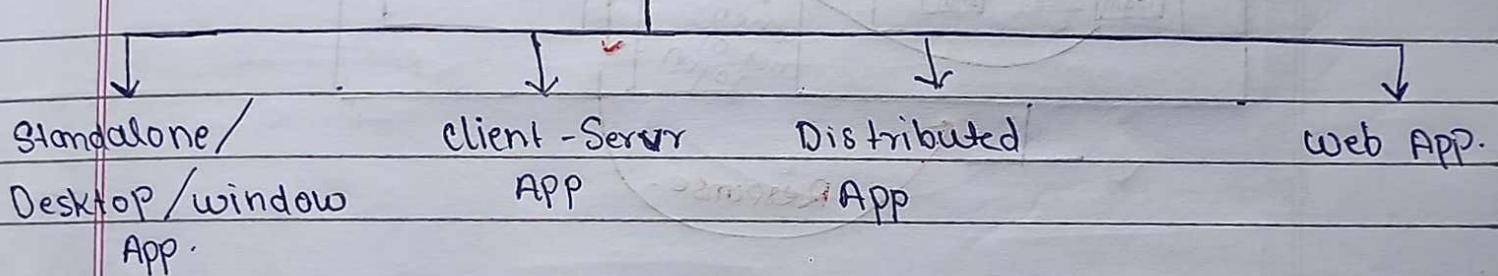


12th Aug

Types of Software:

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

Mainly There are 4 types of software.



[1] Standalone App./ windows / Desktop.

[User machine]

<input type="checkbox"/>	① Word
<input type="checkbox"/>	③ Sticknote
<input checked="" type="checkbox"/> notepad.	
<input type="checkbox"/>	④ Excel
<input type="checkbox"/>	⑤ PPT

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 / Desktop.

Eg: Word , Stickynotes , Excel , PPT.

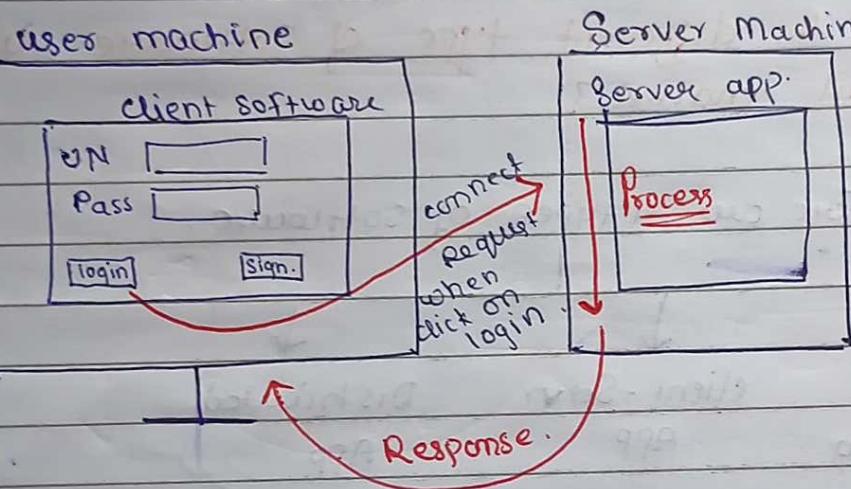
In one Standalone app there is only one Software.

[Q] Client - Server App.

2 - 3 %

(1)

(2)



Serves → which provides needed 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 installed in user machine
- The server software/app should be install 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.

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

I.8 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.

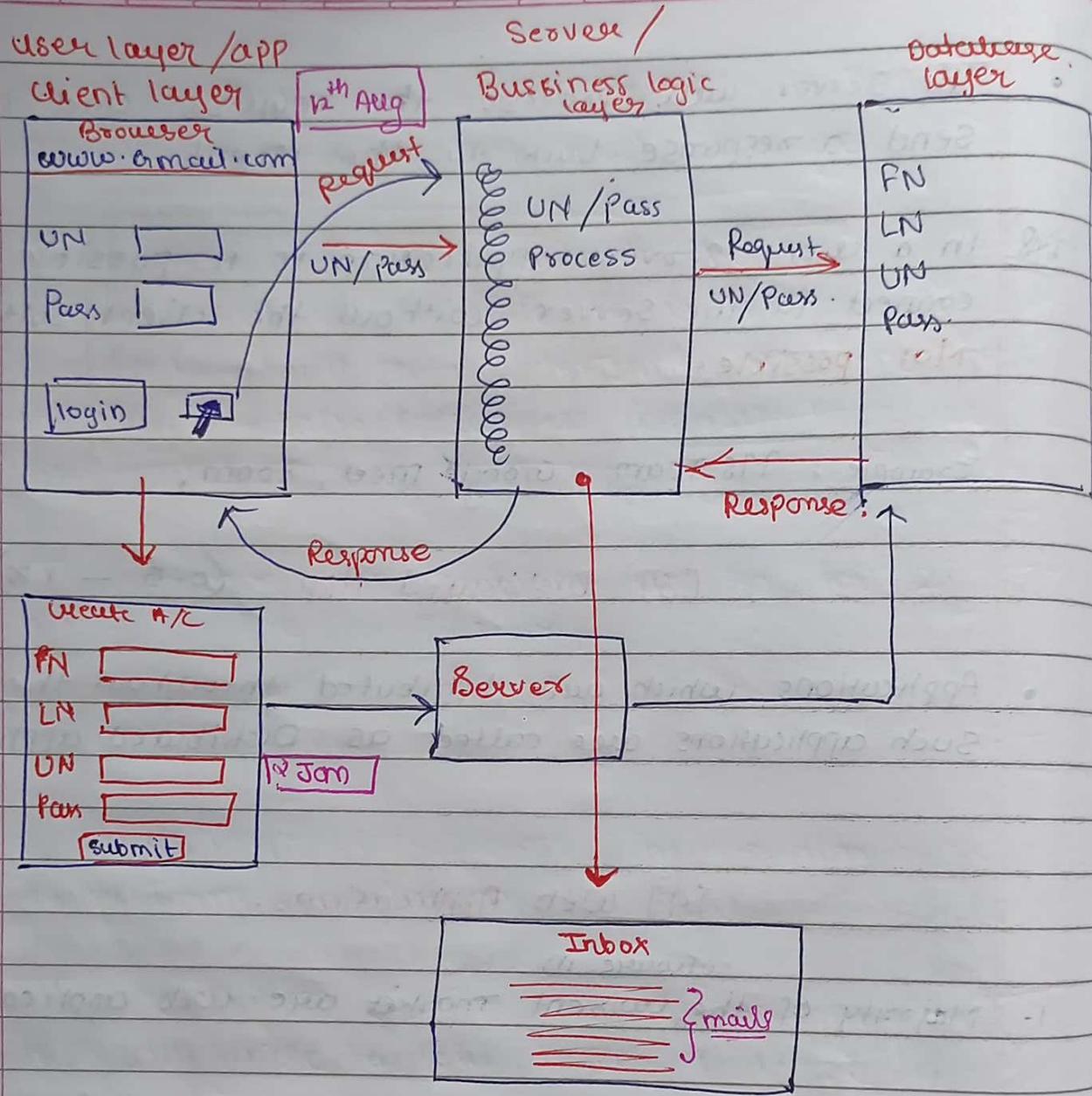
T4] Web Applications.

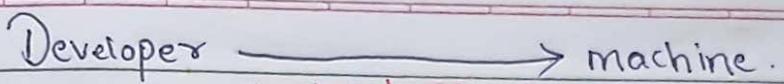
- Majority of the current market are web applications.
- Any Application open through the browser such applications are called as web base application.

