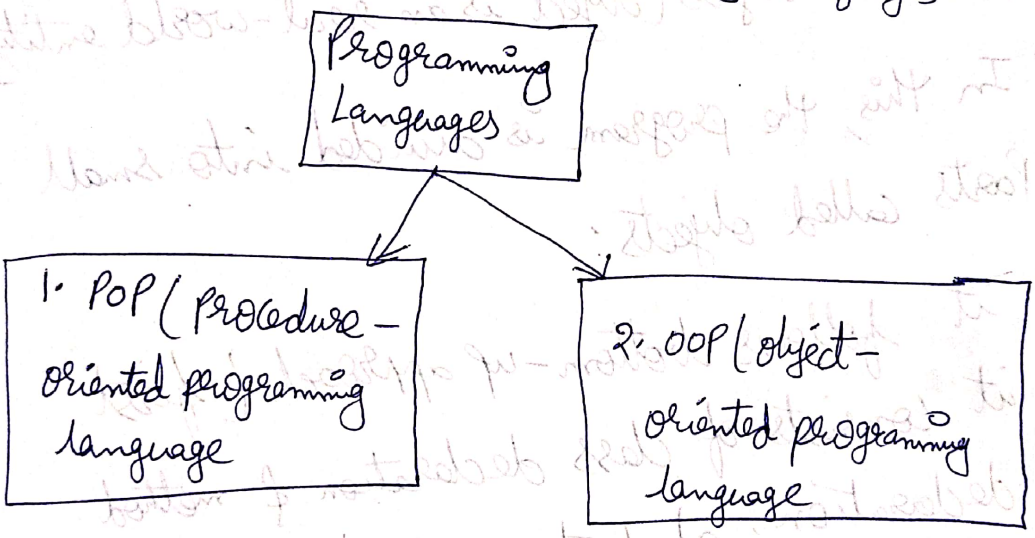
UNIT - I

:- OOP's through JAVA :-

Object - Oriented Programming Language (OOP's) through JAVA :-

⇒ There are 2 types of Programming languages available.

1. POP : Procedure Oriented programming languages.
2. OOP : Object-oriented programming languages.



1. **POP :-** Procedure - oriented programing language depends on the procedures called functions.

→ **function :-** A function is a group of statements that together perform a task.

⇒ Every program has at least one function, which is main().

→ in POP, the total program is divided into number of functions.

→ POP follows top-down approach.

→ where, in c language main function, the execution of program starts from main method. from main method all the statements of program are executed.

2. OOP:- object-oriented programming language depends on objects (object is an real-world entity).

→ In this, the program is divided into small parts called objects.

→ it follows bottom-up approach. (first it consists of class declaration & method declaration, at last we had main method. but execution starts from main method.

Differences b/w POP & OOP:-

POP	OOP
1. Pgm is divided into small parts called as functions.	1. Pgm is divided into small parts called as objects.
2. it follow top-down approach.	2. it follow bottom-up approach.
3. there is no Accesses specifiers (like private, Public, protected)	3. it consists Accesses specifiers (Public, private, protected).
4. no security	4. Provide security
5. Eg:- C, FORTRAN, PASCAL	5. Eg:- C++, Python, Java.

History of Java :-

⇒ Java is an object-oriented programming language mainly designed to develop internet applications, mobile applications, enterprise applications by providing platform independency.

⇒ it was developed by James Gosling at SUN Microsystems in early 1990's.

⇒ its first version is released in January 23/1996.

⇒ Java is descendent of C, C++ programming language. its syntax is similar to C and C++, but Java omits many of the features that make C and C++ confusing, complex and unsafe.

Who developed JAVA?:- "JAMES GOSLING"

⇒ James Gosling developed JAVA at SUN Microsystems.

⇒ He is famous software developer, known as the father of JAVA.

⇒ Before Java language, the software for consumer electronic devices such as Washing machines, micro ovens and micro controllers was developed by C, C++ programming languages.

⇒ C, C++ languages are platform dependent.

⇒ James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan are small team members of SUN Engineers called Green Team.

⇒ firstly, it was called "Green Talk" by James Gosling and the file extension was "-gt". after that, it was called OAK and was developed as a part of the Green project.

OAK Tree :- Java language was initially called as "OAK". OAK is a tree name, that stood outside of James Gosling's office.

⇒ OAK tree represents "Symbol of strength", and chosen as a National Tree of many countries like the U.S.A, France, Germany, Romania, etc.

⇒ In 1995, OAK was re-named as "JAVA" because it was already a "Trade Mark" by OAK Technologies.

→ JAVA name was chosen by Gosling while having cup of coffee with his team members near by his office.

→ JAVA is an Island in Indonsia, JAVA coffee was produced by Espresso Bean, it is a kind of coffee seed, where the first coffee was produced.

→ JAVA was renamed from Java coffee.

Abbreviation of JAVA:-

→ JAVA refers to the hot, aromatic drink coffee.

→ This is the reason Java language icon is coffee cup.

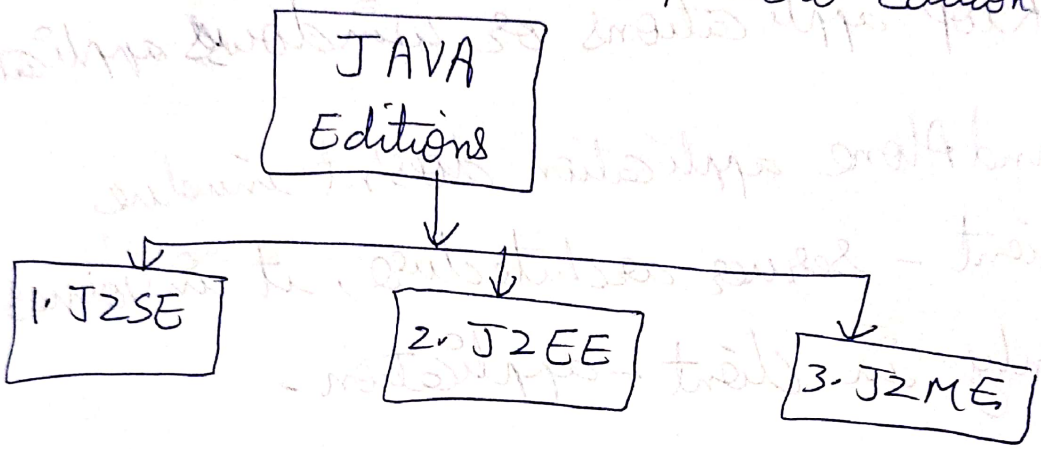
JAVA phrase:- phrase of JAVA is "WORA"

"Write Once Run Anywhere"

→ Once compiled Java code can run on all platforms.

Editions of JAVA:-

1. J2SE :- Java 2 standard Edition
2. J2EE :- Java 2 Enterprise Edition
3. J2ME :- Java 2 Mobile/Micro Edition.

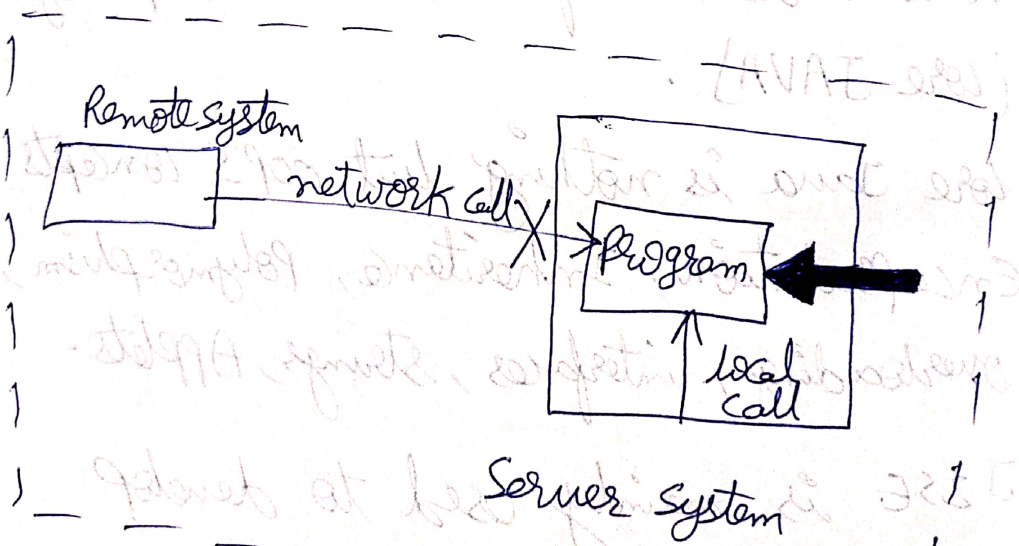


1. J2SE (standard Edition) :- its advantage is,
   - ⇒ it converts the fundamental concepts of JAVA (core JAVA).
   - ⇒ core Java is nothing but OOP's concepts like Encapsulation, Inheritance, Polymorphism, overloading, interfaces, strings, Applets.
   - ⇒ J2SE is mainly used to develop stand-alone applications.

Standalone application :- the applications which are installed in one computer performs actions in the same computer are known as stand Alone Applications (or) Desktop applications or windows applications.

⇒ stand Alone application doesn't involve client - server architecture, it involves only in client - application.

⇒ An application that can only be executed in local system with local call is called stand - alone application.

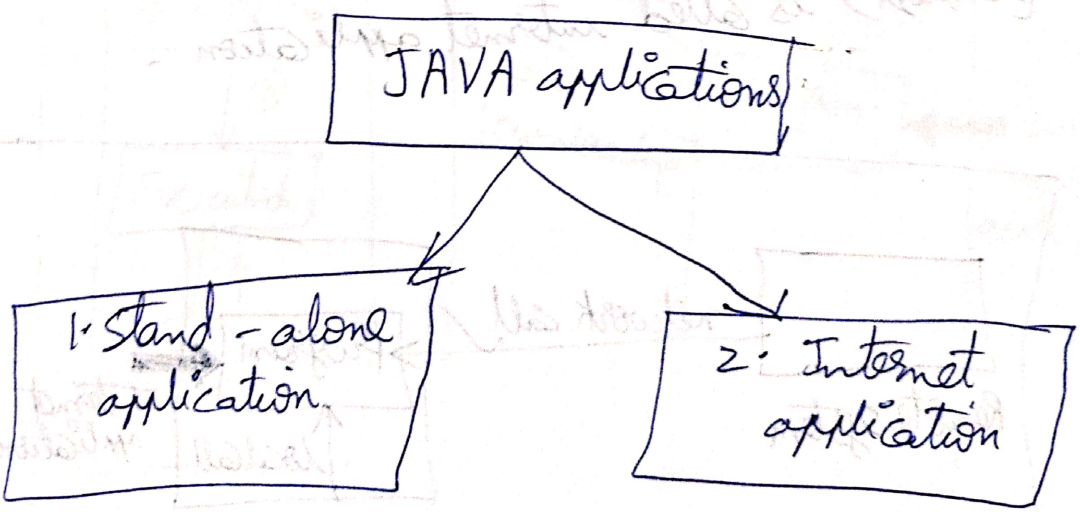


Stand - Alone application.

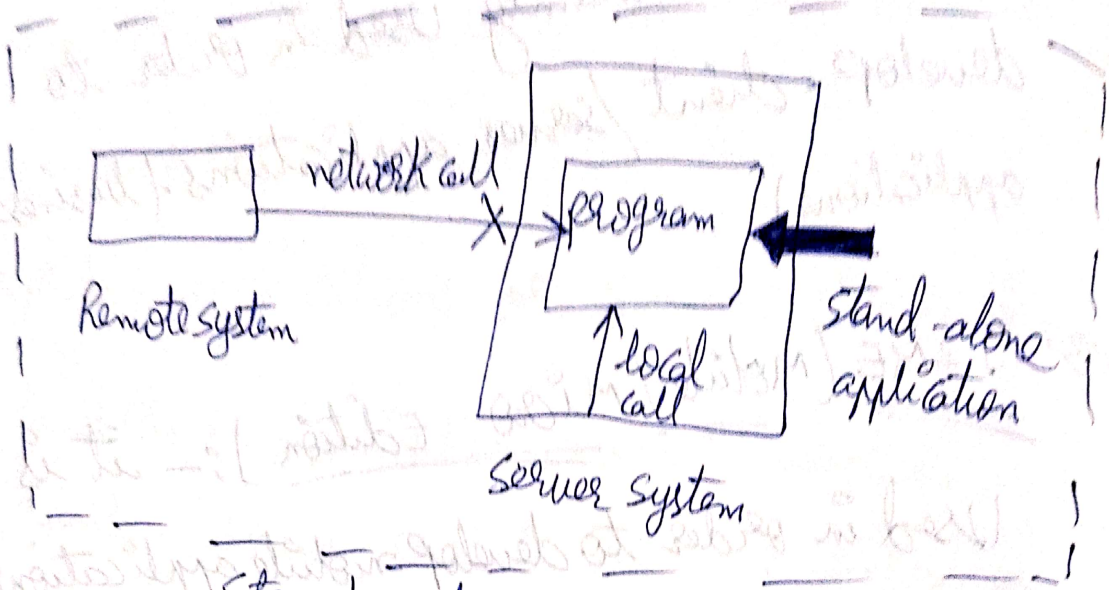
2. J2EE (Enterprise Edition) :- it covers advanced concepts of JAVA like servlets, JSP's. it is mainly used in order to develop client/server applications. (business applications).

3. J2ME (Mobile/Micro Edition) :- it is used in order to develop mobile applications.

JAVA applications :- Based on the way of execution of programs, all available applications are divided into 2 types

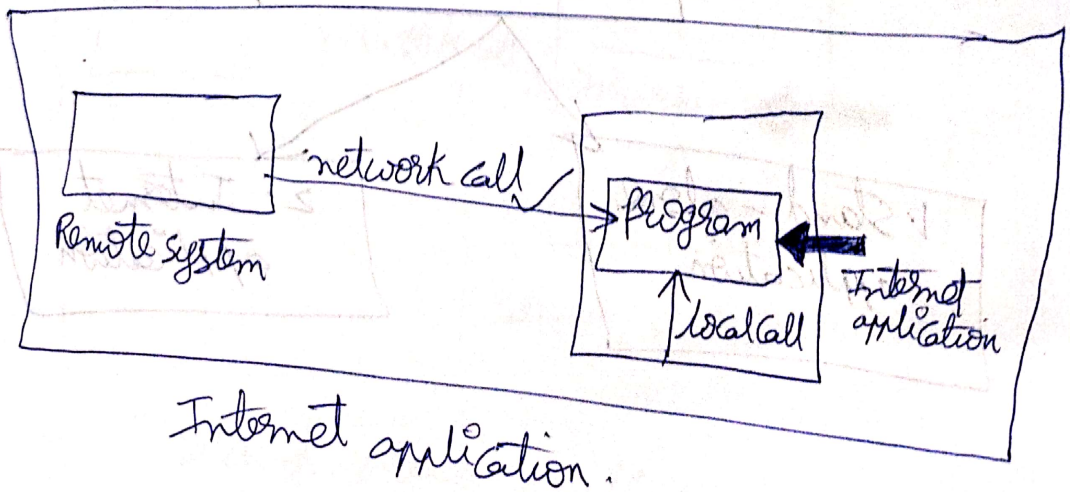


1. Stand-alone applications :- These can only be executed in local system with local call.



Stand-alone application

2. Internet applications :- An application that can be executed in local system with local call and also from remote computer via network call (request) is called internet application.



