05/09

Syntax :

```
class program
{
public static void main (string args[])
{
system.out.print ("Hello World");
system.out.println ("Java programming");
}
}
```

O/P :- Hello world
Java programming

```
class program
{
public static void main (string args[])
{
system.out.print ("Hello world");
system.out.println ("Java Programming");
system.out.print (" MCA");
system.out.println ("BCA");
}
}
```

O/P :- HelloworldJavaProgrammingMCA BCA

* A program can contains more classes.
But one class can only contain the main method and that class is called as main class.

Print() / Print(s)
Println() / Println(s)

} It is a funct? or methods bcz it contains obj inside brackets.

* Java is a pure → 99% object oriented programming language. Each and everything is defined inside a class

advantage over C++

system
↓
out → Print
println

The dot operator specifies that print method is belongs to out object of system class.

Primitives - Built in datatypes

The class that contain the main method, then the file name will be the Classname.JAVA

06/09

1) Class name & File name is same?
2) can we write the program without setting the path?
3) Class Not Found Exception: can't able to Load main class File?

2. yes, we can execute the java program without setting the path in CMD.

> set path = "jdk path
> javac Filename.java
> java Filename

- jdk is present in another place & .java file is another place, the OS will set path = " " between them.

Date
Page

1.

Filename.java
↓ compile
Filename.class → Byte code (It makes the java to platform independent)

- It is not mandatory that the class name & file name must be same. When a java file is compile then it generate the byte code according to its class name (not according to file name).

Java First → is there a byte code file which name is first

Filename :-
```
class First {

}
```

- If the class is public then class name & file name will be same.
- If the class is not public then class name and file name can be different. The byte code is generate according to class name & we can run the program by java classname

Class :- A class is the collection of similar types of object.

- Class is the blueprint of object because it define the state & behaviour of object

or

Class defined the property & behaviour of obj;

Eg:
```
class pen {
    colour, type
    write ();
    draw ();
}
```

Java

Date – 6/09/2024

1. Can we write the program without setting the path?
2. Class name & file name same?
3. Class not found error?

java.exe ⟶ This one is JVM

java.c ⟶ Java compiler

java.class ⟶ Byte code file

(4) OOPs (Object oriented programming concepts) :-

→ Class and Object.

→ Abstraction and Encapsulation.

→ Inheritance and polymorphism.

→ Class :-

- Class define the state and behaviour of an object.
- Class provide you ~~for~~ the blueprint for object.

Eg: class pen {

```
colour, type...
write();
draw();
point();
}
```

→ Object :-

- Object is a real world entity that means it has some state & behaviour.
- object is instance of class.

Eg:

Pen

State :- colour, length, width, type

behaviour :- write, draw

② Abstraction & Encapsulation :-

→ Encapsulation :-

- It means wrapping of data + method for data hiding and the variable is access inside that class.

```
class {
    private variables;
    public methods();
}
```

→ **Abstraction :-**

It means hiding the implementation display functionality.
↓
(Background details)

How the fun^{n} use data & variable of the class?

Data of main class can be access by function call but not shown to user.

How to achieve data Abstraction,

2 types,

1. Abstract class
2. Interface → By using this we can acheive 100% Data Abstraction.

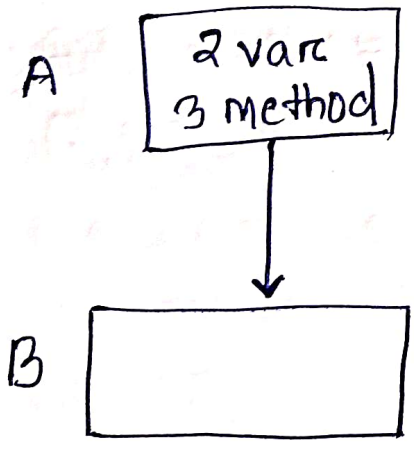
Inheritance & polymorphism :-

↓

- code reusability
- object is going to derive the property.

3 types of inheritance

→ single
→ multilable
→ Hybrid



Supper class or parent

Subclass or child

{ child can access all the property (var, method) of parent class.

- Share same memory.

→ Polymorphism:-

Same method provide different implementation, based on behaviour of object

call during compile time (check syntax)

call during run time (check during execution)

Polymorphism are two types:-

→ Compile time polymorphism (method overloading)

→ Run time polymorphism (method overriding)

Method overloading,

```
Sum();
Sum(a,b);
Sum(a,b,c);
```

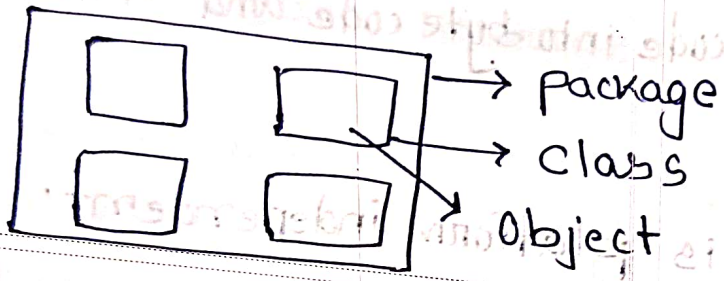
Method overriding,

```
Sum()
 {
  print("Hi");
 }
Sum()
  print("Hello");
 }
```

(4) Dynamic method dispatch.

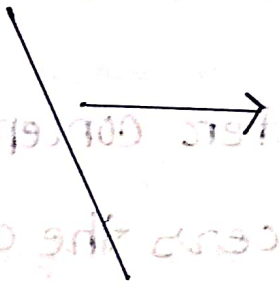
(a) Java Features :

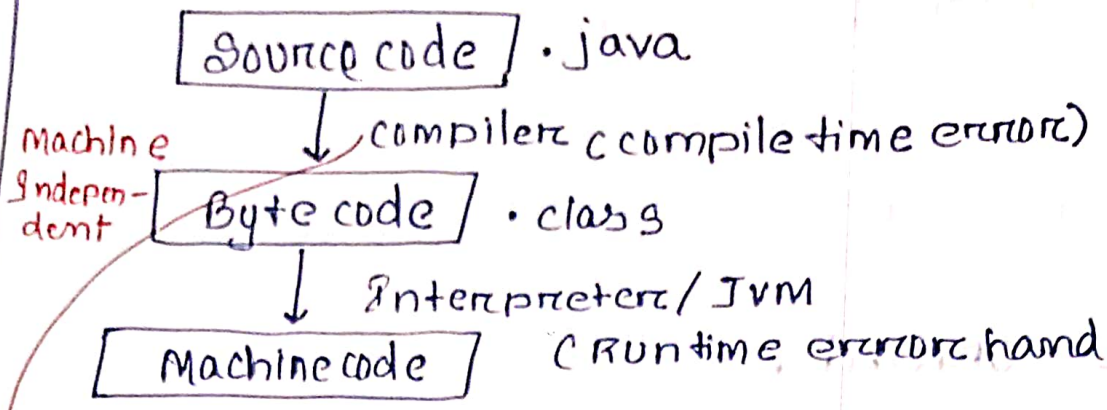
1. Object-orient :-
(Each & every thing in java can be defined inside the class which is the blueprint of Object)



2. Platform independent & portability :-

- Java supportes both compiler and Interpreter
  (byte code) (It translate code line by line).
  (convert high level to low leavel lang.).
- Machine code is machine specific.
- * Byte code is platform independent.





It convert the code into byte code and it is machine-Independent.
(.class)

So that java is platform independent.

- All the primitive data type is platform independent.

3. Robust & Secure:

→ Compile & execute by time checking strictly
→ Sc.close();
eg, delete the space get by scanner class.
→ garbage collection available in the java.

Secure:

→ Don't have pointer concept in java
→ No one can access the address of Memory in Java
→ Data mining

① Can we write the program without setting the path?

Yes, we can execute the java program without setting the path in CMD.

