⇒ Java

→ Difference b/w java and C++:

Note: C++ and java are both the object oriented. but the main difference is:-

→ java

1) Due to byte code, it is platform independent.
Note: Means while compiling it creates the dot file (Byte code) This bytcode (dot file) we can run in any system.

2) Compiled and interpreted language. (by giving command javac) we compile the program and then dot class file then write java.

3) Java does not support the Pointer.
What is Pointer?
Pointer is the special variable which store the other variables adress.

4) Java does not include structure and Union. Note: struct and Union are the both different types of data types

5) what is struck key word:
A structure is a way to group different data types under a single name.
In summary:- struct is a keywood used to declare a structure in C and C++, allowing you to define custom data types that can hold multiple variable with different data types.

→ C++

1) It is platform dependent. 2 compiled language.

3) C++ supports Pointer.

4) it includes structure and union.

the struct keyword explicitly.

The struct and class keywords are often interchangeable in C++.
with the only difference being the default access level (public for "struct" private for "class")

syntax:

```
struct Person{
  std::string name;
  int age;
  float height;
};
```

Java

=> it does not support multiple inheritance.

C++

=> it support M. Inheritance.

=> Java includes automatic garbage collection.

=> it requires explicit memory management

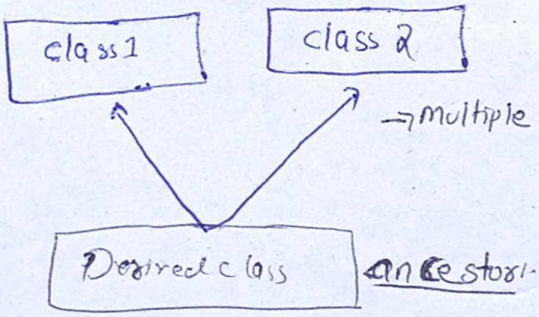
=> Does not support operator overloading.

=> it supports operator overloading.

=> There is no hadder file

=> header file in C++.

=>



→ Java:
It is the heigh level programing Language that is based on the object oriented programing like C and C++, but "C" ~~++~~ doe s not support the OOP.

→ Note: Java has its own run time enviromat called jvm which is used to any where code of Java if it is installed in your computter.

⇒ developed by Jamens Gosling, and his team May, 19, 1995 under the company sun Microsystems. In 1991.

→ Oak (first name of Java)

⇒ now a days (Oracale has the licence of Java) in 2010.

→ Features of Java:- ⇒ it is secure because of JVM.
JVM ensure that is any program perform the unsafe operation if then instead of performing Instruction, it will fetch the data, then such kind of unsafe operation it can not performed.
means, Jvm provide security. it has security mager layer which provide the programes, that untrusted code does not manage to access some Apis and features of the platform.

⇒ Java has access modifiers.

⇒ Jam has Exception handling and it has own memory managmant

→ we say Java is as a robust Language because we can handle even smallest exception.

→ use of Java:
⇒ Android Applications. 99% Applications are made through java. Android has its own Jvm called (Dvm in android) Delvic virtual machine.

2) Financial Sonme industries for huge securities.
3) Java web Applications.
4) Embeded systems like, routes, ATM etc.

→ JDK: (Java Development Kit) for any programer run we need JDK, then we can execute through editors or CMD Command or IDE like (Netbeans)

⇒ In c Drive we check. Inside JDK have different tools like Java compiler, Javac etc. All Development tools. and In JRE All predefined classes and Libraries occurs.

⇒ OOP's

→ stands for object oriented Programming system/structure

→ OOP is a Programing Paradigm/methodology or different ways to achieve the specific task.

There are so many Paradigms like

1. → object oriented ✓
2. → Procedural ✓
3. → functional ✓
4. Logical ✓
5. Structural ✓

→ There are six main Pillars of OOPs:

1. Class & Objects and methods
2. Inheritance
3. Polymorphism
4. Abstraction
5. Encapsulation

→ These concepts are related to real world entity.

note: small talk/Scratch/Phaso, Erlang, Eiffel, are a very pure object oriented programing language.
Java, C# and C++, Python also oop but not pure.

→ Class

1. Class is the collection of objects, and inside the class have methods. we can acces the Methods through objects. 2. it is not a real world entity it is just a template or blue print.
3. class does not occupy memory.

like

```
class Animal {
    void eat() { }
}
```

syntax

```
class ClassName {
}
```

```
dog obj = new dog();
```

by default it has acces modifier default

→ Methods.

↳ A set of codes which perform a particular task.

Advantages: 1↳ Code reusability 2↳ Code optimization

↳ Instead of recoding it every time you need that calculation, you can reuse the same code, this saves time and effort.

it provide efficiency. (reusability)

2↳ it is like finding the fastest route. it take less memory and executes fast. (optimization)

Syntax:

accessmodifier return-type MethodName(List of parameters)

{

}

Method by default access is default.

Animal obj;

obj=new Animal();

⇒ Difference b/w Methods and functions:

1) Functions: A function is a block of code that Perform a specific task. In Java, these are typically called "functions" when they are independent and not tied to a particular class or objects.

2) Methods: In Java you call them methods, when they are part of a class.

⇒ Objects:

1) object is an instance (copy) of class.
2) Object is real world entity.
3) Object occupies memory.

⇒ object consist of:

1) Identity: name and note object always unique.
2) Object have state/Attribute → ex Dog's: breed 2 age 3 Color
3) object have behaviour. Eat, Run, Bark. Note: Behaviour represents the Methods

⇒ ~~How~~ to create objects

1) by using new keyword
2) by using newInstance() method // which is predefined.
3) by using clone() method.
4) deserialization.
5) Factory methods

⇒ Creating object:

1) Declaration: Animal buzo;
2) Instantiation: buzo = new Animal();
3) Initialization:

Animal obj = new Animal();

Note: Every obj has its own behaviour which we can represent through

Dot operator:

obj.method.

obj.run();

buzo.bark();

=> object can be insilize by using

ex:

```
class Animal{
    int colorno;
    Main method
    Animal buzo = new Animal()
      buzo.colorno = 10;
```

=> by using Methodes.

```
class Animal{
    string color;
    int age;
    void initobj(string c, int a ){
        color = c;
        age = a;
        }
    void Display( ){
        sop(color+" "+age);
        }
    }
```

3, Using Constructor.

=> 1. Constructor: it is a block (same as method) having same name as that of class name.

2. does not have any return type, not even void.

3. Public, private, default, protect have modifiers.

4. It executes automatically when we create an object.

=> why we need cons?

```
class Employe {
    string name;
    int emp id;
    Employe(String name, int id) {
        this.name = name;
        this.empid = id;
    }
    main
    Employe ep1 = new Employe("ravi", 101);
                  ("m", 102);
    ep2
```

Note: constructor is not used to create an object. it is used for initilize the object.

Note2: Default constructor creates the compiler not Jvm. and it has default super(); method.

-> why create default?

The default ~~variables~~ constructor initalizes instances variables to default ~~size~~ values (zeros for numeric, null for object reference)

To ensures that an obect is in a valid state when created.

Note => The benefit of default initialization lies in avoiding undefined or unpredictable behaviour that might occur if variable were left uninitialized.

⇒ constructor have no return types why?

1 -> because it is used for initilize the objects:- means when object will initilize there will be no needs for return type.

2 becuse compiler creates const. so thats why it can not judge what return type should be

⇒ Inheritance:- means inheriting the properties of parent class into child class:-

by using extends keyword,

→ And It has parent child relationships

→ we say IS-A relationships

⇒ Advantages:-

1 Code Reusability:-

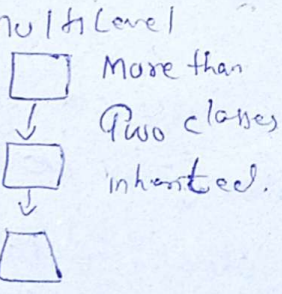
2 we can achieve Polymorphism that is overriding:-

⇒ Disadvantages:- classes are tightly coupled as if it will change one's class properties method all the inherited will be change.