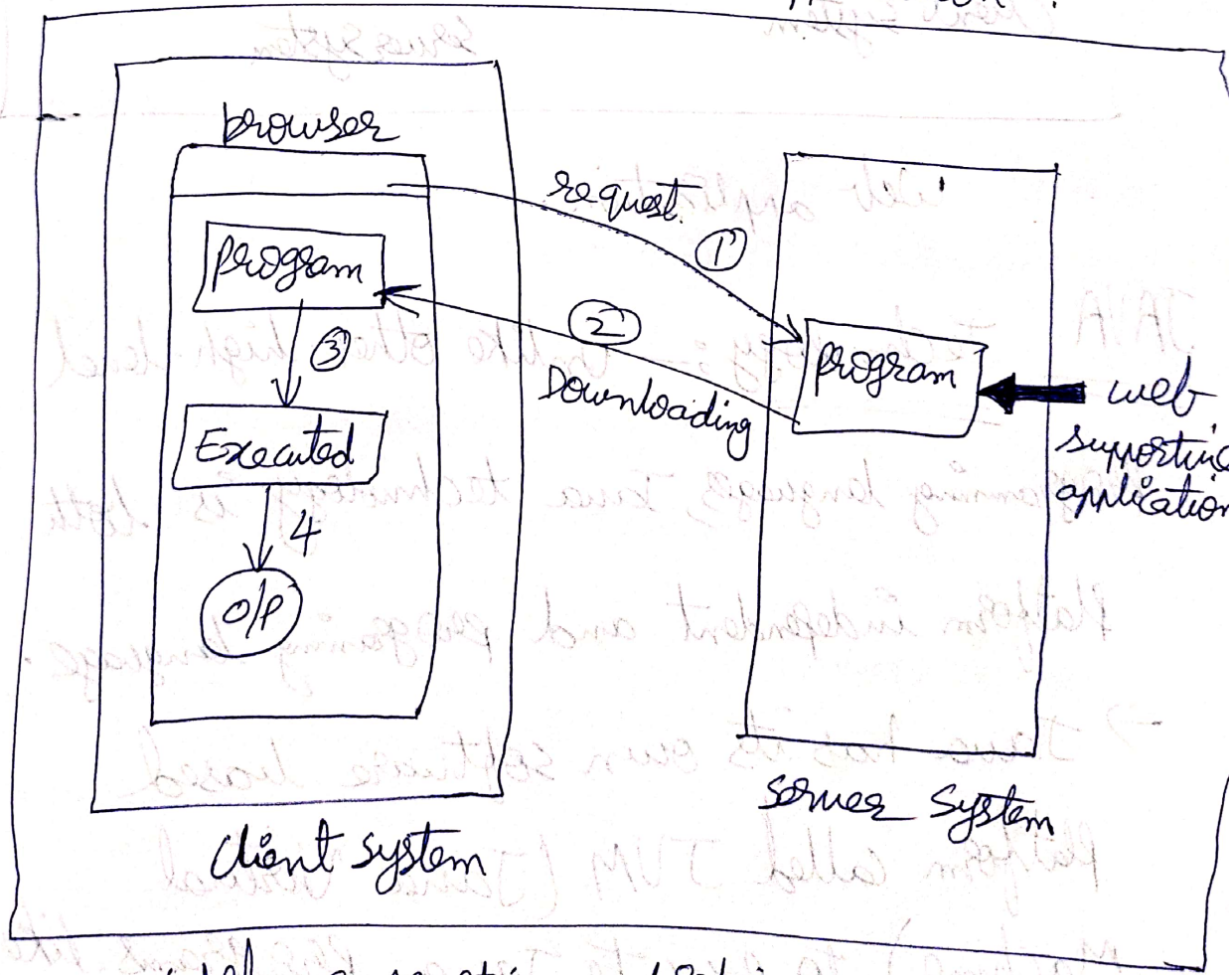
Internet application.

Types of Internet applications :-

⇒ we have two types of internet applications.

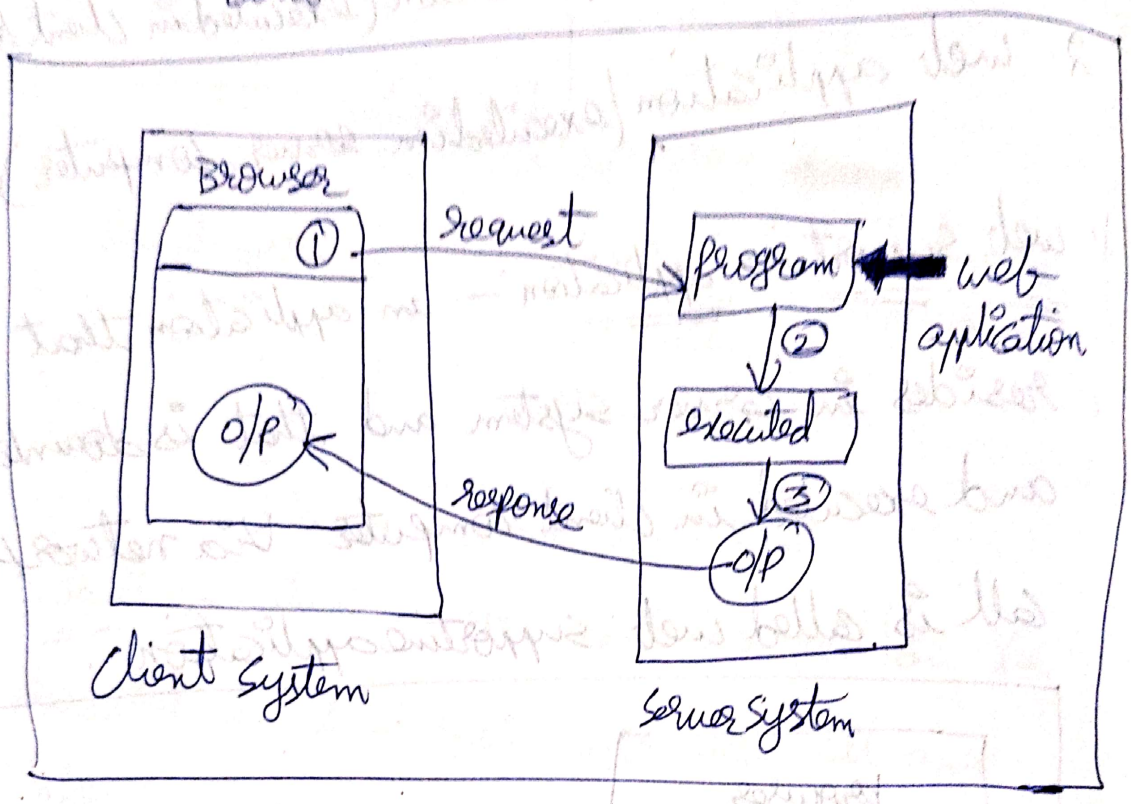
1. web supportive application (executed in client computer)
2. web application (executed in server computer).

1. web supportive application :- an application that resides in server system and that is downloaded and executed in client computer via network all is called web supportive application.



web supportive application.

2. web application :- it resides in server system and that is executed directly in server system via network call and sending response (output) back to client.



Web application.

JAVA Technology :- Unlike other high-level programming languages, Java technology is both platform independent and programing language.

→ Java has its own software based platform called JVM (Java Virtual Machine) to execute Java programs, like

C or C++ programs, Java programs are not directly executed by OS (operating system).

Platform:- it is a hardware or software environment in which programs are executed (runs).

Eg:- Computer platform is (operating system(OS) + Hardware devices).

Platform Dependent:- An application that is compiled in one operating system and run in that system only is called platform dependent. (C, C++).

Platform Independent:- the compiled code application is able to run in different operating system is called platform Independent (Java).

⇒ Java is platform independent programming language because Java program compiled code can run in all operating system (like windows, linux, solaris, etc).

Java Versions :-

1. Java Alpha & Beta (1995)
2. JDK 1.0 (1996)
3. JDK 1.1
4. J2SE 1.2
5. J2SE 1.3
6. J2SE 1.4
7. J2SE 5.0
8. Java SE 6

Sun Microsystem

9. Java SE 7
10. Java SE 8

Oracle

:

13 (2019)

→ In 2010 Oracle own all permissions / rights of Java.

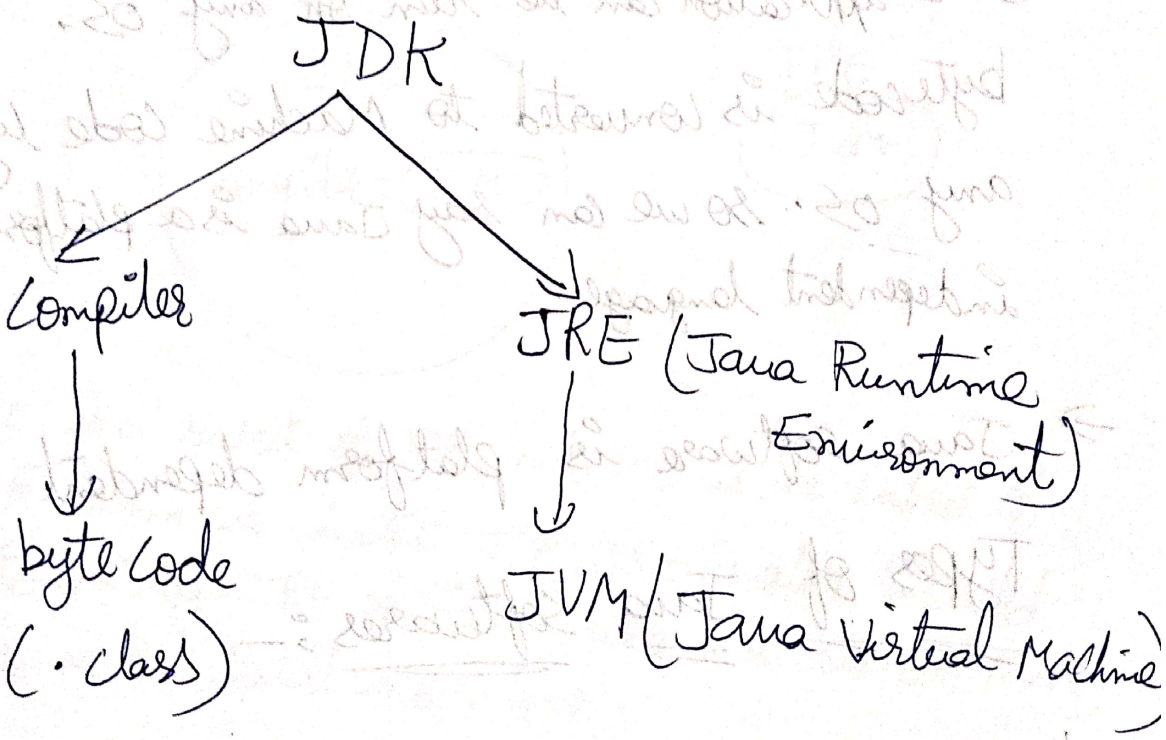
How Java works :-

How Java works:-

Java Development ToolKit (JDK):-

⇒ JDK provides an environment to develop, compile & execute a Java application.

⇒ JDK is a combination of compiler & JRE (Java Runtime Environment).



JRE:- it provides environment for executing Java applications.

JVM :- it executes Java program line-by-line

JVM contains platform dependent

↓

(operating system).

Every operating system has its own JVM (translator)

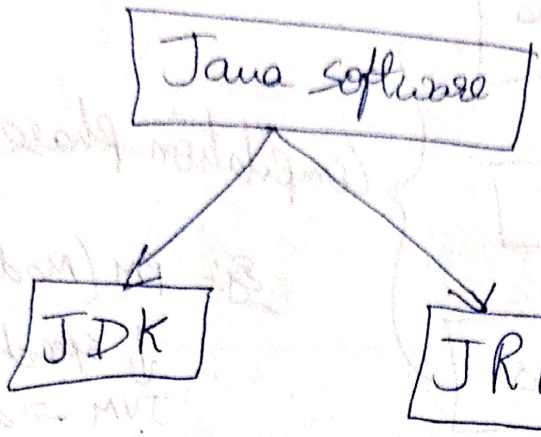
⇒ so Java is a Interpreter based language.

⇒ Java application can be run on any OS. bytecode is converted to Machine code by any OS. so we can say Java is a platform independent language.

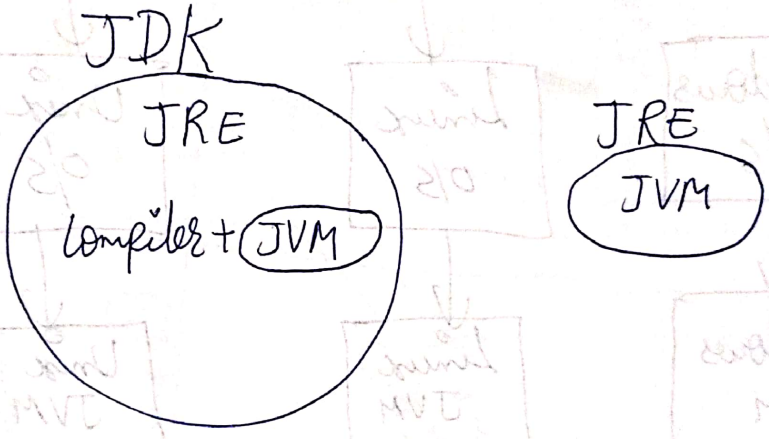
⇒ Java software is platform dependent.