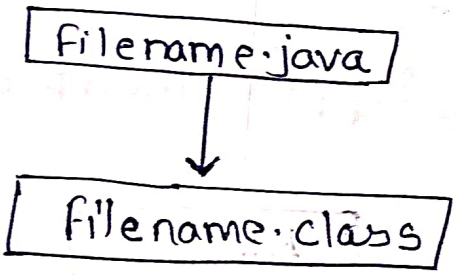
→ set path = jdk path

→ javac filename.java

→ java filename

- jdk is present in another place & .java file is another place, the OS will set path = " " between them.

② Class name & file name same?



→ Byte code
(It makes the java to platform independent)

It is not mandatory that the class name & file name must be same.

When a java file is compile then it generate the byte code according to its class name (not according to the file name).

java First → is there a bytecode file which name is First.

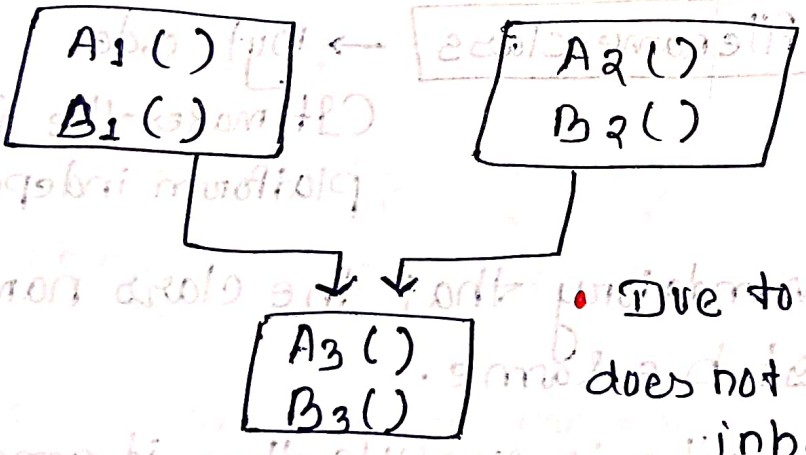
File name :-

```
class First{
}
```

→ If the class is public then class name & file name will be same.

→ If the class is not public then class name and file name can be different. The byte code is generate according to class name & we can run the program by java class name.

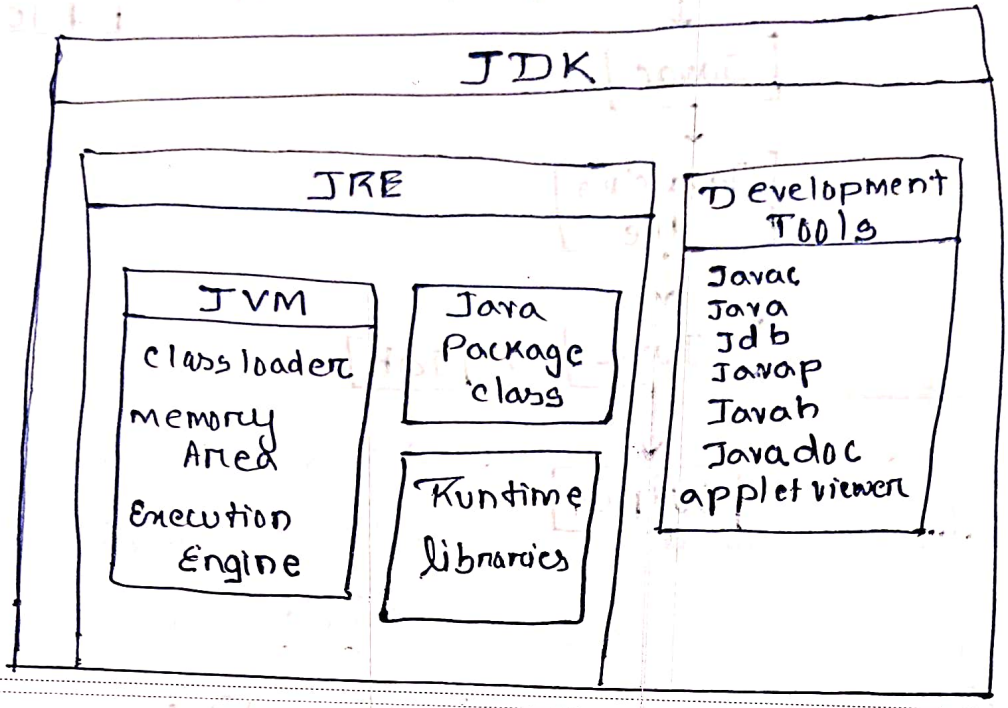
Why java does not support multiple inheritance?



- Due to Ambiguity java does not support multiple inheritance.
- We can achieve multiple inheritance by using Interface.

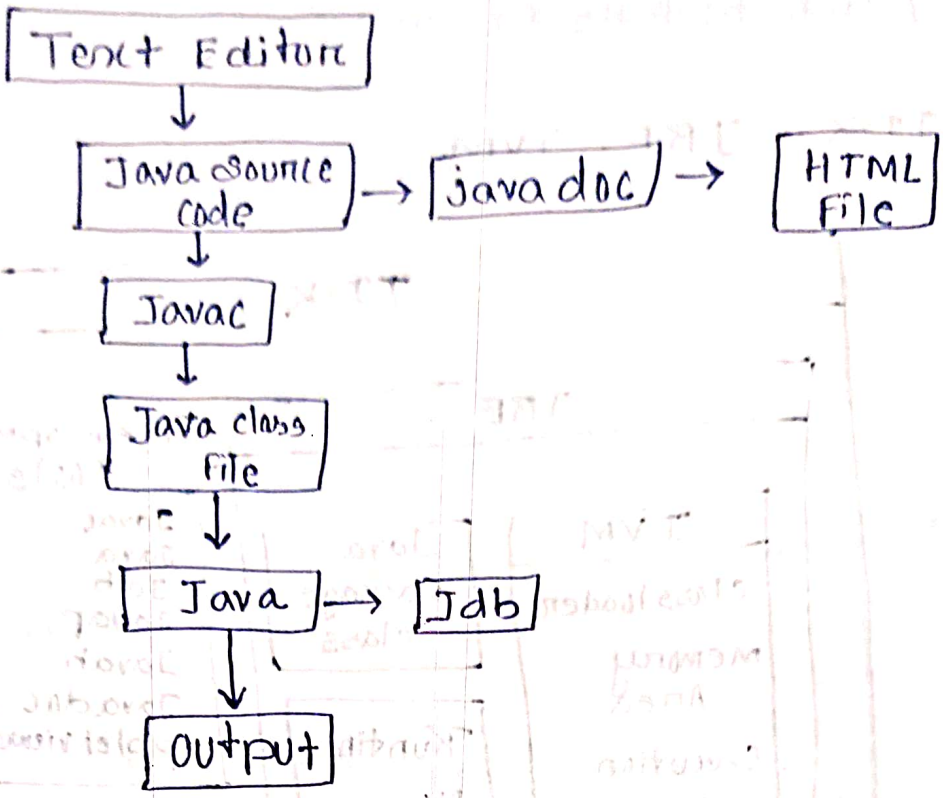
Architecture of java :-

JDK, JRE, JVM



(•) JDK (Java Development Kit) :-

1. Javac → java Compiler → Source → bytecode.
2. Java → java interpretor → execute bytecode & generate o/p.
3. Jdb → find error during program.
4. Javap → To generate source code from the executable code or find orginal source code
   (Java disassembler)
5. Javah → This tool is used to access the c & c++ in java
6. appletviewer → to run java web (HTML) in server.
7. Javadoc → convert the source code to HTML code.



