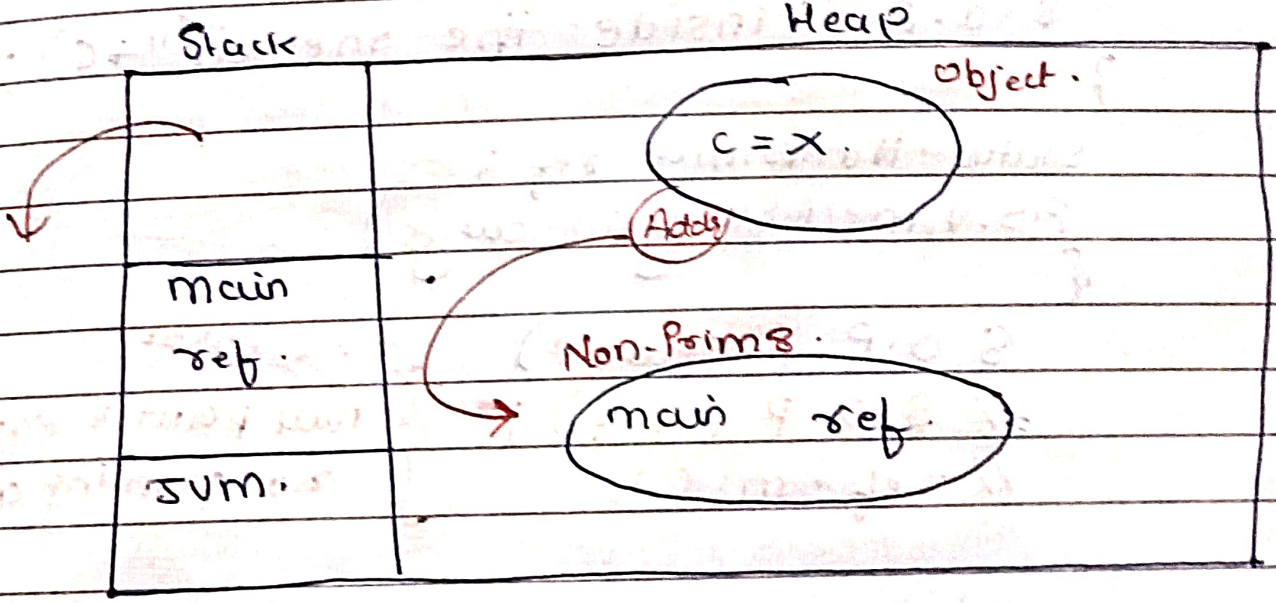
```
class Cyber
{
  Static void m1 ()
  { S.O.P ("Inside m1 method");
    result1 = a+x;
  } }
class Success
{. Static void m2 ()
{ S.O.P ("Inside m2 method");
   result2 =b +y; } }
class Training
{
   void test1 ()
   { a =100;
   }
   void test2 ()
   { b = 200.5f;
   }
}
class Nonprim7
{ x=10; y=20.5f.
  P.S.V.m (String[] args)
  {
    S.O.P ("Start")
    fResult = result1 + result2;
    S.O.P (fResult)
    S.O.P ("Stop")
  }
}
```

Teacher's Signature:

Non-Primitive (Static)

```
class NonPrim8
{
  char c = 'x';
  Static NonPrim8 ref = New NonPrim8();
  Public Static void main (String[] args)
  {
    S.O.P ("Start");
    S.O.P (ref.c);
    S.O.P ("Stop");
  }
}
```

Static ref variable.



Q. Is it possible to create an object before main starts the execution.

A. Yes.

Teacher's Signature:.....