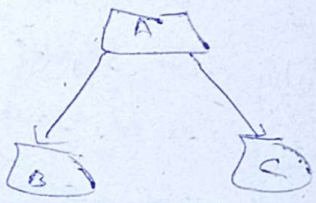
→ Types of inheritance:-

1 Single

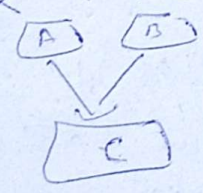
2 Multilevel



3 Heirarical



4 Multiple

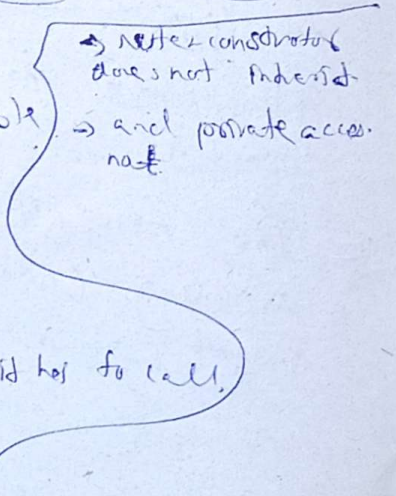


- In multiple inheritance Ambiguity occur that which function it has to call.

X 5 Hybrid: Not possible

→ Outer constructor does not inherited

→ and private access not



⇒ Relationships b/w two classes:

⇒ Types of Relationships b/w classes:
Two types: 1/ Inheritance 2/ Asisociation (has-A)
it has to forms:
1/ Aggregation
2/ composition

⇒ Advantage of Making Relationships:
1/ Code Reuseability
2/ cost cutting 3. Reduce Redundancy (avoid Un-necessary codes)

⇒ inheritance:

```
class A {

}

class B extends A{

}
```

B is-A A

⇒ Association: most commonly used relationships
ex:

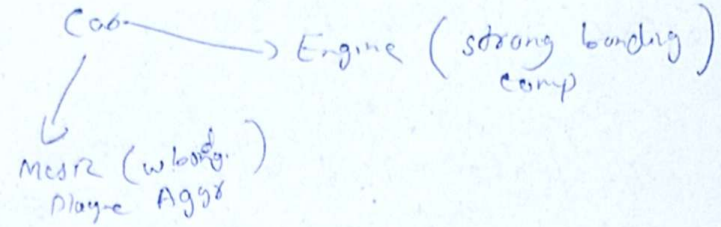
```
class student {
    student name;
    int roll_no;
}
```

becz student has-A name
→ student has-A roll_no.

⇒ Note we can Directly achieve like above syntax but also by using new keyword.
ex:

```
class Engine {

}

class car {

    Engine e = new Engine( ); //although this way we can achive
    specific property of Engine class by using e. modifier. not
    all will be inherited.

}
```

so car has-A Engine ✓: it has the benifit as compare to is-A relationship means If I will change in one class it will not then Impacted on others:



⇒ Association
1. Has-A
2. Non-blood relation.
3. not tightly coupled.

⇒ Polymorphism: Same methods but perform the different tasks, we can achive through the methods
many forms
like water, shapes

⇒ There are Two types of Polymorphism
1. Compile Time Polym — or static through overloading
in handle compiler
2. Runtime or Dynamic Polymorphism — through Overording.
it handle JVM

⇒ method ~~name~~ overloading:
1. Same name
2. Same class
3. Different arguments
   1. No of arg diff
   2. Sey of arg diff
   3. Type of arg diff

Method over riding:
1. Same name
2. different class
3. Same arguments
4. Inheritance Must

ex:-

```
class Test{
    void show (int a, int b){
        sop ("1");
    }

    void show (int a, int b, int c){
        sop ("2");
    }

    main
    Test.show (1, 2) // 1
```

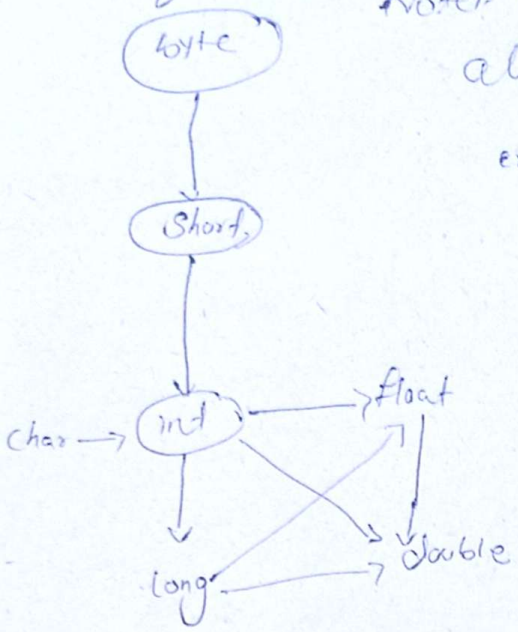
⇒ Questions:-

1) Can we overload Java main() method?

```
class Test{

    Public static void main (String[] args){
        sop ("1");
        Test.main (20);
    }

    Public static void main (int a){
        sop ("2");
    }
```

} Yes we can. because Jvm always calls main() method which recives string array arguments only

⇒ Automatic Promotion: means one type is promoted to another one implicitly if no matching data type is found.

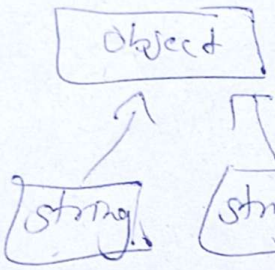


note: Object is the parent class of all the classes in java.

ex: void show (Object a) {
sop ("obj");
}

void show (String a) {
sop ("string");
}

o/p
it will give priority first exact value if available. string will print not object

⇒ 

1. Both are string and Buffer have at same level so "null" cannot be referred, if refer, ambiguous error will occur.

ex
↝ void show (StringBuffer a) {
sop (Buffer);
}

void show (String a) {
sop ("string");
}

main:
t.show ("abc"); ✓ first.
t.show (new StringBuffer ("xyz")); ✓
t.show (null); // ambiguity:

Note: Var arg method
it allow zero or multiple arguments.

void show (int... a)
{
sop ("var");
}

we can pass Multiple arguments before this we use over loaded method or we pass the array, but it was not good approach
1. if no other method will matched the only var arg will call and print its statements if has "0" zero arguments pass.

⇒ Over Riding

```
Class Test{
    void show(){
        sop("1");
    }

class XYZ extends Test
    void show(){
        sop("2");
    }
}

Psvm(){
    t.show() // 1   it depend upon which class's object you are
    x.show() // 2   creating.
}
```

⇒ The benefit is we can change parent class methods implementation then we create child class its own new implemetion.

Means we can change implementation through overriding. by own wish.

→ case 1: Does overriding methods must have same return type? NO!

but note parent method must have higher ~~access bility~~. return type.

ex Object (super) show(){ } // Parent

String (child) show(){ } // Child

and case 2: Child class's methods access modifier must be same or greater as compare to parent.

Case 4 overriding and abstract Method:-

abstract class can have abstract methods, and Not that syntax i. abstract void show( ); this method must be override in child class.

Notes. same situation with interface I1 { void show( ); } that Method must override..

abstract class Test {
abstract void Display( );
}

Note:- we can not create abstract class's object

case 5

→ Note:- through child class object we can call Parent class's methods though super key word.

ex super.show( );

it will print first Parent class method then it's child's method.

class xyz Extends Test {

public void display( ) {
sup( );
}
}

case 6

Note:- final, static and private Methods can not be overriden.

Case-7 The presence of synchronized / static modifier with method have no effect on the rules of overriding.

=> Encapsulation: in Java it is a mechanism of wrapping the data (variables) and code acting the data (methods) together as a single unit.
outer user just use its functionality like car, not about inner machanism.

→ steps to achieve encapsulation:

1. Declare the variables of a class a private.
2. Provide public setter and getter methods to modify and view the variables values.

ex:

```
class Employe {
    private int emp_id; Data_hiding
    public void setEmpId(int empid1) { → setter will modify or set value set karda
        emp_id = empid1
    }
    public int getEmpId() { // getter will view that data
        return emp_id;
    }
}
```

OOPs
- class
→ objects and methods

Inheritance
Polymorphism
→ for code Reuseability

Abstraction
Data hiding
Encapsulation
Tightly coupled classes
→ for security.