Types of Java softwares :-

1. JDK – Java Development kit
2. JRE – Java Runtime Environment



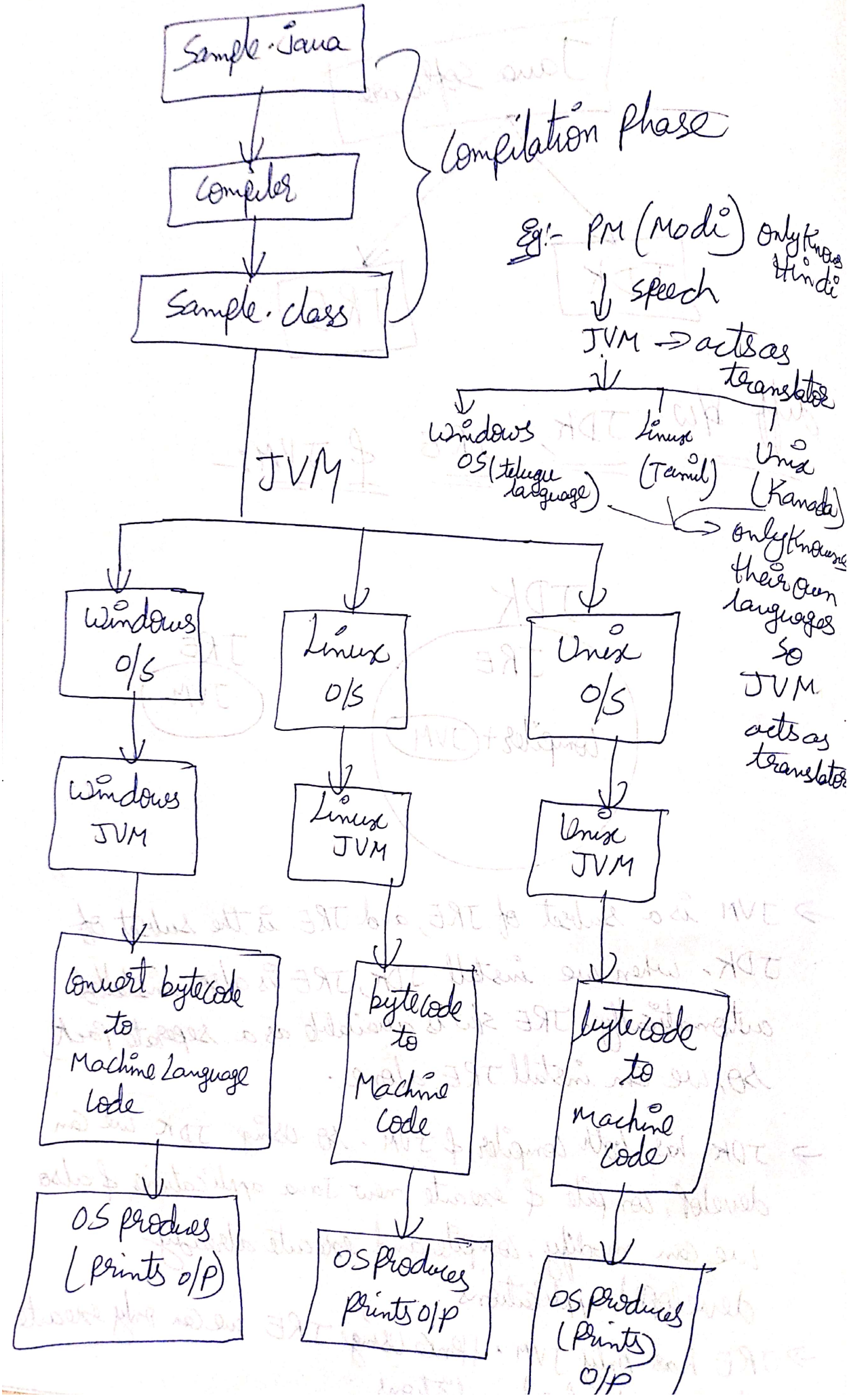
diff b/w JDK, JRE & JVM:-

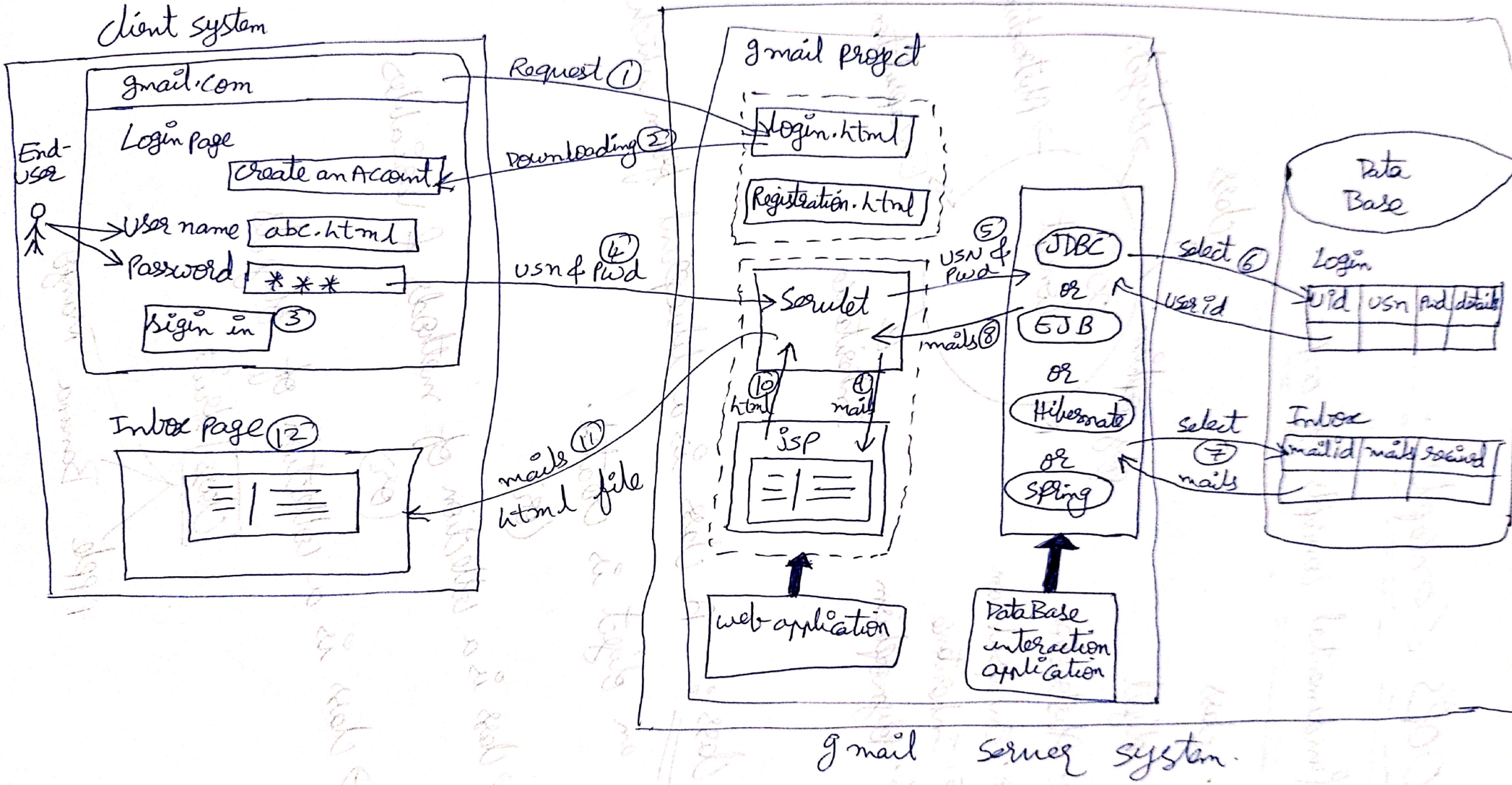


→ JVM is a subset of JRE, and JRE is the subset of JDK. when we install JDK, JRE is also installed automatically. JRE s/w is available as a seperate pack, so, we can install JRE alone.

→ JDK has both compiler & JVM. so using JDK we can develop, compile & execute new Java applications & also we can modify, compile and execute already developed applications.

→ JRE has only JVM. Hence using JRE we can only execute already developed applications.

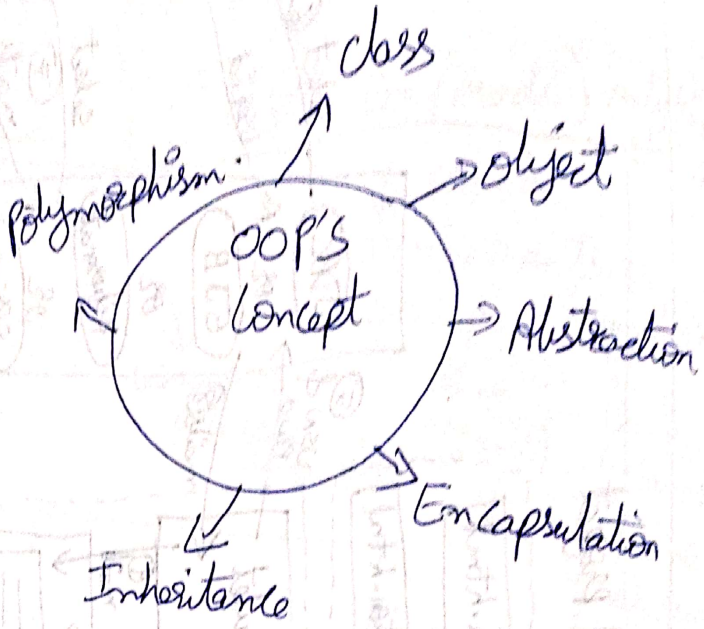




OOP's Concepts :-

Fundamental concepts of oop :-

1. class
2. object
3. Abstraction
4. Encapsulation
5. Inheritance
6. Polymorphism



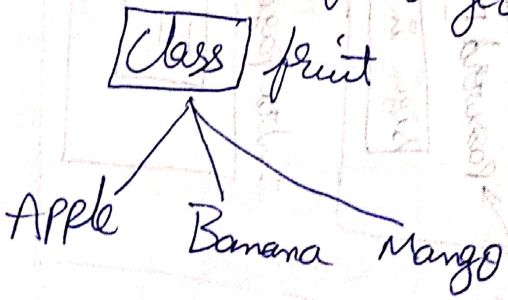
1. **Class** :- A class is a blueprint from which an object is created.

(or)

→ class is a collection of methods & variables

(or)

→ class is a collection of objects.



Syntax:-

```
class class_name
{
  fields/ variables  → Instance variable.
} methods
```

ex:-

```
class student
{
  int xno;
  int marks;
  Void total();
}
```

object :- object is an instance of a class.

→ memory for class variables are allocated only after object is created.

Bench Fan

Dog

Bike Pen Dog.

State	Behaviour	Identity
color	~~breed~~	(name)
name	barking, eating	name.

Syntax:- class_name objectname = new classname();

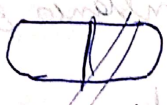
Ex:-
```
Student s = new Student();

s.rollno;
s.marks;
s.total();
```

→ hide (bulb)

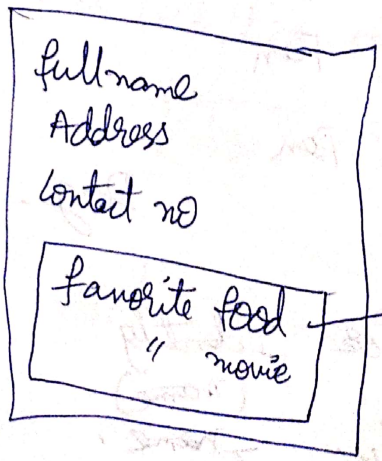
3. Abstraction :- it shows only essential information and hides unnecessary information.

→ the main purpose of abstraction is hiding the unnecessary details from the user.



Capsule.

⇒ Ex:- if we want to create a banking application



→ these are not need for banking application.

⇒ not all of the above information is required to create a banking application. So fetching necessary information & removing Unnecessary information is called abstraction.

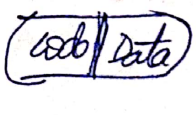
Types of Abstraction :— 2 types of abstractions

1. Data abstraction.
2. Process abstraction.

1. Data abstraction :— hiding the internal representation of data objects & exposing the necessary information.

2. Process abstraction :— hiding the implementation details of a process & providing simplified interface for executing it.

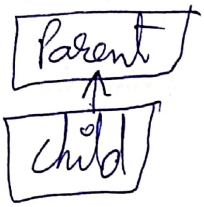
4. Encapsulation :— Combining or binding variables and methods under one unit called class.



capsule

5. Inheritance :- the process of creating a new class from old class. new class is called as derived class/sub class/child class and old class is called as Base class/super class/Parent class

⇒ the main advantage of this is Re-usability. Reusing the properties of one class to another class.



Eg:- human beings are inherited from parent or grand parent features.

Types of Inheritance:-

1. single Inheritance.
2. multiple "
3. multi-level "
4. Hierarchical "
5. Hybrid "

6. Polymorphism :- Representing one thing in many forms (ways) is called Polymorphism.

Poly = many morphs = forms.

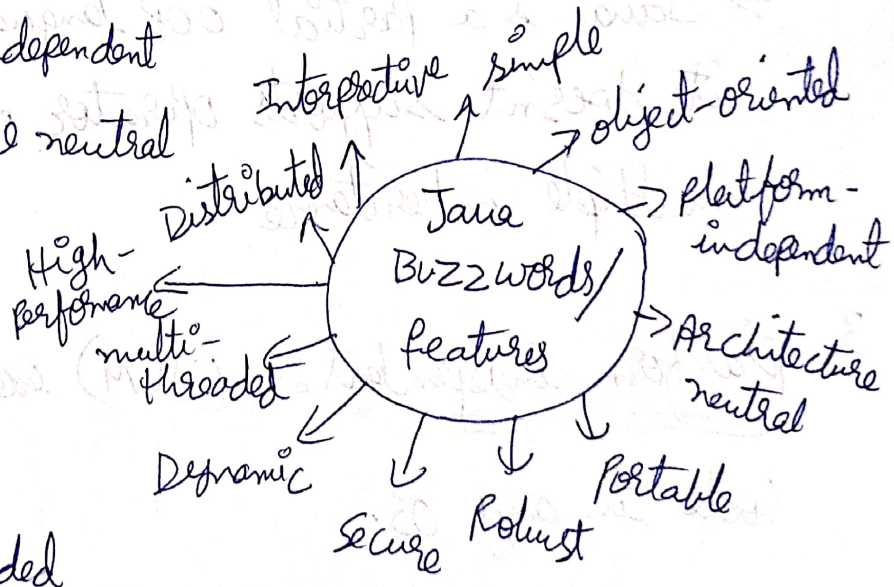
1. compile time :- which method should be called is decided during the compile time. Ex:- method overloading.

2. Runtime :- it executes at runtime.
Ex:- method overiding.

C. T2

⇒ Java BUZZwords / features of Java :-

1. simple
2. object-oriented
3. platform independent
4. Architecture neutral
5. Portable
6. Robust
7. secure
8. Dynamic
9. Multithreaded
10. Distributed
11. Interpretive
12. High performance.



1. Simple :- Java is a simple language, because many syntaxes are same as C languages.

→ many complex concepts are eliminated such as Pointers in C, and in method overloading & multiple inheritance in C++.

→ if an object is no longer used the memory occupied by that object is released by garbage collector.

→ Execution time of Java is less.

2. object - oriented :- execution is depends on object

→ Java is a partial oop language, because it doesn't supports operator overloading, multiple inheritance.

3. platform independent :- (JVM) executes byte code in any OS.

4. Architecture neutral :- we can execute Java program on any configuration, on any OS, on any hardware.

5. Portable :- we can move program from one hardware to another.

⇒ we can move Java program from one hardware to another hardware, from one OS to another OS, from one configuration to another without doing a single modification.

6. Robust :- means strong.

⇒ if a programming language is excellent in memory management and exception handling, we can say that language is Robust programming language.

⇒ In JAVA, memory management is done excellently. Dynamic memory management is there in Java (garbage collector).

7. Secure :- in JVM security manager is those for protection.
→ Java Authentication & authorization service (JAAS)
→ (JAAS) will take care about the security mechanisms.

8. Dynamic :- class file is loaded into memory during the runtime.
→ memory is allocated during execution time.
→ Java is a dynamic programming language.

9. Multi-threaded :- In this program is divided into no. of tasks. each task is called as a thread. Java can executes multiple threads simultaneously.

10. Distributed :- Java is a distributed language Java program can be executed in many systems simultaneously.
→ By Java we can develop network applications.

11. Interpretive :- Java is both compile & interpretive language.

12. High performance :- Java is a high performance language because it is a platform independent, simple, memory management etc.