Web APP \Rightarrow 3 stages.

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

- 1) User Layer / Application layer / client layer.
- 2) Server layer / business logic layer
- 3) Database

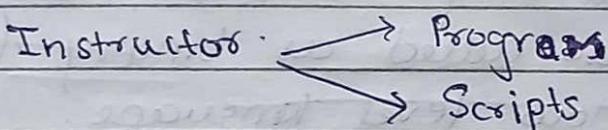




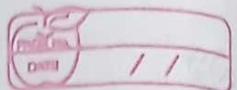
Instructors.

- ① Java
 - ② HTML
 - ③ C/C++

Technical languages



- 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 Software.
 - 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.
 - 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.



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 compiler 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 compiler is a **.java** file, but the output of the compiler is a ~~file~~ **.class** file.
- The **.class** file is given different names, low level language file, also called as binary file, also called as machine level

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

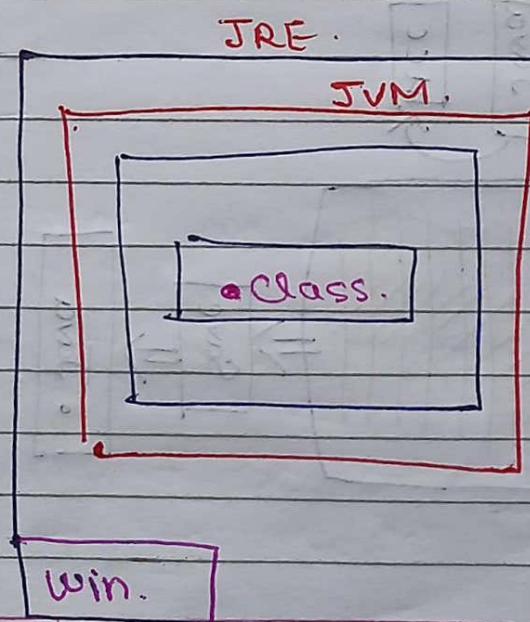
- Once the compilation is over, we then go for execution
- Execution is a process where, we make a .class has to interact with the machine, and to achieve this, java has provided a feature or a component called as JVM.

IMP: JVM is responsible to execute the .class file.

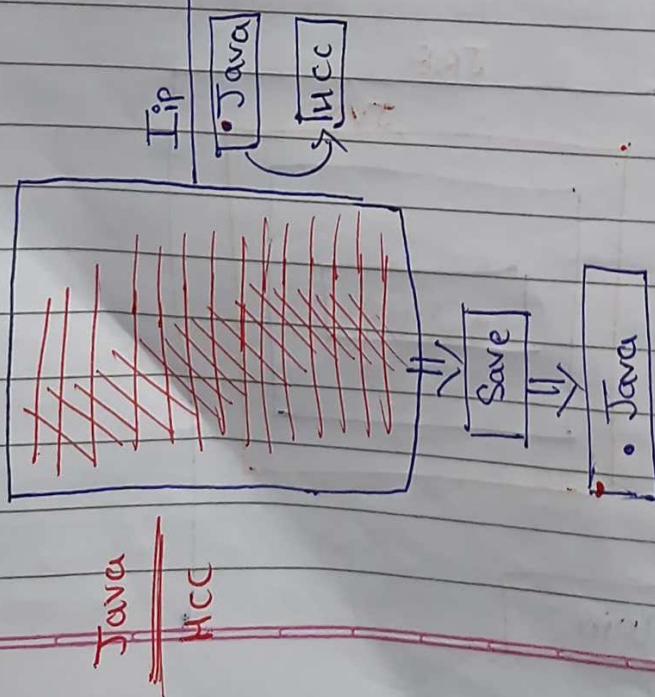
JVM → Java Virtual Machine.

Q: Who is Responsible to execute .class file?

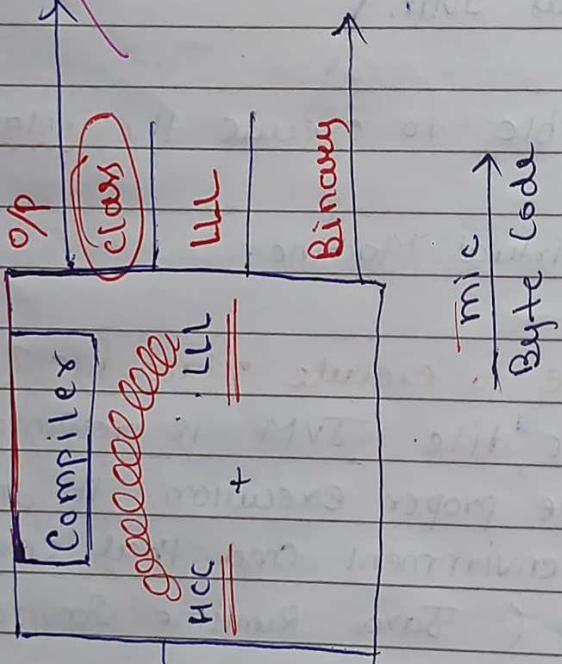
To execute .class file JVM is responsible, but for JVM to have proper execution, it requires an appropriate environment and that environment is given by JRE (Java Runtime Environment).



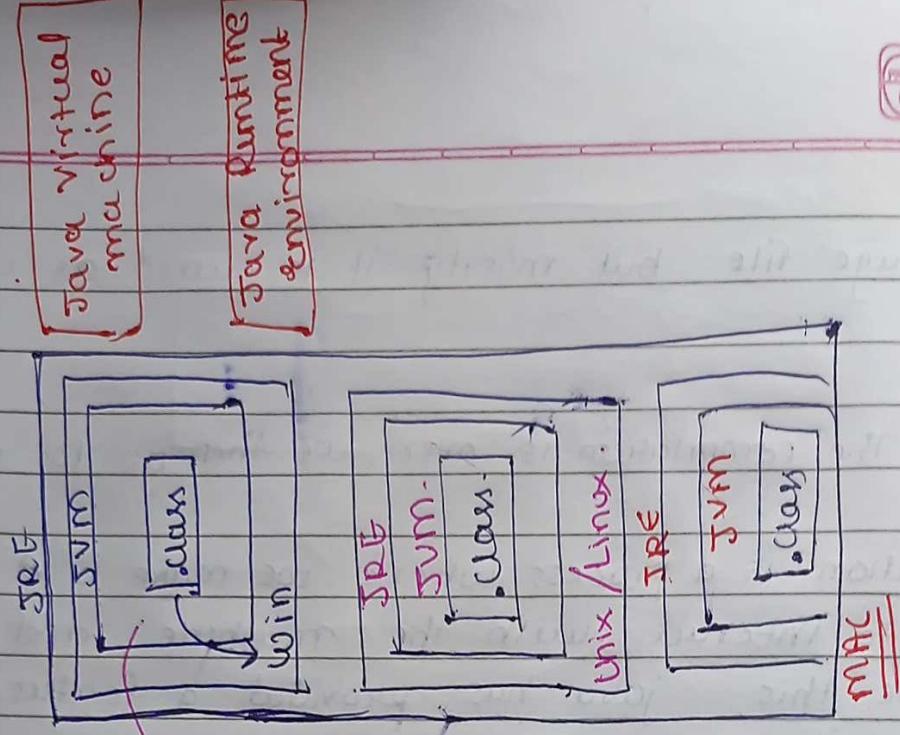
① Writing



② Compilation



③ Executives



- 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.out.println ("First Statement");

System.out.println ("Second Statement");

Sample2.java

- 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 [] args);

{

System.out.println ("Epsilon Demo2");

}

Epsilon.java

- 8) What is the output of following code?

class Sample3

{

Output: JavaC.Sample3.java

Java Sample3

error: more than one method declared in interface

main method not declared.