→ Abstraction :- is Detail hiding and showing main services or it is hiding internal implementation and just highlighting the setup services that we are offering. ex:- Car's. starting, break, but dont need it's external works. necessary is break and internal functionality

Note:- 1) Abstraction is detail hiding (implementation hiding)

2) Encapsulation is data hiding (information hiding)

→ we can achieve Abstraction. by

1) Abstract class (0-100%)

2) through interfaces we can achieve purely (100%)

ex:-

```
Car
↓
no-of-tyres = 4
↳ start ( )
    {
      sop ("with key");
    }
```

```
scooter
↓
no-of-tyres = 2
↳ start ( ) {
  sop ("start with kick");
}
```

then we can Declare like

```
vehicle
↳ no-of-tyres; all main services herghtright
  ↳ star ( );   but internal services ~~heightright~~ hide.
```

ex:

```
class vehicle {
    int no-of-tyres;
    abstract void start();
}

class car extends vehicle
{
    void start() {
        sop ("start with key");
    }
}

class scooter extend vehicles
{
    void start() {
sop ("start with kick");
    }
}
```

Note:

1. A method without body is known as abstract method.

2. If a class has an abstract method, it should be declared as well abstract class too. but if class is abstract then it is not compolsory that it has abstract method. Can have simple method or we can say concrete method mean can have body.
   → If we will give body to method called concrete then that class called 0% abstraction

3. If a regular class extends an abstract class then the class, must have to implement all the abstract methods of abstract class or it has to be declard as well.

4. override concept occur

5. we can not create object of abstract class bcz implementation not occur then then will be not benfit of that.

≈> Interfaces: are similar to Abstract class but having all the methods of abstract type.

Note: Interface are the blueprint of the class, which tells the class what you have to do.

use?

→ it is use to achieve Abstraction.

→ it supports Multiple inheritance.

→ it can be achieve for Loose coupling. mean if we do changing in one thing (class) then other will not impacted.

Syntax:

```
interface Name1{
    void show();
}
```

1) methods // abstract and public → by default

2) fields // public, static, final → by default

Note1: can be concrete method but must has acces Modidefult

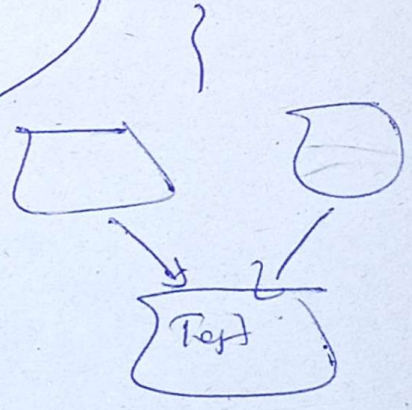
Note2: static method can be create with public acces.

Note3: Can be private methods

```
class Test implements Name1,Name2{

public void show(){
    sop("1");
}

public void display(){
    sop("display");
}
}
```

Multiple

```
interface Name2
{
public void display();
}
```



⇒ Exception handling :- which distrub the normal flow of the program, when/due to unwanted even occurs or An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disturbs the normal flow of the program.

Note ⇒ In this case if unexpected will occur then the we look for alternative and continue that flow in same manner called...

ex: sop (100/0); // this is called Arthmetic exception.

→ when exception will occur the forward code will not run.

⇒ Hierarchy of Exceptions:

note: 1 Object is the parent class of all the classes in Java.

2 Throwable is the parent class of Exception class

3 Throwable (parent class) further inherit two classes:

1 Exception (child) 2 Error (child)

1 Note: Exceptions → are occur due to our programs (due to programming mistake)

2 note: Error → occur bcz of lack of system resources. (weak, slow, pro...)

note: Exceptions are recoverable ✓

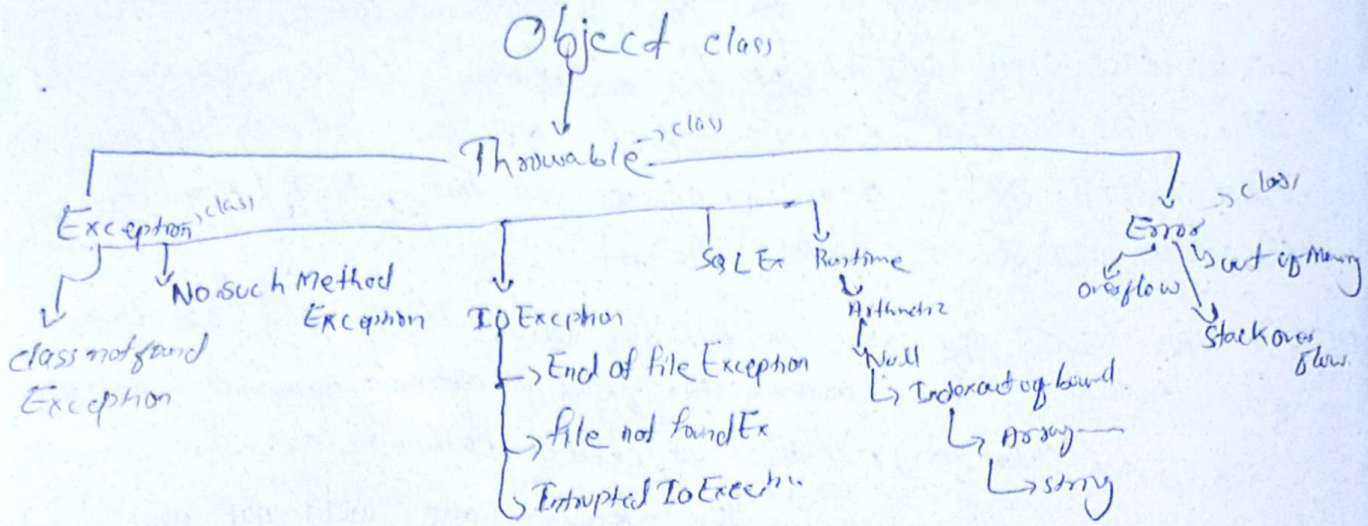
→ errors are not recoverable / programe cant it will do only system Administrator.

→ There are two types of Exceptions:

1 Checked Exception (compile time) 2 Unchecked exception (Run time)

Note: error will occur ONly run time Exceptions = (unchecked

⇒ Hierarchy diagram.



→ Note: No any exception occur while compile time, all occur in run time.

→ compile time exception are those which compiler can check, but few ex which it has no athority then can't found. it will find run time excep...

ex: compile time: Class.forName("com.mysql.jdbc.Driver");
but we can handle it by throw or throws or try catch. using then it will compile easily. then Exception will handle in compile time but run time it will show.
100/0 compiler can't check it will show in run time.

→ Note: Whenever there is exception, the method in which exception occurs will create an object and that object will store 3 things.
1) exception name & description 3) stack trace
and that object ○ → will pass to Jvm the Jvm will check if I will handle then ok other wise it will pass to default Exception handler then that ex will print.

It has name
description
stack trace.

→ we can handle the exception using 5 keywords → try 2, catch 3, finally 4, throw 5, throws

Syntax:

```
try {
    // risky code
}
catch ( Exception e ){
    // handling code  sop (e);
}
```

note: Exception is the parent class of all the exceptions

→ through try catch we can compile the program.

→ note: if try block does not occur inside exception, the it will not go inside the catch block.

Collection framework

~> Collection: It is the single entity or object which can store multiple data.

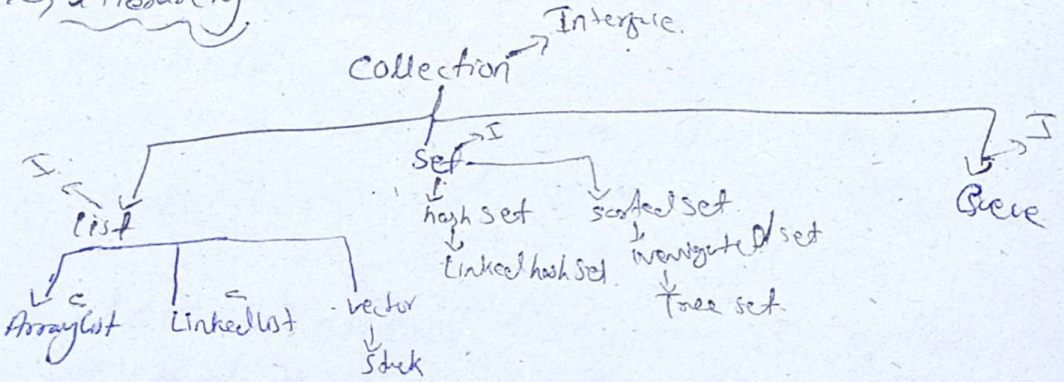
~> framework: it represents the library, interfaces or it is the set of predefined classes and interface which is used to store Mutiple data.