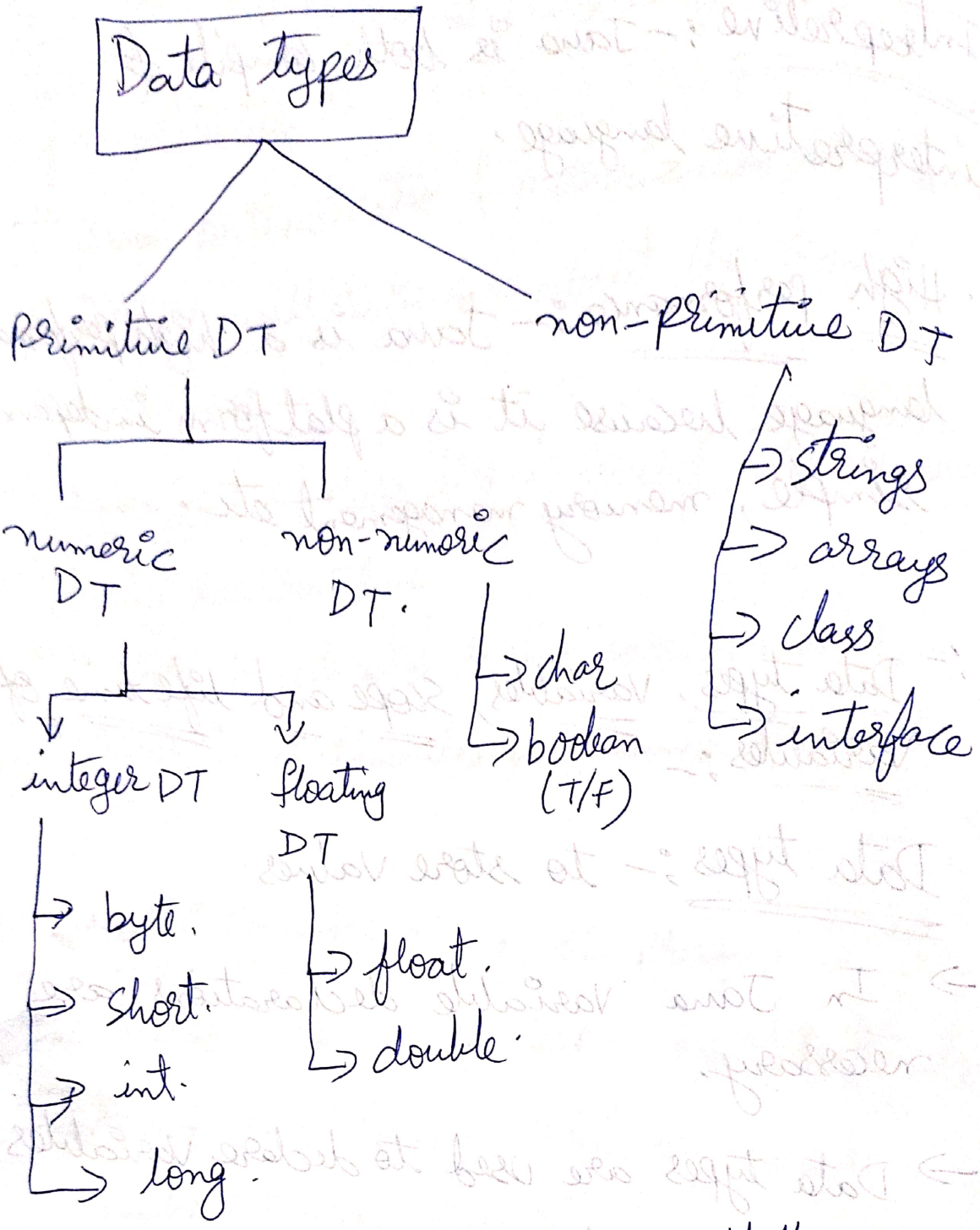
T/3 :- Data types, Variables, Scope and life time of Variables :-

Data types :- to store values.

→ In Java variable declarations are necessary.

→ Data types are used to declare variables to store values.

→



Datatype	size	Range	default Value	ex
byte	1	-128 to 127	0	byte a=10
short	2	-32768 to 32767	0	int a=250
int	4	-2^{31} to $2^{31}-1$	0	int a=657
long	4	-2^{63} to $2^{63}-1$	0	long a=15000
float	4	3.4e 38 to 3.4e -38	0.0	double a=1.25f
boolean	1 bit			

Need of Data types :-

Data types are used to store data temporarily in computer through a program.

⇒ In real world we have different types of data like integer, floating-point, character, boolean and string.

⇒ to store all these types of data in program to perform business required calculation and validations we must use data types concept.

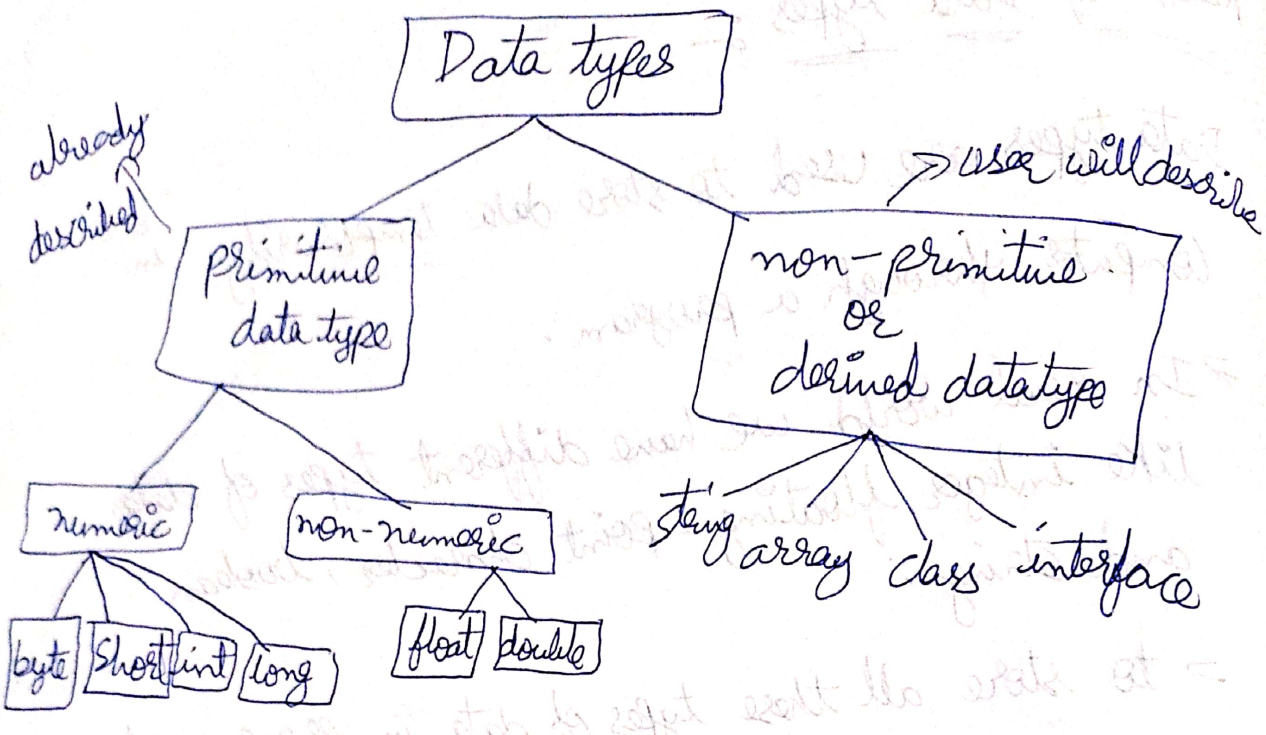
⇒ Definition of data type :-

Data type is something which gives information about

1. size of the memory location and range of data that can be accommodated inside that location.

2. Possible legal operations those can be performed on that location.

Eg:- on boolean data we cannot perform addition operation.



diff b/w float & double:-

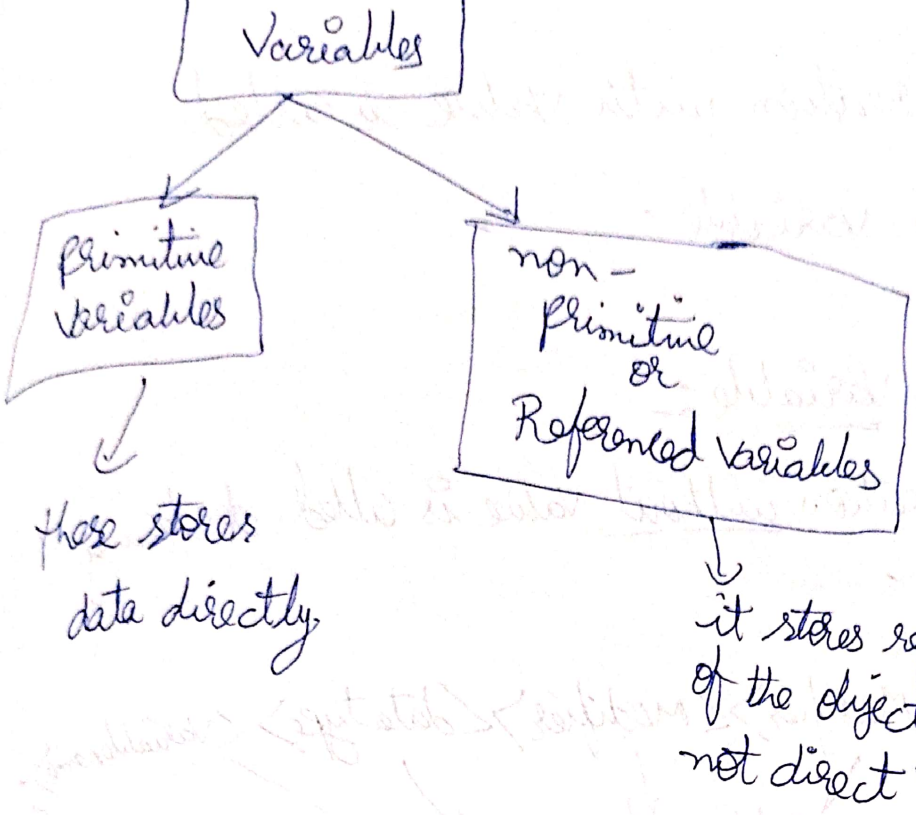
float:- it represent up to 7 digits accurately after decimal point;

double:- whereas double can represent upto 15 digits.

Variables:- Variable is a named memory location used to store data temporarily. During program execution we can modify that data.

→ A variable can be created by using Datatype.

→ As we have 2 types of datatypes, we can create 2 types of variables



→ Variable is a name that used to store a data value.

→ A variable may contain letters, digits & underscore symbol.

Rules:-

→ variable name should not start with digit.

→ Keywords should not be used as variable name.

→ variable name should not contain any special symbol except underscore.

Syntax: datatype <identifier>/

int a; variablename;

defining a variable:- (or)

<Accessibility modifier> <modifier> <datatype> <variablename>

Public static int x = 10;

Public static int x = 10;

→ Variable creation with value is called defining a variable.

Declaring a variable :-

Variable creation without value is called declaring a variable.

<AccessibilityModifier> <modifier> <datatype> <variablename> ;

public static int x ;

Scope and lifetime of a variable :-

Scope of variable :- variables are 3 types.

1. Local variable
2. Instance variable
3. static variable.

Life time of a variable :- Lifetime is the time period between variable creation & destruction.

Eg :- B.Tech course starts 4 years : it starts from 1-st year and automatically destruct after 4th year.

Scope :- it is the region in which it can accessible.

Eg :- 2-1 semester is scope. duration 5 or 6 months.

1. Local Variable :- A Variable is declared inside the body of the method is called local variable. you can use this variable only within that method not other method.

Ex:-
```
class A
{
  m1()
  {
    int a;
  }
  m2()
  {
  }
}
```

defination :- Local Variable gets Life only when its method is called & its variable creation statement is executed. it is destroyed automatically once method execution is completed.

Scope :- it is only within its method after its creation statement.

2. Instance Variable :- A Variable is declared inside the class but out side the body of method

Ex:-
```
class A
{
  int a;
  main()
  {
  }
  m2()
  {
  }
}
```

Instance or non-static :- def :-

Life time :- when object is created it is destroyed when object is destroyed. object is destroyed when its referenced variable is destroyed.

Scope :- it is the scope of object, object is available only if its referenced variable is available.

3. static variable :- A variable which is declared as static is called static variable.

Ex:-

```
class A
{
static int a;
m1()
{
}
m2()
{
}
public static void main()
{
}
}
```

⇒ within the class outside methods with static Keyword.

def:- Lifetime :- when class is loaded. it is destroyed either if class is unloaded from JVM or if JVM is destroyed.

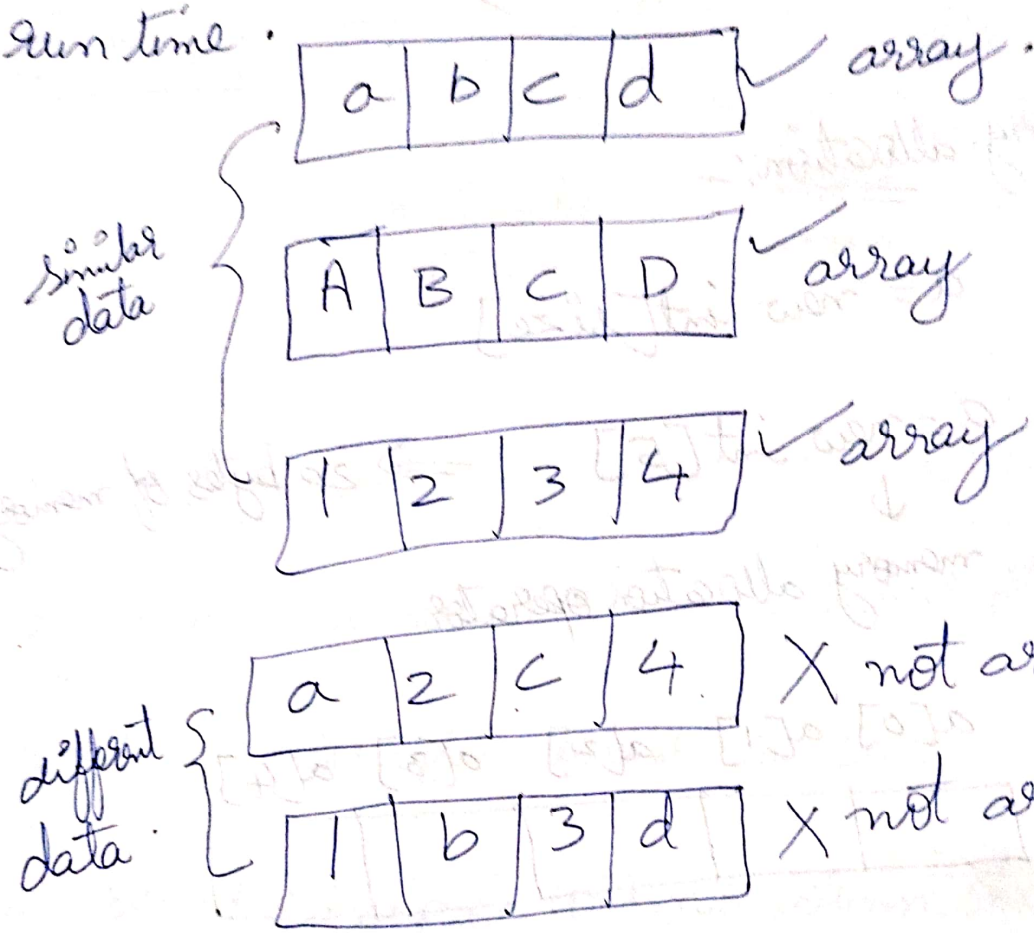
Scope :- is throughout class, & also in place where class is accessible. it means whenever class is available there static variable is available provided it is public.