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

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

Q: 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

T5]

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 [ ] args)
```

{

```
        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 Demo 11
{
    Public static void main (String[],arg)
    {
        System.out.println ("Demo 11");
    }
}

class Sample 11
{
    Public static void main (String[],arg)
    {
        System.out.println ("Sample 11");
    }
}

class Tech 11
{
    Public static void main (String[],arg)
    {
        System.out.println ("Tech 11");
    }
}

```

Save : Demo 11.java

Compile: Javac Demo 11.java

Output: i) Demo 11.java
ii) Sample 11.java
iii) Tech 11.java.

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 Cyber
{
    public static void main (String [] args)
    {
        sys.out.println ("Cyber");
    }
}
```

Success.java X

Cyber1.java ✓

javac Cyber1.java

* 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;

{

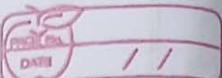
}

It will throw an error because

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

There should be only one public class.

19/Aug



* fundamentals of Java.



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.

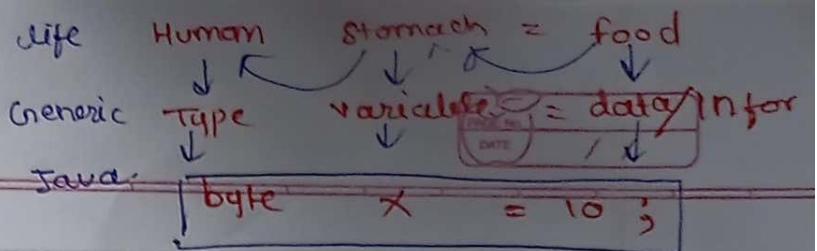
VARIABLES

- 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 meant 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.



- byte datatype :

ex) Class Demo1

```
public static void main(String[] args)
```

{

```
    byte x = 10;
```

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

}

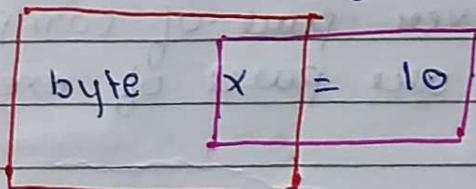
}

compile : javac Demo1.java

execution: java Demo1
(10)

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

- While printing a variable we don't use double quotes (" ") we directly use variable name.
- We use (" ") only when, when we want to display a message or a sentence or words.



declaration.

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

- It is possible to declare and initialize in same line

IQ

Is it mandatory to declare the variable before initializing?

→ Yes it is mandatory

ex-2

class Demo 2

{

psum(str args)

{ x=10; S.D.P(x); }

}

Output : cannot find symbol.

- **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.

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);
 }
}

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

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);
 }
}

- 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.

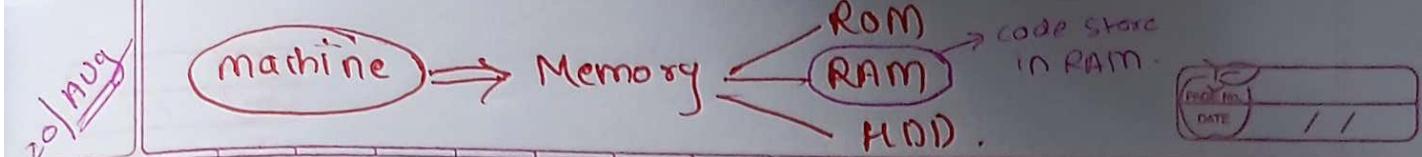
```

class Demo5
{
    public void main (String [] args)
    {
        byte x = 10;
        byte x = 20;
        S.O.P. (x);
    }
}

Output: It will throw an error.

```

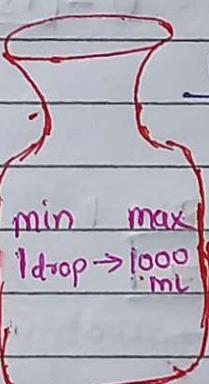
(Note: The code is handwritten. The student has written red annotations on the left side of the code. The annotations say: "cannot declare 2 time for some variable name." An arrow points from this text to the second declaration of 'byte x'.)



Important Note:

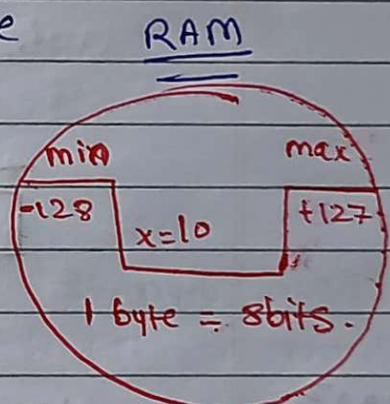
Whenever we execute a program or create a variable both program execution and variable creation happens within a RAM memory.

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

Life example: 

Bottle → Variable
water → Data.

min max
1 drop → 1000 mL
Capacity ⇒ 1L
1L → 1000 mL.
1 byte → 8 bits.



$$-2^n \text{ to } +2^n - 1 \quad 1 \text{ byte} = 8 \text{ bits.}$$

$$-2^7 \text{ to } +2^7 - 1$$

$$-128 \text{ to } +128 - 1$$

$$+127 \text{ to } -128$$

max mix

0 - 75

(n)

e.g.: 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 DataType :

- A short datatype contain 2 bytes of memory
2 bytes = 16 bits

$$2 \text{ bytes} = 8 \times 2 \text{ bits} = 16 \text{ bits}$$

1 - 16
0 - 15 n.

$$-2^n \text{ to } +2^n - 1$$

$$-2^{15} \text{ to } +2^{15} - 1$$

$$-32768 \text{ to } 32768 - 1$$

$$-32768 \text{ to } 32767$$

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;~~ ~~error~~

 SOP (s3)

~~Short s4 = -32769; ~~error~~~~

 SOP (s4)

}

Integer Datatypes

int \rightarrow 4 bytes = $8 \times 4 = 32$ bits.

1 - 32

0 - (31) n.

- 2^n to $2^n - 1$ - 2^{31} to $2^{31} - 1$ -2147483648 to +2147483647

eg: 3

Class Demo 10

{

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 = $8 \times 8 = 64$ bits.
@ 1 - 64
0 - 63 n.

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

class Demo1

{

 public static void main(String[] args)

{

 long var1 = 1000; \rightarrow within range
 S.O.P(var1);

}

String.

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

Class Demo 12

eg: 5)

```

public static void main (String [] args)
{
    long var1 = 1000;
    System.out.println ("The value of var1 is " + 1000);
}

```

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 01 A valid identifier can start with a letter, a currency symbol (\$), and connecting character (+).

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 Demo 14

{

```
P.S.V.M (String [] args) :
```

{

```
int 1 = 10;
```

}

3

Rule 3] We can use numbers just after a letter, or currency symbol (\$) and connecting character (-).

Class Demo 15

{

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 recommended standards, a valid identifier should be between the length 10-15, anything beyond that is not considered as good coding practice.

Class Demo 16

{

Public static void main (String [] args)

{

```
int xxx.....111 = 1000;
SOP (xxx.....111);
```

}

{

Rule 5]

Keywords or reserved words cannot be used as a valid identifier.

class Demo {

{

 Public static void main (String [] args)

{

 int byte = 10;

 SOP (byte);

}

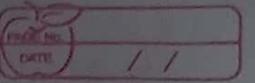
}

error :

Q.