~> collection framework contains 2 main parts

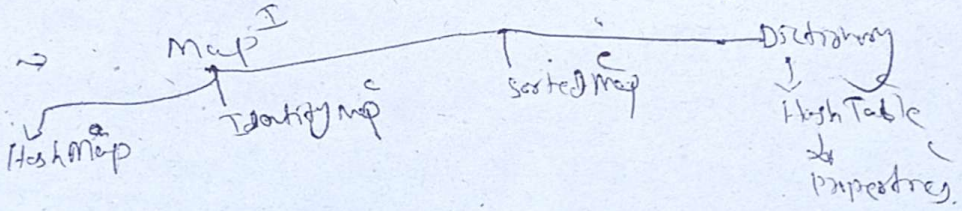
1. Java.util.Collection : in collection we can store the data directly.
2. Java.util.Map: In Map we stores the through key map.

~> Note: we can say that collection framework is an API in which it has predefined classes and interfaces which creates object and multiple data will store.

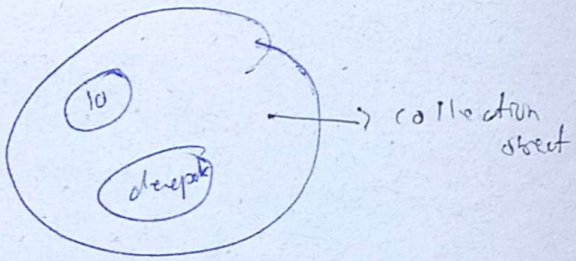
~> Hierarchy



~> The old classes are known as legacy classes.



~> ArrayList al = new ArrayList();
al.add(10);
al.add("deepak");



→ Difference b/w Arrays and collection frameworks:

→ Arrays:

1. it can store primitive and non-primitive (object type) of data can store.
   ex: int[] a = {5, 4, 3};
   ex: Test {
   Test[] t = {obj1, obj2, obj3};

2. Array can store only homogenous type of data.
   (same)
   → In this case it is not better.

3. Array size is fixed, we cannot increase or decrease the size of an array at run time
   → not better

4. Arrays are in built future of java and this we have to develop Algorithams, ex, searching, sorting, Diction

→ C Frameworks:

1. it can store non-primitive type of data.
   ex:
   ArrayList al = new ArrayList();
   al.add(obj1);
   al.add(10); // Integer

? → it is better in 1st case

2. we can store hetrogenous type of data (different)
   ex: al.add('z');
   al.add(2);

3. we can increase or decrease the size of collections at run time

4. it is an API which provides predefined classes and interfaces and methods. like sorting, searching, diction etc are predefined or provided.

→ API :- stands for Application programming interface (API) is like a waiter and menu is like a serves which ~~provides the~~ serves the order you give | menu it is like a source.

→ ArrayList :- is an implemented class of List interface which is present in Java.util.pkg. and List interface is an implement from collection interface.

→ Syntax:-
Package java.util;
class ArrayList implements List {
it has constructors and Methods:
}

→ Properties of ArrayList:

1. it is index base datastructure
2. it can store hetrogenious data types
3. In List we can store the duplicate elements but not in sets.
4. So, ArrayList can store duplicate value.
5. It can store any numbers of null values.
6. it follows the insertion order, and in same order it will retrive
7. it does not follow the sorting orders.
8. it is an unsynchronized.
9. it supports the parallel execution
10. it is not thread safe.
11. it does not give garanty of data consistency

(2) ArrayList is created on the basis of growable array or resizable array. means it is like as array but it is not fixed we can increase or decrease its size

3. Both Array and ArrayList are index base data structure.

→ Diff between ArrayList and LinkedList.

→ AList

→ class ArrayList implements List {
// cons
// Method.
}

→ LList.

→ class LinkedList implements List, Deque
{
//
//
}

2) it acts as List.

3) it store elements in contig memory locations.

4) it is good for retrieval operations (means get the value)

2) it acts as List and Deque.

3) it can't store in contin.

4) it is good for insertion & deletion.

⇒ Legacy classes: Are those classes, ex, vector, stack, Hashtable, Properties in 1.0 JDK version provide ~~to~~ classes and interface like, then 1.2 JDK launch collection frame work then they introduce in it.

⇒ Vector (legacy class) it is an implemented class of List interface and it was introduced in JDK 1.0 version.

⇒ it is present in java.util package

Syntax:
Package java.util;
class vector implements List {
=
=
=
}

Properties
⇒ it is an index based data structure.
2 it can store heterogynes data types
3 we can store duplicate elemts and infinite null values.
4 it follow insertion order and it does not the following the sorting Algorithms.
5 Vectors are synchronize collections.

3 The underline ds of vector is growable "array" or "resizable array".

note: All legacy classes are synchronized.

~> HashSet: it is an implemented class of Set interface which is present in "java.util package".

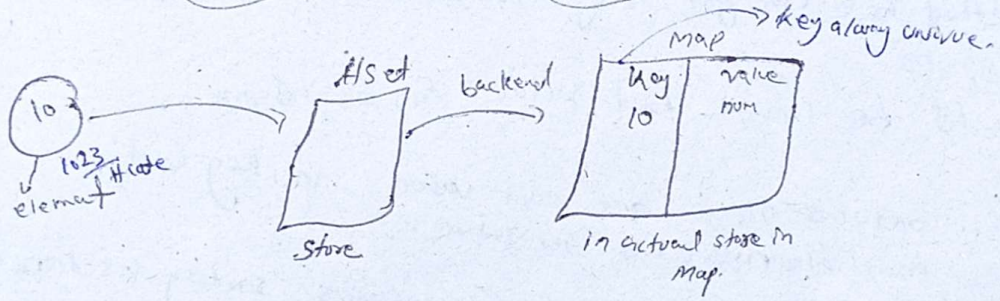
2) Syntax:

```
class HashSet implements Set {
    = Data
    = Methods
}
```

3) Hashtable is the underline datastructure of HashSet. On the backend #Map work behind HashSet.

~> Properties:

1) HashSet are not an index based datastructure. they store the elements according to their "hashcode value" means. Unique integer value we say (Hcode). it is unique value of every objects.



2) it does not store the duplicate elements.

3) not store multiple null value

4) Can store Heterogenous

5) Not follow "insertion order"

6) Not follow sorted order.

7) Non-synchronized.

Note: HashSets are fast but it has security issues.

~> HashMap: HashMap is an implemented class of Map interface. it comes in 1.2 JDK version

syntax:

```
Package java.util;
class HashMap implements Map {
    // (or)
    methods.
}
```

2. its underline data structure is Hashtable.

~> Properties:

1. it store the value in Key-value

Note: One key value is said to be Entry:

Key	value
101	Noor
102	Sagar
103	Deepak

→ Entry and this Entry is called predefined Interface

Note: Entry is the Interface which present in Map interface and one key value pair we called Entry. and Note that, through Entry Method we can get Key and values.

2. Key should be Unique but value can be same.

3. HashMap contains max one null value in key but can store multiple null values in value.

4. it store Hetrog elements. and not following sorting insertion order.

5. it is non-synchronised ds. (multiple threads can pallallely execute) which can enhance performance speed.

7) HashTable :-

1) it is the directe impemented class of Mintegrce and Htable inherit the dictionary class.

syntax

```
class Hashtable implements Map
{
}
```

2) its Underline DS is HTable

→ working of HTable.