Ex:-

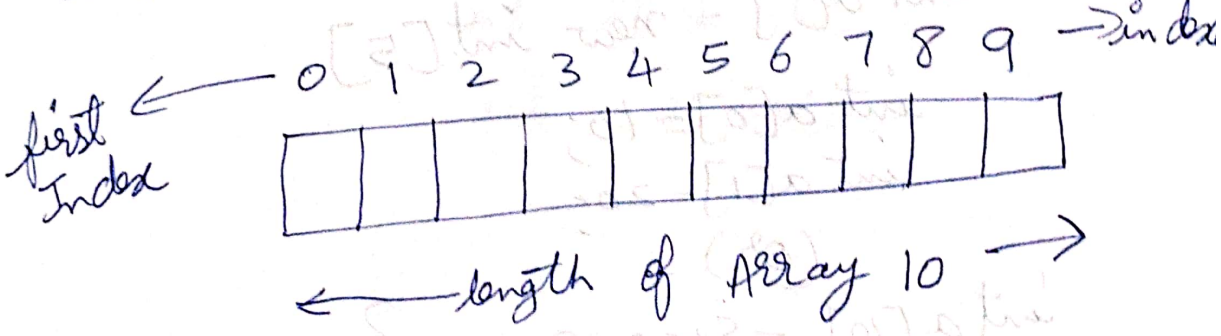
```
class A
{
int data=50; // instance variable
static int m=100; // static variable
void method()
{
int n=90; // local variable
}
} // end of class.
```

T4. Arrays, Operators, expressions :-

Array :- it is a collection of similar type of data, it is fixed in size, means you can't increase the size of array at run time.



→ it stores the value of on the basis of index value.



declaring or 1' Single Array or 1-Darray -

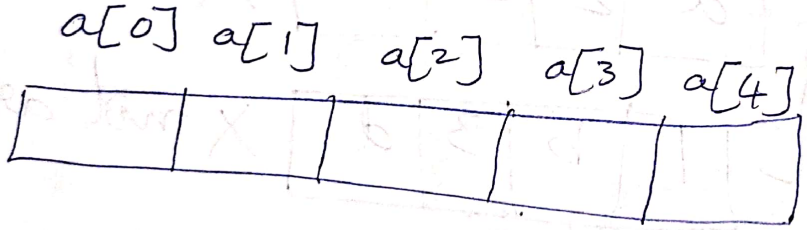
declaration of array :-

```
int a[];
int[] a;
int []a;
```

memory allocation :-

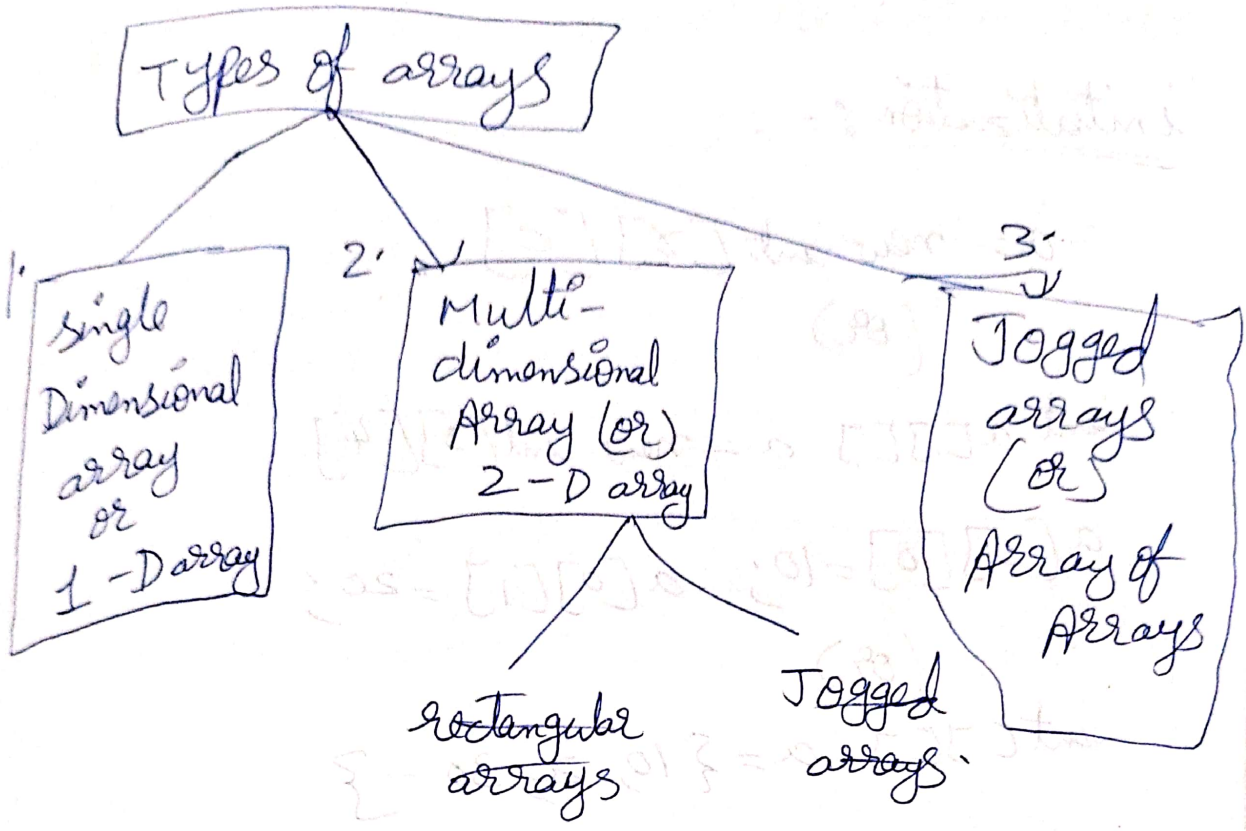
```
a = new int[size]
```

```
a = new int[5]   ⟹ 20 bytes of memory
↓
memory allocation operator
```



Combining declaration & initialization:

```
int a[] = new int[5]
int a[0]=10;
int a[1]=20;
(or)
int a[10] = {10,20,30 ... }
```



2. Multi-dimensional / 2-D array :-

2D arrays :- arrays contains 2 subscriptions.

1. first subscription contains row size.

2. second subscription contains column size.

Declaration :- int a[][];

int[][] a;

int [][]a;

initialization :-

```
a = new int[x][c]
```

(or)

```
int a[][] a = new int[3][4]
a[0][0]=10;  a[0][1]=20;
```

(or)

```
int[][] a={10, 20, 30 ... }
```

3. Jagged Arrays (or) Array of Arrays :-

It is an array with a variable number of columns in each row as different.

Ex :-

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & & \\ 1 & & & \\ 1 & 2 & 3 & \end{bmatrix}$$

Operators:- Various operators are used in java.

1. Arithematic :- +, -, *, /, %
2. Relational :- <, >, <=, >=, ==, !=
3. logical :- && (AND), || (OR)
4. Assignment :- =, +=, -=, *=, <<=, >>=
5. increment ++ } 5. Unary :- A++ ++A --A
6. decrement --

6. Ternary :- ?:
7. Bitwise :- & AND, OR |
8. Shift :- <<, >>

1. Arithematic :-

+ → Add → c = a+b;

- → sub → c = a-b;

* → mul → c = a*b;

/ → div → c = a/b;

% → mod → c = a%b;

2. Relational :-

< → less than → a<4

> → greater than → b >10

<= → less than equal to → b<= 10;

>= → greater than equal to → a >=5

== → equal to → x= 100

!= → not equal to → m! =8

3. logical :-

&& → And operator → 0 && 1

|| → OR operator → 0 || 1

4. Assignment :-

= → is assigned to → a=5

5. Unary :-

1. increment → ++ → increment by 1 → ++i or i++

2. decrement → -- → decrement by 1 → --i or i--

spcial operators:- → instance of - determining whether the object belongs to Particular class or not

∈

• dot :- instance

Ex:- Ganga is instance of River:
if the object Ganga is an object of class River.

• dot :- to access Variable and methods of a class object.

Ex:-

customer.name → // accessing the name of the customer.

customer.ID → accessing ID.

conditional operator :-

is " ?: " the syntax of conditional operator is

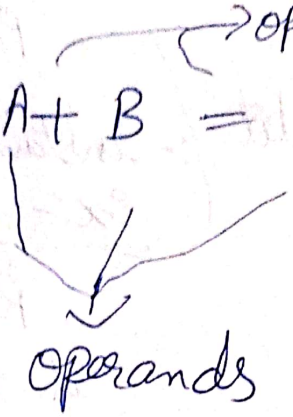
condition ? Expression 1 : expression 2

where exp 1 denotes the True condition
exp 2 denotes the False condition

Ex:- a > b ? true : false

Expressions:- ⇒ An Expression is a combination of variables, operators & methods.

operators & operands in specific manner.

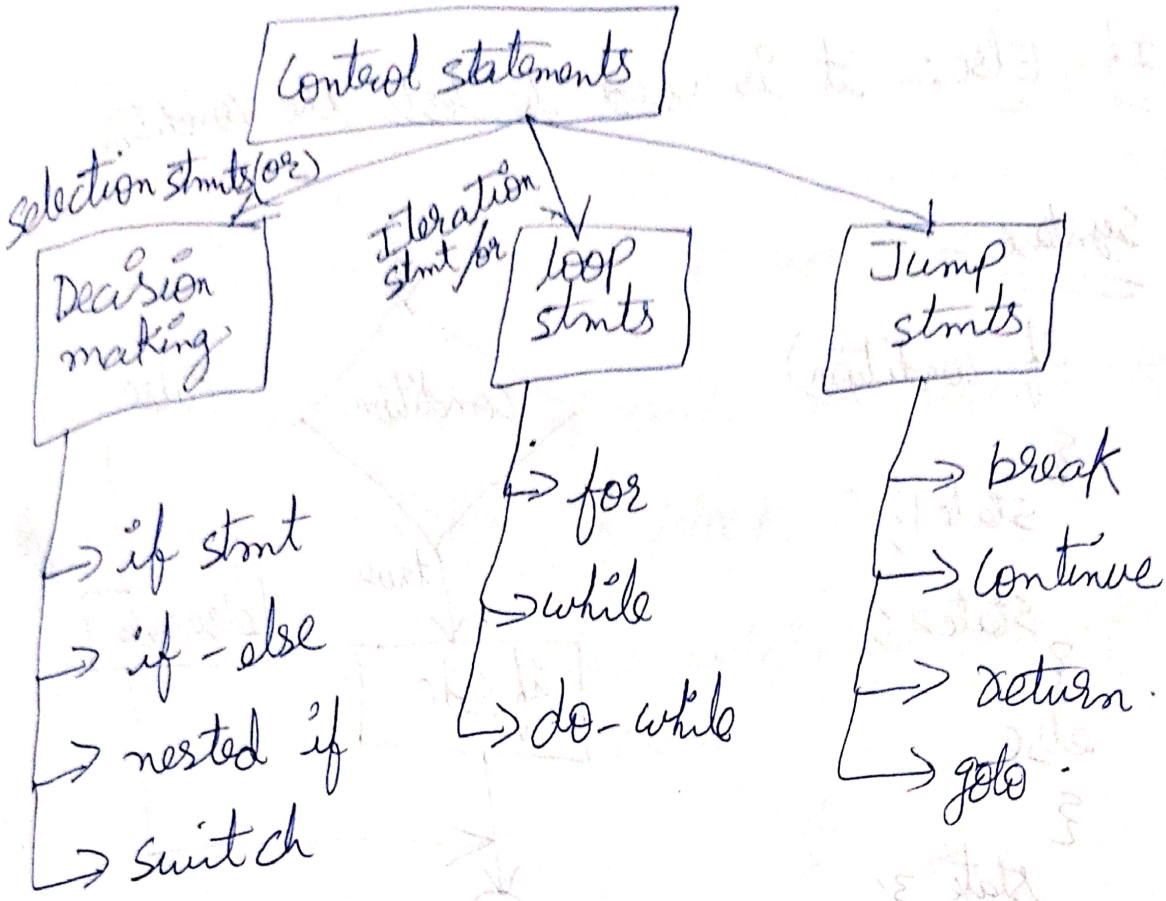


Ex:-

```
c = a + b * c; // Expression with
                  arithmatic operators
(a>b) && (b<c)   // Expression with
                  logical operators.
```

T5:- Control statements :- Type conversion & Casting :-

Control stmts :- are used to control the flow of code

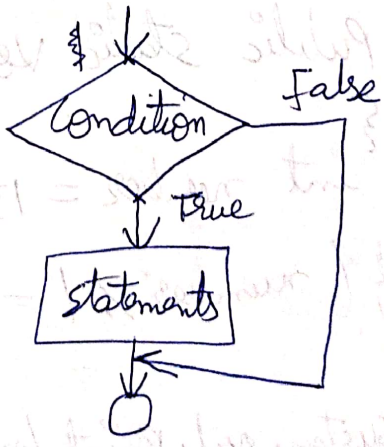


1. selection/Decision making statements:—

1. if:- it is used to test the condition.

Syntax:-

```
if (condition)
{
statement 1;
statement 2;
}
```



Ex:-

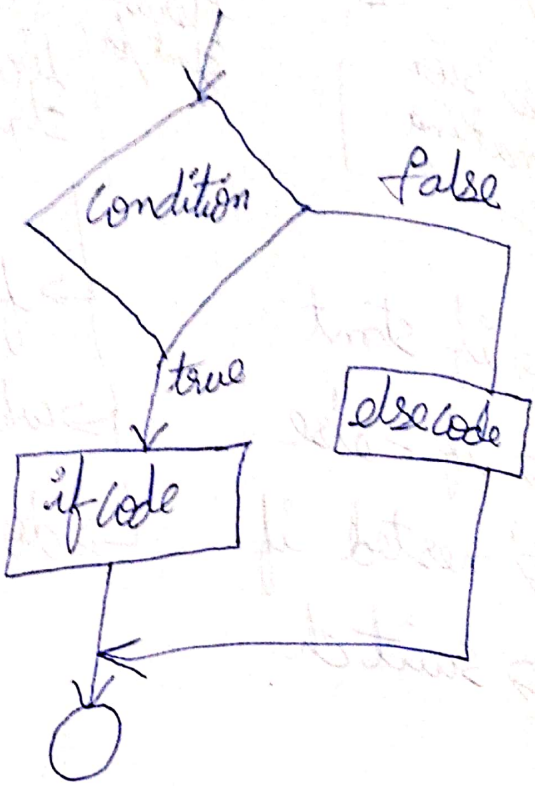
```
class
{
P.S.V.m(String
args[])
{
int age=20;
if (age>18)
{
S.O.Pln("eligible for
Voting");
}}}
```

⇒ if the condition is true, the compiler immediately execute the if block stmts, otherwise skip the condition & skip Pgm.