(•) JRE (Java Runtime Environment) :-

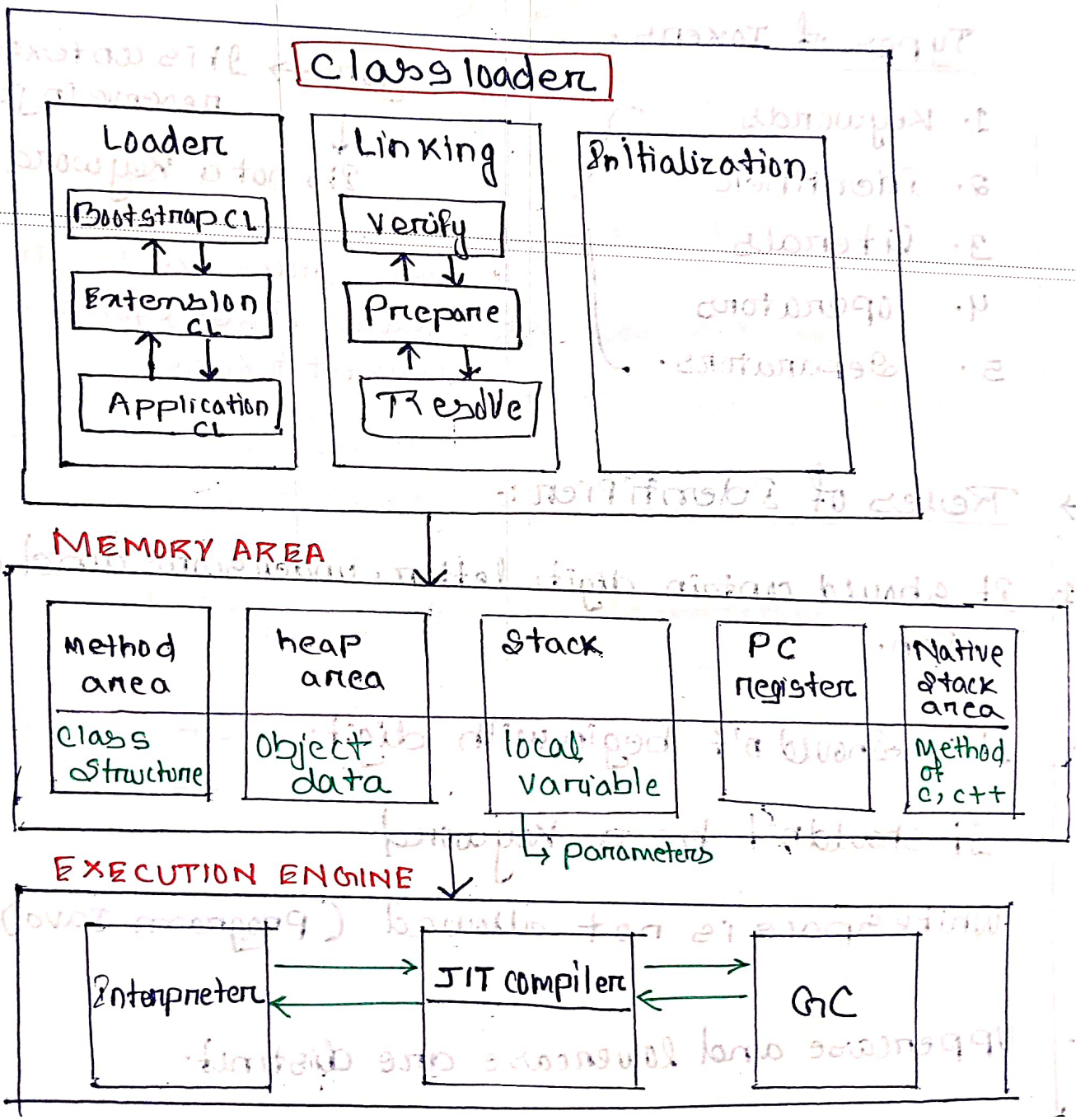
(•) Java Standard Libraries (JSL) :

1. Language support package (Java.lang.*;)
   (~~Java.lang. support~~)
2. Utility package (import java.util.*;)
3. I/O package (
4. AWL (Abstract window Toolkit package)
5. Applet package

(•) JVM (Java virtual machine) :-

- JVM is machine dependent.

1. Class loader
2. Memory Area
3. Execution Engine.



(•) **Tokens :** → Every program is a collection of tokens.

- Smallest individual unit in a program.

```
Class program
{
  public static void main (String args[])
  {
    System.out.print ("Hellow world");
  }
}
```

Types of Tokens :

1. Keywords
2. Identifier
3. literals
4. operators
5. Separators.

Keywords are preserved words.

goto → It is contextually reserve in java
↓
It is not a keyword.

When source code is converted to bytecode, it categorizes into different tokens.

→ **Rules of Identifier :-**

1. It should contain digit, letter, underscore and dollar sign.
2. It is should n't begin with digit.
3. It shouldn't be a keyword
4. White space is not allowed (Program Java) X
5. Uppercase and lowercase are distinct.

6. It can be of any length.

$\checkmark$, 9_ ✗
Program_java ✓
Program java ✗

(•) Literals :-

- These are the fixed values which are assigned to a variable.

Types of literals :-

1. Integer literals →
2. Floating point literals
3. Boolean literals
4. Null literals
5. String literals

For binary - 0b/0B 0101 → Not case sensitive

For decimal numbers, 9, 45, 2089

→ Boolean literals,
- True/false

→ Null literals,
- Stores null value

→ String literals,
ex → "abc", "A@123b"

→ For octal literals - 0

→ For hexadecimal literals - 0x/0X
ex → 0X1F

→ For floating point literals, (default floating point values double)
ex → 1.23, 1.23d, 1.23F