Objet though we can create Hash table ex:-

```
Hashtable ht = new Hashtable();
ht.put(106, "Nood");
```

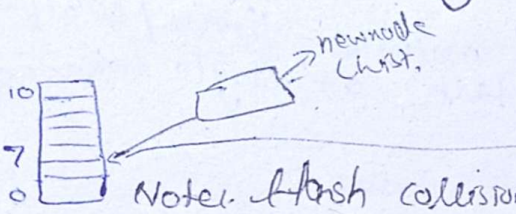
↳ it has own hashcode value

Note:- In java, hash code is a unique integer value that is generated for every object which generate the jvm.

formula: Hascod % 11 = remainder.

106 % 11 = 7

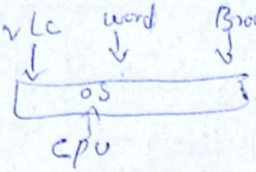
Note:- HashTable's initial capicity is 11.



Note:- Hash collision - occur when same indices value occur then through linked list it will resolve, it will create new node.

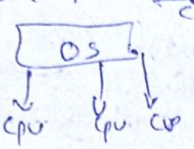
:- Elements will retrive Top to bottom and right to left

~> Multitasking: Performing multiple tasks at single time & it increase the performace of the cpu.



~> we can achive Multitasking by two types: 1) Multiprocessing based 2) Multithreading

~> Multiprocessing: when one system is connected to multiple processor system in order to complete the task.



& it is best suitable on system level or OS level.

~> Multithreading:- Executing multiple threads / sub-process, small tasks at single time

ex: VLC → process/program.
- ↳ video → threading
- ↳ Time → threading
- ↳ progress bar → th

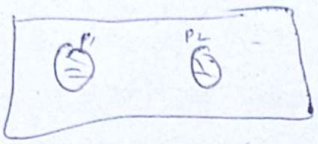
ex:
```
VLC (process, program)
class VLC { psvm()
            { playvid();
              music();
            {
}

class video {
   void playvideo()
        {~
        {
}

class music {
   void startMusic() }
                     }
}
```

~> Note: Multithreading is mostly used in programing level.

~> Note: 2 Java provide predefined APIs for Multithreading

→ Process

1. A program which is in executing state
2. Heavy weight
3. Context switching: A process at a time execute single process when it swith to other with in nano-seconds; we think that cpu perform all the process at a time. in ~~thread~~ process it takes much time



4. Communication: Take more time for interprocess
5. Address space: each process has different address space
6. Are not dependent
7. Synchronization does not req synchronization.
8. More time to terminate

→ Thread

1. it is subpart of process
2. Light weigh
3. it take less time (CSwit).
4. it takes less time.
5. Same adress
6. dependent
7. it requires may be Synchronization.
8. Less tim

Two ways to create thread: 1. Thread(class) 2. Runnable(interface)
if exist in java.lang; package

```
class Thread {
    // constructor
    // meth -
    run()
    start()
    sleep
    getNam
    interrupt, priority
```

}

~~⇒ Creating thread extends~~

⇒ Creating thread → 1st way

step 1

→ class Test extends Thread

{

② override the run method

public void run(){

}

{

main method

③ create an object of class

Test t = new Test();

t.start(); ④ // start the thread.

}

② way

step 1

~~→ class Test Runnable~~ → class Test implements Runnable → it is interface and have only one method Run.

{

step 2 override

public void run(){

sop ("thread task");

}

→ it is better way

psvm

Test t = new Test();

Thread th = new Thread(t); → sop

th.start();

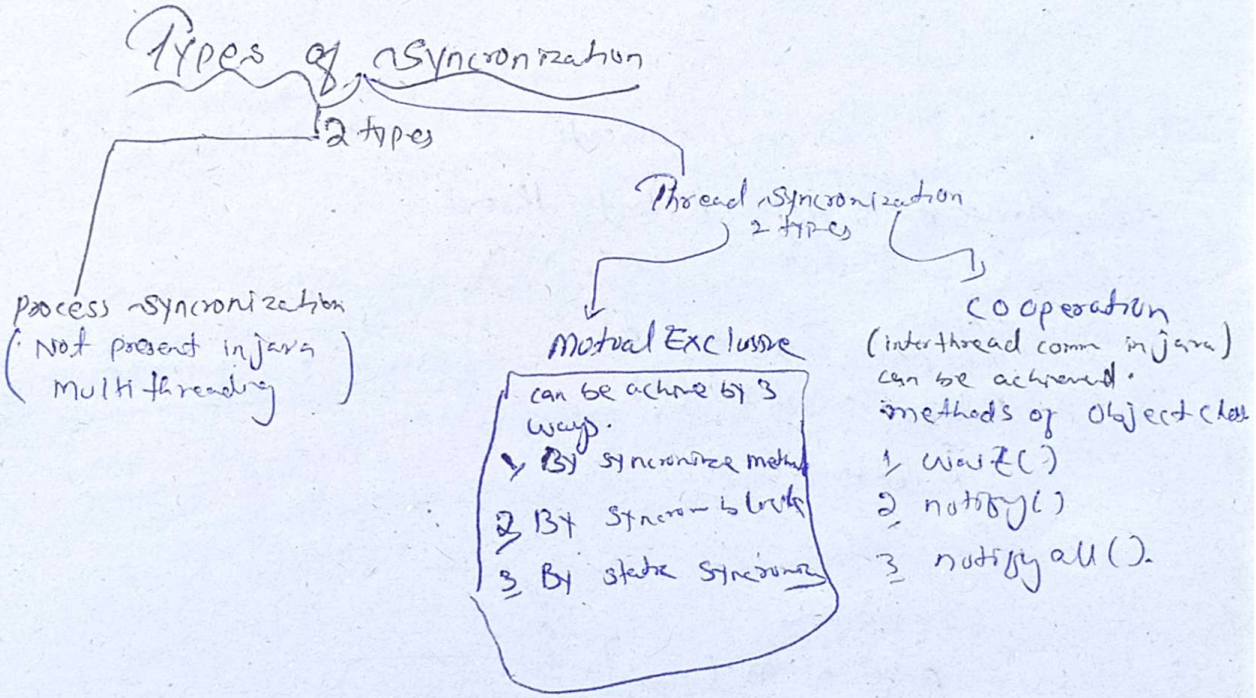
}

~> Synchronization: it can lead to data inconsistant, and does not give always constant output (using thread concept)

Def: it is the process in which through we can multiple threads accessibility do control.
or, it is the process by which we controll the accesibility of multiple threads of a particular resources.

~> Problem which can occur without synchronization
1) Data inconsistency
2) Thread interference

~> Disadv of synchronization, 1) increase the waiting time period of threads 2) create proformance problems.
1) (means if thousand of thread occur then it will wait / to threads.)
due to this performance problem will occur.
for resolving waiting disadvantage java provide package name java.util.concurrent.



→ Problem without using synchronization.

Note :- that by using synchronized key word / using synchronized method we can resolve this issue.

```
class BookTheaterSeat {
    int total_seats = 40;
synchronized void bookSeat(int seats)
    { if (total_seats >= seats)
        { sop("seats booked succesfuly");
          total_seats = total_seats - seats;
          sop("seats left" + total seats);
        }
    else {
          sop("seats can not be booked");
          sop("seats left" + total seats);
        }
    } // now creating threads.
class MovieBookApp extends thread
  {   int seats;
      // reference
      static BookTheaterseat b;
      Public void run(seats)
        { b.bookSeat
        }
      p s v m ( ) {
        b=new BookTheaterSeat();
        MovieBApp deepak = new MBA();
          deepak.seats=7;
          deepak.start(); // for start the run method.
        MBA Amit = new MBA();
          Amit.seats=6;
          amit.start(); }
  }
```

// Every object have two areas sync and no-sync

thread1

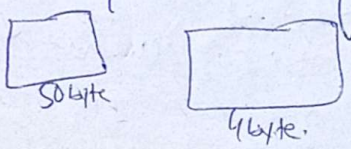
thread2

Length → it is like counting things.
ex: Hello → Length is 5.

Size → it has the space something takes up.
so, Length is about counting, and size is about the space something occupies.