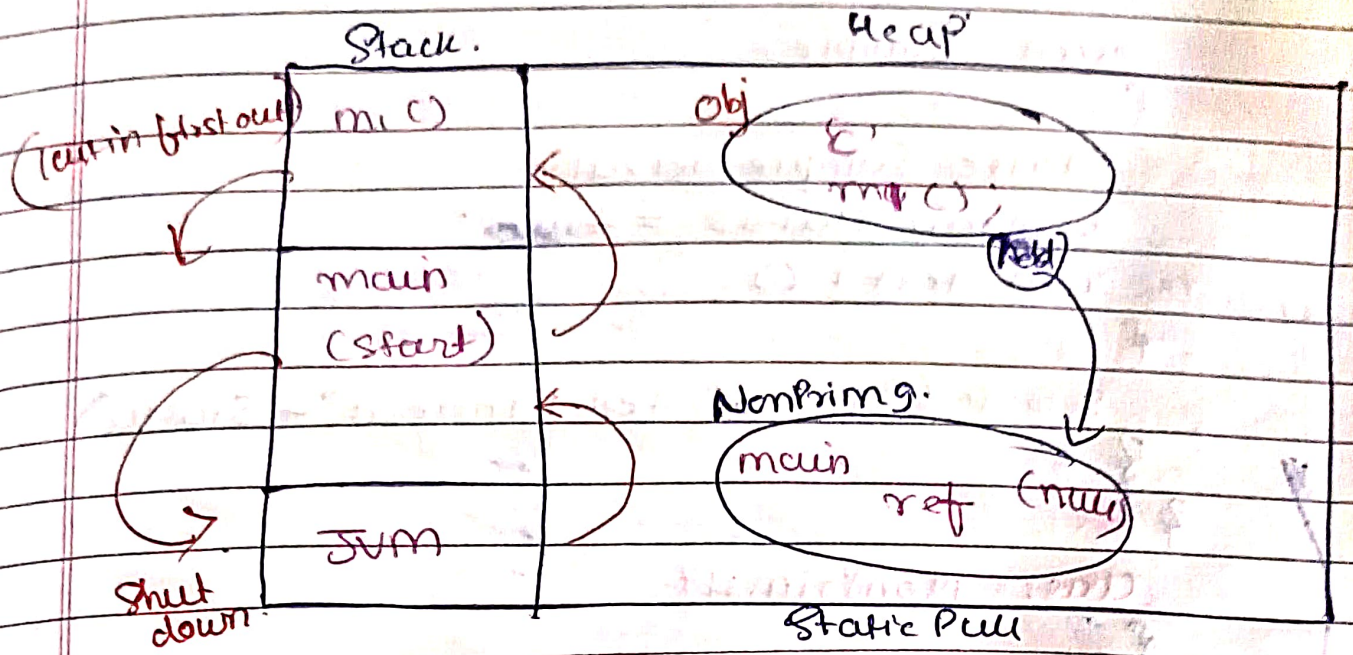
* Default values are assigned to static and Non-static variable based on the type of variable.

* Default values are also assigned to static and non-static reference variable which is null.

```
class NonPrimg
{
  char c = 'x';
  void m1();
  {
    S.O.P ("inside the method"+c);
  }
  Static NonPrimg ref;
  P.S.V.m (String[] args)
  {
    S.O.P ("start");
    // S.O.P (ref);
    // ref.m1();

    ref = new NonPrimg();
    ref.m1();
    S.O.P ("stop");
  }
}
```

error
null pointer exception
not pointing anything

Teacher's Signature:



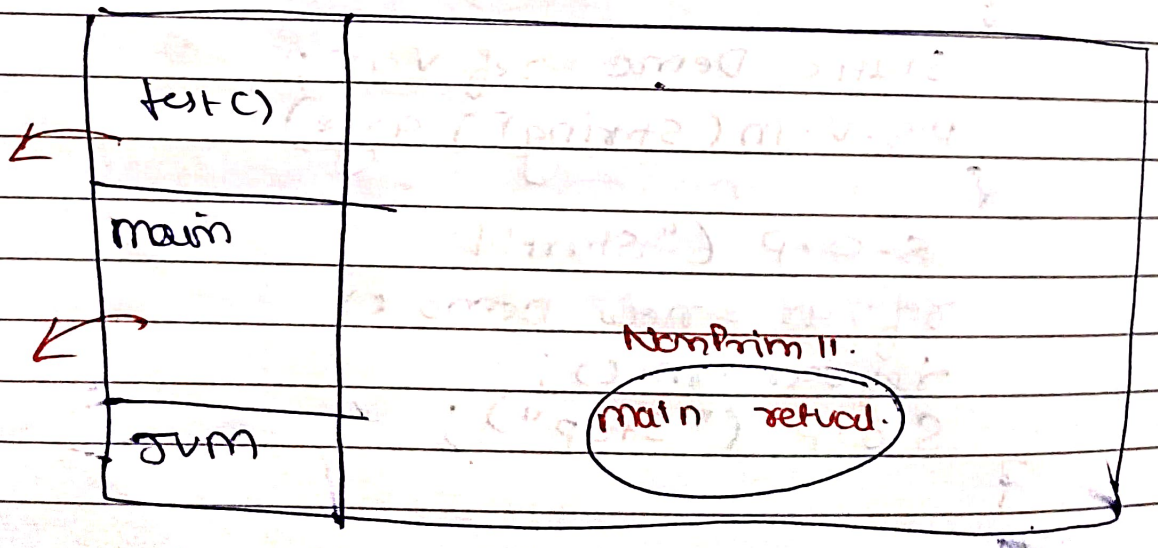
example 10

```
class Demo.
{ float x = 100.5f;
  void m1()
  {
    s.o.p ("inside m1" + x);
  }
{ class Nonprim10;
  {
    Static Demo refVar;
    P.S.V.m(String[] args)
  {
    S.O.P ("start");
    refVar = new Demo()
    refvar.m1();
    S.O.P ("Stop");
  }
  }
```