(•) Operators :-

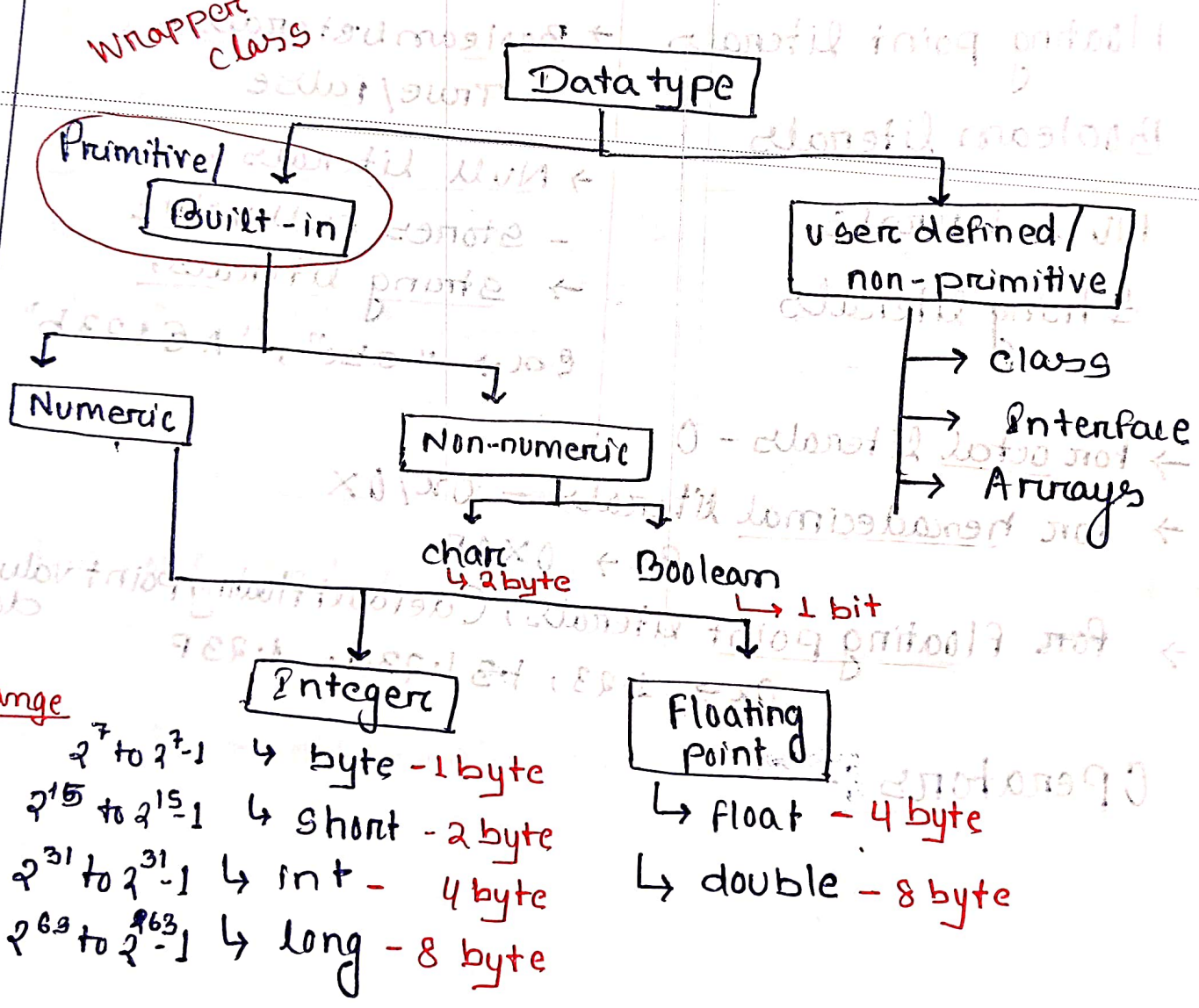
(•) Separator :-

Special symbols used to group/divide different statement.

Ex:- (), {}, [], ;, ,, .

. → To provide access

(•) Data Types :-



Range

2^7 to 2^7-1 → byte - 1 byte

2^{15} to $2^{15}-1$ → short - 2 byte

2^{31} to $2^{31}-1$ → int - 4 byte

2^{63} to $2^{63}-1$ → long - 8 byte

→ float - 4 byte

→ double - 8 byte

```
int a = 5;
byte b = 5;
short s = 5;
long l = 5;
```

00000000 ——— 00000101

(c) Data types belongs to which class? And try to verify how Java is a purely object oriented Programming language?

Ans

- Primitive datatypes doesnot belong to any class So, Java not a fully oop language.
- But for wrapper class, Java is 100% pure OOP language.

(d) To generate random Number :-

1. Random class
2. Math class ⟶ It will a double or floating value by default.

Case :-

1. Any random number
2. (0-n) ⟶ (0-100) ↓ (0-99)
3. (n-m) (15-20)

C

1. Array Indexout of bound exception.
2. Numb e format exception.

1. To generat random number by using random class:-

1. import java.util.Random;
2. create object of Random class
3. ~~we~~ use the random object call nextInt();

2. Now by using Math class:-

1. import java.util.*;
2. Math.random(); → (0.0 to 1.0) ← Its range is from
   ↳ (Returns a double value) → we need to convert it to integer
   (Generate a random no.)

```
import java.util.*;
Public class program10{
  public
      ("Random Number:" + ob.nextInt());
  {
      To get 20,
  ob.nextInt(max-min+1) + min;
```

1. (max-min) + min

2. (max-min+1) + min.

ob. nextInt (max-min) + min

nextInt: function; (max-min): argument

nextInt(10); → (0 to 9)

nextInt(15); → (0 to 14)

1. (max-min) 20-10 = (10)

2+10 = 12

0+10 = 10

9+10 = 19

nextInt (max-min+1) + min

nextInt(11) → 0 to 10

~~1. (max-min) 20-10 = 10~~

2. max-min+1.

= 20-10+1

= 11

(int) (Math.Random() * 50);

(0.0 to 0.9999...)

Instead of this value we can pass our range.

(int) (math.Random() * (max - min))

To generate random number using Random class:

```
import java.util.*;
Public class program10 {
    Public static void main (String [ ] args) {
        Random ob = new Random();
        int random = ob.nextInt();
        System.out.println("Random Number: " +
                                    random);
    }
}
```

* We will get different values when run multiple times.

OUTPUT:-

Random no:- -1378533321

S.out("Random no.:" + ob.nextInt());

↳ Same o/p.

```
import java.util.*;
Public class program10 {
    Public static void main (String[] args) {
        ~~int random = Ob.nextInt();~~
        int random = Math.random()
        System.out.println("Random Number:"+ Ob.next
                                                Int());
    }
}
```

OUTPUT:-

-2127787540

```
import java.util.*;
Public class program10{
 public static void main (String[] args) {
 int random = (int) Math.random();
 System.out.println("Random Number :"+ random);
 }
}
```

```
import java.util.*;
public class program10 {
    public static void main (String[] args) {
     Random ob = new Random();
     int random = ob.nextInt(100); // range is 0 - 99
    System.out.println ("Random Number :" + random);
    }
}
```

OUTPUT :-

Random Number : 0

```
import java.util.*;
public class program10{
    public static void main (String[] args){
     int random = (int) (math.random() * 100);
     System.out.println ("Random Number :" + random);
    }
}
```

OUTPUT :

Random Number : 31

```
import java.util.*;
public class program10{
    public static void main (String[] args){
        int min = 10;
        int max = 20;
        Random ob = new Random();
        int random = ob.nextInt (max - min) + min;
                                    // range is 0-99
        System.out.println ("Random Number:" +
                                    random);
    }
}
```

(6) I/p → Scanner class :-

1. import java.util.Scanner;
2. create object of Scanner class;
3. Use the object and call the method that are present in standard input stream.

↓

Methods are present,
→ nextInt
→ next Float

InputMismatchException: value out of range.

byte range → -128 to 127

Short range → -2^{15} to $2^{15}-1$

Integer range → -2^{31} to $2^{31}-1$

long rang → It will take 10 digit number.

To take String as input we have two Method :-

→ next()
→ nextline()

Note:-

(•) next() and nextline() using same scanner class object.

(•) Symbolic Constants :-

int a=15;

final int MARKS = 500;

final

↓

This keyword is used to make a variable constant.

Symbolic constants

int MARKS = 500;

It marks value of the variables as constant.

23/09

Byte b = 30;
int i = b;
short s = b;
long l = b;
float f = b;
double d = b;

byte → short → int → long → float → double

① int a = 130;
byte b = a;
byte b = (byte) a;
Syntax: (type) variable;

② a = 12;
byte b = (byte) a;

③ a = 297;
byte b = (byte) a;
byte = -128 to 127
= 256

```
public class project11 {
    public static void main(String[] args) {
        int a = 257;
        byte b = (byte) a;
        System.out.println("int value :" +a);
        System.out.println("Byte value :" +b);
    }
}
```

Int value : 257

Byte value :

```
int i = 130;
byte b = (byte) i;
```

00000000

Sign bit

1 (true)	0 (false)
- 2's complement of remaining 7 bits - Convert to Decimal and add the sign bit	→ convert to Decimal → add the sign bit.

```
i = 300;
byte b = (byte) i;
```

2^8 7 6 5 4 3 2 1 0

1 0 0 1 0 1 1 0 0

sign bit

0 0 1 0 1 1 0 0

1 1 0 1 0 0 1 1

+ 1

1 1 0 1 0 1 0 0

2	300	0
2	150	0
2	75	0
2	37	1
2	18	1
2	9	0
2	4	1
2	2	0
	1	0

Byte value : -44

- Can't convert to Boolean Type.

byte and short moved be explicit while converting.
char to long → implicit
Byte to char → explicit
short to char → implicit

```
psum (){
Byte b = 65;
char ch = (char) b;
s.out(ch);
}
```

o/p → A

```
byte b = 125;
char ch = (char)b;
s.out(ch);
```

o/p

```
long b = 125.0f;
char ch = b;
s.out(ch);
```

o/p

Arithmetic Operators

+
-
*
/
%

integer Arithmetic

a op b
a+b = 25
a-b = 15
a*b = 100
a/b = 4
a%b = 0

a = 20
b = 5

20 * -5 = -4
20 % -5 = 4
20 * -5 = -4

14 / -5 = 4

Integer arithmatic Real arithmatic Mixed mode arithmatic.

5 x 2

Arithmatic operation & Bitwise operator supports Type conversion.