2. If-Else:- it is used to tests the condition

Syntax:-

```
if (condition)
{
  state1;
  state2;
}
else
{
  state 3;
  state 4;
}
```



Eg:- class if-else Example

```
{
Public static void main(String args[])
{
int number = 13;
if (number % 1 == 0)
{
System.out.println("even no.");
}
else
{ System.out.println("odd no.");
}}}
```

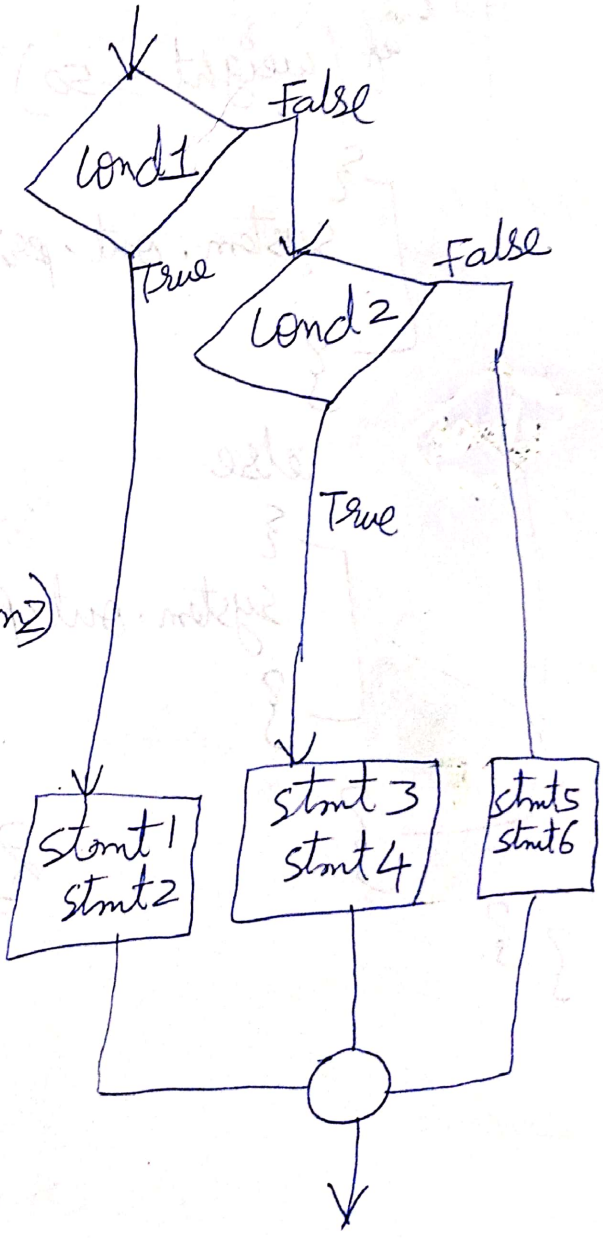
→ if the condition is true, the compiler will be execute the if code otherwise the compiler will execute else block code.

Nested if :- it represents the if block within another if block.

Nested class : A class within another class is called nested class.

Nested if : Syntax :-

```
if(condition1)
{
  statement 1;
  statement 2;
}
else if (condition2)
{
  state 3;
  state 4;
}
else
{
  state 5;
  state 6;
}
```



Ex:-

```
class nested_if_example
{
Public static void main(String args[])
{
int age=20, weight=8;
if (age>=18)
{
if (weight>50)
{
System.out.println("you are eligible to donate blood");
}
else
{
System.out.println("you are not eligible");
}
}
}
}
```

2:-Eg:-

```
if (a>b)
{
if (a>c)
a is greater
else
c is greater
}
```

→ if the condition is true, the compiler will execute if block, otherwise once again the compiler check the else if condition, it is true the compiler will Execute else if block otherwise elseblock will Execute.

Switch:- it Executes one statement from multiple conditions.

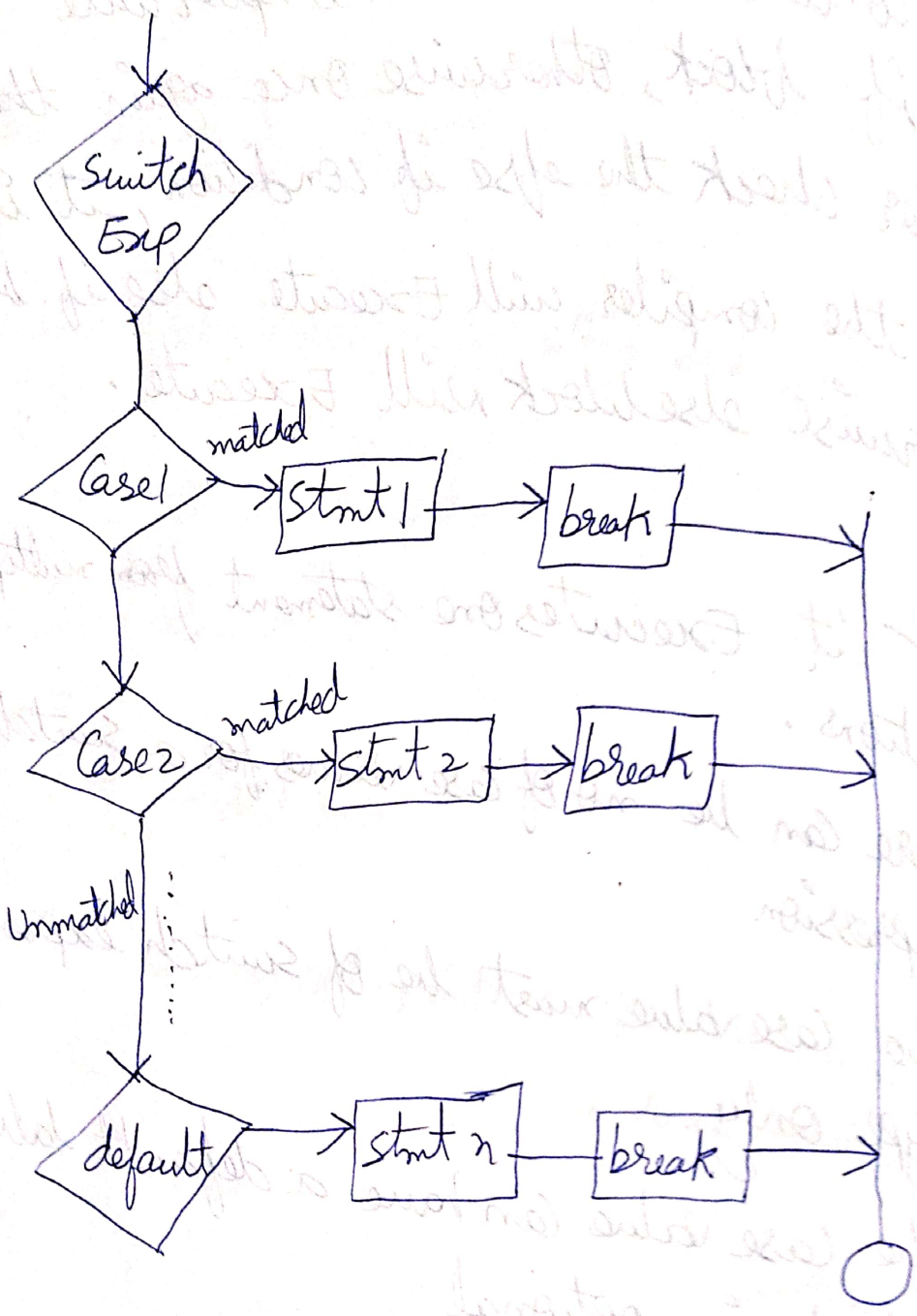
→ there can be no. of case values for a switch expression

→ the case value must be of switch expression type only

→ the case value can have a default label which is optional.

Syntax:-

```
switch (Expression)
{
  case value 1:
      statement 1;
      break;
  case value 2:
      stmt 2;
      break;
  :
  default:
}
```



→ the compiler checks the variable name and case constant if it is matches the compiler Execute the particular case constant statement otherwise break the case by default the compiler will execute default statement.

Ex:-

```
class switchExample
{
PSVM(string args[])
{
int no=20;
switch(no)
{
case 10: sopln("10");
         break;
case 20: sopln("20");
         break
case 30: sopln("30");
         break
   :
   :
default: sopln("not in 10, 20, or 30");
}}}
```

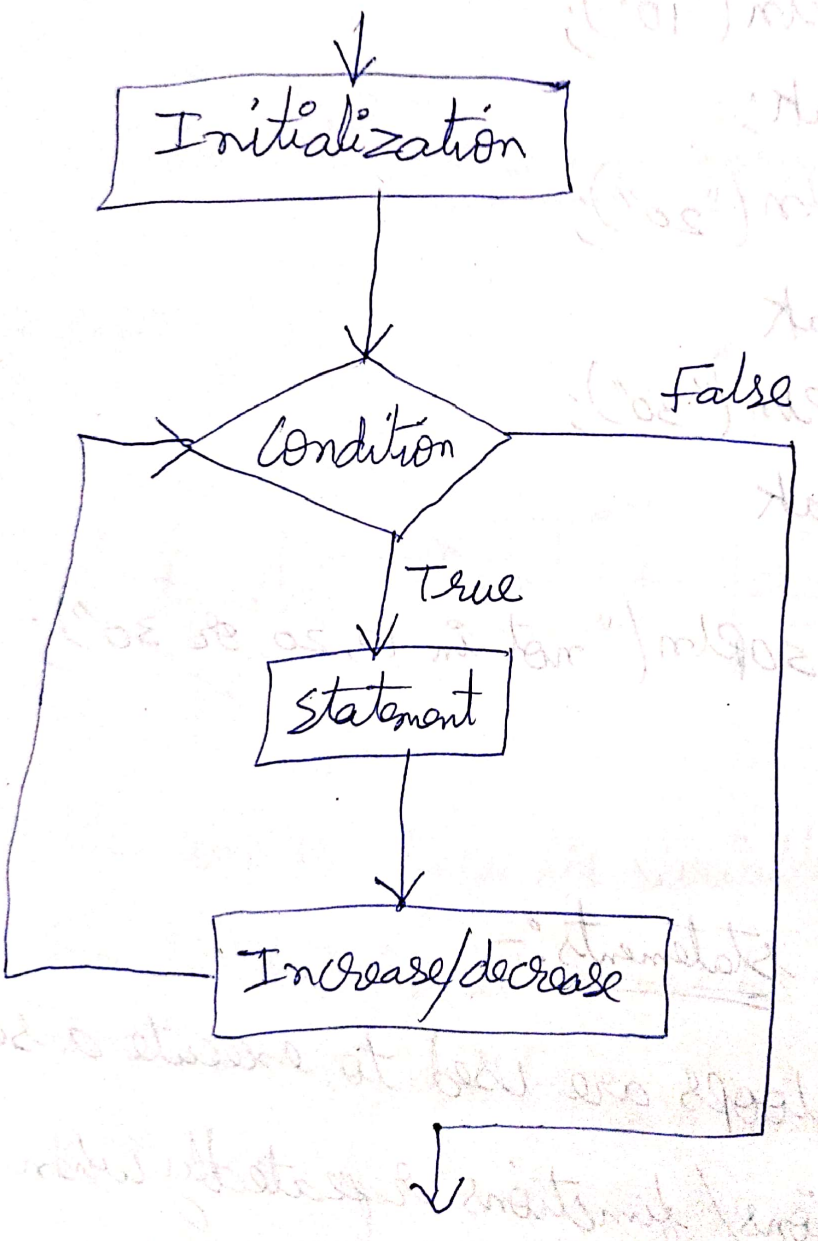
Iteration / loop statements:-

1. for loop:- loops are used to execute a set of instructions / functions repeatedly when some conditions become true.

→ to iterate a part of the program several times.

Syntax:-

```
for(initialization ; condition; increase/decrease)
{
  statement 1;
  statement 2;
}
```



⇒ first the compiler checks the initialization after then it will check the condition, the condition is true the compiler immediately executes inner block statement.

Ex:-

① class ForExample
{
psvm(String args[])
{
int i;
for(i=0; i<=10; i++)
{
sopln(i);
}}}

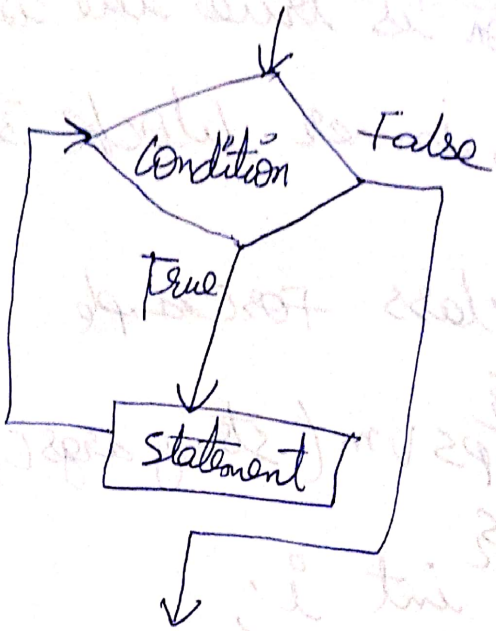
② class factorial
{
psvm(String args[])
{
int i, fact=1; n=5;
for(i=1; i<=5; i++)
{
fact = fact * i;
}
s.o.pln("factorial values in"; +fact).
}}

1 2 6 24

2. while :- it is used to iterate a part of the program several times.

Syntax :-

```
while (condition)
{
  statement 1;
  statement 2;
  increase/decrease;
}
```



Ex :- 1.

```
class whileEx
{
PSVM (String args[])
{ int i=1;
while (i<=10)
{
 sopln (i);
  i++;
}}}
```

⇒ the compiler first check the condition if the condition is true immediatly inner block statements executed otherwise the compiler skip the while loop.

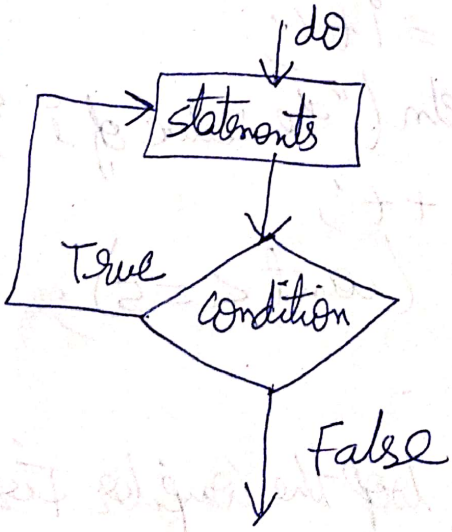
(2). class Factorial using while

```
{
  psvm (String args[])
  {
   int i = 1, fact = 1, no = 5;
   while (i < no)
   {
    fact = fact * i;
    i++;
   }
   s.o.pln ("Factorial value", +fact);
}}
```

3. do-while:- it is used to iterate a part of the program several times. In Java do-while is executed at least once because condition is checked after loop body.

Syntax:-

```
do
{
 stmt 1;
 stmt 2;
 incre/decre;
}
while (condition);
```



Ex: class dowhile

```
{
PSVM (String args[])
{
int i=1;
do
{
S.O.println(i);
i++;
}
while (i<=10);
}}
```

(2)

```
class dowhiledemo
{
PSVM(String args[])
{
int count=1, i=0;
do
{
i=i+1;
S.O.println("the value of i "+i);
count++;
} while (count<=5);
}}
```

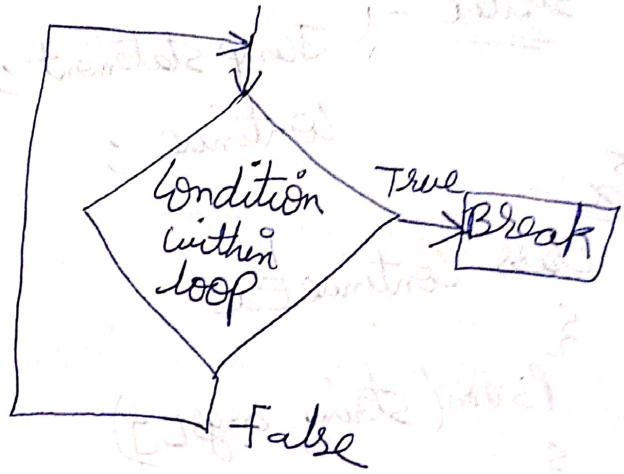
→ in dowhile loop the compiler first process the do statements after that it will check the condition is true the compiler will execute do statement.

Jumping statements:-

1. **Break:-** Break is a predefined keyboard, it is used to Break the current loop.