Teacher's Signature:.....

Example 11

```
class Sample
{
  Static Sample retval;
  boolean status = true;
  void test1 ();
  {
    S.O.P ("Inside test1 method" + Status)
  }
}
class NonPrim11
{
  P.S.V.M (String [] args)
  {
    S.O.P ("Start").
    Sample retval = new Sample ();
    Sample retval . test1 ();
    S.O.P ("Stop");
  }
}
```



Teacher's Signature:.....

Non-Primitive - Non-static

example 12

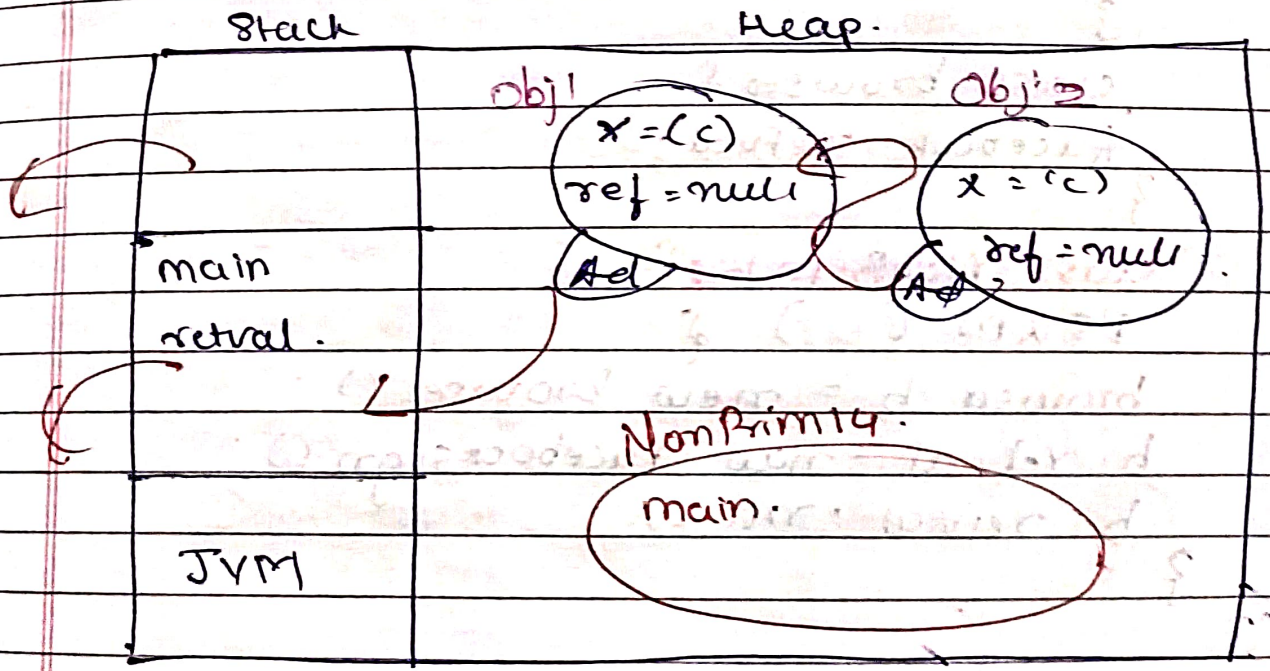
```
class cyber {
  int a = 10;          // Instance variable.
}
class Success {
  void m1() {          // method
    S.O.P ("Inside m1 method");
  }
}
class Demo {           // Reference variables
static cyber refvar1;
static Success refvar2;
}
class NonPrim12 {
P.S.V.M (String[] args)
System.out.Println ("start");

Demo.refvar1 = new cyber();   // object of cyber
Demo.refVar2 = new Success(); // object of Success

System.out.Println (Demo.refvar1.a);

Demo.refvar2.m1();
S.O.P ("Stop");
  }
}
```

Teacher's Signature:.....