Expressions :-

( )
* / %
+ -

{ we will scan the expression from left to right }

x = 9-12/3+3*2-1

= 9 - 4 + 6 - 1

= 5 + 6 - 1

= 11 - 1

= 10

x = 9-12/(3+3) * (2-1)

= 9-12/6 * 1

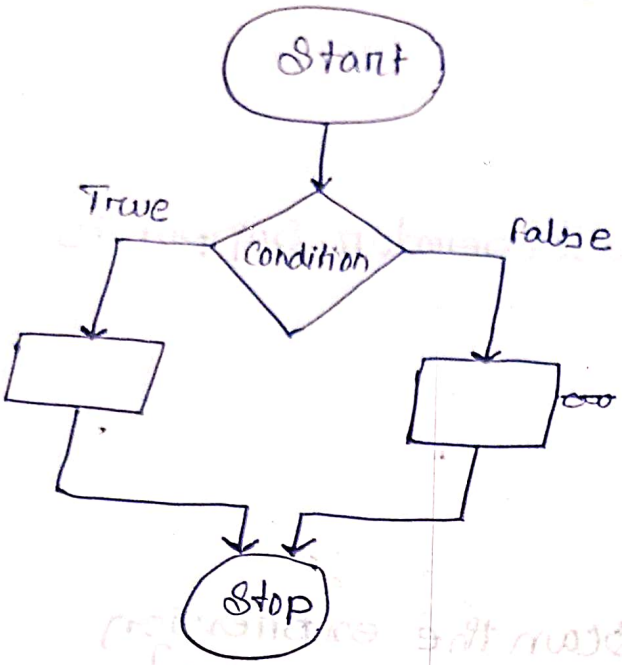
= 9-2*1

= 9-2

= 7

Condition Statement

(•) Flow chart of if block :-



1. Simple If block :-

```
If (condn)
{
==
}
State1;
State2;
```

2. if-else block :-

```
if(condn)
  {
  ==
  }
else
  {
  ==
  }
```

3. Nested if else :-

```
if (condn)
{
    if (condn)
    {
        ==
    }
    else
    {
        ==
    }
}
else
{
    ==
}
```

4. Else - if ladder :-

```
if (condn)
{
    ==
}
elseif (condn)
{
    ==
}
else if (condn)
{
    ==
}
else
{
    ==
}
```

(d) Mark → 0-100

if mark is > 90 and <= 100 → A

> 80 and <= 90 → B

> 70 and <= 80 → C

> 60 and <= 70 → D

< 60 → Fail

Default mark < 0 or > 100 → Invalid input

~~Else~~

(e) Switch case :-

```
Switch (variable)
{
  case value 1 :
      // statem;
  case
```

①

```
float x;
if (x>2)
{
float result = 2*x+5;
System.out.println(result);
}
else
 {
 float sum= 1.5*x+3;
S.O.Pln(sum);
}
```

②

```
float x;
if(x.1=40)
{
    if(x<40)
     {          "salary"+
      S.O.Pln(4** 4 *x+100)
     }
    else
     {          "salary"+
      S.O.Pln(4.5* x+150)
     }
 }
else
 {      "salary"+
 }S.O.Pln(,300);
```

(3) ~~Marks = (x-85)?~~

grades = (marks >= 90) ? 'A' : (marks >= 80 && marks < 90) ? 'B' : (marks >= 70 && marks < 80) ? 'C' : ~~Grade~~ 'F';

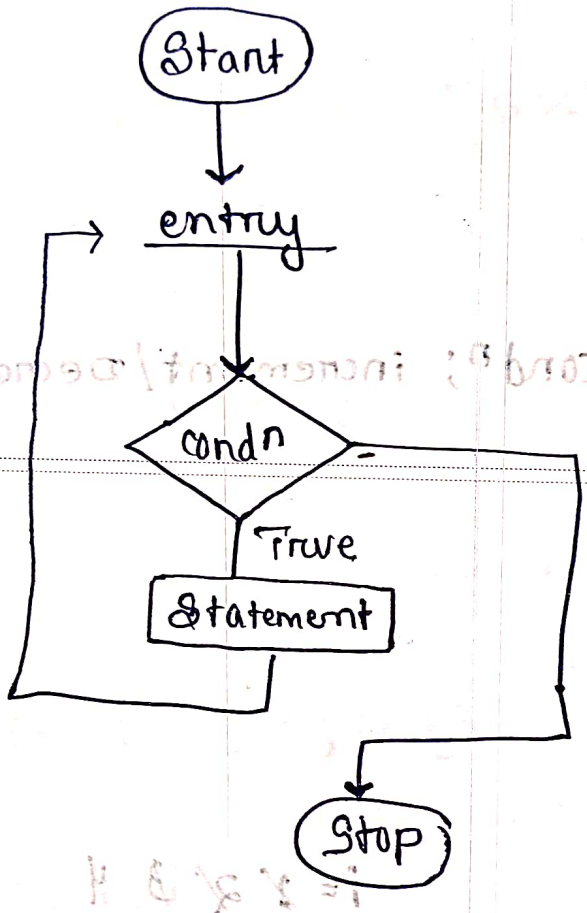
(•) Taking character as input by using Scanner class :-

0	1	2	3	4
H	e	l	l	o

(1) char At(int integer);

System.out.println(args[0])

```
import java.util.Scanner;
Public class dibya{
Public static void main (string[] args){
  Scanner ob = new Scanner();
    if (a>b)
    {
```



(•) While loop :-

Syntax:

```
initialize;
while (condn)
  {
  Statement
  increament/Decrement
  }
```

⊙ do-while loop :-

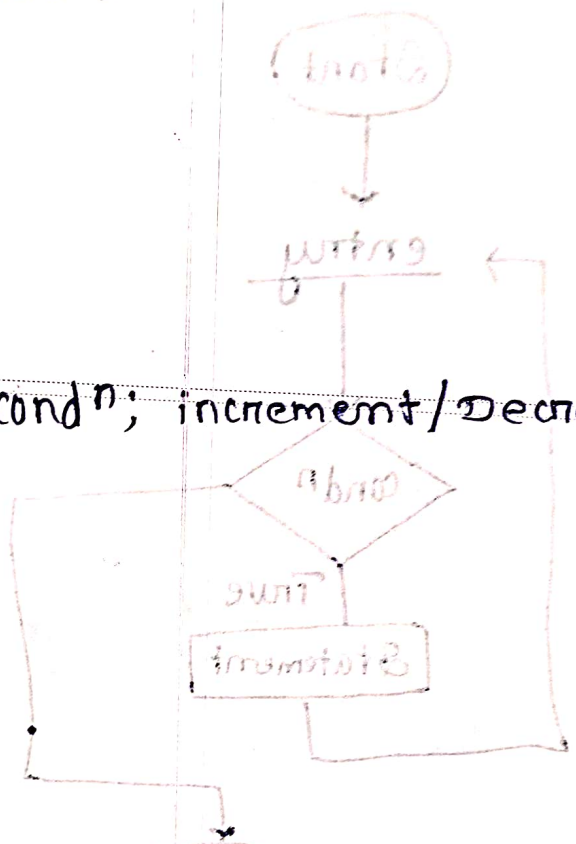
Syntax;

```
initialize;
do
{
statement;
.
.
.
increment/Decreament;
} while(condn);
```

⊙ For loop :-

Syntax :-

```
for (initialization; condn; increment/Decrement);
{
  Statement;
.
.
}
```



```
int i=1 ,j=1;
while (i<5)
{
  j=j*2;
 System.out.println (j);
   i++;
   j++;
}
```

i = ~~1~~ ~~2~~ ~~3~~ 4
j = 1 ~~2~~ ~~3~~ 6

```
int num = 4; i=1;
int
while (i<=10)
 {
  num*
 System.out.println (num*i);
  i++;
 }
System.out.println ("5x"+a+"="+ num*i);
```

S.o.p

```
1
2 3
4 5 6
7 8 9 10
```

```
*
* *
* * *
* * * *
```

```
int i=1
for (int i=1 ; i<=10 ; i++)
 {
  S.o.pln (i);

for (int a=1 ; a<=4 ; a++){
    for (int b=a; b<=a; b++) {
      S.o.pln (" *");
    }
S.o.pln (" ");
}
```

```
*
*
```

(•) Jumps in Java :-

1. break
2. continue
3. return
4. goto
5. labels with break and continue.