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

→ There are 53 keywords



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;
        System.out.println("Value of Demo18 is " + Demo18);
        System.out.println("Value of main is " + 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;
        System.out.println("Value of cyber is " + cyber);
        System.out.println("Value of CYBER is " + 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 CyberSuccess = 10; ✓
- 2) int uCyber = 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; ✗

Long Datatype Concept

- Any Integral value, given to integral Datatype the compiler treats those 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;

 SOP (var5);

}

Output : 2147483648

2147483649

Float & Double Datatype & Variable Store.

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

float

Double.

- In the decimal values; if the decimal values are in the range of 5-7, we go for float. If the decimal values are in range of 12-14, we go for double.
- float can also be called single accuracy or single precision, whereas, double can be defined as double accuracy or double precision.
- float consumes 4 bytes of memory. 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.

1 7

10.1K 10.2K 10.340
 10.44

10.5127572
 10.540
 10.540
 10.540
 10.540

class Demo 21

{

Public static void main (String [] args)

float var1 = 10.5f ;

System.out.println (var1);

}

}

Output : 10.5

~~22) # Double~~

class Demo 22

{

Public static void main (String [] args)

{

double d1 = 180.123;

S.O.P (d1);

double d2 = 250.321D;

S.O.P (d2);

double d3 = 350.321d;

S.O.P (d3);

}

}

Output : 180.123

250.321

350.321

#

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 (i1);

}

#

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.

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);

}

}

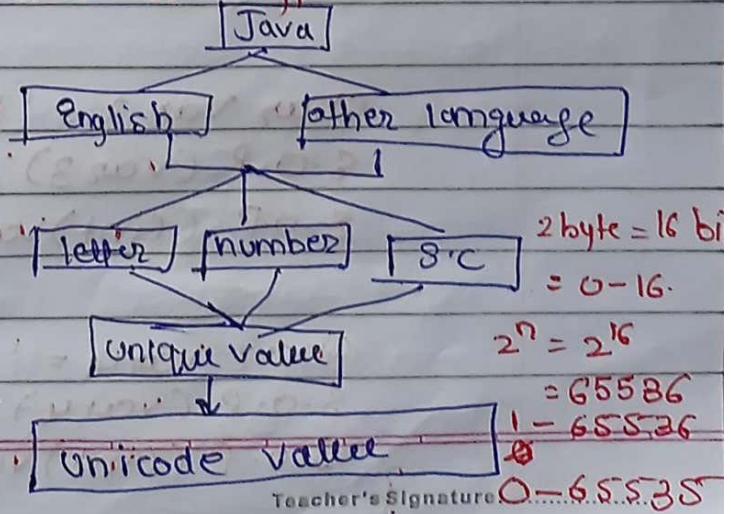
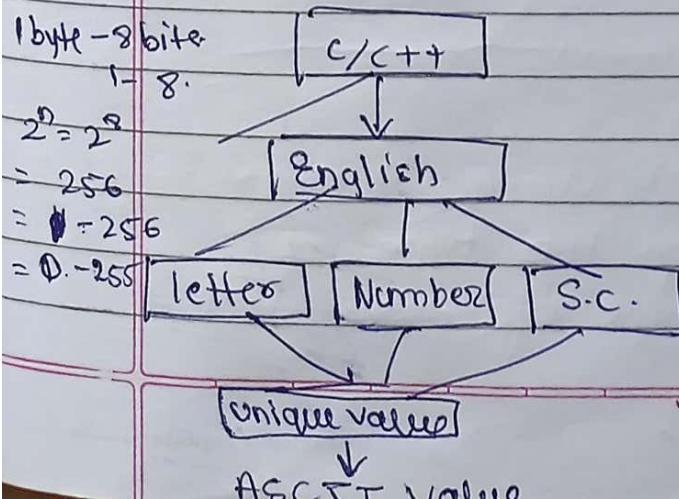
Output :- a

1

#

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

IQ A character datatype consumes one byte of memory in ~~the~~ C/C++ technology but why does java consume 2 bytes for a char datatype?



- 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 Demo 25

```
{ 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';
        System.out.println(x);
        char y = '9';
        System.out.println(y);
    }
}
```

Output : 97
 'b'.

#

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;

System.out.println(status1);

boolean status2 = false;

System.out.println(status2);

}

}

Output : true
false.

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

class Demo28

{

 public static void main (String [] args)
 {

 boolean status1 = 0;

 S.O.P (status1);

}

}

Output : error

23/Pg

OPERATORS.

Operators

- Postfix
- Unary
- Multiplicative
- Additive
- Shift
- Relational
- Equality
- Bitwise AND
- Bitwise Exclusive OR
- Bitwise Inclusive OR
- Logical AND
- Logical OR
- Ternary
- Assignment

Precedence

exp++ , exp--

++exp , --exp , +exp , -exp ~

* % / %

<< >> >>>

< > <= >= instanceof

== !=

&

^

!

ff

!!

? :

= , += , -= , *= , /= , %= , f= , ^= , !=

<<= >>= >>>=

class Operators {

 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);

 // ~~double~~ 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.

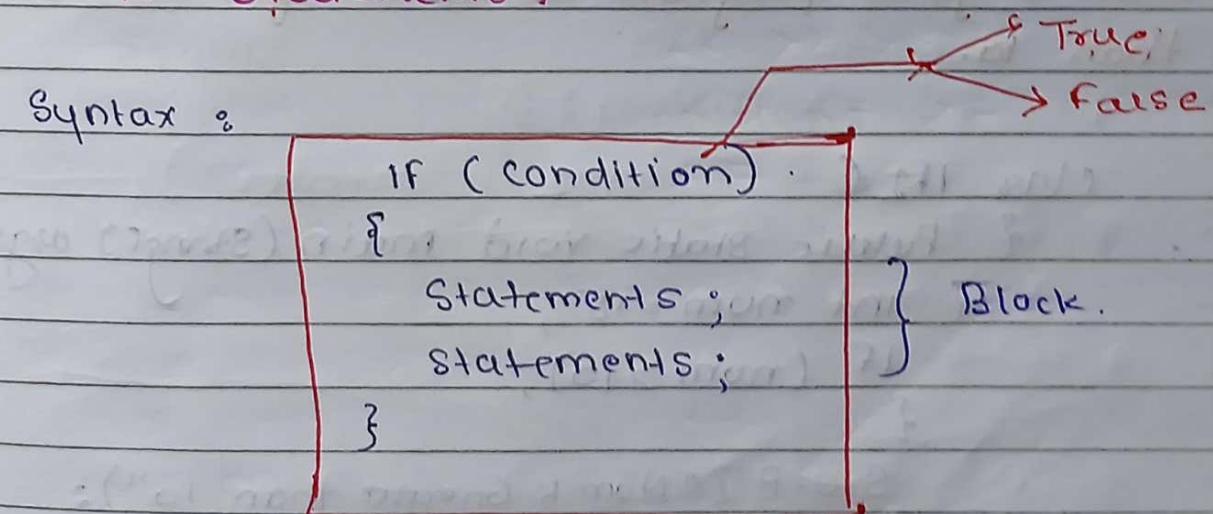
Control flow statements can be categorize mainly in 3 type.

- ① Conditional Statements
- ② Looping Statements
- ③ Transfer Statements.