Example 13

```
class X1
{
  static X1 ref;
    void Z1 (String info)
    {
      S.O.P("This is a method");
    }
}
class NonPrim13.
{
  P.S.V.M (String[] args)
  {
    S.O.P ("start");
    X1.ref = new X1 ();
    X1.ref.Z1 ("This is a method");
    S.O.P ("Stop");
  }
}
```

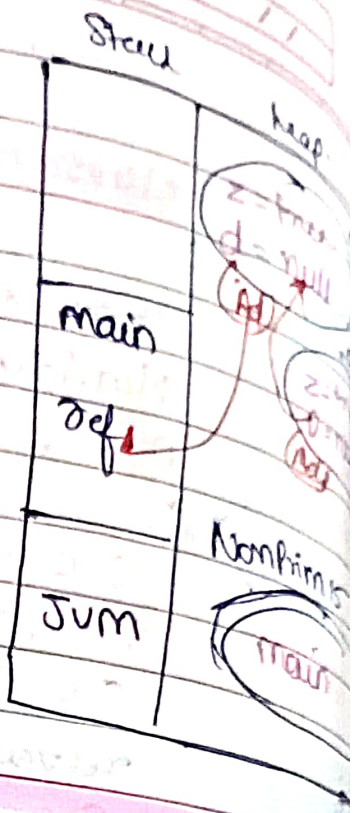
```
class NonPrim14
{
  char x = 'a';
  NonPrim14 ref;
  Public static void main (String [] args)
  {
    S.O.P ("start");
    Nonprim14 retval = new Nonprim14();
    S.O.P (retval.ref);
    retval.ref = new Nonprim14();
    S.O.P (retval.ref.x);
    S.O.P ("Stop");
  }
}
```



Teacher's Signature:.....

23/9/25

```
class Demo{
boolean z=true;
Demo d;
}
class NonPrim15 {
psvm (String[] args) {
Demo ref = new Demo ();
ref.D = new Demo();
S.O.P(ref.D.z);
}
}
```




```
class facebooksign {
int a=10;
void m1() {

}
class browser {
facebook refval;
}
class NonPrim16 {
psvm ( ) {
browser b = new browser();
b.refval = new facebooksign();
b.refval.m1();
}
}
```

* Named object creation & nameless object creation

```
class NonPrim17 {
void m1() {
S.o.P ("Inside m1"); }
psvm (String[] args) {
NonPrim17 n = new NonPrim17 ();
n.m1();
new Nonprim17().m1();
```

Ques 1) How many ways are there to create an object?

→ There are Two ways to create an object.

1) named obj creation & nameless obj creation.

- nameless object creation doesn't mean, we do not store the address, it actually means which store the address in another area.

```
class NonPrim18 {
void m1() {
S.o.P ("    ");}
void m2()
{
S.o.P ("    ");}
Static void test1 (NonPrim18 ref) {
    ref.m1();
    ref.m2();
Psvm (String[] args) {
test1 (new NonPrim18 ());}
```

Teacher's Signature:.....

* This keyword *

- This keyword is when to store or hold current class object address

```
class NonPrim1g {
int a = 10;
void m1 () {
S.O.P (a);
S.O.P (this.a); }
P S V m (String [] args) {
Nonprim1g ref = new Nonprim1g();
ref . m1 (); }}
```

Q. When to be use this keyword ?

→ To differentiate between local var & instance variable. The variables names are same to use this keyword.

```
class Nonprim20 {
int a = 100;
void m1 (int a) {
S.O.P (a);
S.O.P (this.a) }.
P.S.V.M (String [] args) {
new Nonprim20 () . m1 (10);
}
}
```

Teacher's Signature:

24/Sep/25

```
class Hdfc
{ int amountBal ;
  void depositAmount ( int amountBal) ;
  {
    this.amountBal = amountBal ;
    S.O.P ("The amount Deposited Successfully");
  }
  void getBalance ()
  {
    S.O.P ("The Balance is" + amountBal);
  }
}
class Nonprim21
{
  P.S.V.m (String[] args)
  {
    S.O.P ("start"); ①
    Hdfc cust1 = new Hdfc();
    cust1.depositAmount (5000) ;
    cust1 . getBalance () ;
    S.O.P ("Stop");
  }
}
```

5000

5000

Teacher's Signature:.....