1. break :- (It is used to come out from the loop or switch case).

```
for (int i=0 ; i<5 ; i++)
    {
        if (i==2)
        {
            break;
        }
        System.out.println(i);
    }
```

OUTPUT :-

0

1

2. continue :- (used to skip current iteration and print the next interation).

```
for (int i=0; i<=10; i++)
    {
        if (i%2==0)
        {
            continue;
        }
```

→ Skip the even nos

System.out.println(i); ⟶ print all not nos.
}

5. Labels with break and continue :-

```
outer : for (int i=0; i<5; i++)
        {
inner:    for (int j=0; j<5; j++)
          {
            if (i==2 && j==2)
              {
                break outer;
              }
          System.out.println(i+" "+j);
          }
        }
S.O.P ("Helloworld");
```

OUTPUT:-

i=0 , j=0

@

0	0	2	0	4	0
0	1	2	1	4	1
0	2	2	3	4	2
0	3	2	4	4	3
0	4	3	0	4	4
1	0	3	1	5	0
1	1	3	2	5	1
1	2	3	3	5	2
1	3	3	4	5	3
1	4			5	4

Hello world

0 0
0 1
0 2
0 3
0 4

1 0
1 1
1 2
1 3
1 4

2 0
2 1
2 2
2 3
2 4 } → skip

3 0
3 1
3 2
3 4

4 0
4 1
4 2
4 3
4 4

5 0
5 1
5 2
5 3
5 4

Hello world..

* To input a character,

```
Scanner ob = new Scanner();
char ch = ob.next().charAt(0);
```

* To input a string,

```
Scanner ob = new Scanner();
String message = ob.nextLine();
```

(•) Class Object & Method

Date - 18.10.24

Documentation part	optional
Package statement	
* Import statement	
Interface statement	
* class definition	

Main class
{
main method defin^
}
	mandatory part must reanired

(•) How to define one class ?

Syntax of class,

valid identifier

class Class name [Extends Superclass]

{

[fields ;]

[methods;]

}

Note

* What ever written in [ ] braket are all optional.

Ex :-

```
class Rectangle
{
  int length;
  int breadth;
}
```

Field

```
class Example
{
}
```

Note

* No memory is going to allocated for the field because field is present inside the class and the class has no memory.
* Class definition doesn't occupied memory space.

(•) Field declaration :-

Syntax,

type variablename;

return type — name of the variable ;

Eg,

int length;

Note:-

→ No memory has be allocated for the fields that has been defined on a class unless define a class.

Eg,

```
int length;
class Rectangle
{
  int length;
  int breadth;
}
```

Instance variable

* Instance variable :- That define inside a class or member variable.

* Local variable are local to the methods.

(c) Method declaration:-

Define a method on method declaration contain 4 different parts,

1. Name of the method
2. Return type of the method
3. List of parameter
4. Body of the method

Syntax,

```
return type Method name(parameters)
{
  body of method
}
```

```
void compute(int x, int y)
{
  length=x;
  breadth=y;
  S.O.P (length * breadth);
}
```

Note:-

* Return type is the type of value method is going to be return the code inside the body.

Note,

Imp:- All the instance variables & methods are accessible by all the methods in the class but a method can not access a variable defined in another method.

Ex:-

```
Class Eg
{
  int x;  ⟶ Instance variable
  void M1()
  {
    int y=5;      } Scope of the 1st method
    x=y;  ⟶ Local to method M1()
  }
  void M2()
  {
    int z;
    z=10;  ⟶ Local to M2()
    x=z;
    y=z;
  }
}
```

Note:-

* one method calls another Method: or

⇒ All the method can be accessed inside the method class.

Eg:

```
Class Rectangle
{
  int length;
  int breadth;
  void compute (int x, int y)
  { length = x;
    breadth = y;
    int area = length * breadth;
    S.O.Pln ("Area is:" + Area);
}}
```

(c) Creating an Objects :-

* Class name refercance variable-name = new class name();

Rectangle rect1 = new Rectangle();

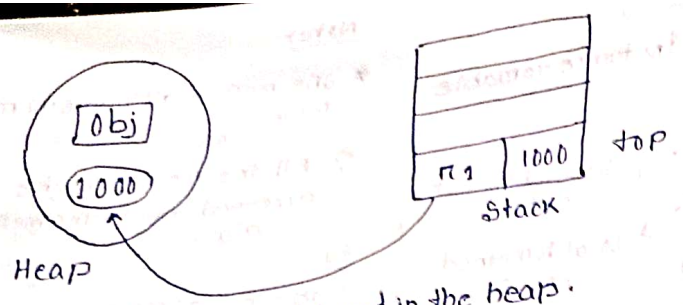
(Inside main method driver class)

Rectangle ret1;
ret1 = new Rectangle();

New :-

Rectangle r1 = new Rectangle();

The new operation creates an object in the heap area of main memory when the object is created, it occupies a specific location in the heap area and that location has address, the new operational operator then this address in (as a reference which is assign to the variable of the class type).



① The object itself is stored in the heap.

② The reference (address) to the object location in the returned & stored in the variable name.

③ The variable then acts as a pointer to the object allowing you to access manuputed object method.

Note:-

Variable does not hold the actual object but rather a reference to where object is located the memory.

```
Class Rectangle
{
  int length;
  int breadth;
  void compute (int x, int y)
  {
    length = x;
    breadth = y;
    int area = length * breadth;
    S.O.P (area);
  }
}
```

```
class Eg
{
  psvm()
  {
    Rectangle r1 = new Rectangle();
    { r1.length = 10;
      r1. breadth = 20;
    r1.compute(10,20);
  }
```

(•) Initializing and Accessing class members :-

1. Direct Access :-

Syntax,

Reference V-name Instance

```
Variable_name . Variable ;
```

```
r1.length;
r1.breadth;
r1.length = 10;
r1.breadth = 20;
```

2. Use Methods :-

Syntax

```
Reference variable = Methodname();
```

```
r1.compute(10,20);
```

```
class Eg
{
  psvm() {
    Rectangle r1 = new Rectangle();
    r1.compute(10,20); ⟶ 200
    r1.length = 15;
    r1.breath = 20;
    S.O.P (r1.length * r1.breadth);
  }
}
```

```
class Eg
{
  psvm(){
   Rectangle r1 = new Rectangle();
  S.O.Pln(r1.length);
  S.O.pln(r1.breadth);
   r1.length = 5;
   r1.breadth = 5;
  int area = r1.length * r1.breadth;
  S.O.p(area);
}
```

2.

* CONSTRUCTOR:-

- Constructor is used to initialize the object.

Properties of construct,

1. The name of the constructor is same as the class name
2. Constructor does not have any return type.
3. Constructor can be public, private, protected.
4. Constructor is automatically called when ~~the~~ one object of a class is created.

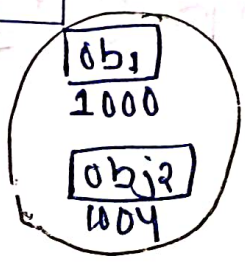
5. A constructor can't be not be a static.

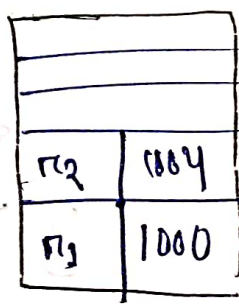
```
tangle()

ength = 10;
readth = 20;
```

```
class Rectangle
{
int length, breadth;
Rectangle()
{
 length = 10;
 breadth = 20;
}
void compute()
 {
  int area = length * breadth;
   S.O.P(area);
 }
}
```

```
class main()
 {
  psvm( )
  {
Rectangle R1 = new Rectangle();
R1.compute(); → 200 o/p
Rectangle r2 = new Rectangle();
 r2.length = 5;
 r2.breadth = 10;
  r2.compute();
 }
```





(•) Note:-

Each object has its own copy of the instance variable of one object this means if any object have no effect on the variable of another.