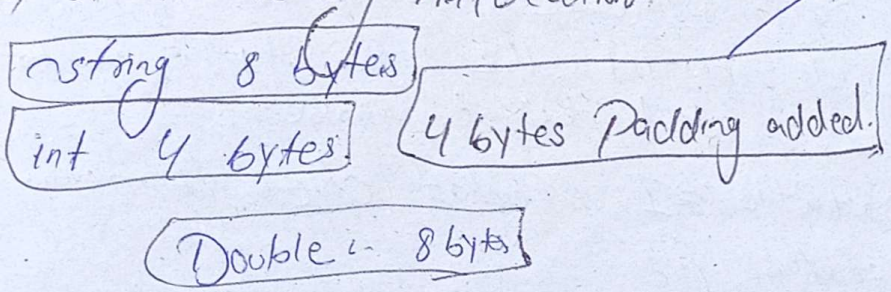
→ Structure: (struct) → it is used when we want to allow different types of data into a single name.
Through (struct) keyword is used to define the structure

syntax:
```
struct structure_name {
    data_type1; char name[50];
    data_type2; int sal;
}
```
note: for every element struct create the separate Memory.



ex: struct emp detail; // we can access through this.
or struct Emp E
e.name;
e.sal;

→ struct Memory Allocation:



→ Union it is the user-defined data type that allows you to store different data types in the same memory location.

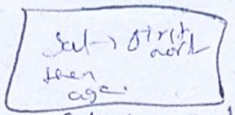
Note: Union keyword is used to define union.

• → size of Union depends on the biggest member of union.

same syntax.

→ first introduce the Union due to store the same location (memory) for all struct key word was introduced

ex
double sal;
int age;



→ dependency occur because first one variable will work than other will execute.
due to same memory some time corrupted value also occur.

→ Enum: (Enumeration) it is a data type that contains fixed set of constants.
or it is used to represent a set of named values or symbolic for a collect of related constants. it provid the clearity easier to understand and intend behind variable without enums!

ex: if status = 1; // what does it mean.

```
enum status {
    success,
    failure,
    Pending;
};
```

ex: Status status = status.Success; // clearly indicates success

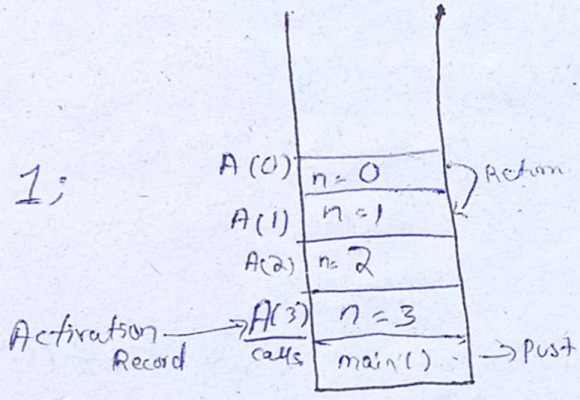
→ Recursion:- When a fun call it self, Base condition is must, use to solve bigges problems after breaking it down in small problems
∴ it is more redable approach, and easy to use
→ it does not reduce the time complexity or the space complexity, even it increase, and it is efficent approach.
note:- If any Problem does not needs the base condition then we can not use the recursive approach.

ex A(n) { infinite then in this case stackoverflow condition show.
    sop ("Hi"); thats why base condition is must.
    A(n-1);
}

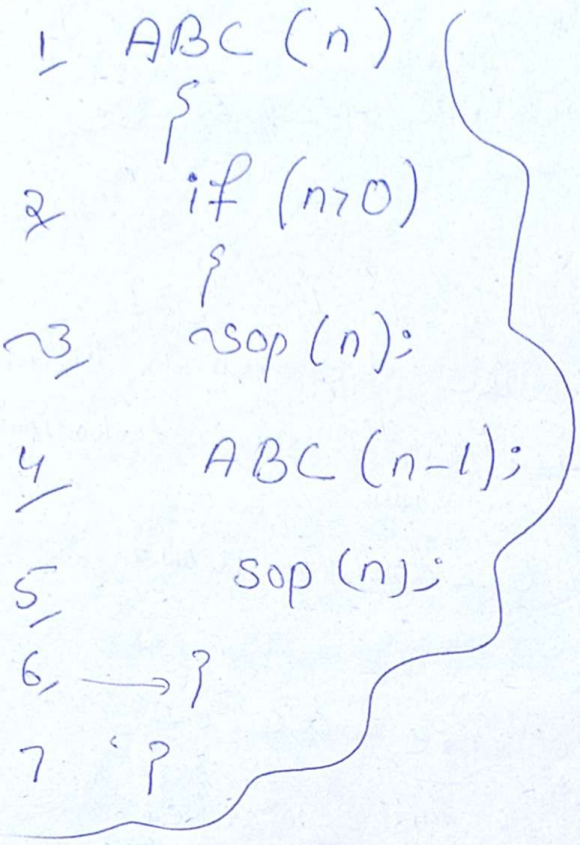
→ Recursive Tree:- we can Trees the recursion by/through two approaches
1 stack 2 Tree.

Using stacks:-

```
A(n)
{ if (n==0) return 1;
    sop(n-1);
    A(n-1);
```



1 → Note first call go to the main function
2 Assume that n=3
3 Jetty b calls lagandy Ixe sab Activation record mai same theendy (on given stack)
4 Jen data push theendi ten stack (memory) increase hundi
5 Jade value return kardaita back wari value khai kando Paaji Place kha
6 Note, Jadeh be Activation Record stack mai push kayo weendo aw ta hik beh nshai be nsave theendi aw Jeh khai as → instruction Pointer chawanda Ayo. it tells if back calls occur then which statement it has to execute.



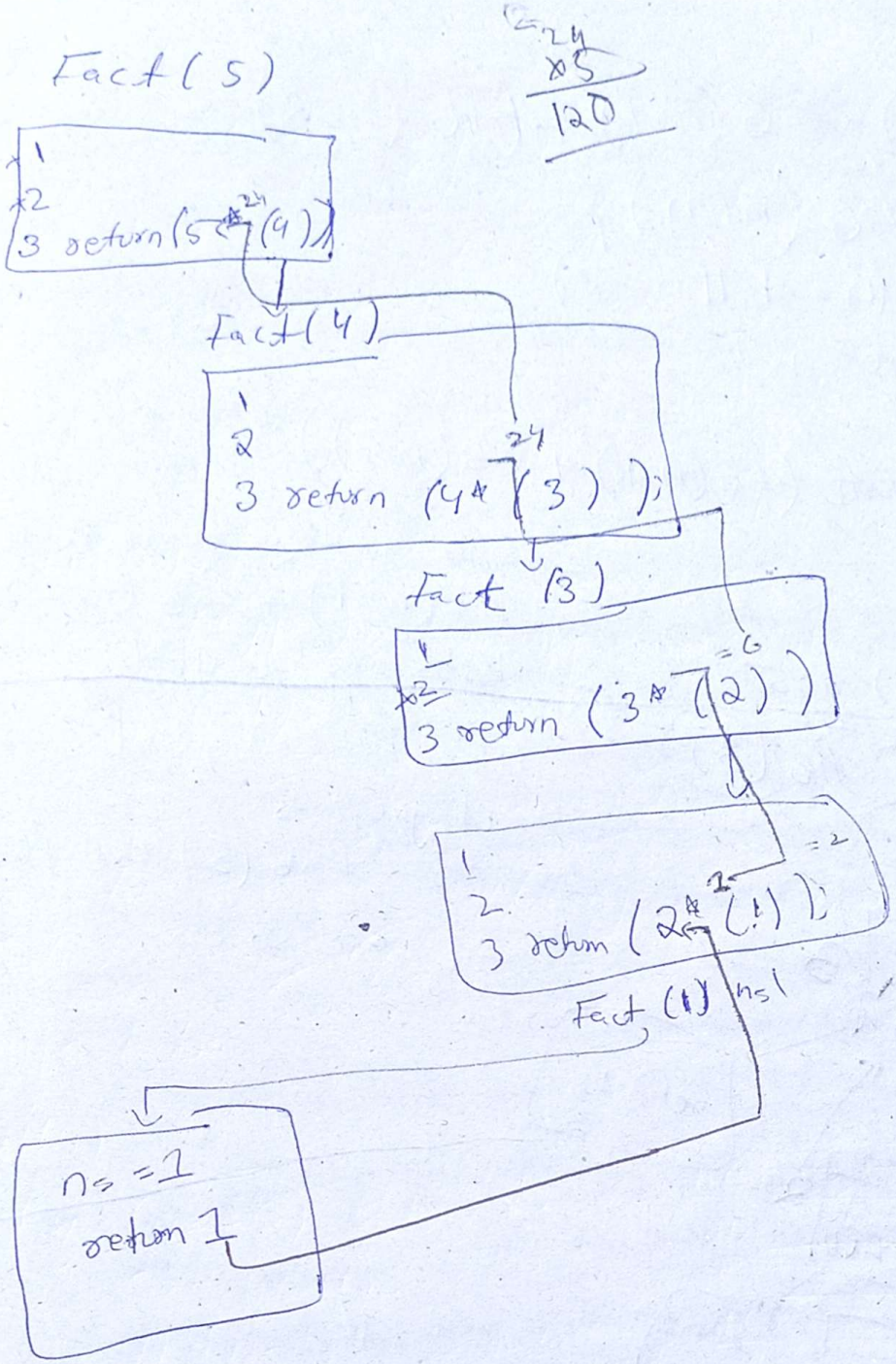
→ Program of factorial using the recursion.

```
int fact (int n) {
    if (n==1) // 1 statement
      return 1;
    else
    return (n* fact(n-1));
        }
```