24/Sep/25

```
class Hdfc
{ int amountBal ;
  void depositAmount ( int amountBal) ;
  {
    this.amountBal = amountBal;
    S.o.p ("The amount Deposited Successfully")
  }
  void getBalance ()
  {
    S.o.p ("The Balance is" + amountBal);
  }
}
class NonPrim21
{
  P.S.V.m (String[] args)
  {
    S.o.p ("start");
    Hdfc cust1 = new Hdfc();
    cust1.depositAmount (5000);
    cust1.getBalance ();
    S.o.p ("Stop");
  }
}
```

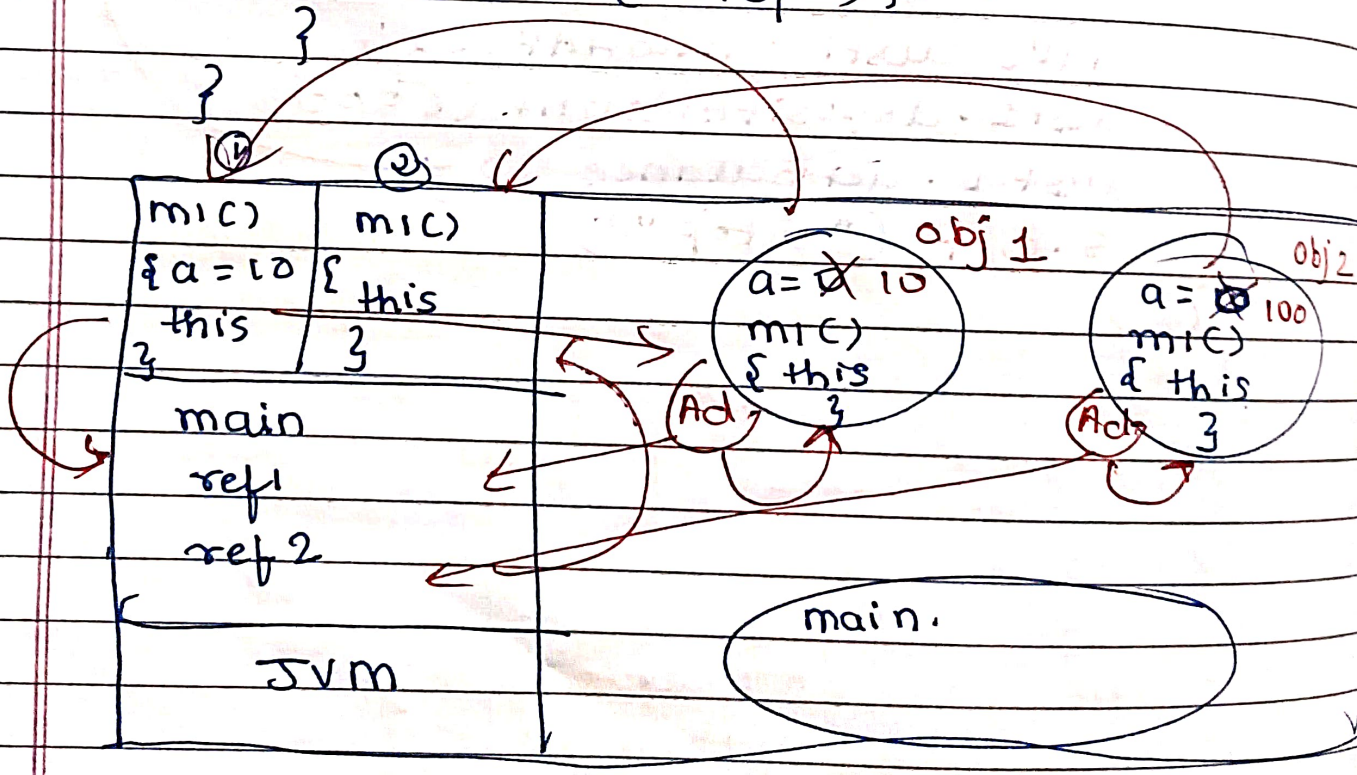
5000

5000

Teacher's Signature:.....