(•) ~~Parameterised~~

(•) Types of constructor:-

1. Default Constructor:-

Define a class student with two intanie variable, name, rollno. of type string and interger respectively, Define a constructor to initialize the object & a method void getinfo to retrive the information. Create a class main to implement class Student by creating one object.

```
Class Student{
{
  String Name;
  int    rollno;
 void getinfo()
 {
  S.O.Pln ("Name of student :" + name);
  S.O.P ("Student Roll Number :" + rollno);
 }
}
```

```
class main()
{
    psvm(string args[])
    {
        Student s1 = new Student();
        s1.name = "dibya";
        s1.rollno = 208;
        s1.getinfo();
    }
}
```

```
class Student
{
    String name;
    int rollno;

    Student()
    {
        String name = "dibya";
        int rollno = 208
    }
    void getinfo()
    {
        S.O.pln(name);
        S.O.p(rollno);
    }
}
```

① First approach,

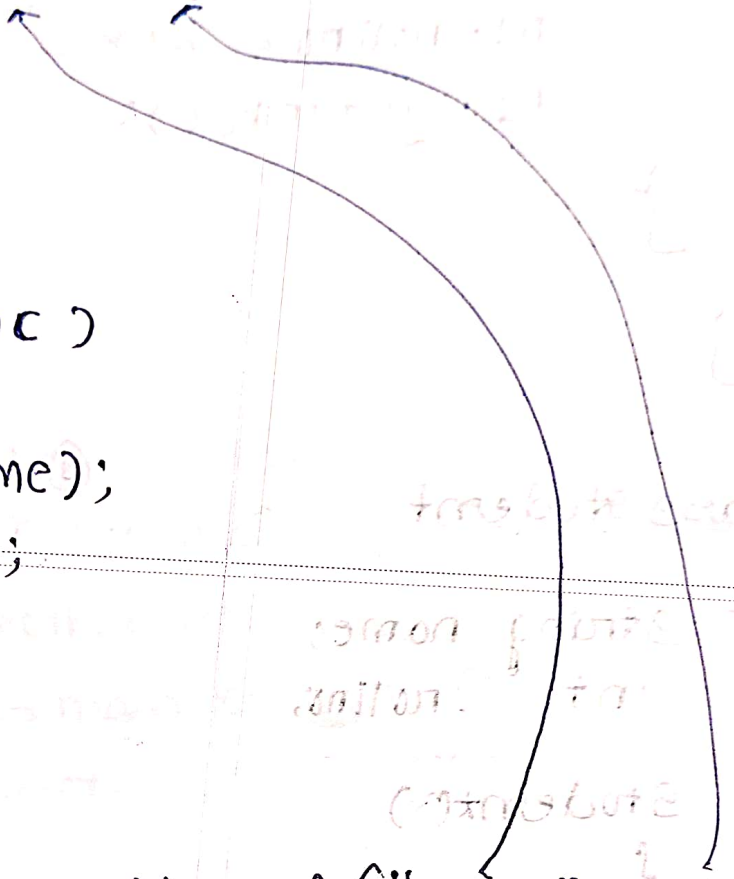
```
class main()
{
    psvm(string args[])
    {
        Student s1 = new Student();
        s1.getinfo();
    }
}
```

② another approach.

```
psvm(string args[])
{
    Student s1 = new Student();
    s1.getInfo();
}
```

② Parameterized Constructor :-

```
class student {
{
  Student ( string n, int r)
  {
     name = n;
     rollno = r;
  }
  void  info getInfo ()
  {
     S.o.pln ( name);
     S.o.p (rollno);
  }
PSVM ( )
{
  Student S1 = new Student ("Dibya", 208);
   S1. getInFo();
}
```



→ this keyword :-

this key is used to refer the current class object.

(c) Usages :-

- this key is used to refer the current class instance variable.
- this is used to invoke the current class method.
- ~~this is used~~ this() is used to invoke the current class constructor.
- this can be parsed as an argument in method call.
- this can be passed as an argument in constructor call.

1. this can be used to refer the current class instance variable,

```
class student
{
  string name;
  int rollno;
  student (string name, int rollno)
  {
    this.name = name;
    this.rollno = rollno;
  }
```

2. this is used to invoke the current class method,

```
class A
{
  void m1()
  {
    S.o.println("M1 is called");
  }
  void m2()
  {
    S.o.println("M2 is called");
    this.m1();
  }
  psvm(String args[])
  {
    new A().m2();
  }
}
```

OUTPUT:-

M2 is called

M1 is called

one more approach to access the method of a class.

```
int x = 10;
```

It means address of 10 is stored in x

```
Student s1 = new Student();
```

- It mean the address of Student object is stored in the reference variable s1.

And x is pointing to the location of 10.

Q) Is it mandatory to used this while invoking the current class Method ?

Q) create a class Employee with instance variables name, employee-id & gross_salary,

1. constructor : Initializes name, eid
2. Gross Salary (int basic) : calculate gross_salary = basic + HRA + DA

HRA is 30% of Basic
DA is 24% of HRA

3. display() : Display employee details.

Write a main class to test a employee class

```
class Employee
 {
  ~~employee(s~~
 String name;
 int eid;
 float gross_salary;

 Employee()
  {
   name = "Dibya";
   eid = 208
   ~~gross_salary = 55000.75;~~
  }
```

```
void Gross_Salary (int basic)
    {
  Float HRA = basic * 30/100;
  Float DA = basic * 24/100;
     gross_salary = basic + HRA + DA;
    }
void display()
     {
      S.O.pln ("Name:" + name);
      S.O.pln ("EmployeeId:" + eid);
      S.O.P ("Gross_Salary:" + gross_salary);
     }
Public static void main (Strin
class main()
  {
    psvm (String args[])
      {
       Employee e1 = new Employee();
         e1.Gross_Salary (10,000);
        e1.display();
      }
  }
```

② Student : Student_name, roll_number and average_mark

implement:

1. constructor : Initializes student_name & roll_number

2. calculate Average (int mark1, int mark2, int mark3): computes the average

3. display() : Shows Student details.

Test the Student class with a main class.

```
class Student
{ String student_name;
{ int roll_number;
{ Float average_mark;
   Student ( String name, int roll)
    {
      student_name = name;
      roll_number = roll;
    }

  void calculateAverage (int mark1, int mark2, int mark3)
    {
     average_mark = mark1 + mark2 + mark3 /3 ;
    }
void display();
    {
     S.O.Pln ("Student Name :" + student_name);
     S.O.Pln ("Student Roll Number :" + roll_number);
     S.O.P ("Average mark :" + average_mark);
}
```

```
class main
{
  psvm(String args[])
  {
    Student S1 = new Student("Dibya", 208);
    S1.calculateAverage(100, 95, 83);
    S1.display();
  }
```

③ Loan: principal, rate, time & simple-interest

implement:

1. Constructor: Initializes principal, rate, & time (use this keyword)
2. calculateInterest(): Computes simple-interest and prints the value.