I] 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 arithmetic operator equality and relational operators.
- It include $>$ $<$ \geq \leq $=$ \neq

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");  
        }  
    }  
}
```

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)
```

```
{
```

```
System.out.println ("Eligible for Voting");
```

```
}
```

```
}
```

2) If - else Statements :

Statements :

↓

IF - else

Syntax :

```
if (condition)
```

```
{ Statements ; }
```

```
Statements ;
```

```
}
```

else

```
{ Statements ; }
```

```
Statements ;
```

```
}
```



- 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.

~~TQ~~

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

(any two out of 3) Q. No. 3

```
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");
```

```
},
```

```
}
```

```
}
```

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

class IfElse2 .

{

 Public Static void main (String [] 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");

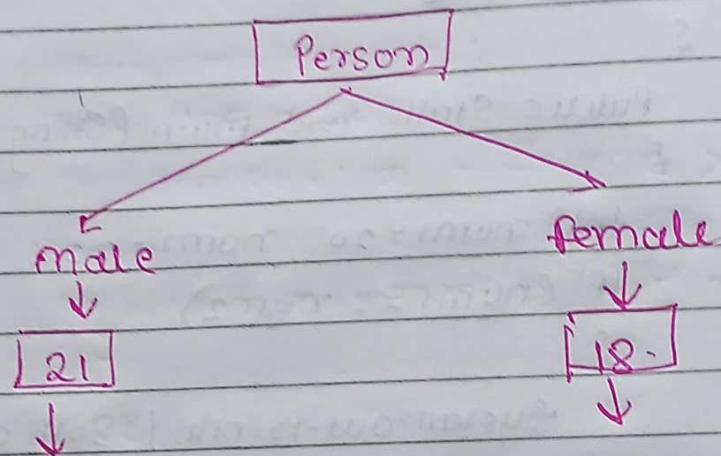
}

}

Output :

Both age are same.

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



char gender = male char gender = female
 int age = 21 int age = 18

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

Condition①

F(0)

T(1)

F(0)

T(1)

Condition②

~~multiplication~~ F(0)

F(0)

T(1)

T(1)

Total condition

F(0)

F(0)

F(0)

T(1)

class IfElse3

{
Public static void main (String [] args)

{
int age = 18;
char gender = 'F';
{ if (age >= 18) && (gender == 'F')
S.O.P ("Eligible for marriage");
}

else

S.O.P ("Not Eligible for marriage");
}
}

}

Output: Eligible for marriage.

24/Aug



class IFEElse :

{

 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") ;

}

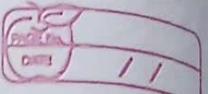
}

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

```
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");
    }
}
```



#

If - else - If .

Syntax ,

IF (condition)

{

// Statement

}

else if (condition)

{

// Statement

}

else if (condition)

{

Statement

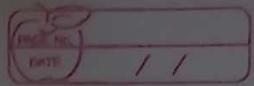
}

else

{

Statement

}



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");

}

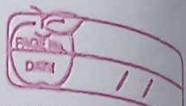
 else if (num1 == num2);

{

 S.O.P ("invalid");

}

}



Ques: write a java program to simulating traffic signals

Class ifelseif

{

Public static void main (String [] args)

{

char sign = 'x'

If (sign == 's')

{

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)
{
    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 don't purchase it the product is mobile & quantity is above 35 then purchase if else Do not Purchase. Purchase.

class Nested

{

 public static void main (String [] args)

{

 char prototype = 'M' ;

 int custQty = 25 ;

 if (prototype == '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 Quantity is too");

}

Switch Case

Syntax:

switch (variable / argument / Parameters)
{

case case value : statements ;

[break]

case case value : statements ;

[break]

default : statements ;

[break]

}

}

- In the switch statement there is no condition match.
- In the switch statements the way it works, the arguments 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.



class Switch1
{

 Public static void main (String [] args)
 {

 System.out.println ("start");
 int a = 2;

 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;

 }

 S.O.P ("stop");

}

}

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

```
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;

}

Q Are cases mandatory within the switch statement?

Cases are not mandatory.

```
int a=1;
switch(a)
{
```

Default :

S.O.P ("Inside Default");

break;

}

Q.

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 either within the Default without that it will throw an error.

```

int a=3;
switch(a);
    {
        s.o.p ("Inside the statement");
    }
}

```

Q. Which are the valid and 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';
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;
}

```

- As break statements are not mandatory without break also, the program will be compiled and executed.
- Without break statement, whichever 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: S.O.P ("Two");  
    case 3: S.O.P ("Three");  
    default : S.O.P ("Invalid signed");  
}
```

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.O.P ("Inside case two");  
    break;  
}
```

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

```
String sStr = "Cyber";  
switch (sStr)  
{  
    case "Cyber" : S.O.P ("Inside Cyber");  
    break;  
    case "Success" : S.O.P ("Inside Success");  
    break;  
    default : S.O.P ("Inside default");  
    break  
}
```

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

```
String str = "Cyber";  
switch (str)  
{
```

```
    case 10: System.out.println("Inside Cyber");  
              break;
```

```
    case 20: System.out.println("Inside Success");  
              break;
```

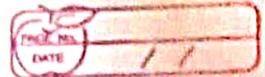
```
    default: System.out.println("Inside Default");  
              break;
```

{}

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)
        {
            System.out.println ("you are eligible to apply for
                a driving license");
        }
        else
        {
            System.out.println ("you are not eligible to
                apply for driving license");
        }
    }
}
```

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

Q. Check if a student has passed or failed.

class Successfail

{

 public static void main (String [] args)

{

 int marks = 25;

 if (marks >= 40)

 {

 System.out.println ("You have passed the exam")

 }

 else

 System.out.println ("You have failed the exam")

 }

}

}

Output: You have failed the exam.

Ques: 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)
```

```
{
```

```
            System.out.println ("You are eligible for senior citizen discount")
```

```
}
```

```
        else
```

```
{
```

```
            System.out.println ("You are not eligible for senior citizen discount")
```

```
}
```

Output : You are eligible for senior citizen discount.

Ques: If - else - if Example. If marks is given
Grade classification.

```
class Gradecheck {
    public static void main (String [] args) {
        int marks = 75;
        if (marks >= 90) {
            System.out.println ("Grade A");
        } else if (marks >= 75) {
            System.out.println ("Grade B");
        } else if (marks >= 60) {
            System.out.println ("Grade C");
        } else if (marks >= 40) {
            System.out.println ("Grade D");
        } else {
            System.out.println ("Fail");
        }
    }
}
```

Output : ~~Grade~~ Grade B

Ques If - else - if example : write a program to calculate Electricity Bill static.

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~~

0-100 units →

₹ 5/unit

else if (units <= 200) units between 101 + 200

101-200 u → ₹7/u

{

bill = (100 * 5) + (units - 100) * 7;

201+units → ₹10/u }

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);

,

}

for 250 units the bill is .

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

Total Electricity Bill : ₹ 1700.0

Teacher's Signature:.....

Ques: Nested If example → ATM withdrawal check

class ATM withdrawal

8

```
public static void main (String [] args)  
{
```

int balance = 5000;

int withdraw = 2000;

boolean atmworking = true; is atm working or not

IF (atm working)

۲

1F (with draw $\chi = \text{balance}$)

۳

S.O.P ("Transaction successful!")