P ~ v m

fact (5);

Fact(5)


24
*5
120
1
2
3 return (5 * (4));
Fact(4)
1
2
3 return (4 * (3));
24
Fact (3)
1
2
3 return (3 * (2));
=6
1
2
3 return (2 * (1));
=2
Fact (1) n=1
n==1
return 1;

Fab recursion.

$$fib(n) = fib(n-1) + fib(n-2)$$

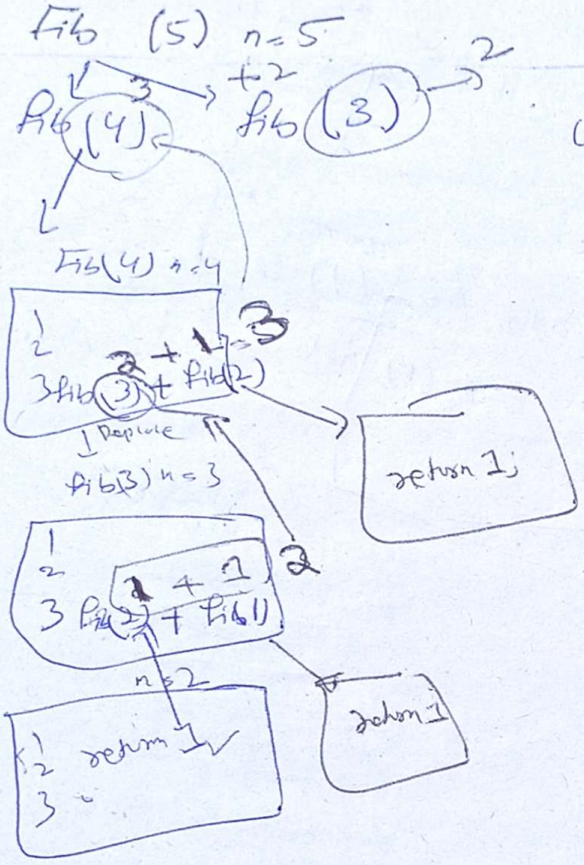
```
int fib (int n) {
1, if (n==1 || n==2)
2.  return 1;
   else
3, return (fib(n-1) + fib(n-2));
   }
```

→ Note: in the recursive call fab(n-1) + fab(num-2) both are called.

like, fab(6) called.

it goes to fab(5) and fab(4)

and so on.

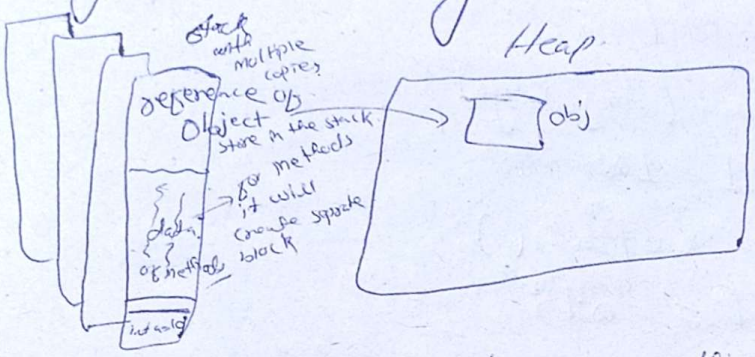


~> Garbage Collection:- Till no we have covered how to create a class, how create an object so on. Now think is that how Memories are allocated to them.

~> There are two types of Memory (Ram) which Java creates. 1) Stack 2) Heap.

~> JVM manages both (stack, Heap) memory, it divide's the memory into two parts (stack, Heap) and store different types of data inside.

~> Note: Heap has more Memory than stacks, and it's only have the one copy.



~> what type of data store inside the stacks?

1) store Temporary variables, and separate Memory blocks for methods. (ex:- if I created a method inside the block I use variables like int a = 10 so it is the temporary, after the brackets it will destroy. so this known as temporary variables.

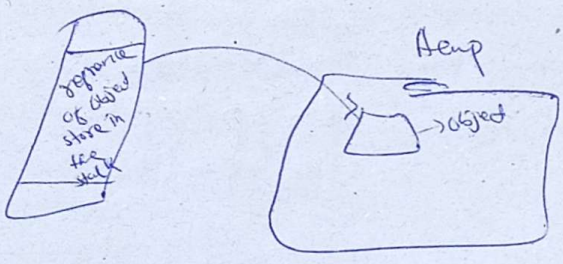
2) store primitive data types. (Note:- Primitive variables doesn't store any reference, so it's store in the stack)

3) store reference of the heap objects.

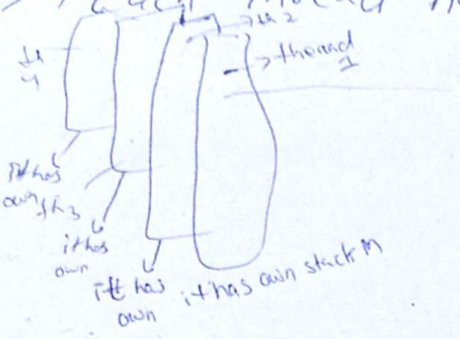
4) Note, when ever we create any object with the new keyword, means when ever we create the object it store in the Heap.

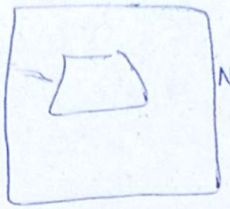
~> Types of reference:-

1) strong Reference
2) Weak reference
3) Soft reference



4→ Each thread has its own stack Memory.

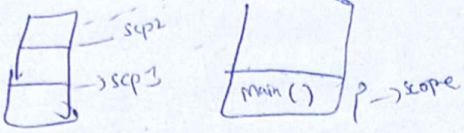




Note and all the the threads share a common heap memory

5→ when stack Memory goes full, it's throws "java.lang.StackOverflowError".