④ Class: Book

1. Attributes:
   - title (String)
   - author (String)
   - Price (double)
2. Constructor: Accepts tittle

```
class Loan
{
  float Principal;
  float rate;
  float time;
  float simple-interest;

  Loan (float p, float r, float t)
    {
this.Principal = Principal;
this.rate = rate;
    float = t;
this.time = time;
    this.calculateInterest();
    }
    void calculateInterest()
      {
        simple-intenest = Principal * rate * time/100;
        S.O.P ("Simple-interest:" + simple-interest)
      }
  class main
    {
      psvm (String args[])
        {
        Loan L1 = new Loan (5000, 8, 30);
    }
}
```

4) Class : BOOK

1.

```
class BOOK
{
    String title;
    String author;
    double price;
    ~~Book (String t, String a)~~
    ~~{~~
    Book ( String title, String author)
    {
        this.title = title;
        this.author = author;
    }
```

```
void setPrice(double price)
  {
    this.price = price;
    this.showDetails();
  }

Void showDetails()
  {
    S.o.pln(title);
    S.o.pln(author);
    S.o.pln(price);
  }

Psvm(String args[])
  {
    Book r1 = new Book("King", "DRSahoo");
    r1.setprice(1050.0);
  }
}
```

(•) this can be used to invoke the current class constructor

```
class A
{ A()
  {
  S.o.pln("constructor called");
  }
  A(int x)
  {
   this();
   S.o.pln("x="+x);
  }
```

```
class B
{
 psvm()
 {
  A obj=new A(5);
  }
 }
```

Note :-

```
A(int x)
{
 S.o.pln("x="+x);
 this();
 }
```

~~this() must be in the~~

this() must be a first statement in constructor

b) calling of one constructor from another constructor is called as constructor chaining.

```
A()
{
 this(5);
 S.o.pln("constructor called");
}
A(int x)
{
 S.o.pln("x ="+x);
}
```

Imp

c) this can be passed as an argument in method call :-

```
class A
{
 int a,b;
  A()
  {
   a=10;
  }b=20;
 void display(A obj)
  {
   S.o.pln(" A ="+obja.a+
           "B="+obj.b);
  }
 void get()
  {
   display(this);
  }
}
```

```
class B
 {
 psvm()
 {
 A object = new A();
 object.get();
 }
}
```

A

(c) this can be passed as an argument in constructer call:

```
class A
{
  B obj;
  A(B obj)
  {
    this.obj = obj;
    obj.display();
  }
}

class B
{
  int x = 10;
  B()
  {
    A obj1 = new A(this);
  }
  void display()
  {
    S.o.p("value of x in class B" + x);
  }
}

class C
{
  public static void main(String args[])
  {
    B obj2 = new B();
  }
}
```

(•) Copy Constructor :-

```
class Person
{
Private String name;
private int age;
Person (String name, int age)
  {
  this.name = name;
}this.age = Age;

person (person Other)
{
  this.name = other.name;
  this.age = other.age;
}
void display()
{ S.o.pln ("Name=" + name + ~~name +~~ "Age" = + age)
}
Public static void main (String args[])
{
  Person P1
```

Q) W.A.P by defining a class called Area with two instance variable length and breadth, define two constructors which accepts length and breadth as an argument and length as a argument, Define a method compute, which compute the area of Savare rectangle accordingly:

```
Class Area
{
 Float length;
 Float breadth;
 Area (Float length, Float breadth)
   {
    this.length = length;
    this.breadth = breadth;
   }
Area (Float length)
 {
  Float breadth;
  this.length = length;
  breadth = length;
 }
Void compute ()
  {
   S = length * length;
   rc = length * breadth;
  Area = length * breadth;
```

(a) Method Overloading :-

Same name with

1. Same method name
2. Different parameters
3. Return type. (overloaded method does not distinguish based on the return type).

4. Compile ~~type~~ time ~~pt OP~~ polymorphism.
method overloading is resolve at compile time.

```
Class Addition
{
  void Add (int a, int b)
  {
   System.out.println (a+b);
  }
  void Add ( int a, int b, int c)
   {
    System.out.println (a+b+c);
   }
 Public Static void main (String args[])
      {
       Addition A1 = new Addition ();
        A1.Add (5,6,7);
      }  A1.Add (5,6);
}
```

d) W.A.I.P to print different Mersages using Method loading

d) Creat a printer clars with an overloaded printMsg() that
Print a default Msg if no. argu are provide.
Print " custom " " a string arg is provide.
" " " Msg multiple time if a string and an integer (no. of repetation are provided).

d) create a text clars & demonstrate the program.

```
Class over
{
  void Print()
  {
    S.o.pln("Hii");
  }
  void print(String m)
    {
      this.m = m;
    S.o.pln(m);
    }
  void print(String name, int roll)
    {
      this.name = name
     this.roll = roll
```

```
        S.O.Pln (name + roll);
    }
    PSVM(

    class printer
      {
        void over()
        {
          S.O.Pln ("No argument is passed");
        }
       void over(String name)
        {
          this.name = name;
          S.O.Pln ("Name is:" + name);
        }
      void over(String name, int roll)
        {
        this.~~String~~ name = name;
        this.~~int~~ roll = roll
        ~~if (r~~
          for (i=0 i<= roll; i++)
            {
              S.O.Pln (name + roll);
            }
        PSVM(String args[])
          {
```

(•) W.A.JP to find the maximum value of two
Create a Maximumfinder class Findmaximum that accepts two of argument
(i) Maximum of two integer, (ii) maximum of two floats.

```
class MaximumFinder
  {
int Findmaximum (int n1, int n2)
    { this.n1 = n1;
      this.n2 = n2;
      if ( n1 > n2)
        {  return n1;
           S.o.pln (" n1 is greatest maximum");
        }
      else
        {    return n2;
           S.o.pln (" n2 is maximum")
        }
    }

doubleFindmaximum (double d1, double d2)
    {
     this.d1 = d1
     this.d2 = d2
    IF (d1 > d2)
     {   return d1;
      S.o.pln (" d1 is greater");
     }
    else
    {    return d2;
     S.o.pln ("d2 is greater");
    }
```

```
float FindMaximum (float f1, float f2)
{
this.f1 = f1
this.f2 = f2

greater = (
if (f1 > f2)
{ return f1;
S.O.pln ("f1 is greater");
}
else
{ return f2;
S.O.pln ("f2 is greater");
}

PSVM (String args[])
{

FMaximumFinder M1 = new MaximumFinder();
S.O.pln (M1.FindMaximum (8, 3));
S.O.pln (M1.FindMaximum (10.0, 15.0));
S.O.pln (M1.FindMaximum (7.5, 4.5));
}
```

(•) Static :-

- It is a key word used for memory management.

```
class Test
{
  int a;
  int b;
 Test(int a, int b)
   {
     this.a = a;
     this.b = b;
   }
  void display()
   {
     S.O.pln ("A=" + a + "B=" + b);
   }
}

class TestDemo
  {
    psvm (String args[])
    {
      Test T1 = new Test (5,6);
      Test T2 = new Test (1,2);
      Test T3 = new Test (3,4);

      T1. display();
      T2. display();
     T3. display();
    }
```

1. Static with variable.
2. Static with Method.
3. Static with a block.
4. Static with a class.

Static variable & Method :-

- A variable defined with static keyword called as a static variable or class variable.
- There variable are share among all the objects of class.
- Java creates only one copy for a static variable.

- The method defined a static keyword called as a static method.
- This method belongs to the entire class.
- static method can be called directly using class name with out need to obj.
- Static method, (~~variable/method~~) can only access other static member (variable/method)
- Static methods can not refer to this and super

```
class Mathoperation{
    Static float multiplication(float a, float b){
        return a*b;
    }
    Static float Division(float a, float b){
        return a/b;
    }
}
Public class Static Ex2{
S.o.pln(Mathoperation.multiplication(4.0f, 5.0f));

S.o.pln(mathoperation.Division(40.0f, 5.0f));
}
}
```

(6) Static block :-

- Static block is also called as Static initialization.
- It is a block of code inside a class that runs only ~~one~~ once when the class is loaded.
- It is often useful and on initializing to Static variables or executing setup code.

```
clars Static Block{
   Static int value;
   Static {
    value = 10;
    S.O.Pln (" Static bloc executed and value intialized to 10");
   }
 Static void display_ value(){
   S.O.Pln ("value =" + value) ;
  }
 Public clars Static Ex3 {
    PSVM (String args[]){
    Static Block. display - value();
}}
```

OUTPUT:

Static block executed and value intialized to 10

value = 10