Remaining Balance: " + (balance -

3

else {

3

18

else

2

5

2

1

二三

1

1

1

Looping statements

- A loop can be defined as ~~repeating a statement~~ or an ~~infinite~~ number of times. This is called as loop or an iteration.
- There are three main looping statements or iteration statements, repeated by Java:
 - **for loop**
 - **while loop**
 - **do-while loop**

For loop

- for loop can be categorized mainly in three types
 - (i) for?
 - (ii) Nested for
 - (iii) for each (Foreach / collection)

Syntax :-

for (Initialization ; Condition ; Increment)

 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 **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 categorized mainly in three types
 - (i) For
 - (ii) Nested for
 - (iii) for each. (Arrays / collections)

Syntax → **for** *i*

for (Declaration, Initialization) ; Condition ; Increment, Decrement)

{

Statements;
Statements;

}

Operators can be categorized into

→ Increment

Pre Inc

Post Inc

→ Decrement

Pre Dec

Post Dec

Increment & Decrement Operators.

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

Increment Operator / Decrement Operator.

Inc & Dec. operator can further be categorized mainly into 2 types.

- (i) Pre Inc. : (ii) Post Inc.
- (i) Pre Dec. : (ii) Post Dec.

Pre & Post Increment operators.

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

Pre \Rightarrow $++i$

Post \Rightarrow $i++$



class IncPostDec {

 public static void main (String [] args) {

 int i = 1;

 System.out.println(i); // 1

 // Preincrement → // First increment the i value
 // by 1 & then Print the value of i.

 System.out.println(++i); // 1 Increment, 2 → Print 2

 System.out.println(i); // 2

 // Postincrement → // First print the value of i
 // & then increment the i value

 System.out.println(i++); // 2 point → 2-Increment 3

 System.out.println(i); // 3

}

Output: 1
2
2
2
3



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

class For 1

```
public static void main (String [] args)  
{  
    Declaration    Initialization   Condition   Increment  
    for (int i=1; i<=3; i++)  
}
```

```
    System.out.println("The value of i is " + i);  
}  
}
```

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

- (1) Dec / Initi $\rightarrow i=1$
- (2) Condition $\leftarrow F \Rightarrow$ Stop
- (3) Statement
- (4) Inc / Dec $\rightarrow i \Rightarrow 2$
- (5) Condition $\leftarrow F =$ Stop
 \downarrow
Statement:
- (6) Inc / Dec $\rightarrow i \Rightarrow 3$
- (7) Condition $\leftarrow F$
 \downarrow
 $3 >= 3$
- (8) Inc / Dec
- (9) Statement
- (10) Inc / Dec $\rightarrow i \Rightarrow 4$
- (11) Condition $\leftarrow F$
 \downarrow
 $4 >= 3$
- (12) If $4 >= 3$ \Rightarrow 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 Condition will check.
- Based on total number of Iteration + 1
- Q. Within the for loop how many times will Inc & Decrement will happen?
- Based on total numbers of Iterations.

Q. Write a java program to print values from 1 to 10 in Descending order.

```
for (int i=3 ; i >=1 ; i--)
```

```
{ S.O.P ("The Descending order is "+i)}
```

Q. write a java program to check if number 3 exists from 1 to 5. If yes print?

```
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 mentioned within for statement.

```
int i=1; → Declaration &
For (i<=5; i++) Initialization are
{           outside the loop
    S.O.P(i);
}
```

Output : 1

2

3

4

5

- We can use minimum one or multiple stages separated by comma (,) in the declaration as well as initialization.

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

• 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.

- 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[])
    {
        for (; ; )
        {
            S.O.P ("Hello");
        }
    }
}
```

Above code will enter into infinite loop

- None of the stages are mandatory, if we do not set any condition, then compiler will itself enter into infinite loop by default.



For Nested loop

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

i=0, j=0	for (int i=0 ; i<=2 ; i++)	*			
0	for (int j=0 ; j<=i ; j++)	*	*		
1	*	*	*		
2	*	*	*		

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

~~eg~~

```
for (int i = 1; i <= 3; i++)
```

{

1		
1	2	
1	2	3

```
    for (int j = 1; j <= i; j++)
```

{

```
        S.O.P ("j");
```

}

```
    S.O.Println();
```

}

i j j

①

```
int temp = 1;
```

```
for (int i = 1; i <= 3; i++)
```

{

```
    for (int j = 1; j <= i; j++)
```

{

1	1	
2	3	
4	5	6

```
        S.O.P (temp);
```

```
        temp++;
```

}

```
    S.O.Println();
```

}

3

3

* *
* *
* *

1 2 3
1 2
1



* While loop: It is a type of loop which continues to execute the code until the condition becomes false.

Syntax: while (condition)
{
 statements;
 statements;
}

while (condition)

{

 statements;
 statements;

}

1 2 3

4 5

- 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)  
{  
    System.out.println(i);  
    i++;  
}
```

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

```
int i = 10;
```

```
while (i >= 1) {
```

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

```
    i--;
```

```
    if (i == 5) {
```

```
        System.out.println("The value of i is " + i);
```

```
        break;
```

```
}
```

```
i--;
```

```
}
```

```
} // This will print all numbers from 10 to 1 except 5
```

Final output

10
9
8
7
6
5

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

```
public static void main(String[] args) {
```

```
int a = 5;
```

```
System.out.println("Factorial of " + a + " is " + fact(a));
```

```
int fact(int n) {
```

```
if (n == 1) return 1;
```

```
return n * fact(n - 1);
```

```
}
```

```
System.out.println("Factorial of " + a + " is " + fact(a));
```

```
int fact(int n) {
```

```
if (n == 1) return 1;
```

```
return n * fact(n - 1);
```

```
}
```

```
System.out.println("Factorial of " + a + " is " + fact(a));
```

```
int fact(int n) {
```

```
if (n == 1) return 1;
```

```
return n * fact(n - 1);
```

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 has given alternative mechanism.

(ii) `Return` statements:

`Return` statement will be teaching ~~when~~ during method, and constructor

(iii) `Try-catch`:

being taught during exception handling

(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)
{
    System.out.println(i);
    break;
}
```

It will throw an error.

(v) continue statement.

- The continue statements act like a skipping statements.
- The ~~skip~~ 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 loop)

```
for (int i=1; i<=5; i++).
{
    if (i == 3)
        System.out.println(i);
    continue; // skip.
    System.out.println(i);
}
```

3/09/25

65

Data Types

Primitive

(8)

int float
byte double
long char
short boolean

Non-Primitive

(9) ~~String~~

~~String~~

(8) ~~Object~~

(1) Local

(2) Static

(3) Non-Static

(1) Local Variable:

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

```
class Localvar {  
    public static void main (String [] args)  
    {  
        // Local var
```

int a=100;

// directly

S.O.P (a);

Output

?

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

A. Yes, it is mandatory.

Class LocalVar2.

{

P.S.V.m(String[] args)

{

int a;

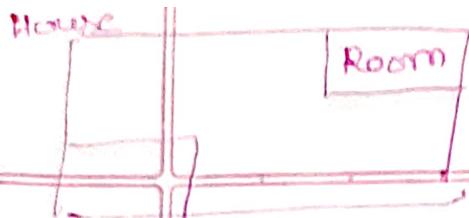
S.O.P(a);

}

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).



House → Room
 classname → Variable
 classname • Variable.



Q

How many ways are there to access static members? How to access static member with or to single class.

There are totally 3 ways to access static members:

- Directly
- classname
- Object creation

class StaticVar

{

static int a=100;

P.S.V.I (String [] args)

{

S.O.P ("Start");

// Directly.

S.O.P (a);

// classname.

S.O.P (StaticVar.a);

will teach during the // object creation.

concept of non-static & from primitive.

S.O.P ("stop");

}

(a) 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

(b) 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);

dot
operator

result can be accessed
by 2 ways
directly and
classname.

result will
be float.