→ when it will store sprate place each place is know is scope in the stack




```
public class MemoryManagement {
→ Public static void main (String[] args) {
    int PrimitiveVariable1 = 10;
    Person PersonObj = new Person();
    String StringLiteral = "24";
    MemoryManagement memObj = new MemoryManagement();
    memObj.MemoryMunagmentTest(PersonObj);
  }
```

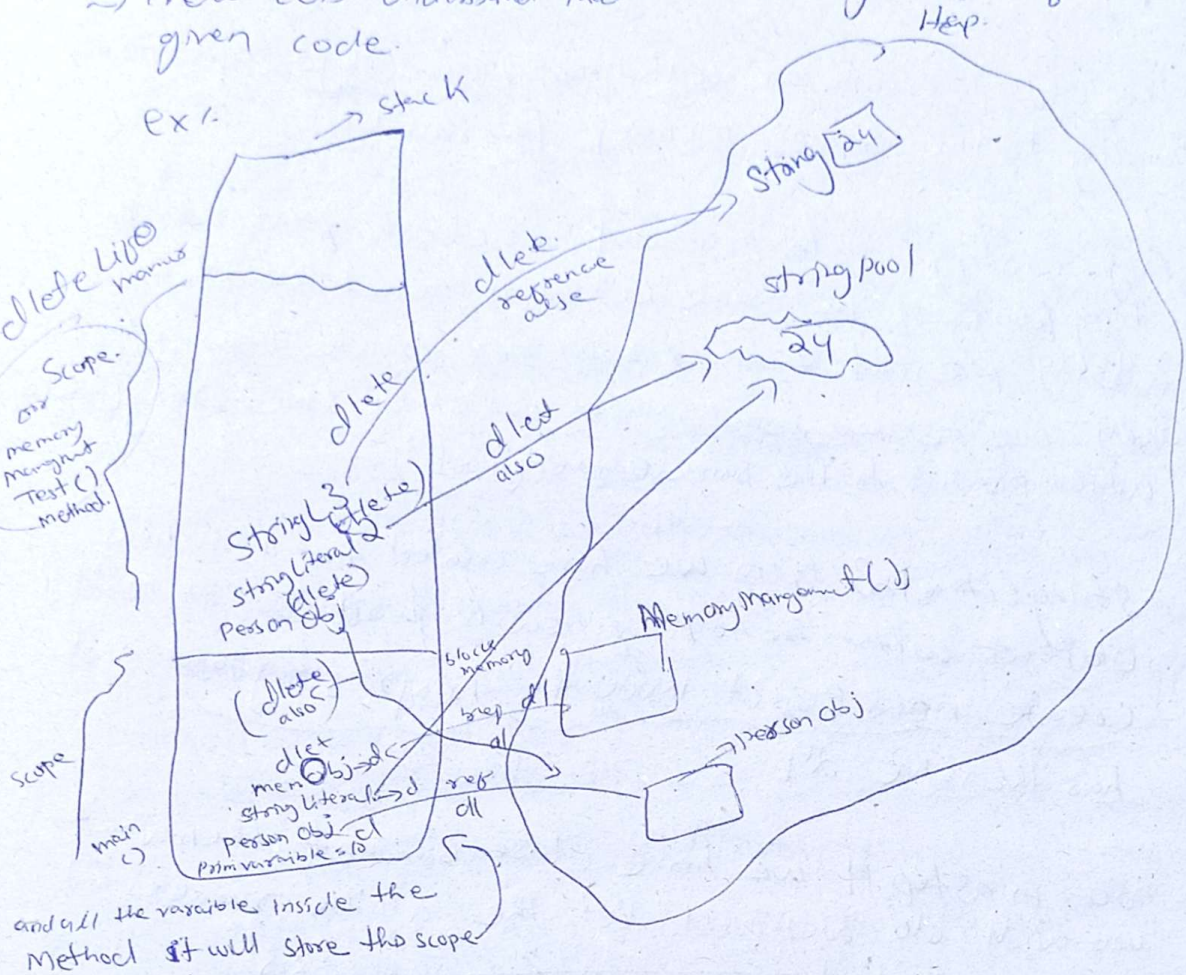
invoke this

```
Private void memoryManagementTest(Person personObj){
    Person PersonObj2 = personObj;
    String stringLiteral2 = "24";
    String stringLiteral3 = new String("24");
}
```

5) Variables with in a scope is only visible and as soon as any variables goes out of the scope it get deleted from the stack (in Lifo Order)

→ let's Understand how the Lifo order is performed and what is meant by scop.

→ now lets understand that how memory is stored for given code.



→ step1. Last main method will invoke, and for each method a block of memory is stored.

→ step2. when method will call, it will store the primitive variables. and p variables does not store any ref thats why store in the stack Method scop.

→ step3. now we have created the new Object. Person it will create in heap with reference of Person Obj and it will store ref in the stack of Person Obj

→ step4, String_Literal store the data into the stringpool. and string Pool store inside the Heap Only. and reference will store inside the stack

step5. we have created one more Object with the name of Memory Mangemet.

step6 Now its time to invoking the Method. name
memoryManagemantTest(personObj); so we are calling/
executing New method so thats why it will create new
scope inside the stack

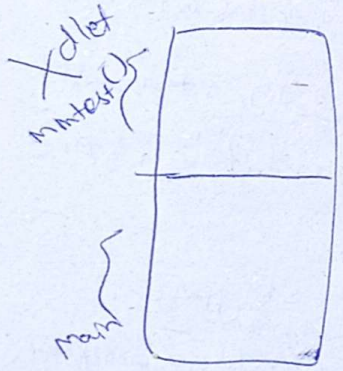
→ 5 personObj inside stack and pass reference ~~his~~ the personObj.

→ 5a Other object create we with name personObj2. and it is
pointing to the same memory PersonObj

→ 5a. Again string Literal2 is created, we check first into the
string pool if that same Literal is present ex 24. is
already present then it will not create new it
will use the same, so again it will stack store string Literal2
which pointed to the same ~~object~~ (pool).

→ 5b. string Literal3 Now we have created but it is new
Object with the help of new keyword. so it will
create new Object inside the heap. string Object. and
has the value 24.

step11. so in step 11 we have closen bracket then what
we will do we will dlet the all the varables and
refrence in that ~~scope in Lifo Order~~ → SO
→ scop.
this scope is finish then it will dlete this scope
in Lifo manner because first we create the main
method. then MMTest last, but we dlet
MMT dlet first,
So step by step it will remove with refrences
also



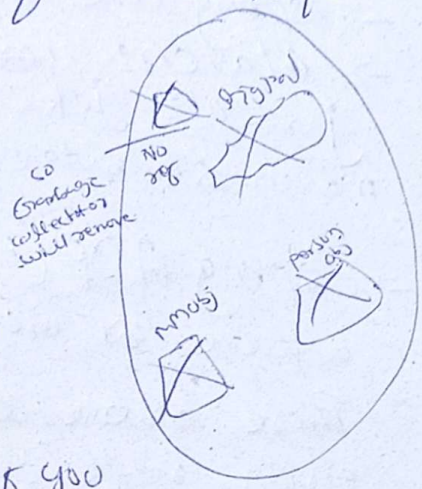
⇒ So we have dleted the ~~se~~ MMT Method
with its refrences, Now control will move to
Main Method. bcz in main method closing
brackets has been reached.

→ So we also remove/dlete the Main Method Sclop.

mem obj with ref dlet
StringLiteral with ref
Person obj with ref
Pram-bk also dleted

So, we have dleted the refrence which were pointing so but now Objects are remaing in the Heap. who will clean? thats why Garbage collection comes.
So

Definitation:- Garbage collector is used to dlete the unreferenced objects from the heap.
This all will be done by Garbage collector
Garbage collector run peoridly, and
Note:- → JVM controls when the run to Garbage Collector. and there is command → System.gc() and it will dlet unrefrence objects. but.



Note:- Instead JVM says that even if you type System.gc(), mean I will run, no, its totally depend upon JVM. when to run, it depend on JVM. So thats why it is called automatic Memorymangment.

→ Spring container:- core component (heart) like in java JVM handle java same as it is spring container spring Application. khai sambaleendo ahy

Responsibilities.

> Transaction Managment

Always Remember that spring ya spring Boot asikhar kha pehly taha khai core java lazim achar gurjy.

→ spring core Module

→ Maven:- ~~core Module~~. it is the build tool. jar file jeka asa insert kareenda ayu Uha internally Maven sambaleendo ahy.

→ spring AOP (Aspect oriented programing) we can achieve by using .xml file or java Inotation.

These above are all the ~~etc~~ through we can easily create dekstop type applications.

≈ spring Boot. Means ass spring ji applications khai Boot Kayu tha.

→ what is spring frame work:-

→ spring:- Open source java frame work.

→ we can develop standalone and Enterprize Apps:

Released in 2003.

→ spring jai through asa jelly b applications thaeendo aayu uha test karar taman easy ahy.

→ Integration capabilities:- means spring khai as ailahi byan frame works sa b as integrate karaye saghu tha like, React, Angular

→ Scalibility:- Spring frame work jai andar jeky b asa applications thaeendo Ayu asan una khai extend karar ya una khai Scal karn Taman easy koondo ahy. (it's is very big advantage of this)

→ Open source:- free to use.