* we can use this keyword in multiple obj

```
Class NonPrim22
int a;
void m1 (int a)
{
    S.o.p ("inside m1 method");
    this.a = a;
}
Public Static void main (String [])
{
    S.o.p ("start");
    NonPrim22 ref1 = new NonPrim22 ()
    ref1.m1 (10);
    S.o.p (ref1.a);
    NonPrim22 ref2 = new NonPrim22 ()
    ref2.m1(100);
    S.o.p (ref2.a);
    S.o.p ("stop");
}
}
```



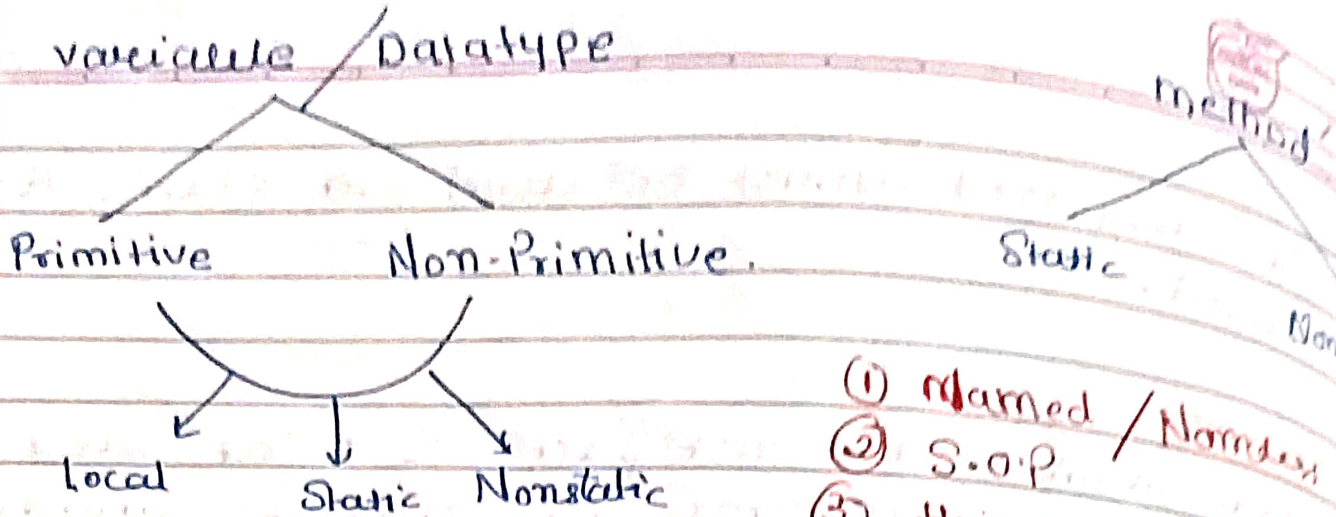
- 'This' keyword cannot be used in static Area / method.

Summary

Static members are not parts of object area. Hence it cannot store the address inside this keyword. Hence, we cannot use this keyword in static area.

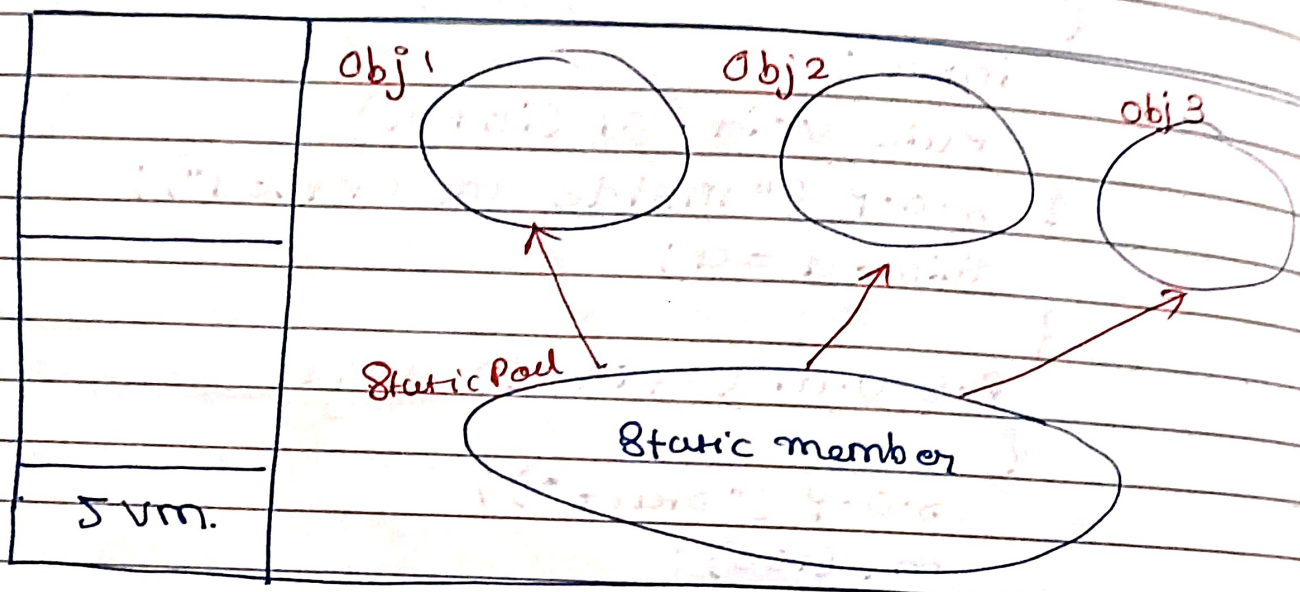
```
class NonPrim23
{
  int a;
  Static void m1 (int a)
{ S.o.P ("Inside m1 method");
  this.a = a;
}
  P.S.V.m (String[] args)
{
   S.O.P ("Start");
   m1 (100);
   S.O.P ("Stop");
}
};
```