Syntax:-

```
Jump_statement;
break;
```



Ex:-

```
class BreakEx
{
PSVM(String args[])
{
for(int i=1;i<=10;i++)
{
  if(i==5)
  {
   break;
  }
  Sopln(i);
}}}
```

Eg:-

```
class BreakWhile
{
PSVM(String args[])
{
 int i=1;
 while(i<=10)
{
if(i==5)
{i++;
break;
}
S.o.pln(i);
i++
}}}
```

continue :- it is a keyword, used to continues the current loop. it continues the current flow of the program.

Syntax :-

```
Jump statement;
continue;
```

Ex :-

```
class ContinueEx
{
psvm(String args[])
{
for(int i=1; i<=10; i++)
{
  if(i==5)
  {
  continue; // it will skip
  }          the test stmt
  sopln(i);
}}}
```

1
2 6
3 7
4 8
9 10

this loop is continued when it reaches to 5.

eg? :-

```
class ContinueWhile
{
psvm(String args[])
{
 int i=1;
 while(i<=10)
 {
  if(i==5)
  {
   i++
  continue; // skip
 }          the test
 sopln(i);  cond
 i++;
}}}
```

Difference between break & continue:-

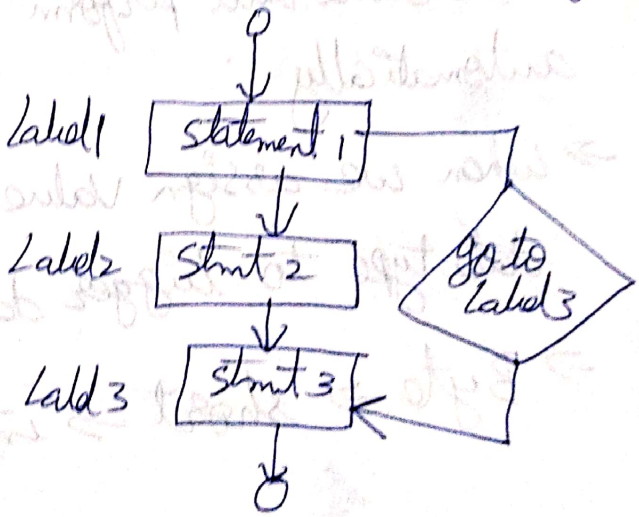
Break	Continue
1. the break is used to terminate the Execution.	1. the continue is not used to terminate the execution.
2. it breaks iteration	2. it skips the iteration
3. Break is used in loops & in switch	3. continue is used only in loop.

goto :- this stmt provides an un-conditional Jump from the goto to a labled stmt in the same Function

⇒ the compiler Jumps from one stmt to another stmt.

Syntax:-

```
goto label;
  ---
  ---
label: statement;
```



Ex:-

```
class goto
{
psvm( )
{
int count=0;
loop: count++;
soptln("count is"; +count);
while(count<=5)
{soptln(count);
goto: loop;
}}}
```

Return :- it is a Keyword, it indicates a function should return a value. the value is either positive or negative.

syntax :- Return Label;

Type conversion :- converting a value from one data type to another data type. it also called Implicit → it is also called automatic conversion
→ it is used when small data type is converted to large.
→ data loss is not there.
→ it doesn't convert numeric data type to character or boolean.

when you assign value of one data type to another. if the data types are compatible then Java will perform the conversion automatically.

→ when we assign value of a smaller data type to bigger data type.

→ Byte → short → int → long → float → double.

Eg:-
```
class Test
{
psvm()
{
 int i=100; //automatic type conversion
 long l=i;
 float f=l;
 soln(""+i); →100
 soln("",+l); →100
 soln("",+f); →100.0
}}
```

⇒ if the conversion is done explicitly by the user.

Type casting:- here we want to assign a value of larger data type to smaller data type.

⇒ at type conversion some data may be lost.

Double ⇒ float ⇒ long ⇒ int ⇒ short ⇒ byte.

Eg:-
```
class simple
{
 psvm(String args[])
 {
  float f=10.5f;
  // int a=f; // compile time error
     int a=(int)f;
     soln(f); →10.5
     soln(a); →10.
}}
```

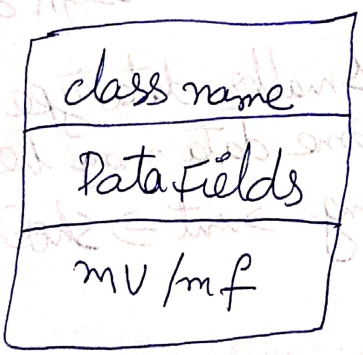
T6:- Concepts of classes, objects, simple Java

Program:-

Class:- Each class is a collection of data & the functions that manipulate the data.

→ the data components of class are called data fields and the function components are called member function and member variables.

→ it is represented as



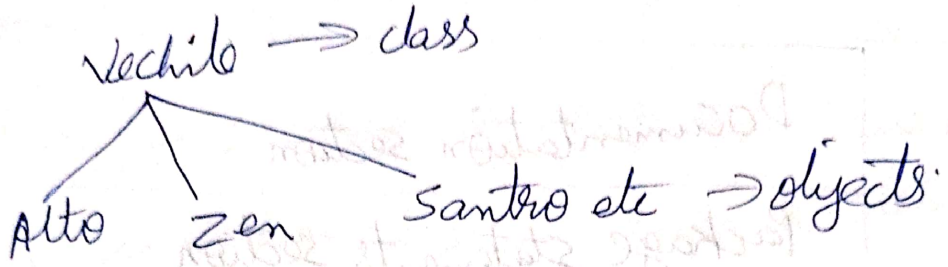
Syntax:-

```
class <class name>
{
  m.v;
  m.f;
}
```

→ in class C should be capital letter.

Object:- object is an instance of a class, the object represents the real world entity.

Eg:-



Syntax:-

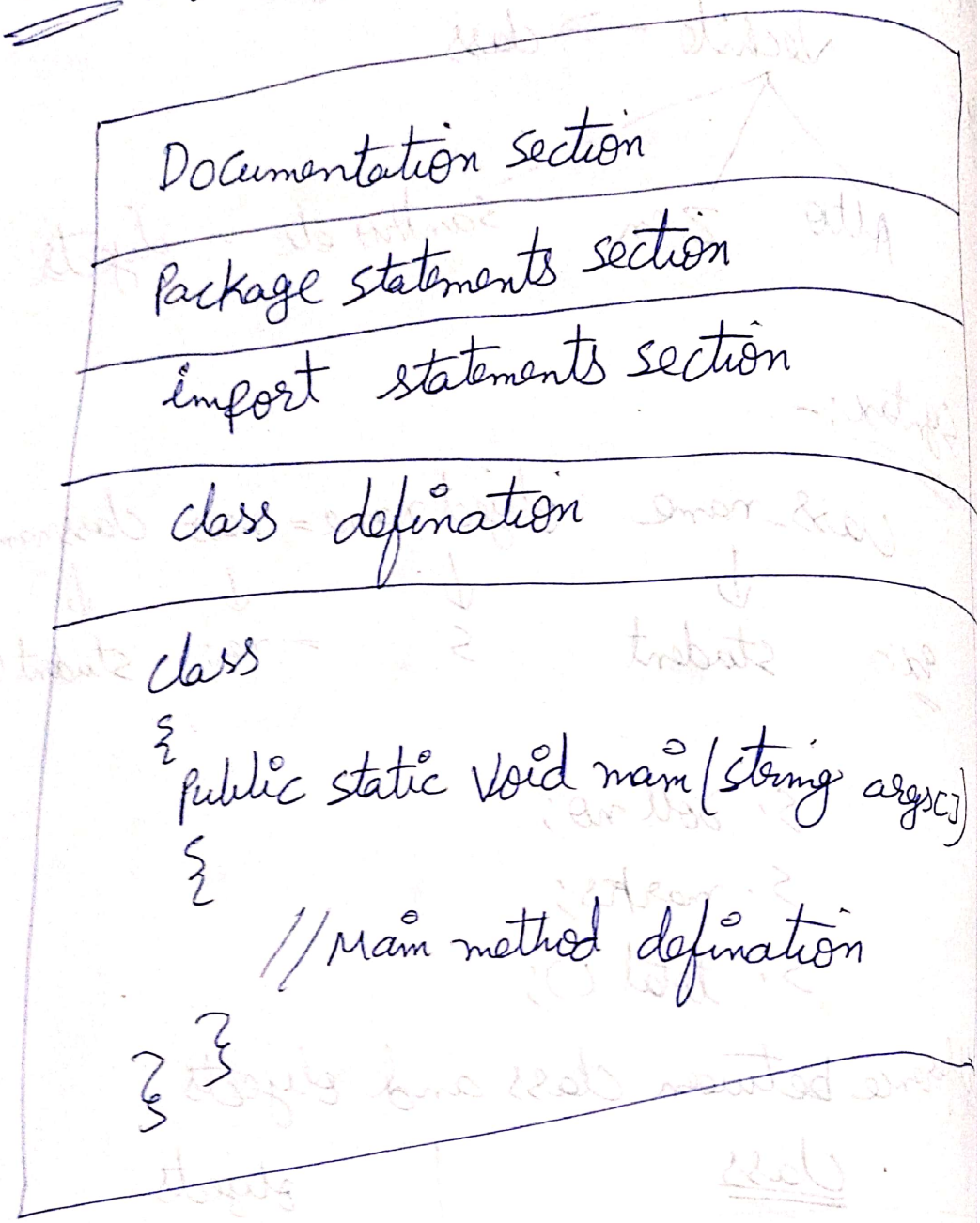
```
class_name  objectname = new classname();
    ↓          ↓      ↓       ↓
ex:- student    s   = new   student();
```

```
s.roll no;
s.marks;
s.total();
```

Difference between class and objects

Class	objects
1. it is collection of data	1. it is an instance of a class
2. it represents the basis for the object	2. it represents the real world entity
3. single class can create multiple objects	3. every object is created from one at only one class
4. ex:- class vechile	4. ex:- Alto, Zen, Santro etc

Structure of Java programs:-



Documentation section:- it provides the information about the source program, everything written in this section is written as comment

Package section :- it consists of the name of the Package by using the keyword "Package"

import section :- All the required Java API can be imported by the import statement.

class section :- this section contains the definition of the class. This class normally contains the data & method manipulating the data.

Writing simple Java program :-

```
/* this is my first Java program */
    class FirstProg
    {
    Public static Void main(String args[])
    {
    System.out.println("This is my first Java
                        program");
    }
    }
```

→ Any Java program can be written using simple text editor like notepad. the filename should be same as the name of the class.

→ the extension to the filename should be.

- Java (Firstprog.Java)

→ the output of this program can be generated on the command prompt using the commands.

⇒ Javac Filename.Java

⇒ Java Filename

Class:- the first statement in Java applicat
it contains collections of member variables
& member function.

⇒ the name of the class & name of your Java program should be same. the class defination should be within the curly brackets.

Public:- it is a Access specifier, it is used for Accessing the values anywhere.

static :- it indicates the compiler checks the main method only once.

void :- it can return a value

Main :- Main is a method or function

String args[] :- it indicates storing all types of arguments in array.

array :- storing the data values.

• (dot) :- it is used for Accessing the value from any ~~any~~ where.

out :- it indicates connected to output console.

println :- it is used for print the correct text.

17. Constructors, methods,

Constructor:- are used to create objects of a class.
it is a special member method.
it will automatically executed by the JVM
whenever an object is created for
placing user defined values inplace of
default values.

1. name is same as classname
2. it has no return type
3. it can't be final, abstract,
synchronized.

→ Constructors are mainly created for initializing the object. In this assigning user defined values at the time of allocation of memory.

Syntax:- Class Name
{
}

Eg:- Rectangle r = new Rectangle();
↓
Constructor

Adv:- a constructor eliminates placing the default values.

How constructor eliminate default values :-

Constructor are mainly used for eliminate default values by user defined values.

ex:-

```
class A
{
int a;
float b;
char ch;
string str;
boolean bl;
```

before using constructor → default values

int → 0
float → 0.0
char → one space
string → null
boolean → false

after using constructor :-

```
A()
{
a=10;
b=20
ch='m'
str="jana";
boolean = true;
}
}
```

10
20.0
m
null jana
False true

Constructors

ex:-
```
class Sum
{
int a,b;
Sum()
{
a=10;
b=20;
}
PSVm()
{
Sum s=new Sum();
c=a+b;
Sopln(c);
}}
```

Rules:-

1. it called automatically when the obj is created.
2. its name must be similar to name of the class
3. it should not return any value.

Methods :- it is a block of code which only runs when it is called.

⇒ the functions defined by the particular class.

def:- Method is a sub block of a class that is used for implementing logic of an object's operation.

Rule:- logic must be placed only inside a method not directly at class level. if we place logic at class level compiler throws error.

⇒ Method must be declared within class followed by paranthesis ( ).

Syntax:-

```
class classname
{
declaration of m.v;
declaration of method()
{
==
}
}
```

Ex:-

```
class myclass
{
static void mymethod()
{
==
}
}
```

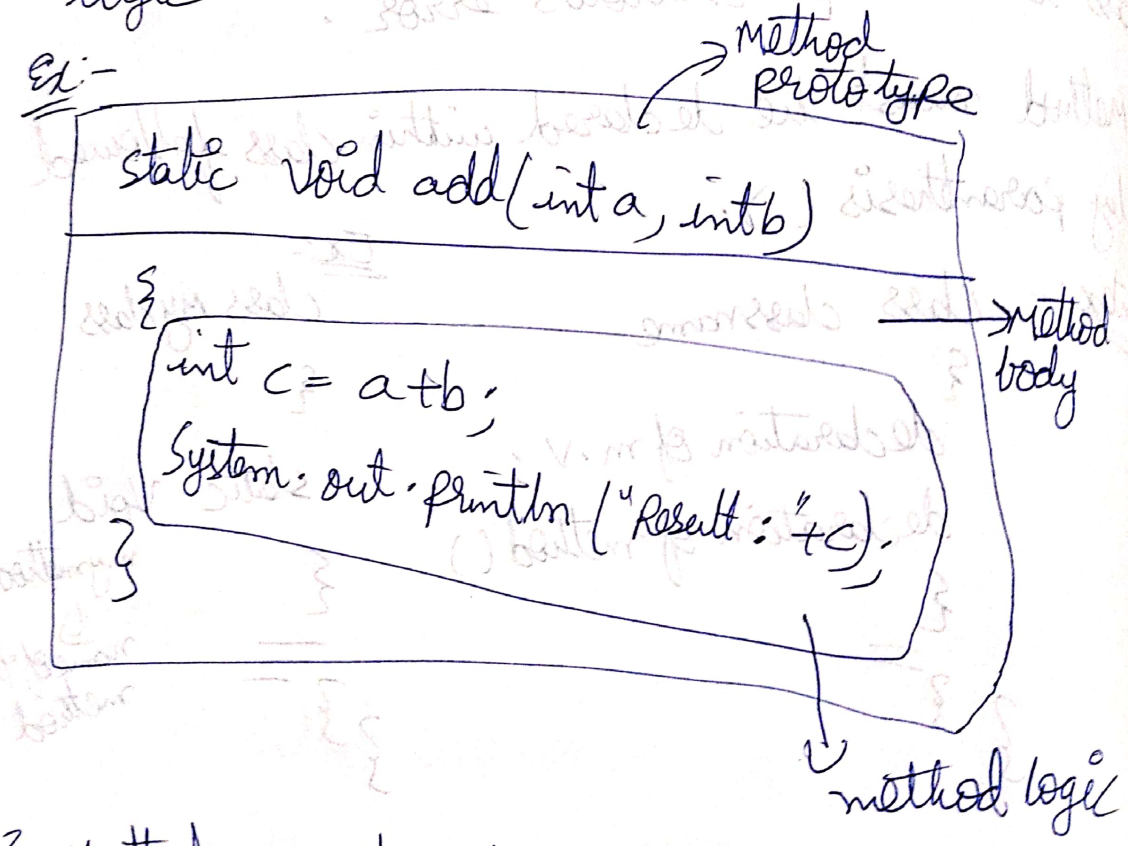
() name of the method

Method Terminology :-

1. Method prototype :- The head portion of the method is called method prototype.

2. Method body & logic :- The " { } " region is called method body, & the statements placed inside method body is called logic.