}

}

Output : 300.5

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

→ There are only 2 approaches - they are

- (i) Classname + . + static member
- (ii) Object creation + static member

```
class Demo {
```

```
    static int x = 100;
```

```
}
```

```
class Static Var4 {
```

```
{
```

```
    public static void main (String [] args)
```

```
{ // Directly
```

```
    System.out.println (x); → Invalid.
```

```
    // classname
```

```
    System.out.println (Demo.x); → Valid.
```

```
    // object → Valid.
```

```
}
```

```
}
```

static var's

```
class Cyber
{
    static float x = 100.5f;
}
```

①

class var.

```
class Success
{
    static int y = 200;
}
```

②

class var.

class StaticVars

{

static float result;

P-S-V. m(String[] args)

{

int z = 300;

result = Cyber.x + Success.y + z

smother
doessmother
var

directly

S.O.P(result);

}

Output: 200.5

}

staticVar's

Ex)

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.

class staticVar2

{

 static int x;

 static float y;

 public static void main (String args)

{

 S.O.P (x);

 S.O.P (y);

}

Output : 0

0.0

~~StaticVar2~~

* We cannot make local var as static var.

class staticVar2

{

 P.S.V.M (String [] args)

 { static int x=10;

 S.O.P (x);

}

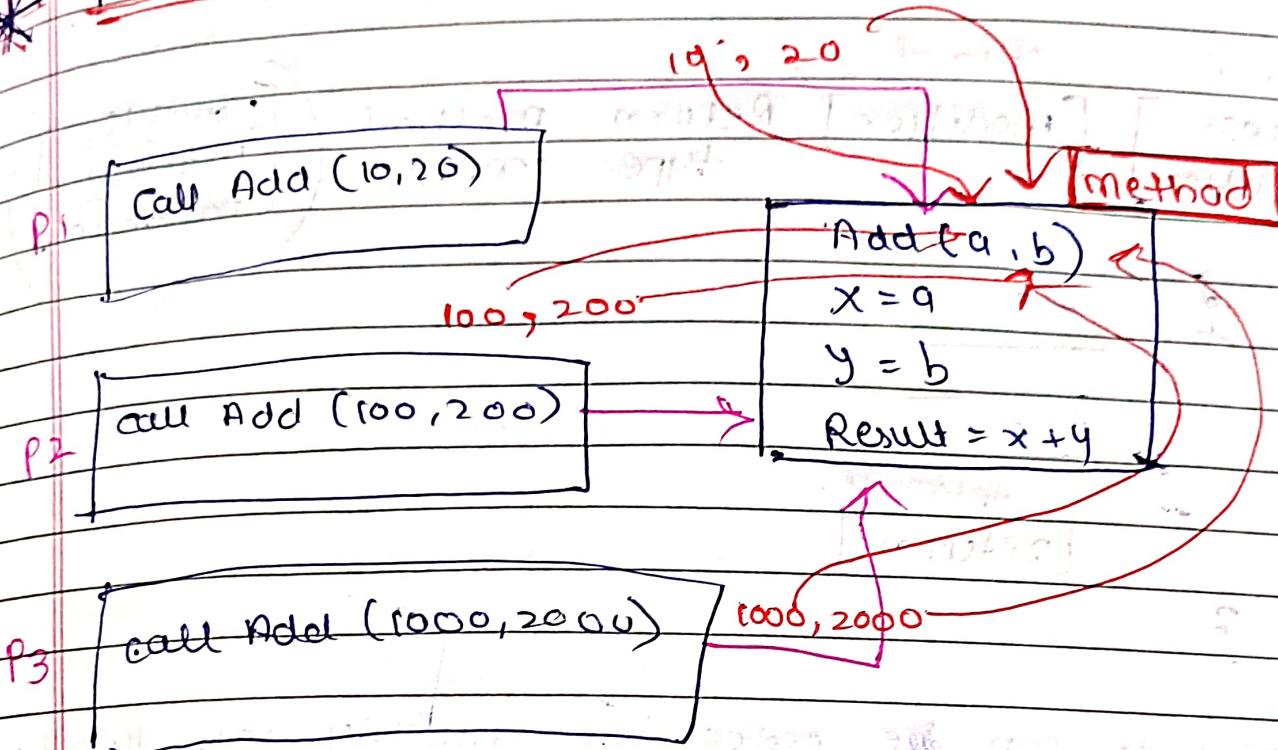
}

Output error.

* Static variable cannot use inside main method.

- To avoid the need for methods:
- ① Duplicate code
 - ② length of code will be more
 - ③ more memory
 - ④ more time for execution.
 - ⑤ Application will be slow

METHODS.

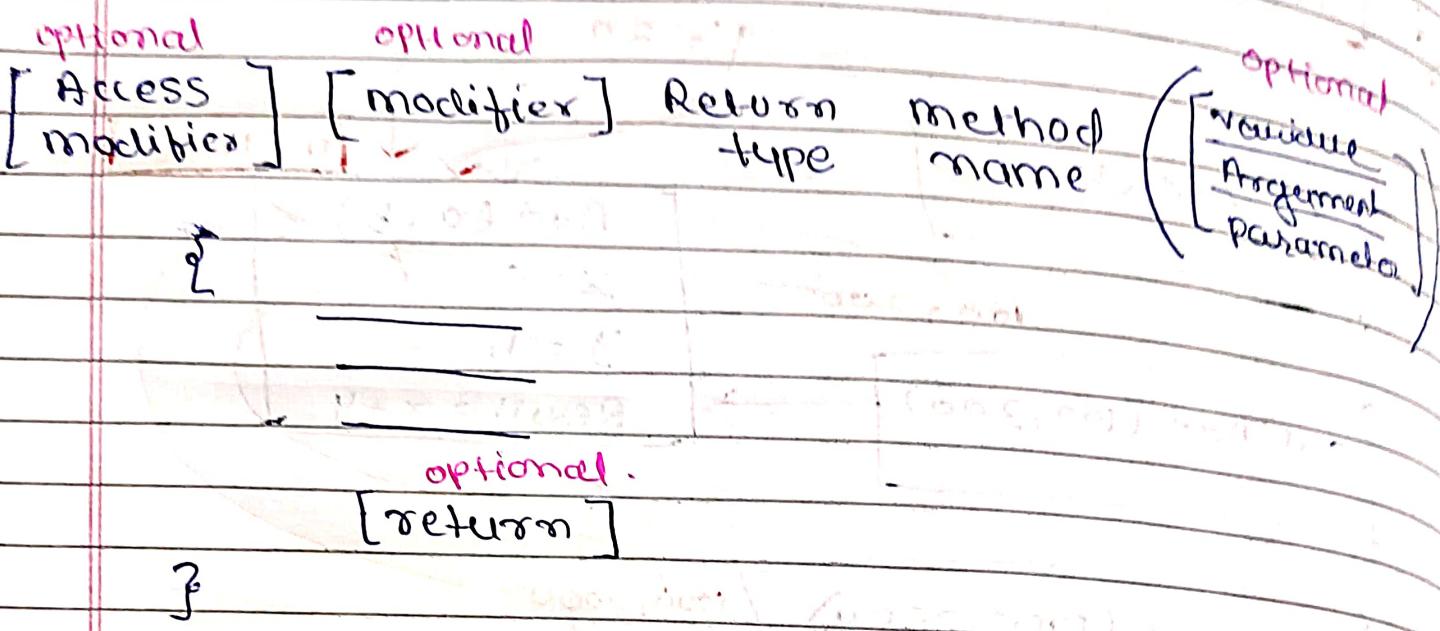


- A method can be defined as ~~state~~ of statement which are meant to perform certain operation.
- Benefits of methods:
 - (i) Due to methods we can reuse the code.
 - (ii) Because of Reusability of code we can avoid duplicate code.
 - (iii) The code length will be less
 - (iv) Will consume less memory.
 - (v) Will take less time for execution.
 - (vi) It will help in increase the speed of the application.

4/09/25

{ } → optional.

* Syntax of Method.



- Methods can be categorized mainly into two types:
 - (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 single class, there are 3 approaches:
 - i) Directly
 - ii) Classname
 - iii) Object creation.

```

class Example1 {
    // called from main function
    public static void main(String[] args) {
        System.out.println("Start");
        // Directly
        System.out.println();
        // classname -
        Example1.main();
        // Object
        System.out.println("stop");
    }
    // called from main function
    static void m1() {
        System.out.println("Inside the m1");
        return;
    }
}
    
```

Output : Start
 Inside the m1
 Inside the m1;
 Stop.

- 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; X
```

```
}
```

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 control 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.

{

public static void main (String [] args)

{

S.O.P ("Start");

test1();

S.O.P ("Stop");

//return;

}

static void test1 ()

{

S.O.P ("Inside the test1 method");

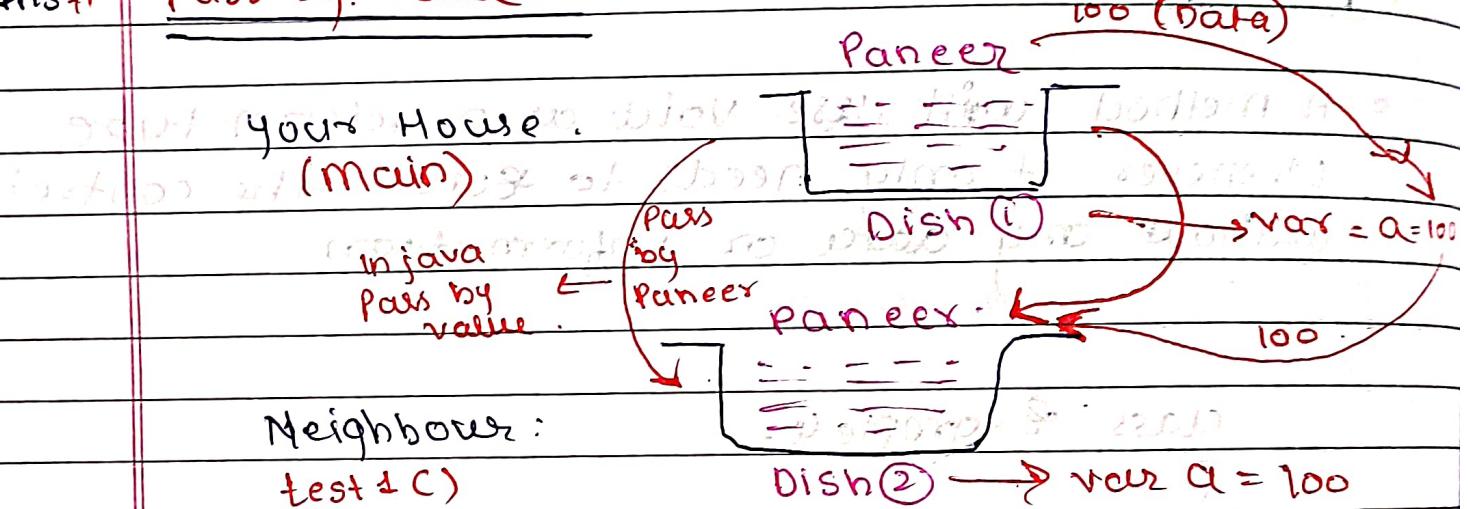
//return;

}

* Imp note:

Whenever we declared local variable within a method, constructor, or block // that variable can only and only be accessed 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.

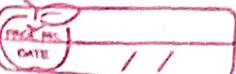


Q. When will you in real time go for pass by value.

→ To access local var from one area to another.

- Another definition of local variable: (final)
- Variables declared within a method or part of the method, constructor, blocks such variables are called as local variables.

Default Pure



Q. What are class Examples

{

p.s.v.m (String(7 args))

① int a=100;

② S.O.P ("Inside main" + a);

(test1(a)); } main

7 } static void test1(int a)

③ { int a; a++; }

S.O.P ("Inside test1 method " + a);

3 return (by default);

Part of a local variable.

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 area of rectangle?

(Lesson 7) Date _____

(S) class Example6 { main() { } }

and C:\Users\Aman\Downloads\9.02.2019

P.S.V.M (String[] args)

public static void

{ int length = 100, breath = 200;

test1 (length, breath);

} (C:\Users\Aman\Downloads\9.02.2019)

3 (C:\Users\Aman\Downloads\9.02.2019)

Static void test1 (int length, int breath)

{

S.O.P ("Area of rec" + (length * breath))

}

(C:\Users\Aman\Downloads\9.02.2019)

3 (C:\Users\Aman\Downloads\9.02.2019)

Q. Area of Triangle:

(Lesson 7) Date _____

(S) class Example7 { main() { } }

and C:\Users\Aman\Downloads\9.02.2019

int breath = 10, height = 20; test1 (breath, height);

} (C:\Users\Aman\Downloads\9.02.2019)

Static void test1 (int a, int b)

{

S.O.P ("Area of triangle" + (0.5 * a * b))

} (C:\Users\Aman\Downloads\9.02.2019)

3 (C:\Users\Aman\Downloads\9.02.2019)

? (C:\Users\Aman\Downloads\9.02.2019)

Q. In Java, methods have default parameters like in Python.

Q. Area of circle in Java without void if.

{ int radius = 10; float pi = 3.14f; }

int r = 10;

final float pi = 3.14f;

test1(r, pi);

int r;

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 * side));

}

}

5/09/25

WEDNESDAY, 5/09/25
SYNOPSIS OF THE DAY



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

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

class Example { }

(1) P.S.V.M (String[] args)

int num1 = 10, num2 = 20;

(2) int result = add (num1, num2); ← (6)
S.O.P ("Result is " + result); ← (7)

(3) static int add (int num1, int num2)

10 20

int result = num1 + num2;

return result;

(4) ← (8) ← (9) ← (10)

Output: Result is 30.

Return type.

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

```
if int status = multi();  
    S.O.P(status);  
{  
    static int multi()  
    {  
        int result = 10 * 20;  
        float result2 = 20.5 - 10.5;  
        return result1, result2;  
    }  
}
```

How to access method outside the main method.

```
class Sample  
{  
    static void m1()  
    {  
        S.O.P("Inside the M1()");  
    }  
}
```

Class Example

```
if P.S.V.M (String [] args)
```

```
{  
    S.O.P("Start ");
```

// class name

```
Sample.m1();
```

// object

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

class Cyber

~~Spotted~~, ~~spotted~~

Static Methods (int a)

~~float x=10.5 f;~~

6) float result1 = x + @.example 10.0
return result1; 10.5 + 100.

110-5

Class Success

get float

Static method 2 (float b).

~~int y=20;~~

~~float result2 = 4+b;~~

~~return result~~
~~220.5~~

class Example10

{ P-S-V-M (String[] args)

of S.O.P ("start")

int a=100;

$$\text{float } b = 200 \cdot 5f;$$

1) casting m①

"call me (2)

float result = Cybers-method 1(a);

~~float result > = Success; method2 (b);~~

~~float finalResult = result1 + result2;~~

S.O.P ("final result" + final result)

§-o · P ("Stop");

3