Teacher's Signature:.....



① Named / Nameless
② S.o.p
③ this keyword
④ Static-Nonstatic (When to use)

When to use Static method.



* If u want to access common data in every object then we will use Static member.

* If u want to access different data in different object.

Teacher's Signature:

11

When to use / declare member as a static or non-static area.

```
class IMS.
{ int StudID;
  String StudName;
  Static String instituteName = "Cyber Success";
  void setStudInfo (int StudID, String StudName)
{
  this.StudID = StudID;
  this.StudName = StudName;
}
  Void getStudInfo ()
{
  S.O.P ("StudID is" + StudID + "StudName is" +
        StudName + " Institute is" + IntituteName)
}
  Static Void attendance ()
{
  }
Class NonPrim 24
{
  P.S.V.M ( String[] args){
  IMS Stud1 = new IMS (); Stud1.setStudInfo
(101, "Rahul"); Stud1.getStudInfo();
  IMS Stud2 = new IMS; Stud2.SetStudInfo
(202, "Lavanya"); Stud2.getStuInfo();
  }
}
```

Teacher's Signature:.....

* Whenever there is a common service or a common data require accross the program, we make such member as static.

* Whenever a service or data is unique by nature we preferred to make it a non-static.

Q. What is a difference between classes & object?

	class	object
(1)	A class can be defined as design / set of char / blueprint	(1) An implementation of design / set of char and blueprint is called as objec
(2)	classes consume generic or Hard Disk memory.	(2) Whereas object consume RAM memory.
(3)	To create a class we have use keyword called as class.	(3) To create an object we have to use new keyw
(4)	A class doesnot depend on object.	(4) An object depend on class.
(5)	In java program we need to create one class minimum.	(5) With the help of one class min we can crea one object. or we can Create multiple obj.

Teacher's Signature:.....

* Whenever there is a common service or a common data require across the program, we make such member as static.

* Whenever a service or data is unique by nature we preferred to make it a non-static.

Q. What is a difference between classes & object?

	class	object
1.	A class can be defined as design / set of char / blueprint	An implementation of design / set of char and blueprint is called as object
2.	classes consume generic or Hard Disk memory.	Whereas object consume RAM memory.
3	To create a class we have use keyword called as class.	To create an object we have to use new keyword
4	A class doesnot depend on object.	An object depend on class.
5	In java program we need to create one class minimum.	With the help of one class min we can create one object. or we can create muttiple obj.

Teacher's Signature:

✳ Constructor ✳

Constructors can be defined as special methods which are meant to perform operations, but do not have a return type.

★ Rules to create a constructor :

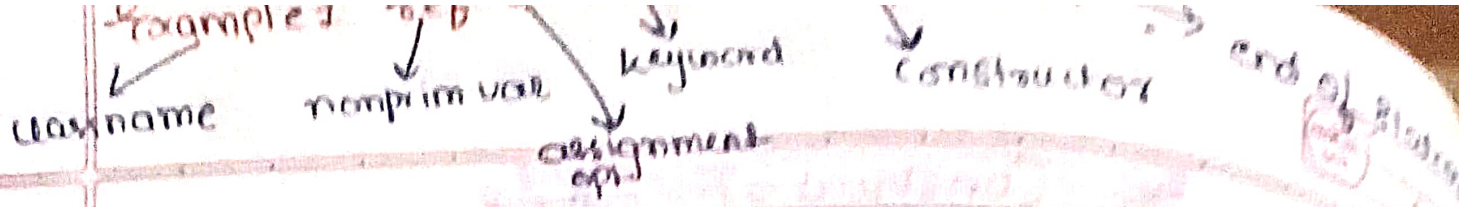
1. Constructor name should be same as classname.
2. Constructors can accept arguments, but constructors do not have a return type.