Ex :-



3. Method parameters & arguments :-

⇒ The variables declared in method Parenthesis " ( ) " are called parameters.

⇒ we can define method with ZERO to 'n' number of Parameters.

⇒ the input values passing to Parameters are called arguments.

Ex:-

```
Void add (int a, float b)   → Method Parameters.
{
}
add(50, 60.0f);   → method arguments.
```

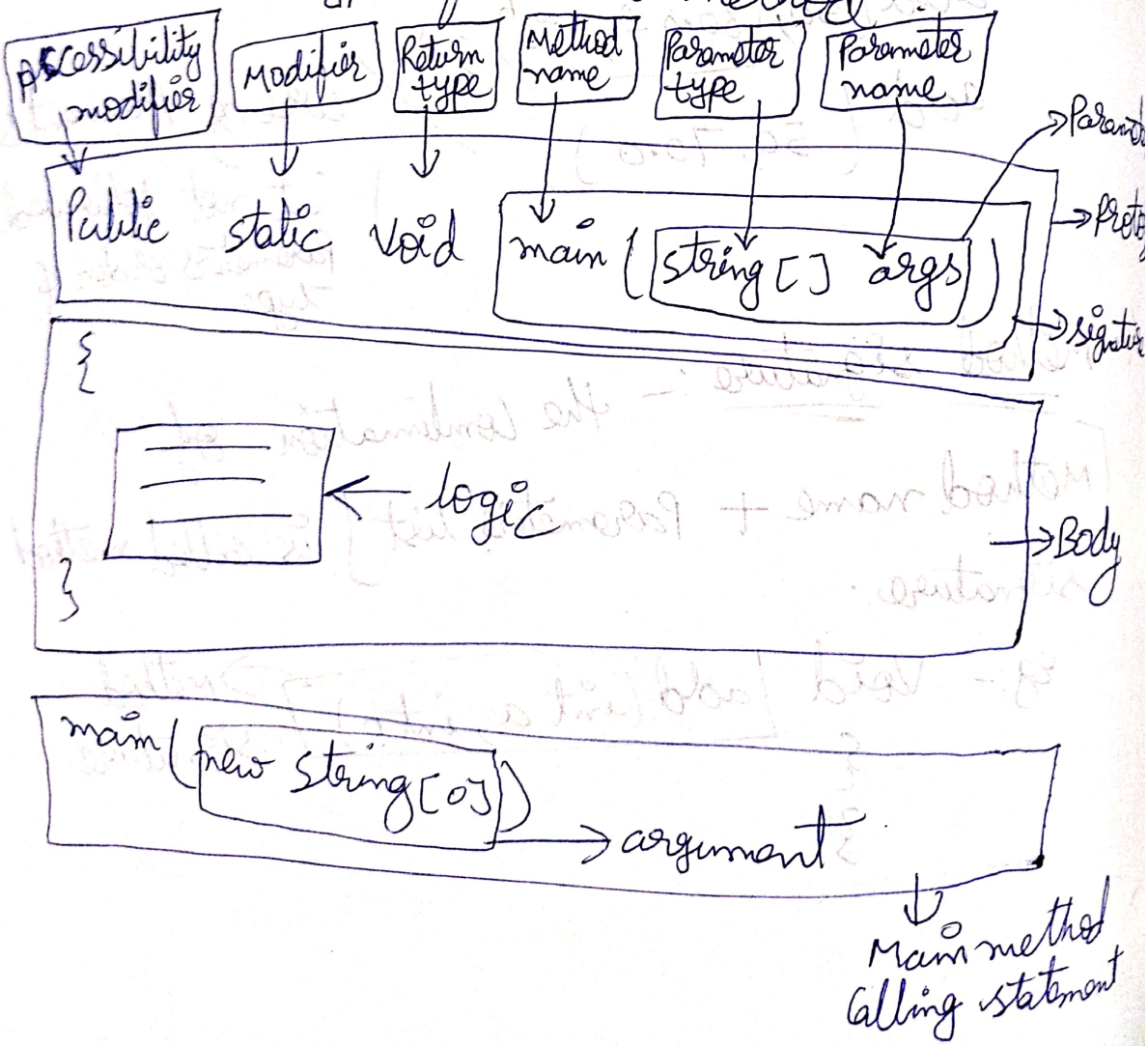
```
add(60.0f, 50);
add(50, 70.0);
```

✗ Wrong Arguments. it not followed Parameters order & type

4. Method signature :- the combination of [Method name + Parameters list] is called method signature.

```
Eg:- Void add(int a, int b)   → Method signature.
     {
     }
```

5. Method return type :- the data type keyword that is placed before methodname is called method return type. it tells the compiler, JVM & developer about the type of the variable is returned from this method after its execution. if we don't want to return value for a method we must use Void keyword as return type for that method.



Advantages:- A

there are 3 types of methods:-

Using → name of the method
→ Return type of the method
→ Parameter passed to the method.

1. Using method:-

```
class A
{int x=10;
void display()
{
S.O.Pln(x);
}
P.S.V.m(String args[])
{
A oa = new A();
oa.display(); →10
oa.x=200;
oa.display() →200
}}
```

Using Returntype:-

```
Class A
{int x=10;
int display()
{
S.O.Pln(x);
return x;
}PSVM(String args[])
{
A oa = new A();
oa.x=20;
int a=oa.display();
oa.x=200;
oa.display();
}}
```

3. Using parameter passing :-

```
class A
{int i;
int display(int x)
{ i=x;
return i;
}
P.S.V.m (String args[])
{
A oa=new A();
int c=a.display(100);
S.o.ln(c);
}}
```

Access control / Access specifiers / Access modifiers :-

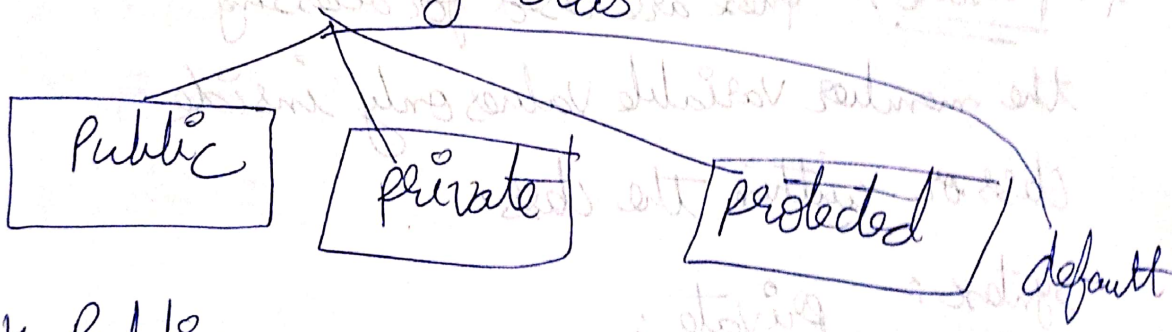
The keywords which define accessibility permissions are called access modifiers.

⇒ Access specifiers are applied before data members or methods. These are used to where to Access & where not to access.

⇒ Java supports 4 accessibility levels to define accessibility permissions at different levels.

1. only within class
2. only within package
3. outside the package but only in subclass by using the same subclass object
4. from all places of project.

⇒ we have 3 keywords.



1. Public
2. private
3. protected
4. default :. if we not mentioned any acc spf; Sys considered as default. if we declare a class as default, that class

1. Public :- it is used for accessing the member variables & member functions inside or outside class defination.

⇒ those members can be accessible from all places of Java application.

Syntax :- public :
statements ;

Note :- these code can be seen by all members / people / public.

2. private :- these are used for accessing the member variable values only inside class or within the class.

Syntax :- private :
statements ;

Note :- Our friends have permission to see the code.

3. protected :- these are used for accessing the member variables & member functions only inside the class.

Syntax :-

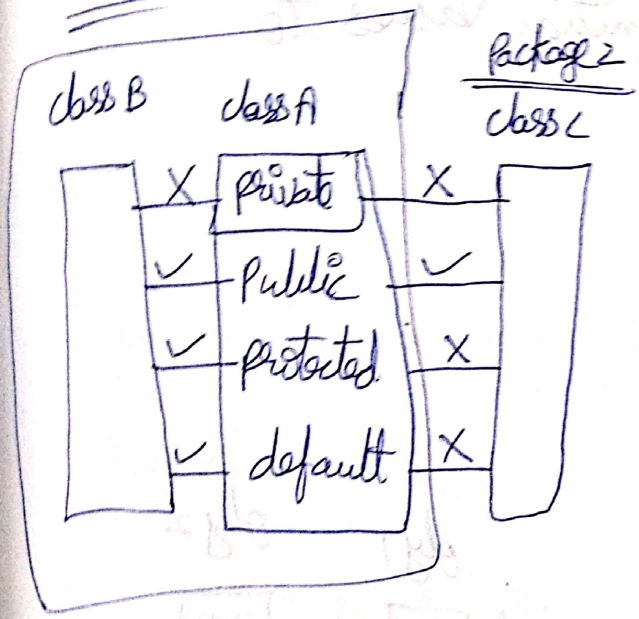
```
protected :
    statements;
```

note :- only we can see the code.

Ex :-

```
class Hello
{
private int a=20;
private void show()
{
 s.o.pln("HelloJava");
}
Public class demo
{
 PSVm()
{
Hello obj=new Hello
s.o.pln(obj.a);
obj.show();
}}
```

Package1



T8. This Keyword ; Garbage collection :

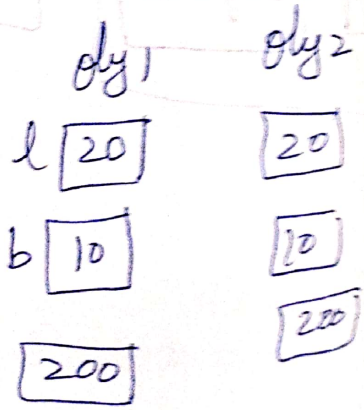
Constructors :- 2 types

1. default constructor (doesn't contains any arguments)
2. Parameterized

→ if any constructor is not defined in class, compiler will define default constructor impliatly & provide initial values to variables.

Ex :-

```
class Rectangle
{
double length;
double breadth;
Rectangle()
{
length = 20;
breadth = 10;
}
```



```
double area()
{ return length * breadth;
}
PSVm ( String args[])
{
Rectangle obj1 = new Rectangle();
Rectangle obj2 = new Rectangle();

S.O.P("Area of obj1 = " + obj1.area());
S.O.P("Area of obj2 = " + obj2.area());
}
}
```

O/P is same for all objects.

Parameterized constructor :- constructor with parameters are called parameterized constructor.

Eg:-

```
class Rectangle
{
  double length;
  double breadth;
Rectangle(double l, double b)
```

```
{
  length = l;
  breadth = b;
}
double area()
{
  return length * breadth;
}
PSVm (String args[])
{
Rectangle r1 = new Rectangle(10,20);
    // r2 = //  //  (30,40);
S.O.P("area of r1" + r1.area());
S.O.P("area of r2" + r2.area());
}
}
```

this Keyword:- it refers current object inside a method or constructor.

Ex:-
```
class A
{
}
A x = new A();
```

⇒ for every class a unique reference id is generated that unique reference id is refered by object. this keyword is also refer the reference id of the class.

Ex:-
```
class A
{
psvm()
{
  A x = new A();
  s.o.p(x);
}
}
```

o/p
01843

```
classA
{
void show()
{
s.o.p(this);
}
psvm()
{
A x = new A();
s.o.p(x);
x.show();
}
```

01843 → x
01843 → this

⇒ this is used to call defined default constructor as well as parameterized constructor.

this is a keyword used for accessing the instance values inside the constructor.

Syntax :- this.variablename ;

1. we cannot use this keyword in static methods it leads to compile time error; since it is non-static variable.

```
class ex
{
  int x = 10;
  int y = 20;
  void m1()
  {
    System.out.println(this.x);
    System.out.println(this.y);
  }
  Public static void main(String[] args)
  {
    System.out.println(this.x);
    System.out.println(this.y);
  }
}
```

Garbage collector :- In Java when an object is no longer in use or when an object is un-referenced then garbage collector automatically destroys the object & release memory at that object.

Overloading :- it is the concept of defining multiple methods with the same name but with different signatures.

Parameter Passing :- there are 2 different ways to pass parameter to methods.

1. pass by value

```
A() ⇒ caller method
{
  B() ⇒ called method
}
B()
{
}
```

2. call by reference: changing of called methods will not effect the calling method.

call by reference :- objects are passed as a reference.

Recursion :- is a process in which a function calls itself directly or indirectly.

Exploring string class :-

String :- collection of characters.

java.lang Package should be used.

declaring string :-

a) String s1;
   s1 = "hello";

b) String s1 = "hello";

String s1 = new String ("hello");

methods of string class :-

1. concat() :- concatenate 2 strings

Eg:- String s1 = "hello";
String s2 = "world";
String s3 = s1.concat(s2);

o/p :- helloworld

Exploring string class:-

String :- is a collection of characters.

Exploring string class:- the string class supports several constructors. to create an empty string, you call the default constructor. Eg:- string s= new string();

declaring string:- a) string s1;
s1 = "hello";

b). string s1="hello";
string s1= new string("hello");

methods of string class:-

1. concat():- used to concatenate 2 strings.

Eg:- string s1= "hello";
string s2 = "world";
string s3= s1.concat(s2);

O/P:- hello world.

2. length() ⇒ display length of the string.

Syn:- int length();

Eg:- string s1= "hello"; →5
int len = s1.length();

O/P:- 5.

3. charAt():- it return a character at specified position.

Syn:- char charAt(int i);

Ex:- string s= "hello";
char ch= s.charAt(1);

O/P:- e

A1, A3, A4
B1, B4, B7, B9
T07, 10.

4. strcmp():- it compress 2 strings. if 2 strings are equal it returns 0.
if s1>s2 +ve
if s1<s2 -ve

Eg:- s1 = Hi
Hi it returns 0 value

O/P:- 0.

Parameter passing:- There are 2 ways that to pass parameter to methods.

- Call-by Value
- Call by reference.

1. Call-by-Value:- this copies the value of an argument into the format parameter. changes made to the parameter will have no effect on the argument.

2. Call-by-reference:- a reference to an argument is passed to the parameter. changes made to parameter will effect the argument.

name → call directly Id number → reference.

```
A()  → caller method
{
 B()  → called method
}

B()
{
}
```

Recursion:- it is a process in which a function calls itself directly or indirectly.

Ex:- Pgm to print sum of n 10 no's:-

```
class recursion
{ PSVM
  { int result = sum(10);
    S.O.P(result);
  }
  PS int sum(int k)
  { if (k > 0)
    { return k + sum(k-1);
    }
    else return 0;
  }
}
```