1) Example 1

Constructors are always called during object creation.

```
Class Example1
{ Example1()
  { S.O.P("Inside 0 Arg Const");
  }
  Public static void main(String[] args)
  { S.O.P("Start");
    Example1 ref = new Example1();
    S.O.P("Stop");
  }
}
```

O/P → Start
Inside 0 Arg Const.
Stop.



Example : 2

```
class Example2
{ Example2 ()
  {
     S.O.P ("Inside 0 Arg Const");
     return;
  }
  P.S.V.M (String [] args)
  {
     S.O.P ("Start");
     Example2 ref = new Example2 ();
     S.O.P ("Stop");
  }
}
```

O/P :- Start.
Inside 0 Arg Const
Stop.

1) To return the control from the constructor we use return statement.

2) If we do not add the return statement then the compiler will add the return statement.

Teacher's Signature:.....

Example : 3

```
class Example3
{ Example3()
  { S.O.P ("Inside 0 Arg const");
  }
  Public Static void main (String[] args).
  { S.O.P ("Start");
    Example3 ref1 = new Example3();
    Example3 ref2 = new Example3();
    Example3 ref3 = new Example3();
    S.O.P ("Stop");
  }
}
```

O/p → Start.
Inside 0 Arg const.
—"—"—"—
—"—"—"—
Stop.

Q. How many times can a constructor be called?

→ Based on number of time we create an object or based on number of times we use new keyword. that many times the constructor will get executed.

Teacher's Signature:.....

* Example4

```
class Example4
{ Example4()
  { S.O.P("Inside 0 Args Const");
  }
  Example4(int a)
  { S.O.P("Inside 1 Args Const");
  }
  P.S.V.M(String[] args)
  {
    S.O.P("Start");
    Example4 ref1 = new Example4();
    Example4 ref2 = new Example(10);
    S.O.P("Stop");
  }
}
```

O/P → Staut.
Inside 0 Arg Const
Inside 1 Arg Const
Stop

We can create multiple constructors and at min one constructor within the java class.

Examples

```
class Examples
{
  /* Examples()
  {
  } */
  P.S.V.m (String[] args)
  {
    S.O.P ("Start") ;
    Examples ref1 = new Examples();
    S.O.P ("Stop") ;
  }
}
```

1. Default
2. Empty Imp
3. 0 Arg Const.

i) If we do not create any constructor, only then the java compiler will create a constructor

ii) Whenever the java compiler create the constructor it will always create default, zero argument empty implementation constructor.

Teacher's Signature:.....

* Example 6.

```
class Example6
{
  Example6()
  { S.o.p("Inside o arg const - one");
  }
  Example6()
  {
      S.o.p("Inside o arg cons+ - Two");
  }
  Public Static void main(String[] args)
  {
     S.o.p("Start");
     Example6 ref1 = new Example6();
     S.o.p("Stop");
  }
}
```

* Duplicate constructors are not allowed within a class.

* Example 7

```
class Sample
{ Sample()
  { S.O.P("Inside Sample 0 Arg Const");
  }
}
class Demo
{ Demo()
  {
    S.O.P("Inside Demo 0 Arg Const");
  }
}
class Example7
{ P.S.V.M (String[] args)
  { S.O.P("Start");
    Sample S1 = new Sample();
    Demo d1 = new Demo();
    S.O.P("Stop");
  }
}
```

Q. Is it possible to call constructor from another class?

→ Yes, by creation of object of respect class.

Teacher's Signature:

Example 8

```
class Example8
{ int a;
  Example8()
  {
    System.out.println("Inside 0 Arg"+a);
    a=10;
  }
  void m1()
  {
    S.o.P("Inside m1 method"+a);
  }
  Public static void main(String[] ar
  { S.o.P("start");
    Example8 ref = new Example8();
    ref.m1();
    S.o.P("stop");
  }
}
```

Q. What are the different types of constructors?

→ There are mainly 3 types of constructors

1. Default Const
2. Zero argument Const
3. Parameterized Arg const

Teacher's Signature: