lect 48:

seen: 19-7-23

* Constructor ##

class {

① Variables
- NOT static
- static

② methods <

③ Constructors

④ static Block

⑤ - - - - → coming soon.

class Core2web {

int x = 10;
static int y = 20;
void fun(){
int z = 30;
}
}

bipush
byte integer push

1. class

class Core2web{
int x;
Core2web(){
super(); inv bipush
x = 10;
}

static block

static variable

global

```
static void fun (){
        int z = 30 ; #bipush.
}

static {
        int y = 20 ; # bipush.
```

Constructors

→ CPP → favourite question
what is copy constructor ?

java → copy constructor

private

byte code

performance

method Area

backend → kitchen.

API

frontend

interior

designer

hoff

Constructor

Compiler → default

Programmer → No argument constructor.

```
class Core2Web{
    Core2Web(){
        SOP("In Core2Web");
    }
    public static void main(String[] args){
    }
}
```

this → 1st implicit parameter

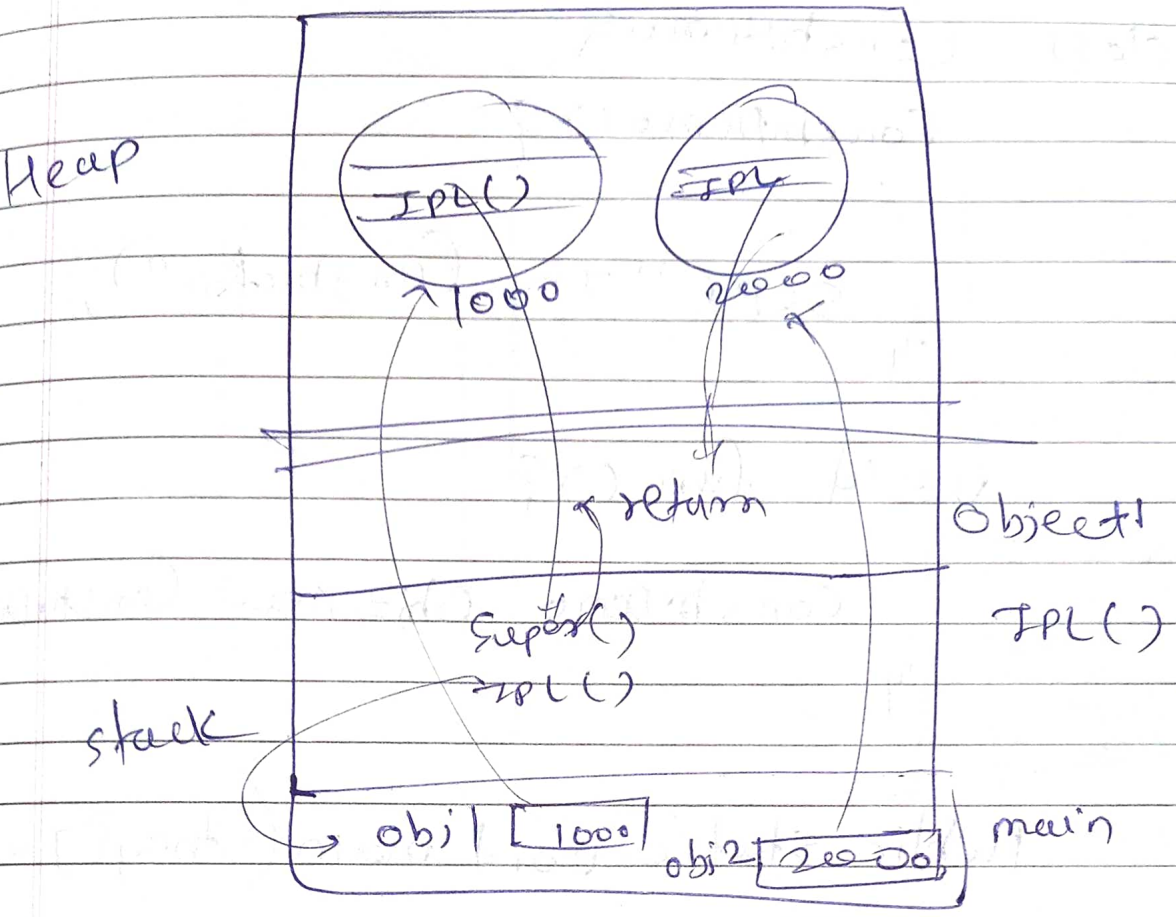
Date

```
class Core2Web {

    Core2Web() {
        sop("In Core2Web");

    }

    int x = 10;
    Public static void main(String[] args){

        sop("In main");
```

object बनाना, constructor का call जाना.

```
        Core2Web obj = new Core2Web();

        sop( obj.x);
        sop(obj.y);
```

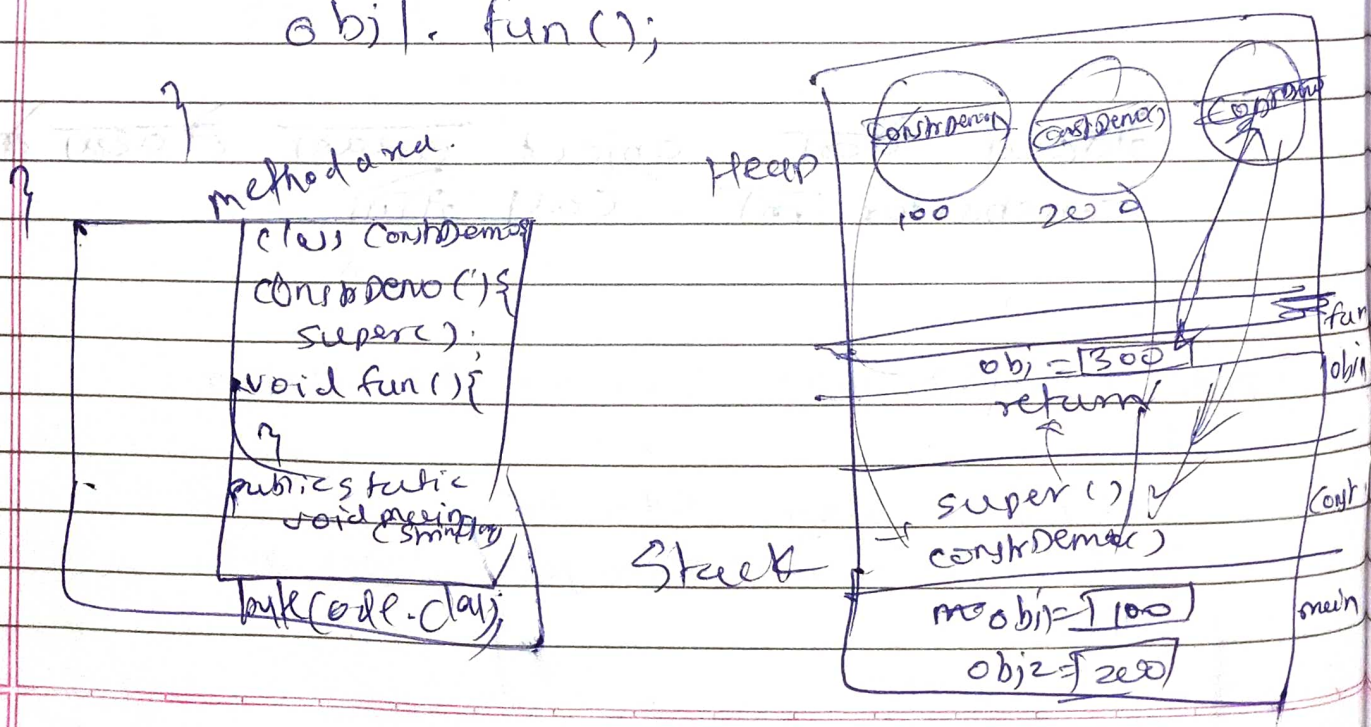
```
class IPL {
    IPL(){
        sop("RCB");
    }

    Public static void main(String[] args){

        IPL obj1= new IPL();
    }   IPL obj2 = new IPL();
```



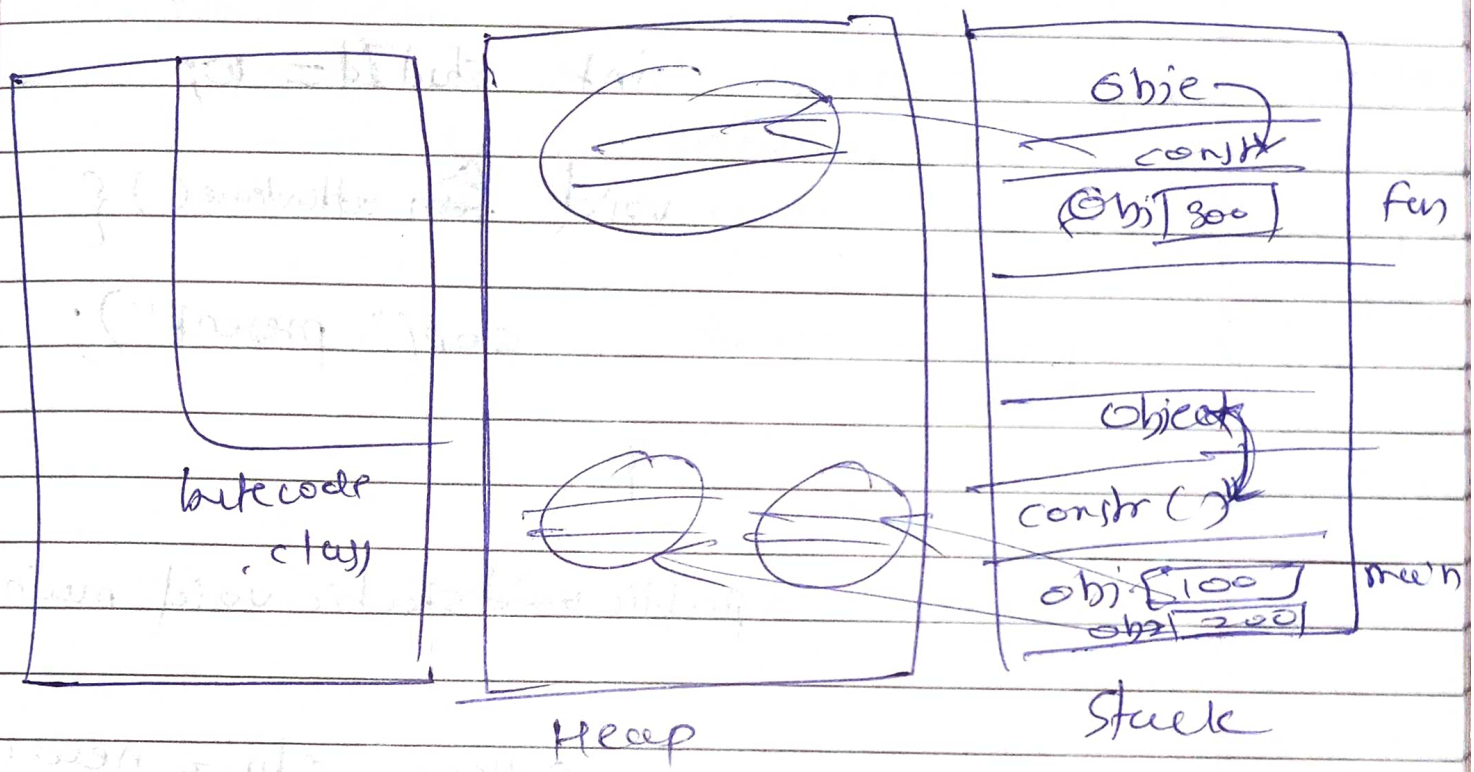
O/P :⇒ RCB
RCB

गेवढ्या वेळा object बनवला तेवढ्या वेळा constructor ला call गेला.

```
class ConstrDemo{
    ConstrDemo(){
        SOP("In Constructor");
    }
    void fun(){
        ConstrDemo Obj = new ConstrDemo();
    }
    public static void main(String[] args){
        ConstrDemo obj1 = new ConstrDemo();
        ConstrDemo obj2 = new ConstrDemo();
        obj1.fun();
    }
}
```



O/P:- In Constructor
In Constructor
In Constructor



check the
Sop (obj) =

seen 20-7-2023

Object 49 Constructor 2

1) → real time

```
class college {

    int studId = 10;

    void attendance() {

        sop(" present");

    }

    public static void main(String[] args) {

        College obj = new College();

        sop(obj.attendance());
        sop(obj.studId);
    }
```

नाम → book, Car, College, student

काम / Kahani → pollitics, Hospitals, cricket

class Company {

No. of stakeholders.

empolycount	stock prize	type
No project	used software	compan mode
dept ;	designation	no. of Devop
branches	location	No. of Sprint
time	name	No. of Investors
salary	CEO	
	Parent Company	

void development() {

}

culture() {

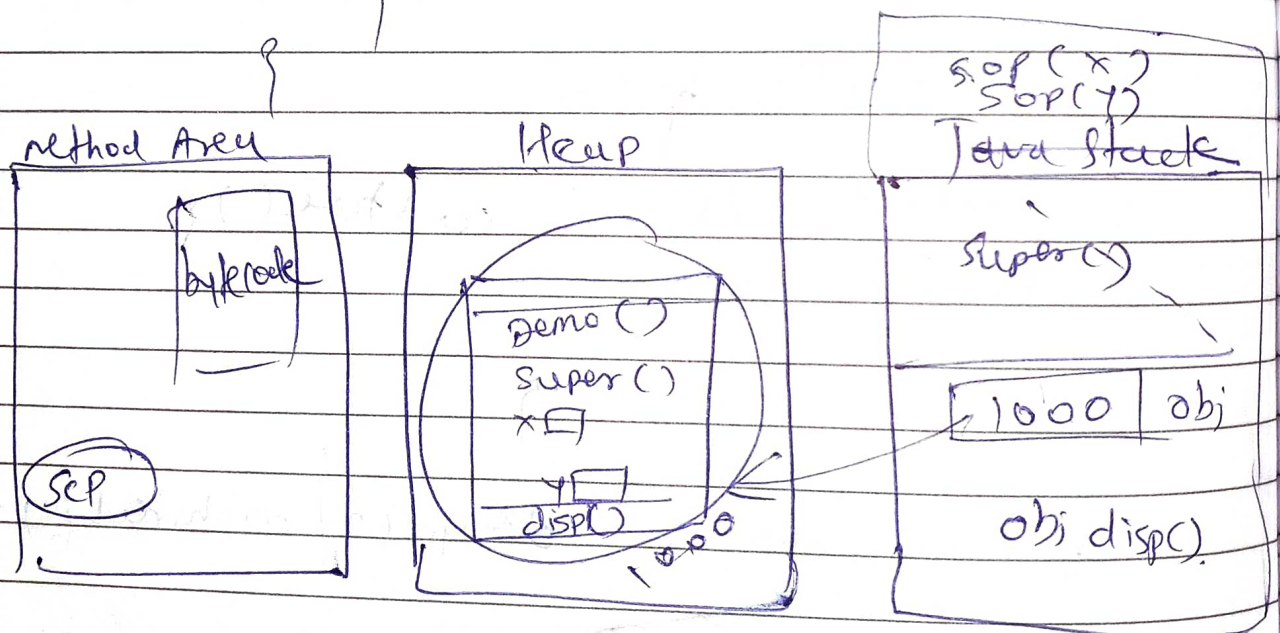
}

design() / architect() { }

Search on L). jira software

Sprint

```
class Demo{
    int x=10;
    int y=20;
    void display(){
        sop(x);
        sop(y);
    }
    p s void main(String[] args)
    {
        Demo obj = new Demo();
        obj.disp();
    }
}
```

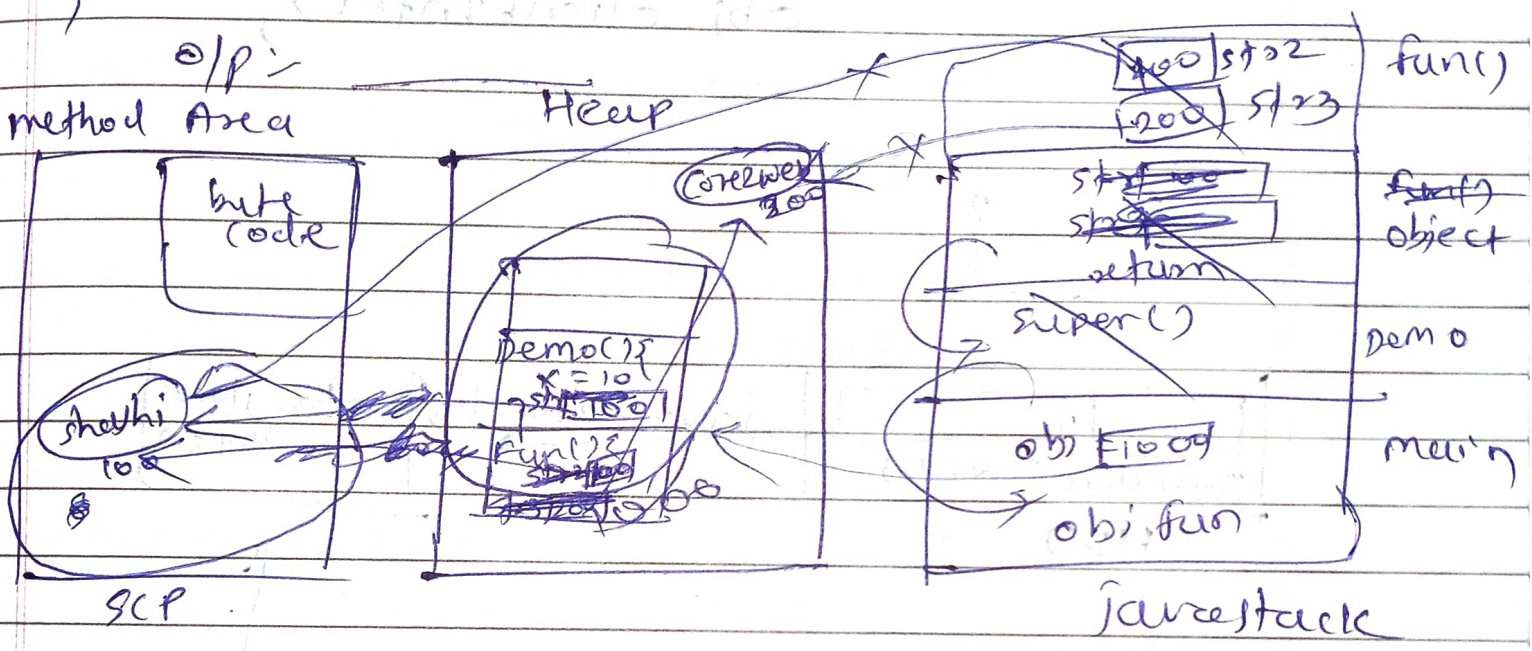



```
class Demo {
    int x= 10;
    String str1= "shaikh";
    void fun() {
        String str2= "shaikh";
    }
    String str3= new String(Capgemini);
```

~~Class~~

```
Public static void main (String[] args) {
    Demo obj = new Demo()
    obj.fun().
}
}
```

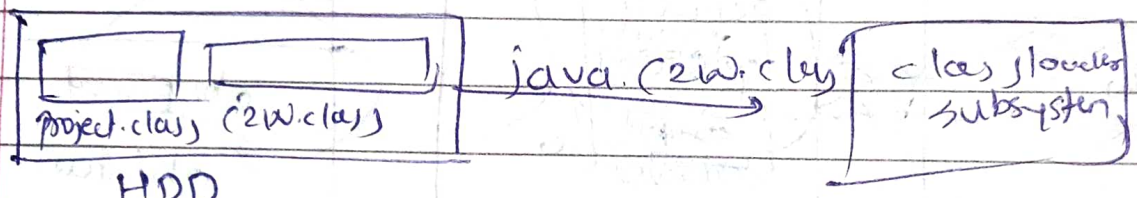
o/p :-

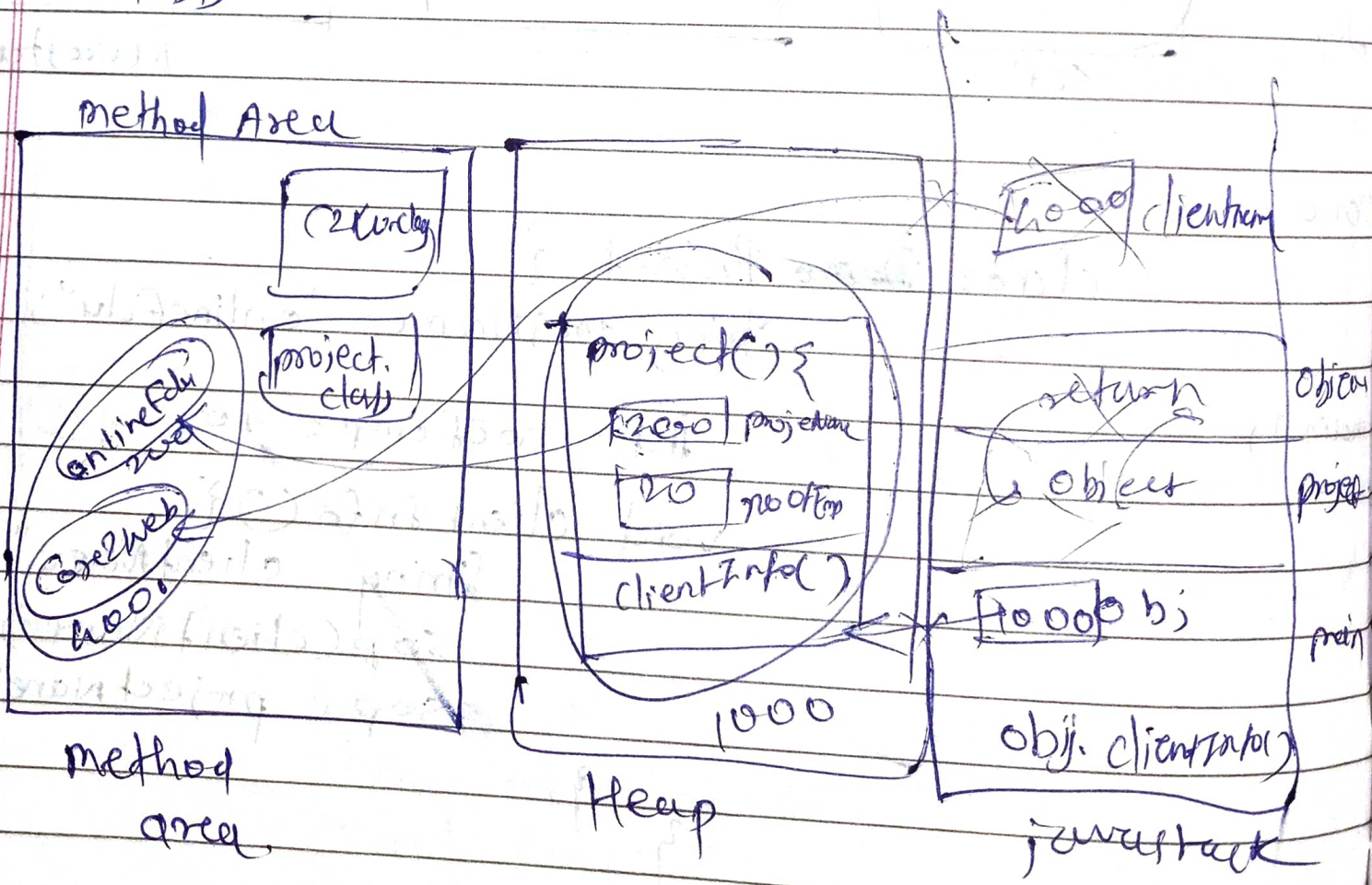



```
class ~~Demo~~ Project {
    String projName = "onlineEdu";

    int noOfEmp = 20;

    void clientInfo () {
        String clientName = "Core2web";
        sop(clientName);
        sop(projectName);
    }
}
```

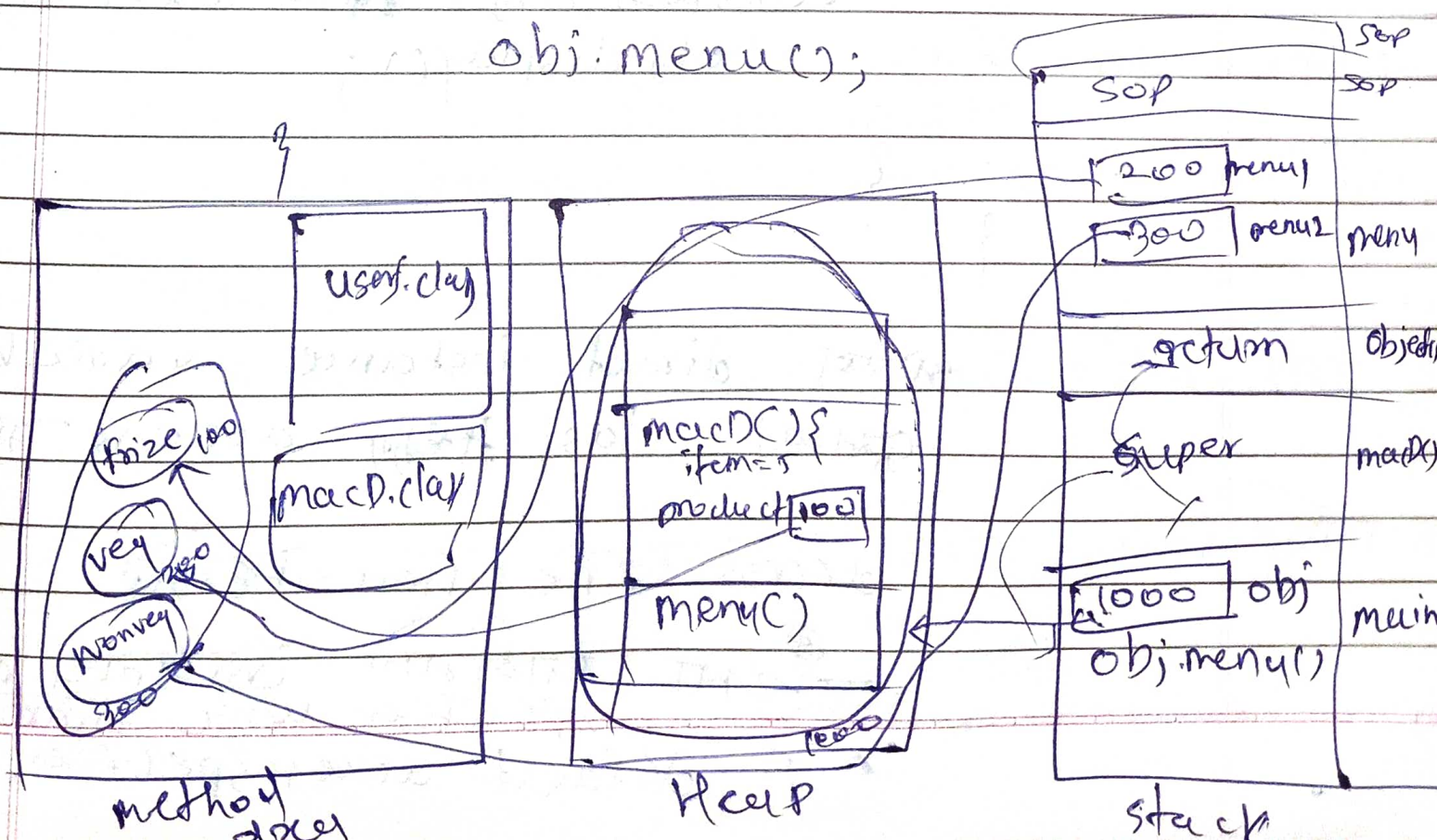
enduser

```
class C2W {
    p s v m (String [] args)
    {
        project obj = new project();
        obj.clientInfo();
    }
}
```





```
class MacD {

    int item = 5;
    String product = "Frize";

    void menu(){

        String menu1 = "veg";
        String menu2 = "Nonveg";

        sop(item);
        sop(product);

    }

class Users {

    public static void main(String[] args){

        MacD obj = new MacD();

        obj.menu();

    }
```



real time examples
write & practice it.

Page No.
Date

iect 50: constructor 03

```
class Core2Web {

    int noofCourses = 9;

    String favorite = "Java";

    void display() {

        sop(no-of couree);
        sop(favorite);

    }

class users {

    public static void main(String[]args){

        Core2Web obj = new Core2Web;
        obj.display();

    }

}
```

direct instance variable class access

access specifier

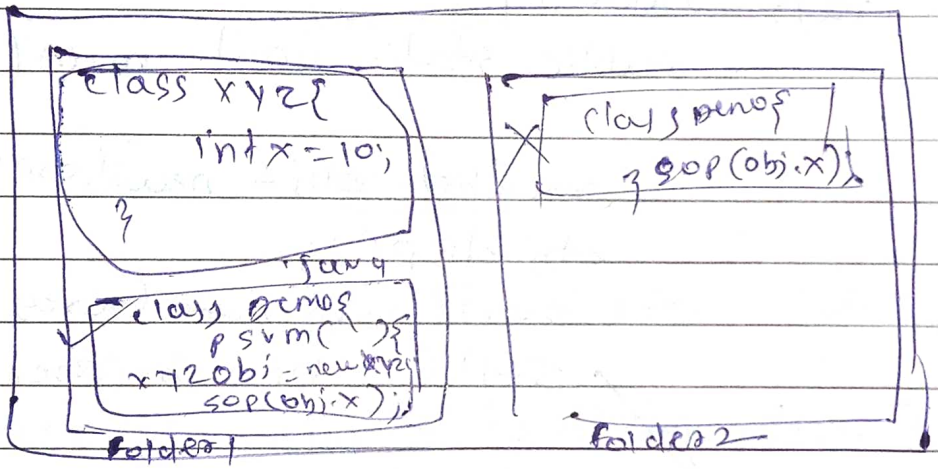
default access specifier

fiverr
medical application

Access specifiers

default private public protected.

main method multiple लिहू शकतो But entry point फक्त असतो जो JVM ला सापडतो तो फक्त कोणत्यातरी class मध्ये लिहायचा असतो.



default access specifier फक्त folder मध्ये :-
- फक्त file मध्ये पण चालतो.
- दोन वेगवेगळ्या file मधून पण access होतो.

But.
वेगळ्या folder मधून access होत नाही.

Private ला अजिबात लहान scope असतो.

Access Specifiers

```
class Core2Web {
    int numCourses = 8;
    private String favCourse = "CPP";

    void disp() {
        sop(numCourses);
        sop(favCourse);
    }
}

class Student {
    public static void main(String[] args) {
        Core2Web obj = new Core2Web();
        obj.disp();
        sop(obj.numCourses);
        sop(obj.favCourse);
    }
}
```

error : favCourse has private access in Core2Web class

```
sop(obj.favCourse);
```

→ String class मा access करता येतो आणि object पण बनवू शकतो.

→ परंतु System class मा access करता येतो पण object बनवू शकत नाही कारण त्याचा constructor private आहे.

System s = new System();

error: System() has private access in System

System s = new System();
^

design pattern (HLD, LLD) → low level design.
→ आपल्या concepts
→ architecture

example : instagram

server कायम चालू असतो.

Same class चा object कितीही वेळा बनवला तरी तो एकच गोळा बनवतो आणि नवीन गोळे बनवतो पण address तोच देतो. तो एक design pattern त्याचं नाव singltan design pattern → यालाच server लिहीतात. वापरतात.

नवीन गोळे बनलेच नाही पाहिजेत

Singletan design pattern मध्ये आपल्या class चा constructor private केला जातो

तिथं वेगळ्या पद्धतीनं object बनवला जातो.

Lect 51 seen :- 20-7-23

Access specifiers - 01

class साठी ~~public~~ package बनवायचा लागतो.

```
class Demo {
        int x = 10;
        private int y = 20;
        void fun(){
                sop(x);
                sop(y);
        }
}

class MainDemo{
    public static void main(String[] args){
        Demo obj = new Demo();
        obj.fun();
        sop(obj.x);
        sop(obj.y); → error, private access
        sop(x); error →
        sop(y); error →
    }
}
```

हा variable हा class मध्ये दिसतो (← private int y)

error → ~~non static~~ variable, non ~~static~~ variable — cannot find symbol — दोघांकडे class

कंपनीमध्ये 99% डेव्हलपर्स use केले जाणार.

example: eept o , crypto

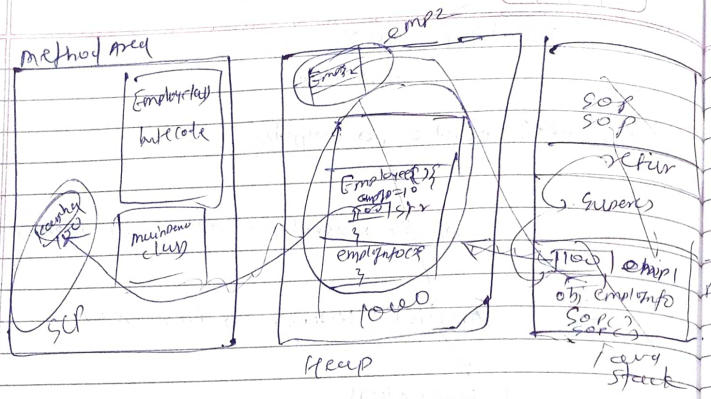
swift → developer ला खूप मागणी.

onclick listener → in project

Parameterized inner class.

```
class Employee {

    int empId = 10;
    String str = "Kanha";

    void EmpInfo {
        sop("Id= " + empId);
        sop("Name=" + str);
    }
}

class MainDemo {

    ps v main(String [] args){
        Employee empl = new Employee();
        empl.EmpInfo();
        sop(empl.empId);
        sop(empl.empId);
    }
```

VC → venchar capital
→ investors



object मध्ये data तरी same असला तरी ते instance variable स्वतःचे असतात.

गोष्ट की द्वारामध्ये mobile असतो प्रत्येकाचा.

Black rock → company

Pankaj याचा company
zepto ⇒ 20-age
Pratty
Silicon valley
founder पिचर
Uber story
Super pub
Social network on zuker berg.
wulf of wall state

shashank redom
titanic
Stefan

ज्याला ज्याला जागा असते त्याला object असतो.

class मध्ये जागा लागते object ला असते.

```
class Employee {

    int empId=10;
    String name = "kanha";

    void empInfo(){

        Sop("Id=" + empId);
        Sop("Name=" + name);
    }
```

```
class main {

    p s v main(String [] args){

        Employee emp1 = new Employee();
        Employee emp2 = new Employee();

        sop(emp1.empId);
        Sop(emp2,
          emp1.empInfo();
          emp2.empInfo();
          { emp2.empId= 20;
changed by  { emp2.name = "Rahul";
second      {
object.       emp1.empInfo();
              emp2.empInfo();

    }
```

→ static variable सगळ्यांसाठी common असतात.

→ समजा कोणत्याही object ने static variable ला change केला तर तो सगळ्यांना object मध्ये change दिसणार आहे.

→ static variable method area

```
class Employee {
    int empId = 10;
    String name = "kanha";          } Instance var → object
    static int y = 50;              } common for all obj

    void empInfo(){
        sop("Id=" + empId);
        sop("Name=" + name);
        sop("y=" + y);
    }

class mainDemo{
    p s v main(String[] args){

    Employee emp1 = new Employee();
    Employee emp2 = new Employee();

    emp1.empInfo();
    emp2.empInfo();
    sop("-----");
    emp2.empId = 20;
    emp2.name = "Rahul";
    emp2.y = 5000;
    emp1.empInfo();
    emp2.empInfo();

}
```

O/P:

```
10
kanha
50
10
kanha
50
-----
10
kanha
5000
20
Rahul
5000
```

examples.

Boolean value

sir or Board ~~के~~ static

student ⇒ ~~के~~ instance variable

static variable mummy.

static variable jessi

instance variable ⇒ jerseyNo.

class watsappGrp {

static admin.

}

class petrolPump. {

static prize =>

~~static~~ petrol => instance

class library {

person id

book static

class Aircraft {

static stairs

class TPL {

static constable

instance variable results

class farewell {

no. of stud =

static JPLvar =

seen → 21-7-25

lect 52 Access specifiers - 02

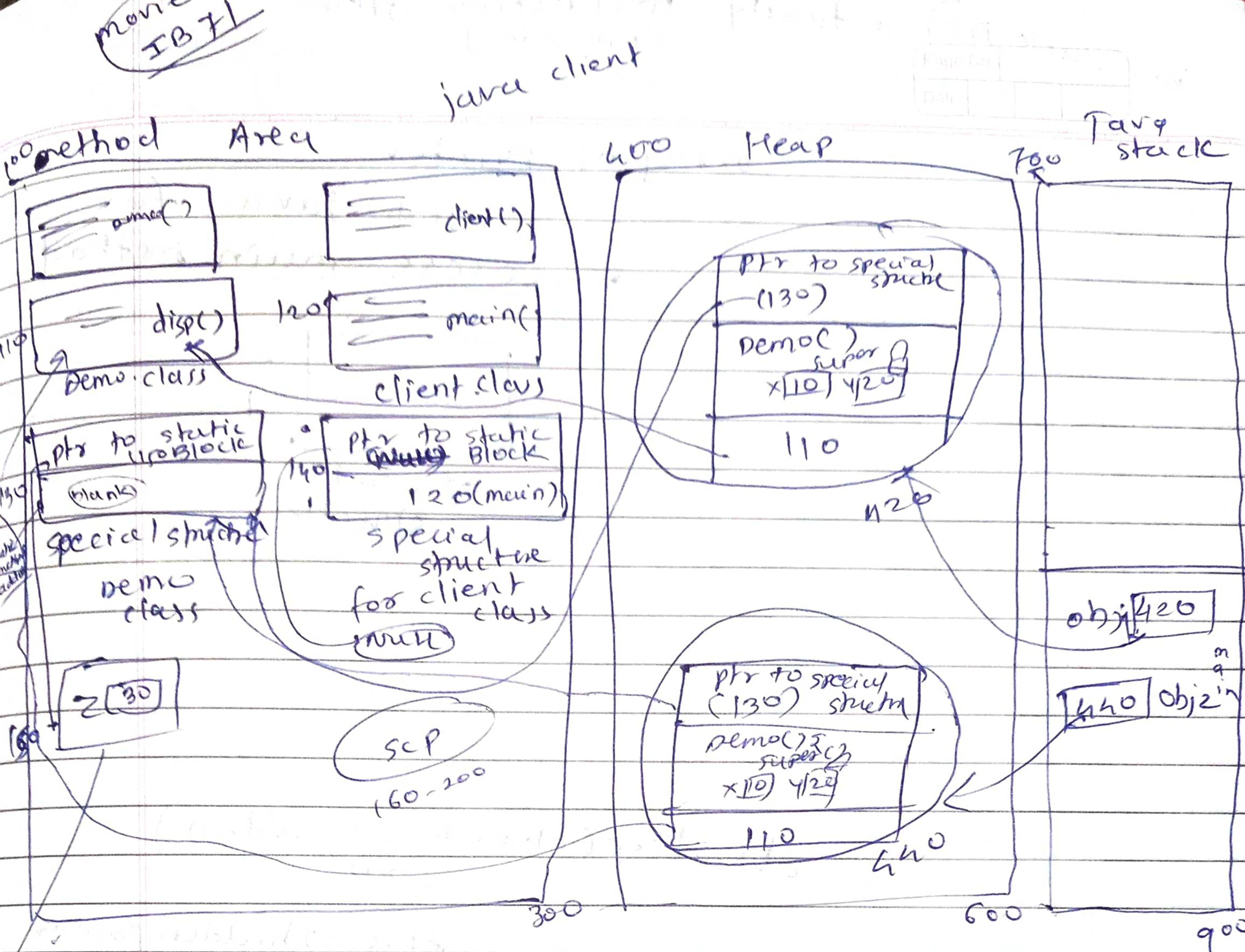
```
class Demo{
        int x=10;
        private int y=20;
        static int z=30;
        void disp(){

                sop(x);
                sop(y);
                sop(z);
        }

class Client{
        p s v main(String[] args){

        Demo obj1 = new Demo();
        Demo obj2 = new Demo();
        obj1.disp();
        obj1.x=100;
        obj2.z=300;

        obj1.disp();
        obj2.disp();

}
```

static variable ∈ class variable

Demo.z=40

without object

change



java म्हणा की मागच्या सगळ्यात आधी static गोष्टी load होतात.

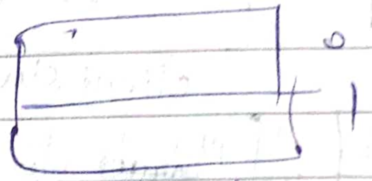
→ सगळं असं internally sequence नाही नसतं सगळं विखुरलेलं असतं त्यांना address नी बांधून ठेवलेलं असतं

→ हे mem शिकण्यासाठी representation आहे

→ method Area कुठेपण असू शकतो, java stack पण कुठेपण असू शकतो (असं नाही की java stack चा Box उशाच असतो).

java stack मध्ये actually instruction जातात (byte code).

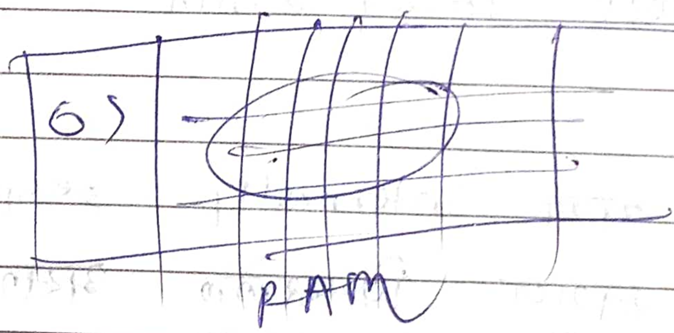
array of pointer
used java for
to access main method



special structure.

obj disp(object चा address)
420

this → hidden pointer.



(52) है ख्य भारी स्तोम (double कहीय ये ये)

lect 13 seen → 21-7-22

static variables and static methods

६ diagrams springboot शिकताना उपयोगी पडणार आहेत.

```
class staticDemo{
    static int x=10;
    static int y=20;
}
```

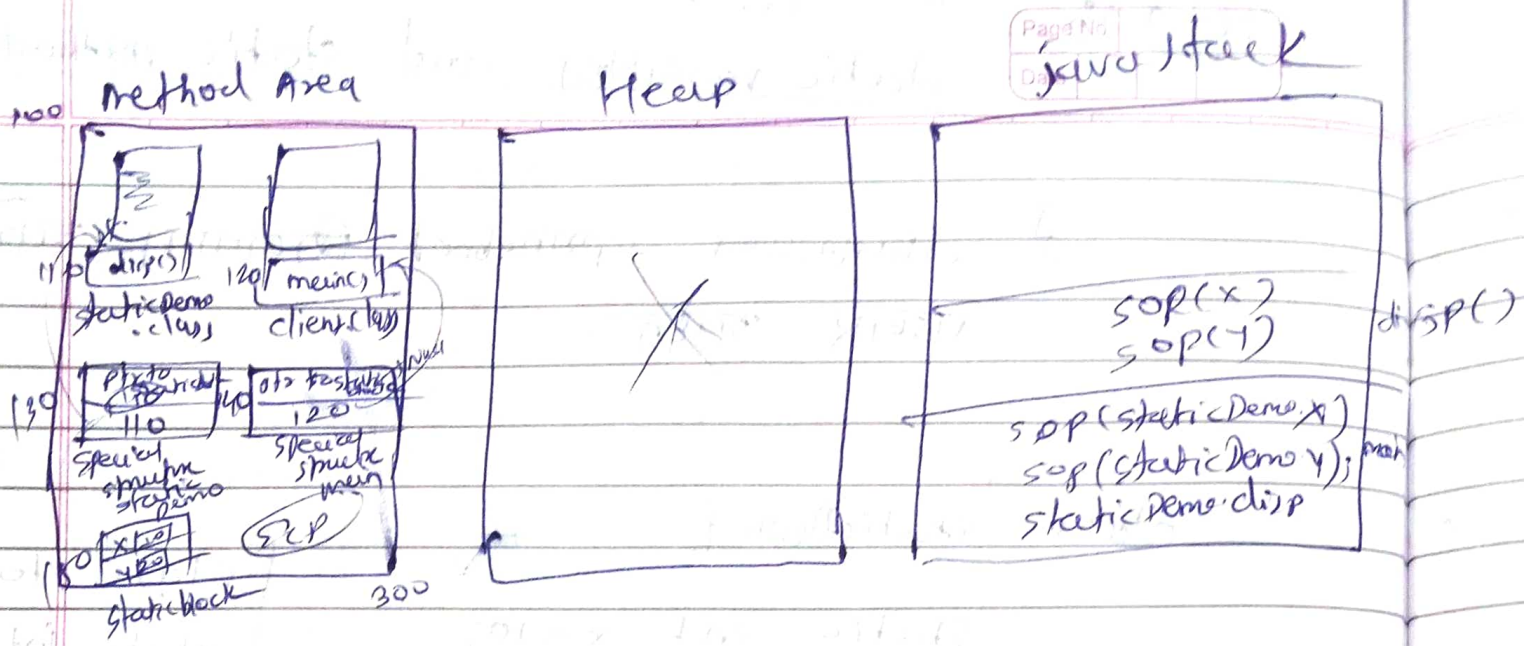
⇒ Compiler ⇒

```
class staticDemo{
    static int x;
    static int y;
    staticDemo(){
        super()
    }
    static{
        x=10
        y=20
    }
}
```

1.5 पर्यंत ~~main~~ without main चालत असत code लिहीत होते.

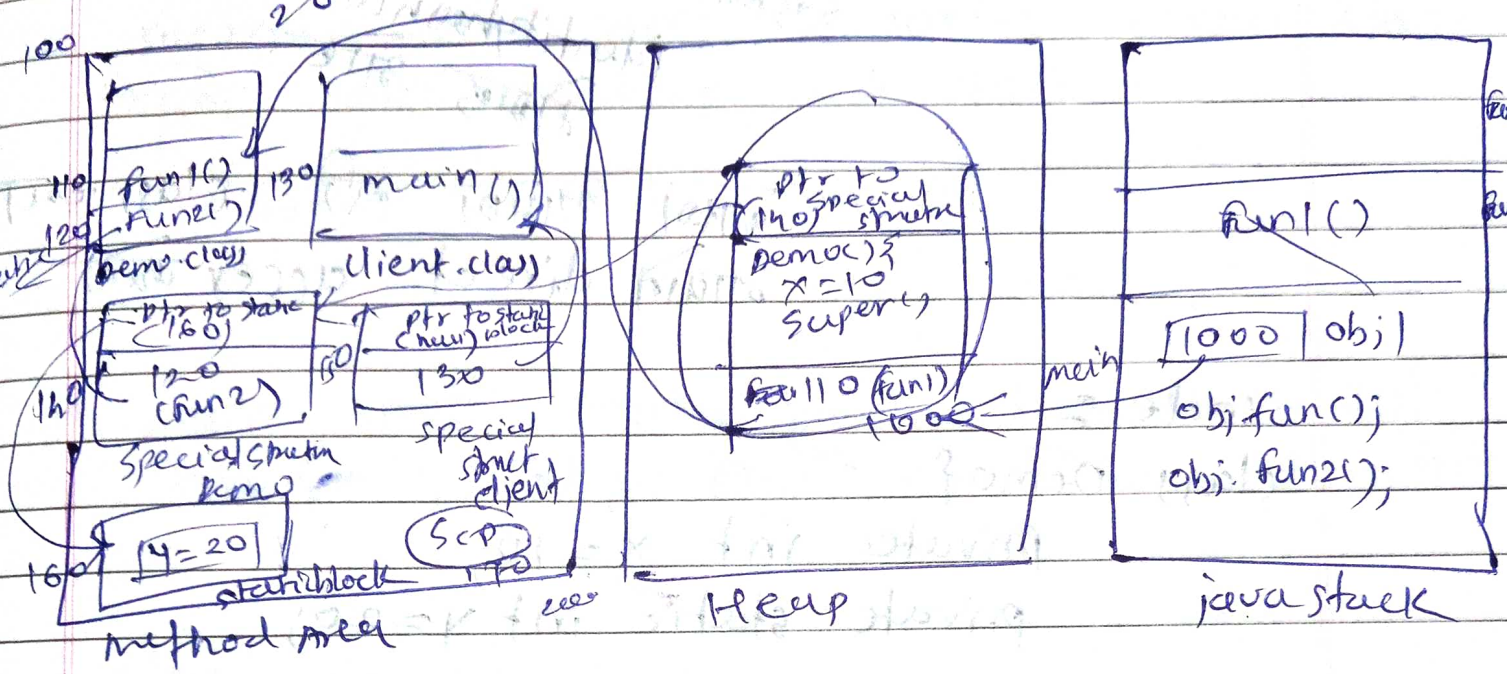
static block नव्हते.

```
class StaticDemo{
    static int x=10;
    static int y=20;
    static void disp(){
        sop(x);
        sop(y);
    }
class client{
    p s v main(String[] args){
        sop(StaticDemo.x);
        sop(StaticDemo.y);
        staticDemo.disp();
    }
```




```
class Demo{
    int x=10;
    static int y=20;
    void fun1(){
        sop(x);
        sop(y)
    }

    static void fun2(){
        sop(y);
    }

class Client {
    p s v main(String[] args){
        Demo obj1 = new Demo();
        obj.fun1();
        obj.fun2();
        sop(obj1.x);
        sop(obj1.y);
    }
```

o/p ⇒ 10
20
20
10
20



write real time example.

1-454

sesn:- 24-7-23

static method

System class मधल्या identityhashcode ही static आहे.
~~मेथड~~

~~कोणा~~ आपण ती class च्या नावाने direct access करतो.

code:

```
class Demo{
    private int x=10;
    private static int y=20;

    void fun1(){

        sop(x);
        sop(y);
    }
    static void fun2(){
        sop(y);
}   }

class Client {
    public static void main(String[] args){
        Demo obj = new Demo();
        obj.fun1();
        obj.fun2();
    }
            // Demo.fun1(); error //
            // Demo.fun2();
```

↓ non static method cannot be referenced

static block

example ⇒ concert of singers

method area वरती सगळ्यांचे address असण setup असते म्हणजेच.

↳ म्हणजे initialization नाही.

⇒ main आल्यावर static block java stack वरती जाऊन initialize होतो pop होऊन return येतो.

```
class Demo{
    static{
        sop("static block");
    }
public static void main(String[] args){
    sop("main method");
}
}
```

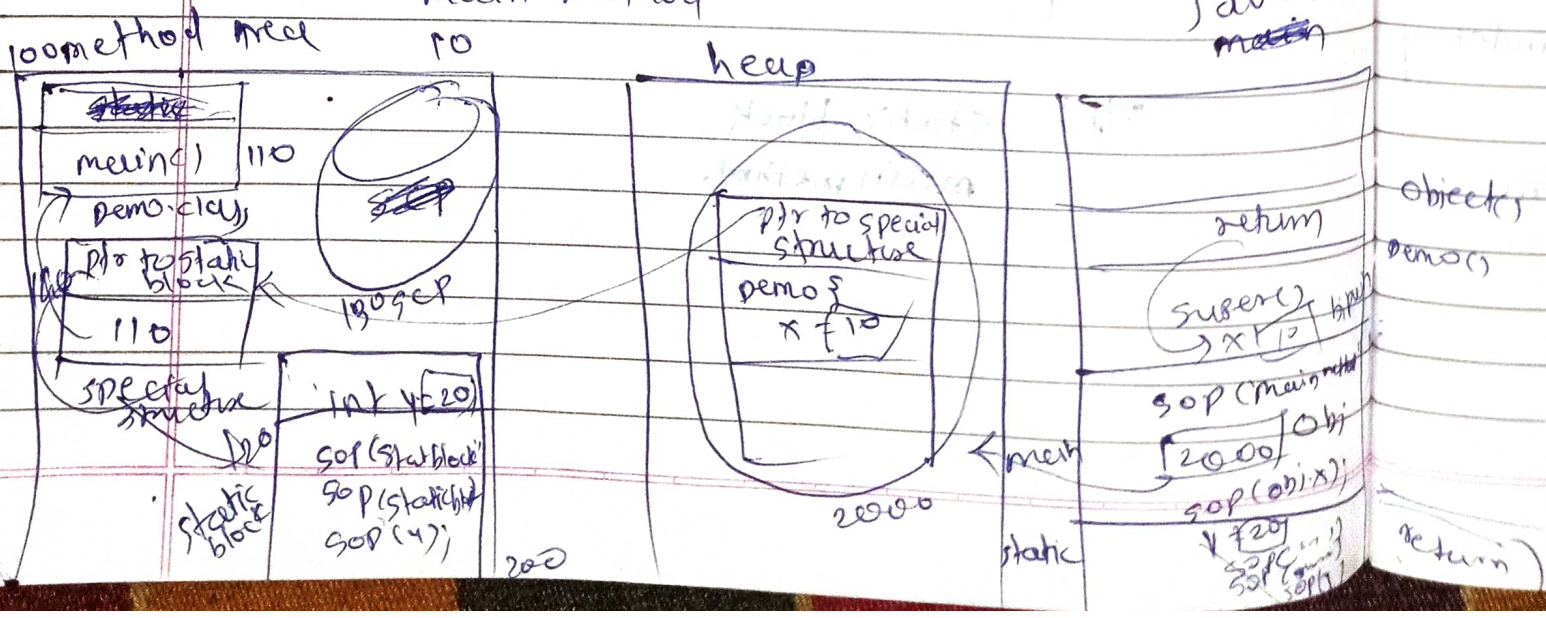
o/p: static block
main method.

Cabinet

```
class Demo{
    int x = 10;
    static int y = 20;
    static{
        sop("static block 1");
    }
    public static void main(String[] args){
        sop("main method");
        Demo obj = new Demo();
        sop(obj.x);
    }
    static {
        sop("static block 2");
        sop(y);
    }
}
```

o/p:-

static block 1
static block 2
20
main method
10



static block वर return jvm ला (परतवतो / return करतो)

जर दोन्ही class मध्ये static block असतील तर कोणता आधी run होणार ?
My it home

static block का येतो ?

→ जर तो static variable ला ✱ नाही static block मध्ये मिळतो. कारण static variable हे class variable असतो.

```
static {
    sop("Hellow world");
    System.exit(0);
}
```
→ static method

// error: for main method.

```
p s v main(String[] args){
    sop("In main method");
}
```

op: Hello world.

lect55 seen 25-7-23

static block

static कोणत्याच पेरान लिहू शकत नाही
तो फक्त class मध्ये पेरान लिहू शकतो.

```
static {

    int x = 10;

}
```

static block कशासाठी असतो?
→ it is used for like → connection established
→ socket connection establish
→ data read करण्यासाठी → file open
→ JDBC connection
1. static block main method च्या पण आधी होतो execute
2. There is no need of main method
3. multiple static block are merged (sequencewise) but for start flow code main method is mandatory
4. static variable automatically in static block
5. static block main शिवाय उपयोगी ठरतात programming मधला setup ready करण्यासाठी केला जातो.

```
class Demo{
    static{
        sop("static Block1");
    }
    Public static void main(String[] args){
        sop("In Demo main");
    }
class client {
    ~~static block~~
    static{
        sop("static Block2");
    }
    Public static void main(String[] args){
        sop("In main");
    }
    static{
        sop("static Block3");
    }
```

→ java client
o/p → staticblock2
staticblock3
In main

→ Static block array class एकच्या असल्या class कोड inherit होत नाही.

→ java Demo
→ Static Block1
In Demo main

initialization नंतरच तेव्हा stack push होते.

तीन च गोष्टी static असतात
① static methods
② static variables
③ static block
④ static inner class.

```
class Demo {
        static int x=10;
        static {
            static int y=20;        ← error:- illigal start of expression
        }
        void fun() {
            static int z=20;        ← error:- illigal start
        }
        static void gun() {
            static int z=20;        error:- illigal start
        }
}
```

हा घाट येतो त्यामुळे हा error होतो.

because → of java supports global static variable
global static ↓ class variable

→ हेच कारण top ला static variable आहे.
priority

ipl च्या तनक कमीन मोदि ओढ ☺

static variable ~~स्टेज~~ initialized in static block होतो.

* **Instance**
  (class, object)
  - instance variable (Global non static)
  - instance block
  - instance method
    - constructor → implicit call / automatic call
    - non static method → आपल्याला उचलावा लागतो

- static
  - static variable (class variable)
  - static block → main → implicit call
  - static method → static methods

→ instance method ला आपण सहज स्वताच्या गोष्टी access होतात कारण ती यांच्या आहेत स्वताच्या ~~गोष्टी येतात~~.

→ ~~Instance~~ method ची code ⇒

```
class Demo{
    int x=10;
    void fun(){
        sop(x);
    }
    void gun(){
        fun();
    }
```

~~Instance block~~ :→
↳ सारखी concept थोडीशी constructor सारखा असतो आहे:

```
{
    sop("instance Block");
}
```

→ instance block constructor मध्ये merge होतो ⇒ invokespecial आणि instance variable ~~नंतर~~ नंतर.

```
class Demo {
    int x = 10;
    Demo() {
        sop("Constructor");
    }
    {  // instance block
        Sop("Instance Block1");
    }
    public static void main(String[] args){
        Demo obj = new Demo();
        sop("main");
    }
    {
        Sop("Instance Block2");
    }
```

See the Bytecode

sequence :→
1. static variable
2. static block
3. static method
4. instance var
5. instance block
6. constructor
7. instance method.

```
class Demo {
    int x=10;
    static int y=20;
    Demo(){
        sop("In Constructor");
    static{
        sop("In staticblock1");
    }
    {
        sop("In instance block1");
    }
    public static void main(String[] args){
        Demo obj = new Demo();
    }
static{
    sop("In static block2");
    }
    {
        sop("In Instance block2");
    }
```

O/P :-
In static block 1
In static block 2
In Instance block 1
In Instance block 2
In constructor

write or Draw Diagram of code

lect 56 seen 26-7-23

static context

heading: **Constructor**

① constructor f

② constructor ला return type नसतो even void पण नसतो. जर void दिला तो normal method सारखा treat करतो.

→ is special method.

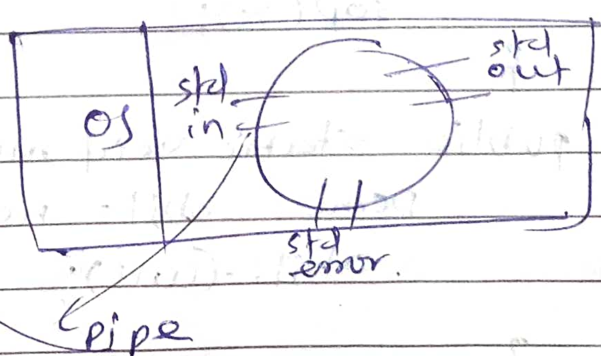
→ class च नाव आणि method नाव same असू शकतं. (void Demo() { })

→ जर resume मध्ये CPP पण लिहली असेल तर cpp चा constructor आणि java च्या constructor difference विचारू शकतात.

→ java मध्ये destructor का नाही कारण Garbage collector आहे.

→ Phenolix आणला पाणी शोषत. पिवळी सोन्यासारखी

जाहिरात



ज्या गोष्टी run ~~stack~~ होण्यासाठी stack frame वर जातात त्यांना method म्हणतात. constructor, static block method आहेत.

example सांगितला

interview मध्ये ← ही diagram काढण्यासाठी os कडे interviewer ला नेण्यासाठी.

→ main नंतर जे कोणी येतात ना त्या सगळ्या methods हा concept आला आहेत. सगळे कंपनीत discussion मधून

→ Constructor main च्या आधी पण आला static block मधून object बनवून शकतो. ही jvm architecture मध्ये लिहिलंय.

→ But protocol — main ही पहिली method मानली.

→ कोणत्याही कंपन्यामध्ये static block मधून object बनवून देत नाहीत.

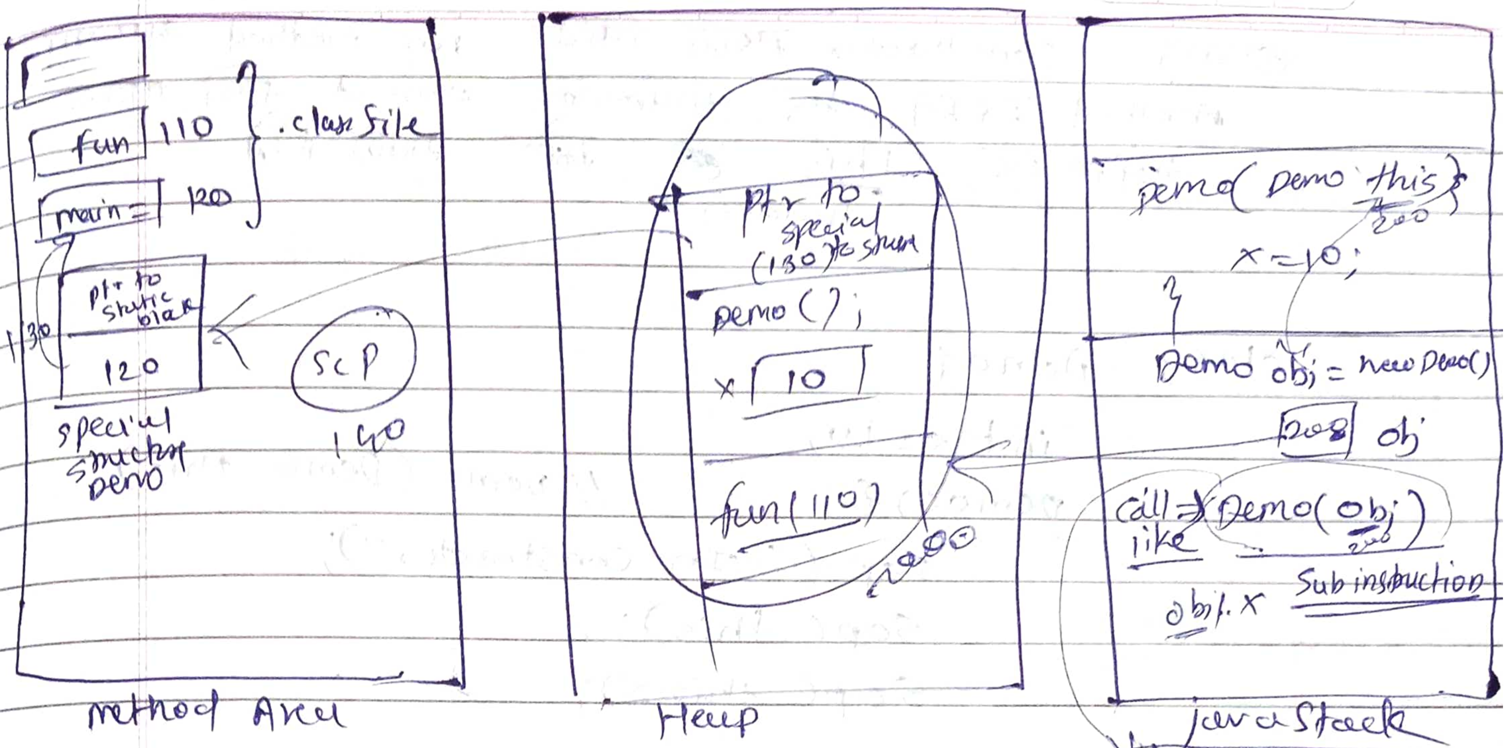
→ main च्या आधी येतात ना ते block असतात. हे protocol मध्ये असतं आहे.

System class open करून बघा त्या मध्ये (bytecode) त्याने आपलं नाव आलं → it imports
- printstream class
- io class

```
class Demo{
        int x = 10;
        Demo(){
                sop("In constructor");
        }
        void fun(){              // fun(Demo this);
                sop(x);          this.x
        }
        public static void main(String[] args){
                Demo obj1 = new Demo();
                obj1.fun();   // call like
                              fun(obj1)
        }
}
```

Diagram.

new / duplicate } calls, when object created.



→ setup ~~directly~~ असतो → method area.
→ runtime ला त्याला request पाठवत असतात }

sop(this.x);

* Hidden this reference (pointer)

fun(int var)

int x = 10;

fun(x)

fun(Demo var)

Demo obj = new Demo();

fun(obj)

this मध्ये जे nonstatic आहेत त्यांना लागतो.

→ static मध्ये नसतो

javascript मध्ये DSA करतो येत

→ आपले समजले instanceof nonstatic असतात त्यांना this pointer parameter असतो.

① main मेथड मधून आपण directly object पर्यंत जाऊ शकतो कारण त्याचा address आपल्याकडे आहे (obj).

② पण constructor मधून किंवा fun method सारख्या method मधून जर आपल्याला object मध्ये जायचं असेल तर This मुळे शक्य होते
hidden

```
class Demo{
        int x=10;
        Demo(){                    // Demo(Demo this)
            sop("In constructor");
            sop(this);
            sop(this.x);
        };

        void fun(){                 // fun (Demo this).
            sop(this);
            sop(x);
        }.

        Public static void main(String[] args){
            Demo obj1 = new Demo(); // Demo(obj1)

            sop(obj1);
            Obj.fun() ;             // fun(obj1)
        }
}
```

→ core2web application खूप मोठं आहे.
→ सरांनी बाहेरच्या देशातील Healthcare application वर काम केलंत त्याचे त्या कंपनीला 70-80 कोटी भेटले होते. त्या level चे core2web application आहे.

DSA चे 12 topic आहेत.

→ This data type हे class चे नाव असते.

→ local variables जन्माला त्याच्या stack frame वर येतात आणि मरतात त्याच्या stack frame वरती.

```
static {
    int x = 10;   → local variable
}
main (
    (x)           → error not accessible
)
```

⇒ Pointer मुळे आपला code (robust) राहत नाही.

private variable pointer ने change करू शकतो. in CPP म्हणून java मध्ये pointer concept नाही

⇒ static method ला call केला असता this parameter नसतो.

1. class चे name & constructor name is same.
2. object बनवल्यानंतर constructor ला call जातो.
3. constructor ला instance variable initialize करण्यासाठी पाठवला जातो.
4. constructor ला return type नसतो.
5. implicitly call जातो.
6. पहिली line invokespecial असते.
7. compiler ने दिलेल्या constructor ला default + आणि आपण दिलेल्या constructor No argument constructor म्हणतात

actually java में parameterized constructor इसलिए type का आता है because of this reference

(8)

Dem( ) {

}

Demo (int x);   (this, int x)

Demo obj = new Demo(10)
call like
Demo(obj, 10);

Constructor

```
class Demo{
    Demo(){
        sop("No args");
    }
    Demo (int x) {
        sop("para");
    }
    Demo(Demo xyz) {
        sop(para Demo)
    }
    public static void main(String [] args){
        Demo obj1 = new Demo();
        Demo obj2 = new Dem(10);
        Demo obj3 = new Demo(obj1);
    }
```

this → xyz, obj1 → Demo (same)

o/p:- No args
para
para Demo.

Lect 57

Sun: 27-7-23

this reference

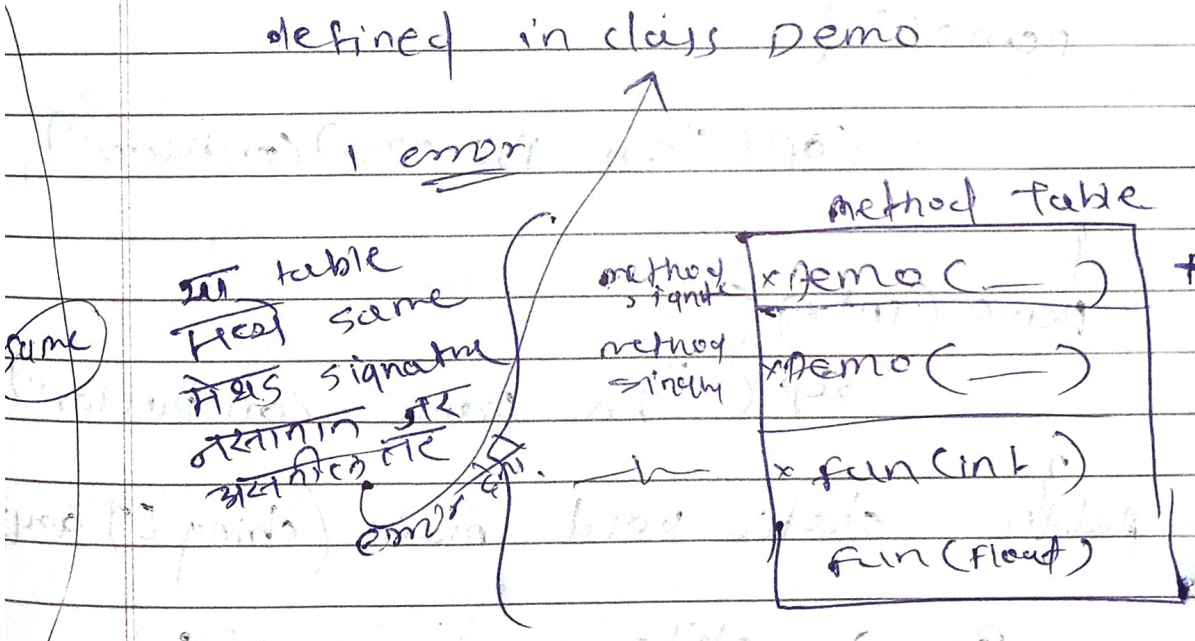
* method signature ⇒ (table)

```
class Demo{
    int x=10;
    Demo(){
        sop("In constructor");
        sop("x = " + x);
    }
    Demo(){
        sop("In constructor 2");
        sop("x = " + x);
    }
}
```

after compilation ⇒ error

constructor Demo is already defined in class Demo

1 error



→ method table compile time ला बनवला जातो & त्यात सर्व method signature ठेवल्या जातात.

→ method signature मध्ये method चे नाव आणि parameter असतात. ret तिथे return type असतो

C मध्ये overloding नाही.
- CPP मध्ये ~~आणि~~ → name mangling concept
- Java → method signature concept

⇒ ambiguty error :-
fun(int x, float)
fun(float, int)

call like fun(10, 10)
हा data दोन्ही method मध्ये जातो ambiguty error

* * Hidden this reference (parameter) * *

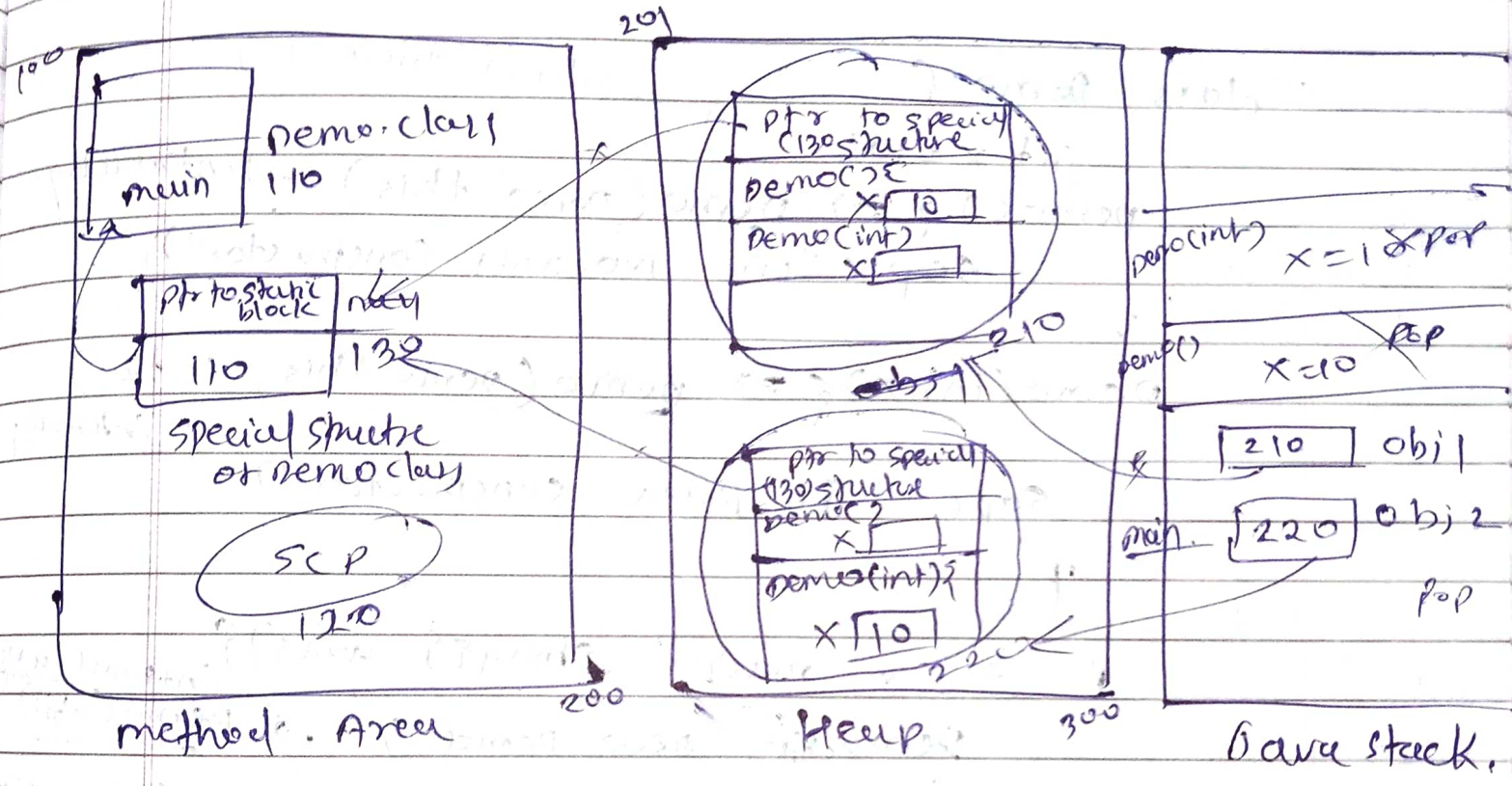
→ instance variable जेव्हा constructor बनवले असतील तेव्हाच मध्ये initialize होतो.

```
class Demo{
    int x=10;
    Demo(){
        sop("In No-args Constructor");
    }
    Demo(int x){
        sop("In Para constructor");
    }
    public static void main(String[] args){
        Demo obj1= new Demo();
        Demo obj2= new Demo(10);
    }
}
```

x → means after pop connections ~~stacks~~ are closed then garbage collector collect the no reference objects & variables

O/P:- In No-arg Constructor:
In Para Constructor:

Diagram ⇒



⇒ instance variable x=10 हो दोन्ही constructor मध्ये असतो But एका वेळी एकच stack frame push होते. तर x=10 हा त्याचा object असेल त्यांच्याकडे तो initialize होतो. नंतर obj2 बनला की तो तिथे initialize होतो.

⇒ नीतीन कामेट ⇒ 20% गोष्टी काम chat GPT करतय.

⇒ Content writer ⇒ ~~write~~ zero positions

cpp मध्ये this ची देखील * असतो.

Page No.
Date

* Hidden this reference ⇒ *

↳ internally pointer च असते

↳ variable name by java

```
class Demo {                          internally ⇒ this x = 10;
    int x = 10;
    Demo() {  ⇒ Demo(Demo this)  → internally
        sop("In No-args Constructor");
    }
    Demo(int x) { ⇒ Demo(Demo this, int x)
                                          internally
        sop("In para constructor");
    }

    p s v main(String[] args){
        Demo obj1 = new Demo();   → Demo(obj1)   internal call
        Demo obj2 = new Demo(10); → Demo(obj2, 10);
    }
```

* Two use of this ⇒ *

① instance variable initialize करण्यासाठी

② एका constructor मध्ये same class मधील दुसरा constructor ला call करण्यासाठी.

* this चा अर्थच होतो ज्या object ची instance म्हणजे object instance variable → object-ची variable

→ constructor instance variable initialize करण्यासाठी this reference ची मदत होते.

```
class Demo {
    int x = 10;
    Demo() {
        sop(this.x);
        sop(x);
        sop("In No-arg Constructor");
    }

    Demo(int x){
        this.x = x;
        this();   //→ error: call this statement first
        sop(this.x);
        sop(x);
        sop("In para Constructor");
    }

    P S V main(String[] arg){
        // Demo obj1 = new Demo();
        Demo obj1 = new Demo(50);
    }
```

→ जर आपल्याला एकच object बनवून दोन्ही Constructor ला call करायचा असेल तर this() चा वापर केला जातो.

→ यामुळे दुसरा object बनण्यापासून वाचतो म्हणजेच heavy operation वाचते.

```
class Demo{
    int x=10;

    Demo(){
        sop("In No-args Constructor");
    }
    Demo(int x){
        this();
        sop("In para Constructor");
    }
    p s v main(String[] args){
        Demo obj2 = new Demo(50);
    }
}
```

⇒ instance method में this का प्रयोग

```
Demo(){
    this(70);
}
Demo(int x){
    this();
}
```

Error : recursive constructor invokation.

⇒ See the bytecode if call the constructor by this ⇒

⇒ ज्या constructor मधून आपण call करतो तेथून त्याच्या parent class च्या constructor ला call जात नाही म्हणजेच object class ला.

⇒ super() व this() ही एका constructor मध्ये लिहू शकत नाही कारण super मुळे parent constructor ला call जातो पण this मुळे दुसऱ्या constructor ला कॉल जातो त्यामुळे error येते

super() } हे दोघांना पण
this() } पहिल्याच लाईन ला लिहावे लागते.

```
class Demo{
        private int x=10;
        Demo(){

        }
        void fun(){
                sop(x);
        }
}
class client{
        p s v main(String [] args){
                Demo obj1 = new Demo();
                obj1.fun();
        }
```

bytecode ← मध्ये declare च होत नाही

method ला private करून bytecode बघा.

Private variable, method असेल तर ती bytecode मधून गायब होतात म्हणजेच ते दुसऱ्या class available नसतात.

lect58 : seen 27-7-23

Inheritance Introduction

~~tech~~ constructor ~~ka~~ chaining → This() call
Technical

⇒ Pure oop oriented lang java
it follows classes

class मध्ये आता मध्ये 8 च point असतं शकतात.

* आपण compulsory केला method लिहिला
setter-getter methods
⇒ ° Constructors() ← ↳ it change variables ↳ it return/print values.

invoke special → call to parent's constructor
object.class constructor
↳ हा काही return करत नाही

object class मध्ये common method असतात
like toString()

int = 0;
String name = null; → in java
⇒ NULL; → in c

```
class Player{
        private int jerNO = 0;
        private String name = null;

    Player(){
        sop(" In Constructor");
    }
    void info(){
        sop (jerNO + " = " +name);

    }
}
class Client{
    public static void main(String[] args){

    Player obj1 = new player (18, "Virat");
    obj1.info();

    player obj2 = new player (7, "CSK");
     Obj2.info();

}   }
```

error: : Constructor Player in class player cannot be applied to given types:

Player obj2 = new Player(7, "MSD");

required: no arguments

found : int, String

reason: actual & formal argument lists differ in length.

2 errors.

```
class Player {
    private int jerNo = 0;
    private String name = null;
    Player(int jNo, String pName){

        jerNo = jNo;
        name = pName;
        Sop("In Constructor");
    }
    void info() {
        Sop(jerNo + "=" + name);
    }
}
class client {
    p s v main (String [] args){

        Player obj1 = new player(18, "Virat");
        obj1.info();
        player obj2 = new player(7, "msd");
        obj2.info();
        Player obj3 = new player(45, "Rohit");
        obj3.info();
    }
}
```

O/P: In Constructor
18 = Virat
In Constructor
7 = msd
In Constructor
45 = Rohit.

constructor स्था आतमध्ये initialization होते

initialization म्हणजे जेव्हा तुम्हाला मागा मिळते तेव्हाच कुणाला String assign होत

Istan code

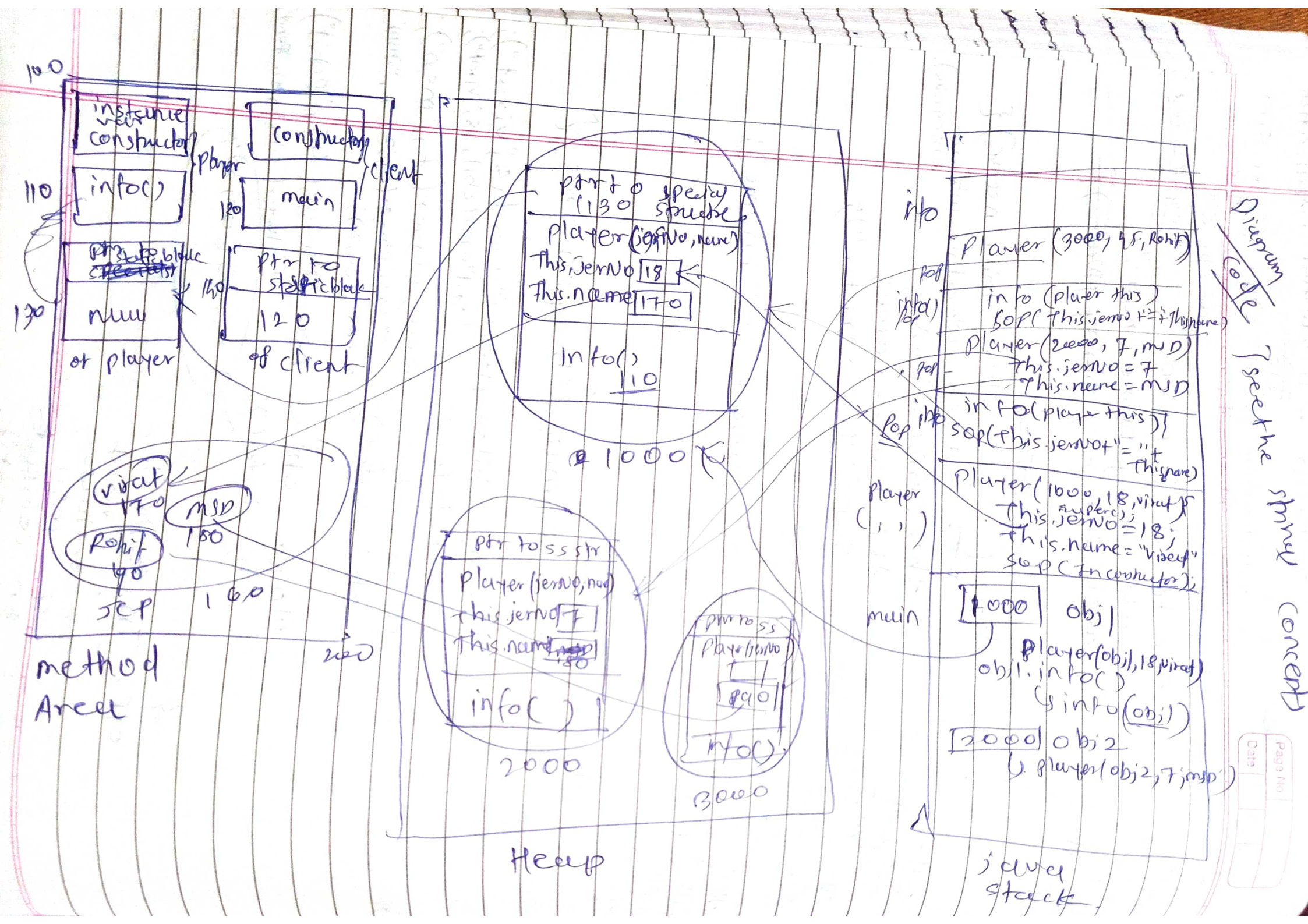
constructor initialization करतो नाही तर assignment.

```
class player{
    private int jerNo=0;              String str= "virat";
    private String name=null;
    ~~Player( int jNo, String pName){~~

    ~~this.jerNo=~~
    player( int jerNO, String name){
        // player(Player this, int jerNo, String name);
        this.jerNO= jerNO;
        this.name = name;
        sop("In constructor");
    }
    void info(){
        sop( jerNo + " = " +name);
    }
}
class Client{
    p s v main(String[] args){

        Player obj1= new Player(18,"virat");
        obj1.info();         call // Player(obj1, 18, "virat");
        player obj2= new Player(7, "MSD");
        obj2.info();         call // Player(obj2, 7, "MSD");
                             info(obj2);
        player obj3= new Player( 45, "Rohit");
        obj3.info();         // Player( obj3, 45, "Rohit");
                             info(obj3);
    }
}
```

setter method: ज्याच्या मध्ये data ची माहिती असते जरी private variable असले तरी class मध्ये data update करते

getter method: it prints instance variable of class

O/P: In Constructor
18 = virat
~~In constructor~~
7 = MSD
In constructor
45 = Rohit.


100
instance variable
constructor
info()
Player
110
120
main
Client
ptr to static block
nuu
120
130
140
of player
of client
ptr to special square
Player(jerNo, name)
this.jerNo 18
this.name 170
info()
110
@ 1000
virat
170
MSD
150
Rohit
90
JCP
160
200
method Area
ptr to ssstr
Player(jerNo, nam)
this.jerNo 7
this.name 150
info()
2000
ptr to ss
Player(jerNo)
@90
info()
3000
Heap
info
Player (3000, 45, Rohit)
pop
info()
info (Player this)
SOP(this.jerNo+"="+this.name)
Player(2000, 7, MSD)
this.jerNo = 7
this.name = MSD
info(Player this)
pop
info
SOP(this.jerNo+"="+this.name)
Player
(,)
Player(1000, 18, virat)
this.jerNo = 18;
this.name = "virat"
SOP(In constructor);
main
1000 obj1
Player(obj1, 18, virat)
obj1.info()
info(obj1)
2000 obj2
Player(obj2, 7, msd)
jvm stack
Diagram Code Speed the story (concept)

Inheritance

→ जर constructor private असेल तर बाहेरून त्याचा object बनत नाही.

→ Inheritance मध्ये Access specifier मोठा role play करतात.

→ class public केला तर file च नाव same असावं नाहीतर

error :⇒ class Demo is public, should be declared in a file named Demo.java

```
public class Demo{
       ^
```

1 error

reason :⇒ access the folder access

* IDE वरती class by default public असतात.

→ ज्या type चा class चा access specifier असतो त्याच type चा access specifier चा Constructor चा असतो.

* class वरतीही private नसतो.

```
private class IPL{

}
```

error : modifier private not allowed here

```
private class IPL{
      ^
```

1 error.

MNC → multinational company

protected class IPL {

}

error: same as private

static class IPL {

}

error → modifier static not allowed here.

⇒ class ला modifier default and public हे दोनच असतात.

⇒ byte code हा बघण्यासाठी representation असतो → जर private constructor बनवला तर byte code मध्ये दिसत नाही. याचा अर्थ असा नाही की तो बनतो. तो त्या class च्या members मध्ये दिसत असतो.

⇒ protected constructor बनवला त्याचा access फक्त child class कडेच असतो.

⇒ जर static constructor बनवला तर चालत नाही.

⇒ error: modifier static not allowed here

static IPL(){

| error

ICC → International cricket council

relative path
absolute path

in //
method javallang/object .<init>

method ICC. "Unity:

Inheritance ⟹

```
class ICC{
    ICCC){
        Sop(In ICC constructor");
    }
}
class BCCI extends ICC{
    BCCI(){
        Sop("In BCCI constructor");
    }
}
class client {
    public static void main(String[] args){
        BCCI jayshah = new BCCI();
    }
}
```

O/P ⟹ In ICC constructor
In BCCI constructor

lect 59 seen:- 28-7-23

Inheritance - 01

Blue learn → founder → Harish

oop → real time मादे.

→ parent class चे सगळे access child मध्ये होऊ शकतो पण child class चे parent मध्ये access नाही होत

example:- Food पासून पुढे काय करणार हे वडीलांना माहीत असते.

⇒ inheritance redundancy / repeat code कमी करणे.

→ BCCI → Board of control for cricket in India

example:- BMW / BMW versions

Mercedies / versions

Apple → iPhone

Alphabet → Google

example:- class team {

team name =
~~state of team~~;

}

class player extend team {

jerno

nam

}

Class Batsman extend player {

class Bowler extend player {

class Google {

into provide

}

class Hospital {

class Singhad Institute. {

library name

class ST<W& extends Singhad Institute

class Defence {

Army, Navy

class university {

class Supreme court {

child → state court {

class LIC

types of Insurance.

class [illegible] {

Rajysabha

class Reliences

jio

class coursewala

batches

class linux

ubuntu kali linux

class microsoft

word

class India {

states

class meta

facebook Insta facebook

class cloud

AWS Azure GCP

Parent class C langs

child class

oop language

Process scheduling

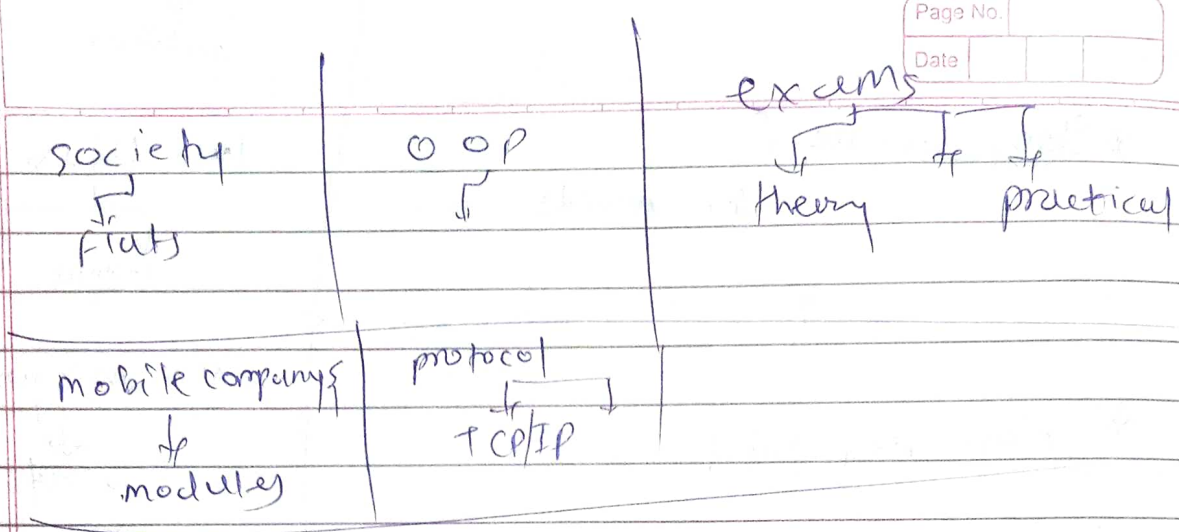
FCFS round job

class Amazon

amazon prime

ecomerce

server AWS



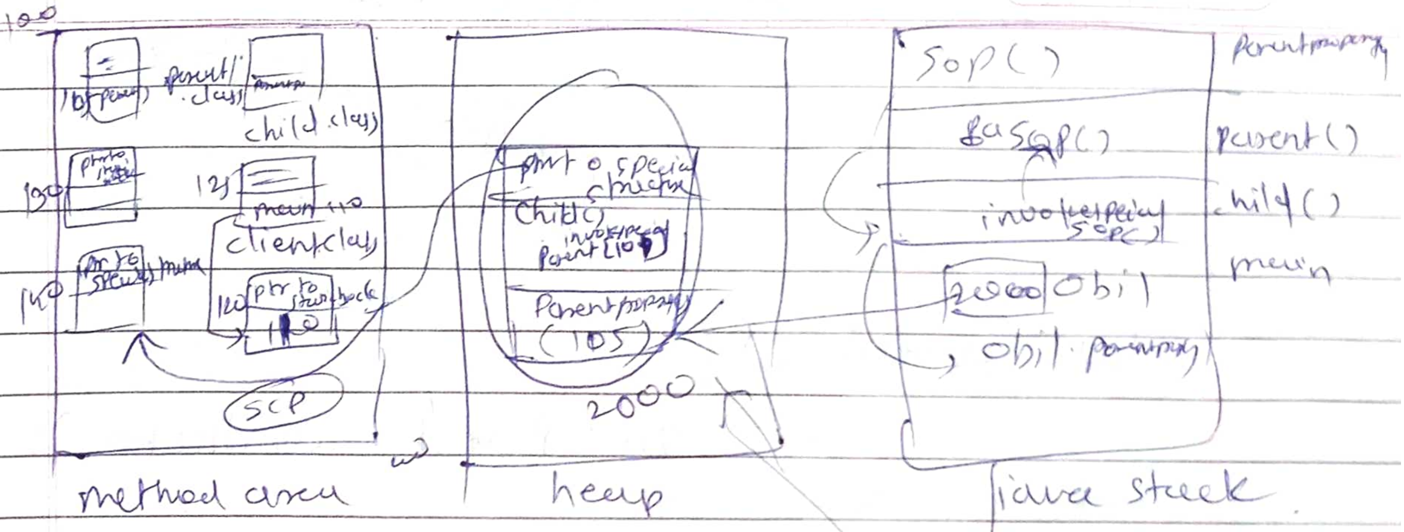

```
class Parent {
    Parent() {
        sop("In parent constructor");
    }
    void parentProperty() {
        sop("flat, car, Gold");
    }
}
class child extends Parent {
    child() {
        sop("In child constructor");
    }
}
class client {
    p s v main(String[] args) {
        child obj1 = new child();
        obj1.parentProperty();
    }
```

spring boot

Gaurav Sain

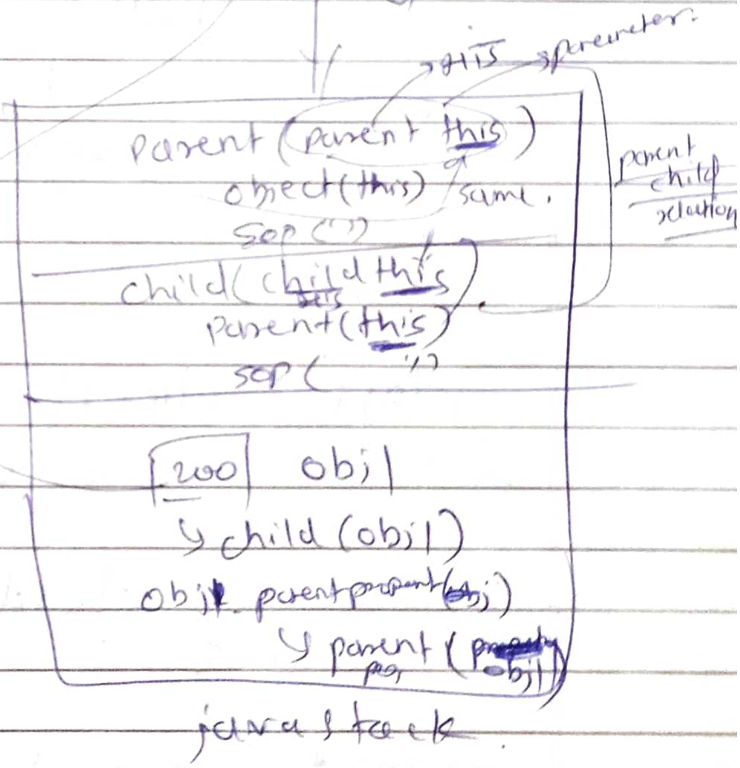
g design pattern

inheritance ला आहेत composition होते
↳ marie object बनवा



child obj = new child();
Parent obj = new child();
this

updated



spring boot

Gaurav sain → system design

check this

9 design pattern

Object obj = new child()

class आमची

Object directly & indirectly class ला parent असतो

parent child relation

→ java मध्ये parent ला reference आणि child ला object आमची

rect 60% seen 30-7-23

Inheritance - 02

init → initializer

```
class Parent (){
        sop("In Parent Constructor");
    }
}
class child extends Parent{
    child{
        this(); super();
        sop("In child Constructor");
    }
}
class Client{
    P s v main(String [] args){
        child obj = new child()
    }}
```

error: call to super must be first statement in constructor.

this(); super();

error: recursive Constructor invocation

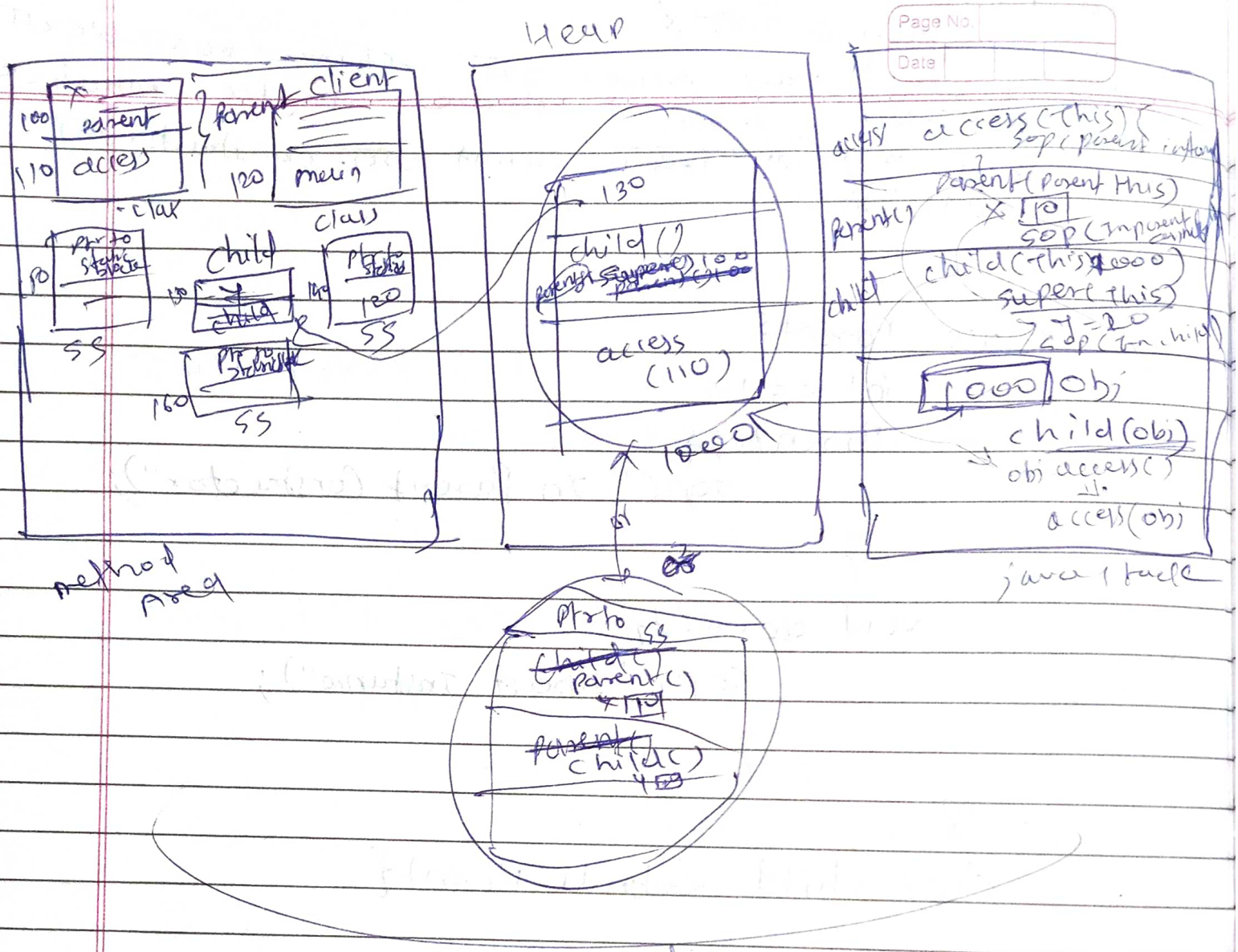
child(){

2 error

child class चा object, automatically parent चा constructor ही call का होतो?

→ कारण child class parent चा instance variable चा access असतो. म्हणून ते instance variable initialize होवे साठी parent चा constructor ही call होतो

```
class Parent{
    int x = 10;
    Parent(){
        sop("In Parent Constructor");
    }
    void access(){
        sop("Parent Instance");
    }
}
class child extends Parent{
    int y = 20;
    Child{
        sop("In child Constructor");
        sop(x);
        sop(y);
    }
}
class
    p s void main(String[] arg){
        Child obj = new child();
        obj.access()
    }
}
```



* static In Parent class

```
class Parent{
        static{
            sop("In Parent static Block");
        }
}
class child extends Parent{
        static{
            sop("In child static Block");
        }
```

child class च्या bytecode मध्ये main parent ला call दिसतो त्या वेळेस access parent होतो.

```
class Client{
    public static void main(String[] args){
        child obj=new child();
    }
}
```

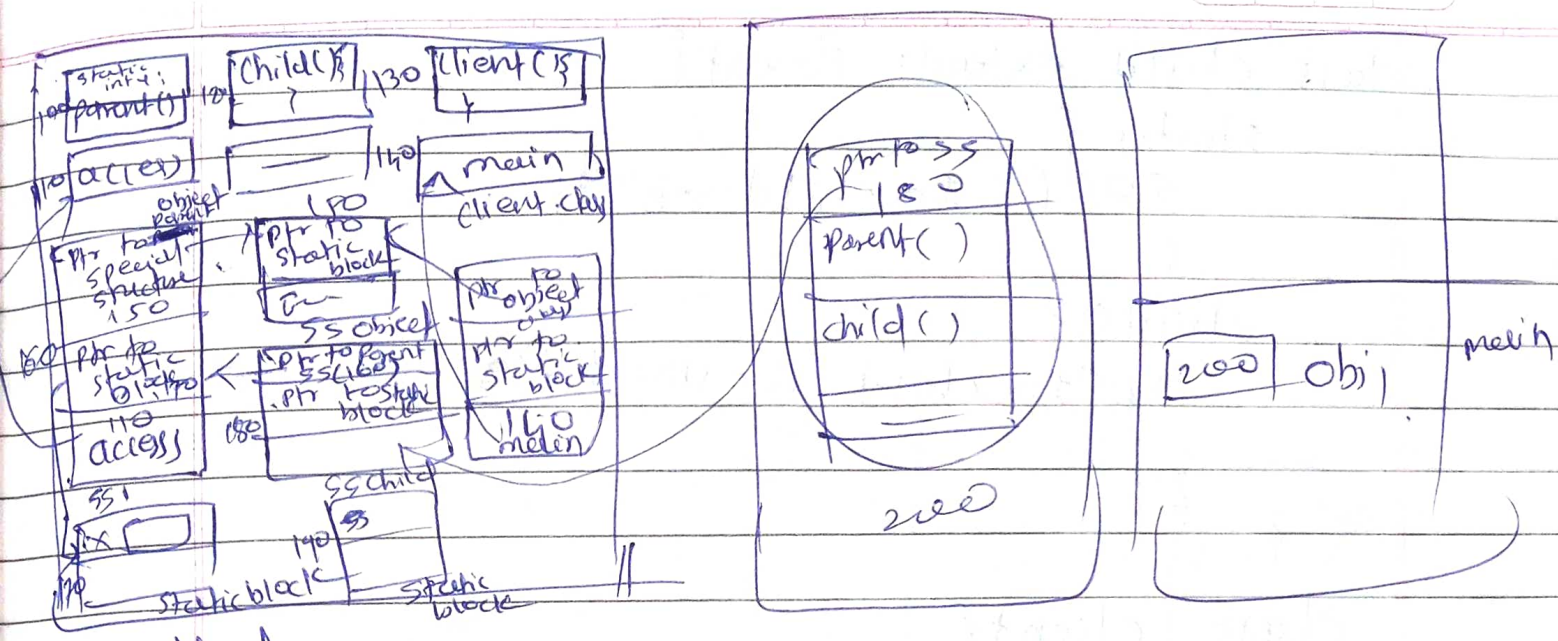
```
class parent{
    static int x=10;
    static{
        sop("In parent staticblock");
    }
}
class child extends Parent{
    static{
        sop('' In child static block);
    }
    void main(String [] args){

    }
}
```

o/p: In parent staticblock
In child staticblock

→ Parent चा static block का जातो कारण त्याचा reference असतो jvm ला.

→ parent static variables initialize करायचे मागितल्या नंतर child मध्ये use करायचे आवश्यक म्हणून तो parent का जातो

```
class Parent{
    static int x=10;
    static {
        sop("In Parent static block");
    }
    static void access(){
        sop(x);
    }
}
class child extends Parent{
    static {
        sop("In child static block");
        sop(x);
        access();
    }
}
class Client{
    Public static void main(String[] args){
        sop("In main");
        Child obj = new Child();
    }
}
```



method

inheritance

→ आई नाही ये तर का! की Parent च्या गोष्टी child कडे येतात. Then access असतो.

→ SOP मध्ये super access करू नाही देत.

class Parent {

int x = 10;

static int y = 20;

~~Parent~~

static {

sop("Parent static block");

}

Parent() {

sop("In Constructor");

}

void methodOne() {

sop(x); sop(y);

}

~~static void methodTwo()~~ {

sop(y);

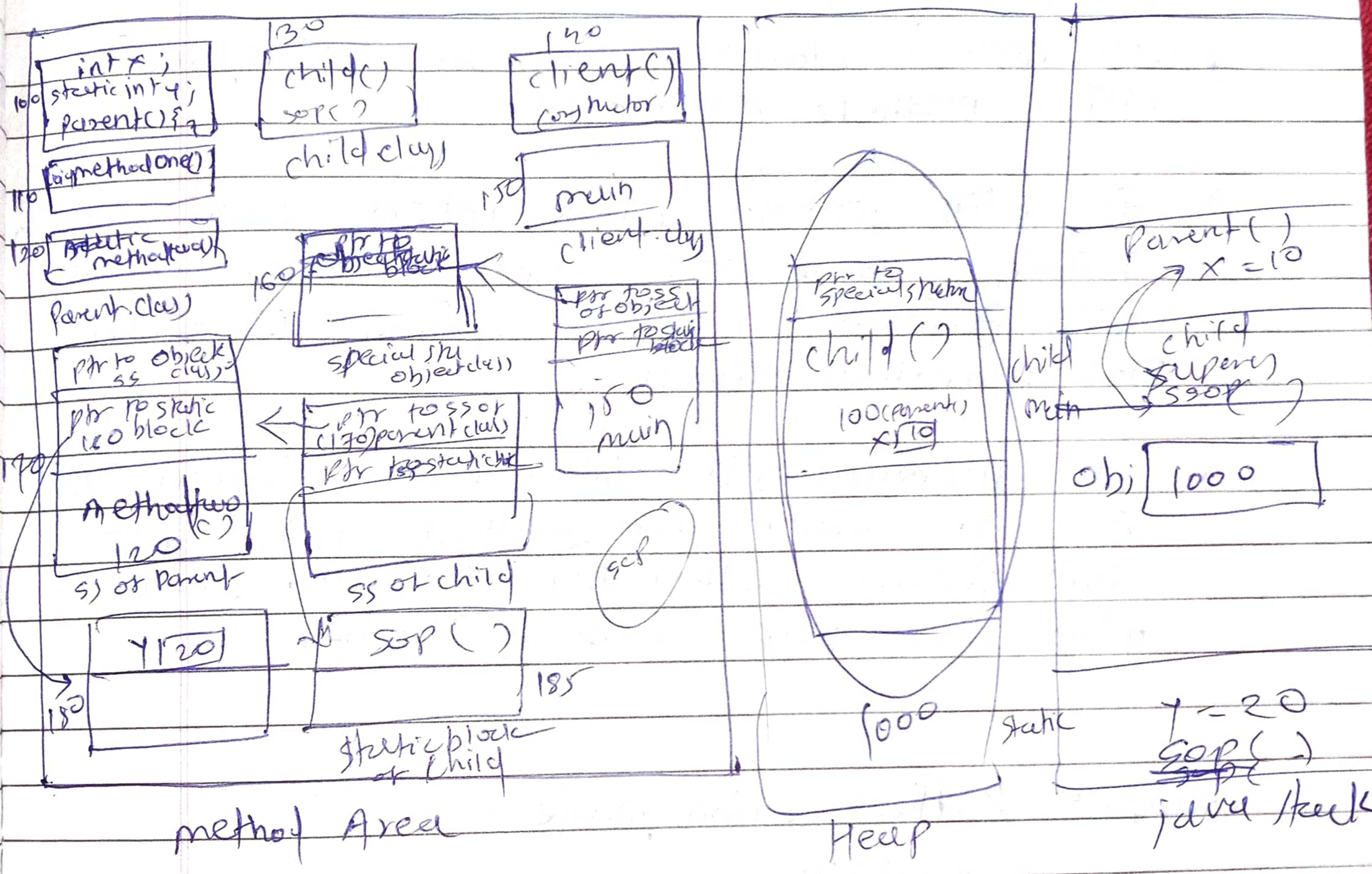
}

}

```
class child extends Parent{
    static{
        sop(" In child SB");
    }
    child(){
        sop("In child constructor");
    }
}

class client{
    p s v main(String[] args)
    {
        child obj = new child();
        obj.methodone();
        obj.methodtwo();
    }
}
```

(HW) o/p : with diagram.

```
Parent static block
In parent Constructor
In child SB
In child constructor
10
20
20
```


int x;
static int y;
parent(){}
100
methodOne()
110
static methodtwo()
120
Parent.class
130
child()
sop()
child class
140
client()
constructor
150
main
client.class
160
ptr to object
special str
object class
ptr to object class
ptr to static 120 block
170
methodtwo()
120
ss of Parent
ptr to ss of (170) parent class
ptr to static child 185
ss of child
ptr to ss of object
ptr to static block
150
main
scp
Y 20
180
sop()
185
static block of child
method Area
ptr to special structure
child()
100 (parent)
x 10
child
main
1000
static
Heap
Parent()
x = 10
child
super()
sop()
obj
1000
Y = 20
sop()
java stack

lect 61 :→
seen : 2-8-23 → uploaded
↳ 31-7-23

Inheritance 03

→ आपल्या जावा मध्ये inheritance आहे.

→ कधीही कोणतीही class लिहा करताना त्या object class च्या 11 methods लागतात.

→ जर तीन वेगवेगळे class लिहले आणि त्यामध्ये प्रत्येकात same दोन method आहेत आणि दोन वेगवेगळ्या आहेत म्हणजेच प्रत्येक class मध्ये 4 methods झाल्या.

→ जर आपल्याला इथे inheritance वापरायचा असेल तर त्या same दोन method आहेत त्या एका वेगळ्या class मध्ये लिहायच्या आणि त्याला प्रत्येक class साठी extend करायचं.

example :- Authentication page
- security.
banking - जीथे जीथे common call आहेत जो ते already लिहून ठेवलेले असतात त्या सगळ्या module साठी वापरतात.

~~In sh~~ finally :→ जो code common आहे जो तो inheritance मध्ये parent class कडे दिला जातो.

inheritance
→ लिहताना उलटा लिहला जातो. आधी सगळा code लिहला जातो मग त्या common goshtin-चा parent class बनवला जातो.

→ Defense class example

→ macD, Dominos franchiese

→ class JavaScript
↓ Angularjs, ↓ nodjs, ↓ reactjs

→ class UPSC
↓ mpsc, ↓ exam/attorney

(central ministry
↓ state ministry

→ real time example on inheritance or polymorphism
100 %

→ Ram class Hey Polymorphism → overloading
→ दोनों → overriding

```
class Parent{
        int x=10;
        static int y=20;
        Parent(){
            sop("Parent");
        }
}
class child extends Parent{
        int x=100;
        static int y=200;
        child(){
            sop("child");
        }
        void access(){
            sop(super.x);
            sop(super.y);
            sop(x);
            sop(y);
        }
}
class client{
        public static void main(String[] args){
            child obj=new child();
            obj.access();
        }
}
```

sop(Parent.x)
sop(Parent.y)

CPP - 10th constructor / CPP चे सगळे double coat वाले topic आहेत

o/p: Parent
child
10
20
100
200

Q) super कुठून येतो ? कुठून आला (शोधा.

→ super print करू शकत नाही sop(super)

→ This ——— शकतो sop(this)

→ see the bytecode of above code to see super.x & sop(super.y).

→ jvm पर्यंत super नावाचे काय माहीत नाही

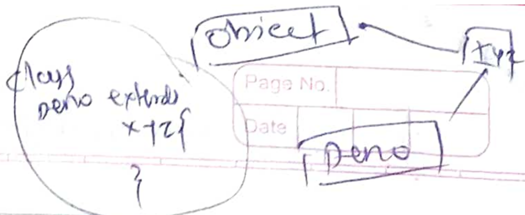
→ super() : → internally invokespecial().

→ super variable → nonstatic आहे.

→ hint → super चा संबंध direct parent class शी असतो.

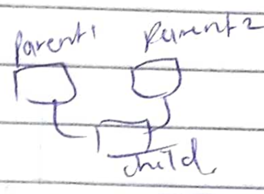
→ पण method पण parent मध्ये आणि same नावाचा असतील तर parent चा method ला call करायचा असेल तर super.method() हे करू शकतो.

Polymorphism :-

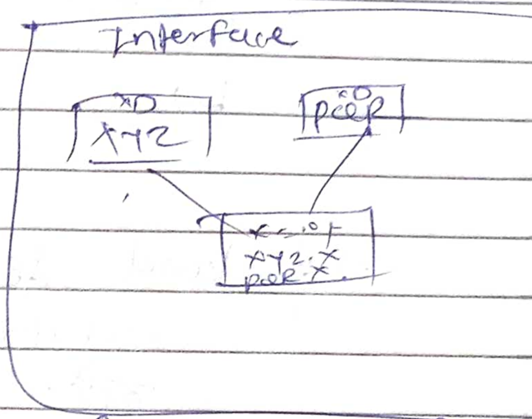


* multiple inheritance → completed in interface.
covered →

types
1. single inheritance
2. multilevel inheritance
3. multiple inheritance →




```
class Parent1{
    int x=10;
}
class Parent2{
    int x=20;

}
class child extends Parent1, Parent2{
    (super.x)
}
```



confusion
होईल ambiguity येणार
आहे

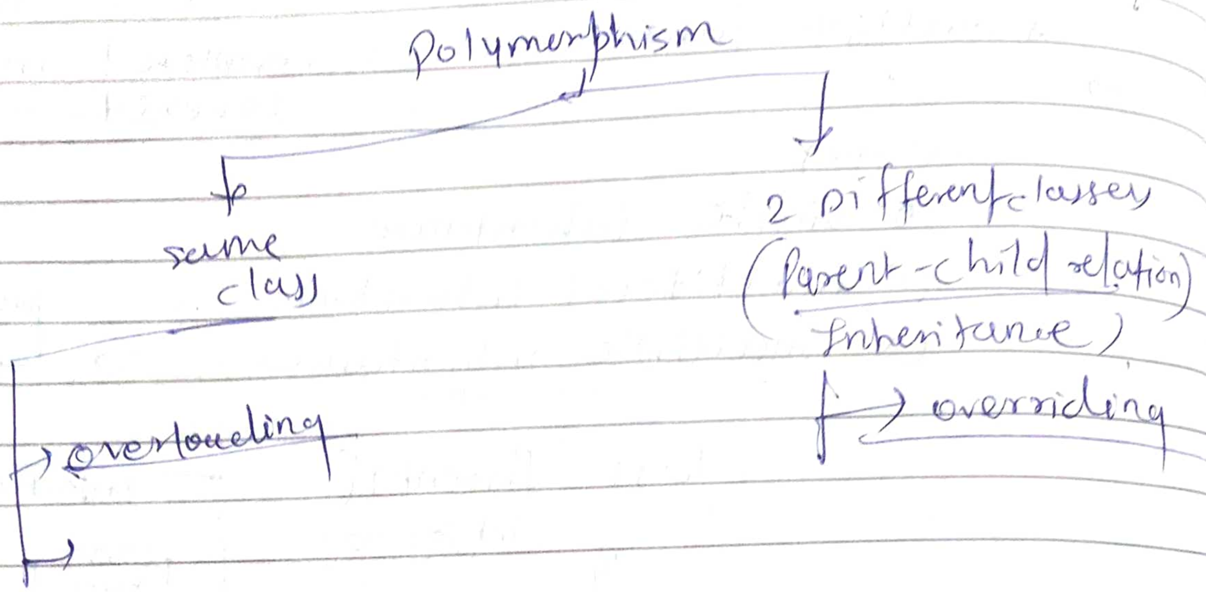
→ म्हणून हा syntax च चालत नाही।
→ multiple inheritance java मध्ये चालत नाही.
→ CPP मध्ये चालतो. Solution pointer.
→ याच्यामुळे CPP वाले लोक java ला बोलत असतात की java lan. oo. pure oop lang असू शकत ?

oracle → कुठे नाही कंपनी स्वत: च्या कुठल्या उदाहरण सांगत सांगत नाही (comma ला error का असं नाही सांगत की multiple inheritance चालत नाही

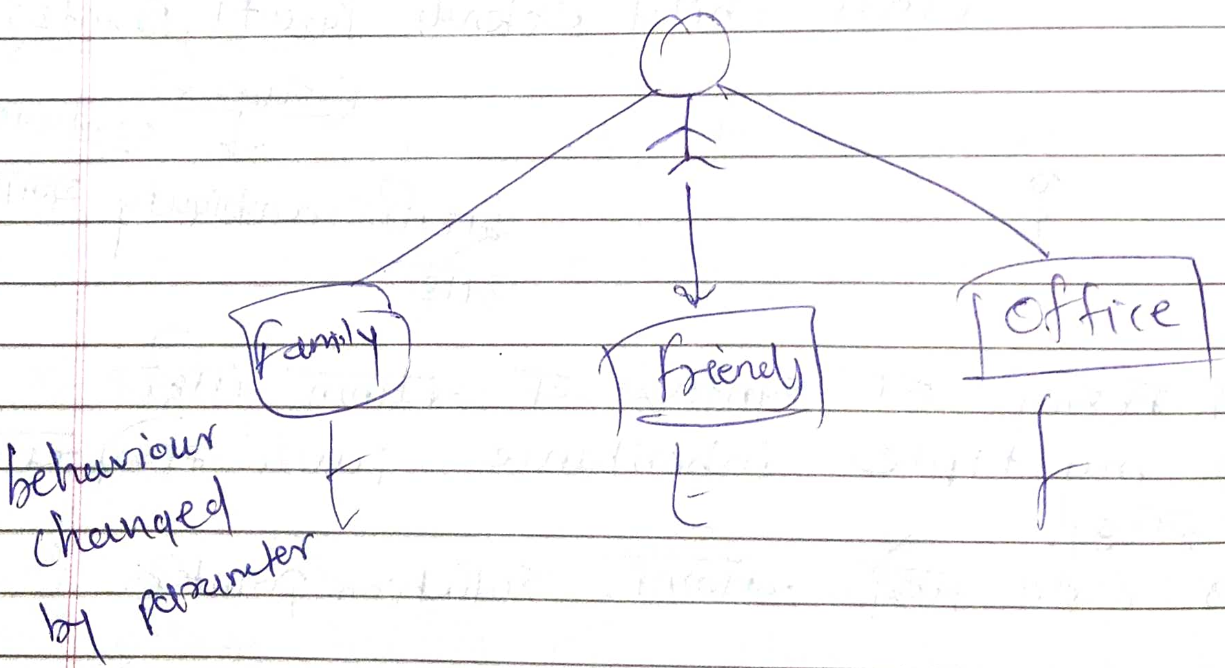
****

✱ Polymorphism ✱

→ poly morphic



→ Backend developer → आवश्यक होतं overloading & overriding वर रेकॉर्ड आहेत.



✱ constructor overloading चं उत्तम example

⇒ same name many forms

⇒ कंपनीमध्ये पण काम same करायचं पण वेगवेगळ्या data वरती करायचं. (polymorphism

best example of polymorphism.

① println.

↳ varies ↳ print overloaded (int, String, ...)

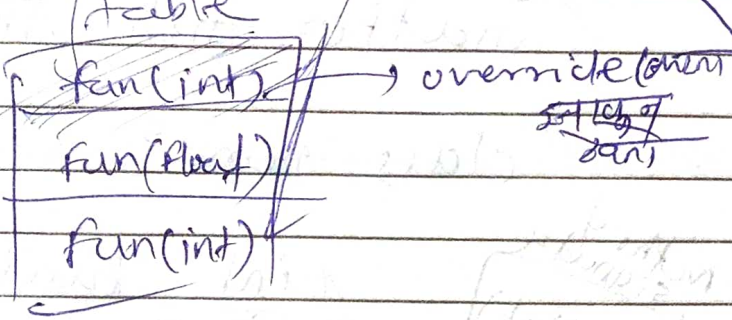
```
class Demo{
    void fun(int x){
    }
    void fun(int y){
    }
```

error: method signatures are same.

```
class Demo{
void fun(int x){
}
void fun(float y){
}
}
```

```
class X72 extends Demo{
{
void fun(int x){
}
}
```

obj.fun(10)




```
parent{
void marry(deepika){
}
}
```

```
child
void marry(katrina){
}
```

overriding concept

```
void
```

abstract class Parent {
    void marry();
    without body
    void property() {
    }
};

child {
    marry() {
    kamina
}

→ abstract class चा object बनवता येत नाही.

interface parent {
    void marry();
    void property();
}
method मी body नाही.

class child implements Parent {
    void marry() {
    }
    void property() {
    }

→ overloading मध्ये scenario मध्ये return type matter करत नाही.

class Demo {
    int fun(int x) {
    }
    float fun(int y) {
    }
}

method signature
fun(int)
fun(int)

return type method signature मध्ये नाही.

Error: method signature same

class Demo {

void fun(int x){ // fun(int) ← method signature
sop(x)
}
void fun(int x){ // fun(int) method signature.
sop(x)
}

⇒ एका class मध्ये same method signature असू शकत नाही.

error : method fun(int) is already defined in class Demo
void fun(int x){

}error

⇒ return type overloading मध्ये consider केला जात नाही.

⇒ overriding मध्ये return type बदलता येतो.

```
class Demo{
    void fun(int x){
        sop(x);
    }
    void fun(float y){
        sop(y);
    }
    void fun(Demo obj){
        sop("In Demo Par");
        sop(obj);
    }
    Public static void main(String[] args){
        Demo obj = new Demo();
        Obj.fun(10);
        obj.fun(10.5F);
        obj.fun(10);
        Demo obj1 = new Demo();
        obj1.fun(obj);
    }
```

```
10
10.5
In Demo Par
address Demo@4a574682
```

System.out.println()

↓

static

lang package

class System {

constructor is private.

static final PrintStream out;

io package

class PrintStream {

void println(int x){

}

void println(float x){

}

void println(String str){

}

void println(Object obj){

}

}

class client{

p s void main(String args[]){

System.out.println("The Core Web");

}

System → system class

out → PrintStream static Object

println → println method.

example of method overloading

Lect 62 scanner 8-25

Polymorphism

dev tool → Azure

Overriding

method मा body भएन → concrete मेथड
नभएन → abstract method

⇒ Static methods override होन नसकिने किनकी यदि दुवै class मा method भयो -

```
class Parent
    Parent(){
        sop("Parent Constructor");
    }
    void property(){
        sop("Home, car, Gold");
    }
    void marry(){
        sop("Deepika Padukone
    }
}
class child extends Parent{
    child{
        sop("child constructor");
    }
    void marry()
        sop("Alia Bhatt");
    }
```

override ← void marry()

```
class client{
    Public Static void main (String[] args){

        child obj = new child();
        obj. property();
        obj. marry();

    }
```

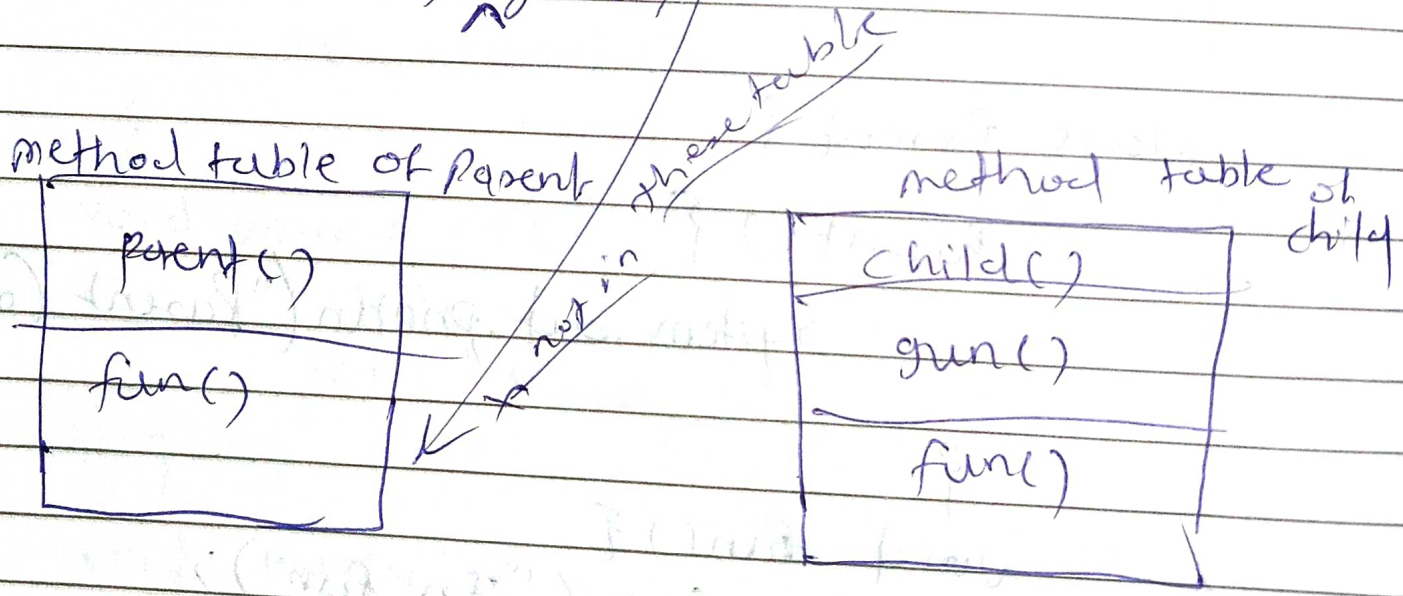
o/p → parent Constructor
child Constructor
Home, car, Gold
Alia Bhatt.

```
class Parent {

    Parent() {
        System.out.println("Parent Constructor");

    }
    void fun() {
        sop ("In fun");

}   }
class child extends Parent {
        child {
            sop ("child Constructor");
        void gun() {
            sop ("in gun");

        }
class}
```

CPP → virtual function or pure virtual function

```
class Client (){
    Public static void main(String [] args){

        Child obj1 = new child();
        obj1.fun();
        obj1.gun();

        Parent obj2 = new new Parent();
        obj2.fun();
        obj2.gun();
```

error → cannot find symbol
obj2.gun();



if we make parent method is private then it ~~will~~ does is not going in child's method table.

primary object का असली

child obj1 = new child();

reference — compiletime

compiler ही वाढ़ून देतो

object → runtime होत heap वरती memory तयार होते

compiler syntax check करतो पण execute करत नाही.

Parent obj1 = new child();
Child obj2 = new Parent();

error: incompatible type: Parent cannot be converted to child

child obj2 = new Parent();

error.

→ ~~कंपनीमध्ये simple ज्याचा reference त्याचा object कधीच दिसत नाही.~~

→ ~~जास्त करून~~ Parent चा reference आणि child चा object दिसतो.

example: Sujata Mastani (parent) — franchises
○ mango mastani

★ Surveys Hotel (pure-nonveg).

```
class Parent {
    Parent (){
        sop("Parent (on)structor");
    }
    void func(){ //void func(int x);
        sop("In Parent fun");
    }
class child extends Parent{
    child (){
        sop("child constructor");
    }
    void fun(){
        sop("In ~~Parent~~ In child fun");
    }
}
class client{
    public static void main(String [] args){
        Parent obj1 = new child();
        obj1.fun();
    }
}
```

~~ज्याचा~~ reference ~~आहे~~ ~~त्याला~~ method table check करतो method signature ~~साठी~~.

① case

```
class Parent {
    Parent() {
        sop("Parent Constructor");
    }
    void fun(int x) {
        sop("In Parent fun");
    }
}
class child extends Parent {
    child() {
        sop("child constructor")
    }
    void fun() {
        sop("In child fun");
    }
}
client {
    p s v main(String [] args){
        Parent obj = new child();
        obj.fun();
    }
}
```

o/p:

error ⇒ Parent class applied

fun(int x)

required: int

found: no argument.

reason: actual & formal parameter are differ in length.

Case2

```
class Parent {
    Parent() {
        SOP("In Parent constructor");
    }
    void fun() {
        SOP("In Parent fun");
    }
}
class child extends Parent {
    child() {
        SOP("In child constructor");
    }
    void fun(int x) {
        SOP("In child fun");
    }
}
class client {
    P S V main(String[] arg)
    Parent obj = new child()
    obj.fun(); // obj.fun(10)
}
```

fun() => inherit to child

o/p =>
In Parent constructor
In child constructor
In Parent fun.

override म्हणजे, जर method signature same असेल तर parent ची method child ला येत नाही.

(priority) → ती method ला दिली जाते

→ Compulsory parent मध्ये ती method आहे & override करायची ती method exactly same पाहिजेत.

lect 63 seen 4-8-23

Polymorphism:

⇒ real time example सांगा ना cricket च पहीजे सांगा.

realtime example of overloading

```
class IPL {
    void matchInfo(String team1, String team2){
        sop(team1 + "vs" + team2);
    }
    void matchInfo(String team1, String team2, String venue){
        System.out.println(team1 + "vs" + team2);
        sop("venue =" + venue);
    }
}
class client{
    public static void main(String[] args){
        IPL ipl2023 = new IPL();
        ipl2023.matchInfo("GT", "CSK");
        ipl2023.matchInfo("GT", "CSK", "NMSA");
    }
```

NMSA → Narendra modi stadium Ahmadabad

- International level matchy
- Domenstic level → IPL

```
class match{
    void matchType(){
        sop("T20/oneDay/Test");
    }
}
class IPLmatch extends match{
    void matchType(){
        sop("T20");
    }
}
class TestWorldCup extends match{
    void matchType(){
        sop("Test");
    }
}
class Client{
    public static void main(String[] args){

        match type1 = new IPLMatch();
        type1.matchType();

        Match type2 = new TestWorldCup();
        type2.matchType();

    }
```

o/p:→ T20
Test

90 over maximum
3 session मध्ये.

```
class Demo{

    void fun (int x){
            sop("int Para");
    }
    void fun(char ch){                    // fun(float a)
            sop("char para");
    }   }
class client{
    public static void main (String[] arg){
            Demo obj = new Demo();
            obj.fun('A');
    }
}
```

o/p:→ char para

```
class Demo{
    void fun(int x){
            sop("int Para");
    }
    void fun(float ch){
            sop(float para);
    }   }
class client {
    p s v main(String[] arg){
            Demo obj = new Demo();
            obj.fun(10.1);
    }
```

error: no suitable method found for fun(double)
obj.fun(10.1);
method Demo.fun(int is not applicable
argument mismatch; possible loosy conversion from double to int
method Demo.fun(float) is not applicable

1 (argument mismatch; possible lossy conversion from double to float.

```
class Demo{
    void fun(int x, float y){
        Sop("Int-float para");
    }
    void fun(float x, int y){
        Sop("Float-int para");
    }
}
class Client {
    public static void main(String[] args)
    {
        Demo obj = new Demo();
        obj.fun(10,10);
    }
```

error: reference to fun is ambiguous
obj.fun(10,10);
both method fun(int, float) in Demo and method fun(float, int) in Demo match.

error

→ compiler method table check करतात। कोणती method bind करतो तर दोन्ही पण method bind होत असतील तर तो ambiguity error देतो.

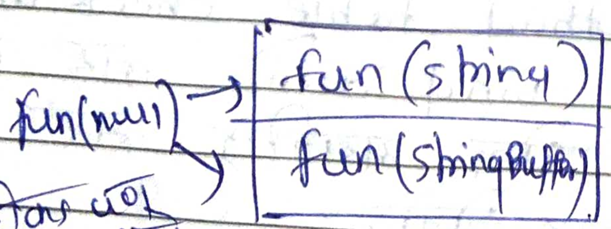
```
class Demo{
    void fun(String str){
        sop("String");
    }
    void fun(StringBuffer str1){
        sop("String Buffer");
    }
}
class client {
    p s v main(String[] args){
    Demo obj = new Demo();
     obj.fun("Core2Web"); // String
     obj.fun(new StringBuffer("Core2Web")); // String Buffer
    }
}
```

o/p :⇒ String
StringBuffer

// ⇒ String str1 = null; ✓
// StringBuffer sb = null; ✓

} [illegible] null pointer.

// call like obj.fun(null) → ambiguity error.



[illegible] error.

object arr[] = {10, 'A', 10.5, 20.5F};

object च्या array मध्ये सगळ्या type चा data घ्यायचा असतो तो ठेवतो, class चा parent आहे.

```
class Demo{
    void fun (object obj){
        sop ("object");
    }
    /*
    void fun(String str){
        sop("String");
    } */
}
class Client{
    Public static void main(String [] args){
        Demo obj = new Demo()
        obj.fun("Core2web");
    }
}
```

o/p: object.

```
class Demo{
    void fun (object obj){
        sop("object");
    }
    void fun (String str){
        sop("String");
    }
}
class Client{
    p s v main (String [] args){
        Demo obj = new Demo();
String  <- obj.fun("Core2web");
object  <- obj.fun (new StringBuffer ("Core2web"));
String  <- obj.fun (null);
}
```

child ला priority असते

match नाही झाले तर मेल जाते.

आधी sum: job ला मग द्यायचं वेगा.

nice system call in OS

nice value

o/p: string
object
string

Question on overloading

① overloading कशासाठी वापरायचं ?

② → same fn'lity करायची आहे त्यासाठी from same method आणि वेगवेगळी कामे करून घेण्यासाठी

example:→ Addition of numbers
Add (two numbers)
Add (three number)
parameter change.

③ real time example?
→ match example.

④

Overriding:

@ When overriding is used?

→ overriding हे दोन वेगवेगळ्या class मध्ये होत असतं आहे त्यामध्ये parent-child relation असायला पाहिजे.

② तेहा parent ची method हवी आहे पण आपला त्या method सहमत नाही त्यावेळी overriding वापरायचं

void पण primitive data type आहे.
→ primitive (int, float, char, double, void)

```
class Parent {
        void fun(){
                sop("Parent fun");
        }
}
class child extends Parent{
        void fun(){
                sop("child fun");
        }
}
class client{
        p s v main (String [] args){
            Parent p = new child()
                p.fun();
}}
```

O/P :⇒ child fun.

⇒ overriding मध्ये (return type classes बद्दल बोलत होते sir. (String, StringBuffer, System)

Compulsory 1.4 पर्यंत java चे मत होते की return type parent च्या method चा आणि child च्या method चा same असला पाहिजे.

⇒ 1.4 नंतर return type मध्ये parent आणि child relation असलं तरी चालतं.

⇒ primitive return type आहेत तर exactly same मागतात.

⇒ जी method override होते तिला overriden.

(covarent return method classes में maintain.)

```
class Parent {
    int fun(){
        sop("parent fun");
        return 10;
    }
}
class child extends Parent{
    char fun(){
        sop("child fun");
        return 'A';
    }
}
class client{
    P s v main(String [] args){
        Parent obj = new child()
        obj.fun();
    }
```

error : fun() in child cannot override fun() in Parent

```
char fun(){
```

return type char is not compatible with int.
|error.

check ① my question why returntype is same for overriding

→ strict question ~~after~~.

Covarent return types.

Page No.
Date

Covarent return type

Constructor

parent child relation

```
class Parent {
    Object fun() {
        object obj = new Object();
        sop("Parent fun");
        return obj;          // return new object
    }
}
class child extends Parent {
    String fun() {
        sop("child fun");
        return "shashi";
    }
}
class client {
    Parent p = new child()
    p.fun();
}
```

ref 64
seen 5-8-23

~~Abstract classes~~

overloading

- method signature → not same.
→ Return type ~~[illegible]~~ ધ્યાનમાં રાખતા નથી.
→ parameter
  ↳ length
  ↳ type
→ parameter
  ↳ argument → primitive → char → int
    → byte → int →
  ↳ argument → classes → Parent object / child obj

Overriding

str1 == str2 → address
str1.equals(str2) address
↑ checks

- example → object class ની equals method
  + String ની equals method
  overriding example આપે. → content checks
- → Parent-child relation in classes
- → method signature is same
- → return type
  ↳ primitive same
  ↳ classes → (covariant return type)

eg tech → વિદ્યાર્થીઓ ને કે કયા class વિશે આવે છે ?
→ System class, object class
  System class → unique constructor private આપ્યા
  object class → object class માં 11 methods.

super → this मध्ये typecast करतो parent class मध्ये.
→ super instance variable स्वतःच आहे.
→ super हा non static आहे.

⇒ Parent(this) & super()

⇒ wrapper classes →
Integer → parent class Number → object

Arithmetic exception → runtimeexception → Exception → Object
class child ← Parent

→ final String class extend करता येत नाही
override करता येत नाही.
javap java.lang.String

→ String class चा direct parent object class आहे.

⇒ ↳ same for StringBuffer.

offcampus Interview

```
class Parent{
    Object fun(){
        return new object();
        or
        return new String();
        or
        return obj
    }
}
class child extends Parent{
    String fun(){
        return new String
    }
```

Object obj = new object
हे तेव्हाच लिहावे लागेल जेव्हा obj वापरायचा आहे.

Example:- मी lift च 5 नंबर च बटण दाबल होत आत मध्ये काय होतय मला पण नाही माहीत. यालाच abstraction म्हणतात.

⇒ यासाठी एक design pattern आहे त्याच नाव आहे. Abstractfactory design pattern
→ हा कंपनी मध्ये जास्त इकडून वापरला जातो.

```
class Parent {
        String fun(){          → child
            return new String();
        }
}

class child extends Parent{
        object fun(){          → parent
            return new Object();
        }
}
```

error: fun() in child cannot override fun() in parent
object fun(){
return type object is not compatible with String
|error.

कारण :→ Parent ला child कडच्या गोष्टी दिसत नाहीत.

⇒ Parent चा reference आणि child चा object चालतो.

⇒ Parent च्या reference ~~वरती~~ वरती child च्या गोष्टी ~~कॉल~~ त्याच call करता येतात ज्या Parent मधील पण आहेत आणि child मध्ये पण आहेत.

void → nothing
↳ याचा wrapper class आहे.

Page No.
Date

overloading → compile time polymorphism
compile time binding
or
early binding.

overriding → runtime polymorphism
↳ late binding
runtime binding.

म्हणजे compile time ला कोणती method bind होणार आणि त्यानंतर जेव्हा runtime येईल तेव्हा तो कोणाचा o/p देणार.

↓ Access Specifier in overriding ↓

```
class Parent {
    public void fun(){
        sop("parent fun");
    }
}
class child extends Parent{
// default → public void fun(){
        sop("child fun");
    }
}
```

error: fun() in child cannot override fun() in Parent
void fun(){
attempting to assign weaker access privileges; was public
|error.

```
class Parent {
default void fun(){

    }
}
class child extends Parent{
    public void fun() {

    }
}
```

o/p:- no error ✓

⇒ Private method override करायचे येत नाही.

```
class Parent {
    private void fun() {
        sop("Parent fun");
    }
}
class child extends Parent{
    void fun(){
        sop("child fun");
    }
}
class client{
    p s void main(String [] args){
        Parent obj = new child();
        obj.fun();
    }
}
```

```
class Parent{
        void fun(){
                sop("Parent fun");
        }
}
class child extends Parent{
        private void fun(){
                sop("child fun");
        }
}
class Client{
        p s v main(String [] args){
                Parent obj = new child();
                obj.fun();
        }
```

o/p: error : fun() in child cannot override fun() in Parent private void fun(){ attempting to assign weaker access privileges;

error – way package → default ठेवूनच राहत folder म्हणजे package.

→ जर parent च्या method child कडे आले आणि child कडच्या method जर त्याच्या child कडे टाकून देण्याच्या नसतील तर त्या method ला final केले जातात.

constant

→ final हा modifier आहे.

Final modifier in overriding

```
class Parent{
    final void fun(){
        sop("parent fun");
    }
}
class child extends Parent{
    void{
        sop("child fun");
    };
```

error → fun() in child cannot override fun() in Parent
void fun()
overriden method is final
↳ parent class method override करायचं नहीं.

modifier → ① static
② final

→ class ने extend केले म्हणजे direct override नाही होत.

```
class Parent{
    static void
        void fun
            sop("In Parent fun");
    }
class child extends Parent{
    void fun(){
        sop("In child fun");
    }
}
```

PTC questions

1. C/CPP (recursion, linked list,)
2. swap
3. prime number
4. Anagram string (shashig, ashish)
5. An array find second largest
6. Binary search, linear search.
7. static & nonstatic.

→ error: func() in child cannot overrid func()
in Parent void func(){
overriden method is static
}error.

→ static हा फक्त class ला bind असतो. & overriding मध्ये फक्त दोन class महत्त्वाचे होते. त्यामध्ये static overriding महत्त्वाचा होत नाही.

→
```
class Parent {
    static void fun(){
        sop("Parent");
    }
}
class child extends Parent{
    static void fun(){
        sop("child");
    }
}
class client{
    p s v main(String [] args){
        Parent obj3 = new child()
        obj3.fun();
    }
}
```

O/P: In Parent ⇒

Rdn: strictFP
strictfloating point

method hidding

→ फक्त दिसताना दोन्हीकडे जर same static methods असतील तर child class हा parent च्या method ला hide करण्यासारखा ~~वाटतो~~ वाटतो।

→ ही overriding सारखी दिसते पण हे overriding नसते

→ का केला जातो? method hidding.

→ ~~कारण जेव्हा आपल्याला parent ची method~~
काही method parent मध्ये असतात त्या आपल्याला override पण करू द्यायच्या नसतात ~~अ~~ पण same to same implementation parent मधले पण पाहिजे असते. त्या method classes च्या असल्या पाहिजेत object वर depend नसल्या पाहिजेत.

→ final आणि static same असल्यासारखे आहेत पण static वापरल्यामुळे आपल्या object बनवायला लागत नाही.

OOP

① classes & object → building without area (blueprint); actual जागा लागते (car class, object
② Encapsulation → class असतो.
③ Inheritance
④ Polymorphism
⑤ Abstraction →

Lect 65: **Abstract classes**

seen :- 7-8-23

framework → abstraction मध्ये येत.

Abstraction

→ Abstract class

→ interface.

प्रश्न : समजा माझ्याकडे एक class आहे. त्यात दोन मेथड्स त्यातली एक method override करू शकतो child मध्ये मी ती method लिहिली पाहिजे की नाहि ?

→ ① एकतर ती parent ने लिहावी (or)
② body नाहि द्यायची abstract करायची child class त्याला body देणार.

* Actually class ला abstract का केलं जातं जर कुणाला त्या class चा object बनवू द्यायचा नसेल तर.

* जर class मध्ये सगळ्या method ला body असेल तरी पण आपण class abstract बनवू शकतो.

* This type absbstruction → शक्य एक abstraction

→ HDFC एक मोठी bank आहे

rules ← RBI → all bank follow rules of RBI but → rate of interest ← abstract

rate of interest is different or depend on bank like HDFC or SBI

commercial & Home lightbill is different.

↓ expensive per unit

↓ less expensive

```
class Parent{
    void career(){
        sop("Doctor");
    }
    void marry();
}
```

error: missing body, or, declare as abstract
method marry,
void marry()
^

```
abstract class Parent{
    void career(){
        sop("Doctor");
    }
    abstract void marry();
}
```

error:- Parent is not abstract and does not override abstract method marry() in Parent.
class Parent{
^
(error.

→ Interface में 100% abstraction होता.

```
abstract class Parent{
    void Career(){
        sop("Doctor");
    }
    abstract void marry();   → incomplete method
}
class client{
    public static void main(String[] args){
        Parent obj = new Parent();
    }
}
```

incomplete method ला access करू शकत नाही ←

जर मी obj.marry() method access करायचा प्रयत्न केला तर काय print होणार आहे

म्हणून JVM ला incomplete class आहे असे कळते नाही.

[that why incomplete class]

error: Parent is abstract, cannot be instantiated

→ abstract class चा जरी object बनत नसला तरी त्याला constructor असतो.
कशासाठी असतो तर constructor chaining साठी असतो.

→ कारण जरी हा abstract असला तरी आपण त्यावरून class inherit करतो मग त्या child class चा object बनतो कारण आपण जी method आपण abstract केली होती तिला child मध्ये override करावं लागणार ⇒ Error

⇒ त्यामुळे आपण child object त्यावरून त्याच्या constructor ला call होऊन त्याच्या parent च्या constructor ला पण call होतो.

obj => null वरती आपण call करू शकत नाही.

```
abstract class Parent{
    void Career(){
        Sop("Doctor");
    }
    abstract void marry();
}
class Child extends Parent{
    void marry(){
        Sop("Alka kubal");
    }
}
class Client{
    p s v main(String[] args){
        Child obj2 = new Child();   ✓
        obj2.career();
        obj2.marry();
    }
}
```

```
Parent obj2 = new Child();   ✓
obj2.Carreer();
obj2.marry
```

parent class जरी abstract असला तरी त्याला reference आणि child चा obj बनवताना ती reference करण्याचे कारण method table मध्ये बघतो.

O/p: Doctor
Alka kubal

Q. जर abstract class चा object बनवता येत नाही तरी abstract class मध्ये constructor का असतो?

→ Constructor chaining
need of constructor chaining?
→ जर constructor वरती वरती call करत गेला तर वरच्या class मधले instance variable वगैरे आहेत जे खालच्या class मध्ये लागणार आहेत ते initialize होण्यासाठी constructor chaining ची गरज असते.

write real time example of abstract class with abstract method

Page No.
Date

① abstract class PTC {

abstract method project / Role

}

② abstract class India {

concrete → central goverment Budget

abstract → state goverment Budget.

}

③ ab class BCCI {

Concrete → funding

abstract → auction.

}

④ ab class shop {

concrete → total amount

abstract → mode of payment

⑤ ab class KFC {

Concrete → price

abstract → infrastructure.

⑥ abstra class Luriton {

concrete → Lavi bags

abstract → products.

⑦ abstract class India goverment {

election

abstract void seats

}

(Constructor मरी private असेल तरी त्या class चा object बनवू शकतो. पण त्यासाठी त्या class चा member ची मदत घ्यावी लागेल (singleton design pattern).

```
ab class WTC {
        concrete → matchinfo
        abstract → winner
    }

ab class filmIndustry {
        concrete filmInfo();
        abstract TicketPrize.
    }

abstract class SPPU {
        Concrete → Syllabus
        abstract → college place
    }

abstract class Bakery {
        Concrete → Cake Flavors
        abstract → prize.
    }
child: Cakeshops.

abstract class placement registration {
        Concrete → registration duration
        abstract → no. of registration.
    }

abstract git {
        concrete → functioning
        abstract → CEO & staff
}
```

abstract society {
    concrete society colour
    abstract flat colour.
}

abstract franchise {
    concrete → product quality
    abstract → price
}

try this
↓ make methods private, static etc.
to 0 input method.

Interface

Singleton Design Pattern

singletone → single object.

```
class singleton {
    singleton() {
        Sop("constructor");
    }
}
class client {
    ps v main(String[] args) {
        singleton obj1 = new singleton();
        Singleton obj2 = new
        S  obj3 = n  ();
    }
}
```

o/p: Constructor
Constructor
Constructor

```
class Singleton {
    static Singleton obj = new Singleton();
    private Singleton() {
        Sop("constructor");
    }
    static Singleton getobject() {
        return obj;
    }
}
```

book
↳ design pattern by dummies
gang of four.
(Microsoft / Amazon, interview & java based code):

```
class Client{
    p s v main(String [] args){

        Singleton obj1 = Singleton.getobject();
        sop(obj1); 100
        Singleton obj2 = Singleton.getobject();
        sop(obj2); 100
        Singleton obj3 = singleton.getobject();
        sop(obj3); 100
    }
}
```

→ आता nonstatic method call करू शकतो
→ Constructor private करून हा रचना design pattern येतो under येतो singleton design pattern

⇒ object static को बनाया जाता ?
→ कारण static प्रत्येक आत मध्ये static वर access होते. method पण static केल्यामुळे without object ने call होते.

Q. How interface is class?
→ जेव्हा class असताना त्याची .class file बनते

** abstract class & interface किस तरह child class बनाया जाता

→ Interface मध्ये constructor नसतो म्हणून interface चा Parent class object class नसतो.

Interface (100% abstraction)

→ java मध्ये multiple inheritance support करत नाहि म्हणून ती support करण्यासाठी concept आणली म्हणजे Interface.

→ 1.8 मध्ये interface method ला body देता येते.

→ interface मध्ये सगळ्या method abstract असतात.

Q. interface ची .class file बनत असते का?
→ yes.

Q. interface मध्ये constructor असतो का?
→ नसतो.

→ interface चा object बनत नाहि → कारण अर्धवट गोष्टींचा object बनवता येत नाहि.

The interface methods are by default public abstract असतात.

→ Architecture & programming lang सोबत develop केले जातात.

→ java language लिहीला ~~लिहीला~~ jvm लिहीला आहे.

→ Interface आल्यावर method ला call करायचं म्हणून की child class आणावाच लागेल.

```
interface Demo{
        void fun();
        void gun();
}
class client{
    p s v main(String [] args){
        Demo obj = new Demo();
}}
```

error: Demo is abstract; cannot be intantiated
Demo obj = new Demo();
1error.

lect 66 seen → 8-8-23

OCJP
oracle certified programmer

Interface -01

zerodha [Nikhil Kamat]
without funding
[illegible]
bootstrap company

```
Interface Demo{
        void m1();

}
```

Bytecode

```
Intrface Demo{
        public abstract void m1();

}
```

public abstract → automatic added by compiler.

```
Interface Demo{
        void m1(){

        }
}
```

error: interface abstract methods cannot have body
void m1(){
|error

→ ~~Ambani statics~~
interface reference [illegible] [illegible] [illegible]. [illegible]
Parent obj = new child();

```
interface Demo{
        void fun();
        void gun();
}
class Demochild implements Demo{
    void fun(){
            sop("In Fun");
    }
    void gun(){
            sop("In gun");
    }
```

o/p: error : gun() in DemoChild cannot implement gun() in Demo

void gun(){

attempting to assign weaker access privileges;

was public → only interface मध्ये असताना method by default public असतात.

same error for fun();

```
interface Demo{
    void fun();
    void gun();
}
class DemoChild implements Demo{
    public void fun(){
        sop("In fun");
    }
    public void gun(){
        sop("In gun");
    }
}
class client{
    p s v main(String[] args){
        Demo obj = new DemoChild();
        obj.fun();
        obj.gun();
    }
}
```

public abstract

o/p:→ In fun
In gun.

```java
interface Demo{
    void fun();
    void gun();
}
abstract class DemoChild implements Demo{
    public void gun(){
            sop("In gun");
    }
}
class DemoChild1 extends DemoChild{
    public void fun(){
            sop("In fun");
    }
}
class Client{
    public static void main(String[] args){
    Demo obj = new DemoChild1();
    obj.fun();
    obj.gun();
    }
}
```

वरचे सगळे abstract class

complete class

complete class कारण इथपर्यंत सगळ्या method ला body मिळाली

~~वरच्या~~ ~~scenario~~ ~~असा~~ actual scenario मध्ये वापरला जातो.

→ adaptor class चा जन्म या concept मधून होतो.
   ↳ is design pattern

Q. interface केव्हा वापरले जाते?

→ जेव्हा हे फक्त दोन scenario असतात
   ① जेव्हा 100% abstraction असले पाहिजे तेव्हा
   ② जेव्हा आपल्याला दोन दोन class च्या functionality एका class कडे आणायच्या असतात direct तेव्हा interface वापरले जाते.

Page No.	
Date	

```
interface Demo1{
        void fun();

}
interface Demo2{
        void gun();

}
class DemoChild implements Demo2, Demo1{

}
```

जो class पहीला असतो त्याच्या method ला implement करावा लागतो.

error: DemoChild is not abstract and does not override abstract method gun() in Demo2

```
class DemoChild implements Demo2, Demo1{
^
error.
```

```
interface Demo1{
        void fun();

}
interface Demo2{
        void fun();

}
class DemoChild implements Demo1, Demo2{

}
```

```
class Demo1 {
        void fun(){

        }
}
class Demo2 {
        void fun(){
        }
class DemoChild extends
        Demo1, Demo2{

}
```

अशावेळेस ambiguity येते म्हणून java मध्ये multiple inheritance चालत नाही.

→ interface मध्ये multiple inheritance चा concept का चालतो.?

→ कारण आपण जरी method ची नाव same लिहले तरी ambiguity येत नाही कारण method ला body च नसते.

for example in real time:

असं एखाद्या वेळेस दोन login method लिहायची आपल्याला त्यावेळेस हा scenario येतो.

→

→ sudo update-alternatives --config javac
or
java

to switch

Multiple Inheritance in Java

```
interface Demo1 {
        void fun();
}
interface Demo2 {
        void fun();
}
class DemoChild implements Demo1, Demo2{
        public void fun(){
                System.out.println("In fun child");
        }
}
class Client {
        public static void main(String[] args){
                Demo1 obj1 = new DemoChild();
                obj1.fun();
                Demo2 obj2 = new DemoChild();
                obj2.fun();
        }
}
```

O/P: In fun child

→ interface मध्ये method ला body देऊ शकतो 1.7 नंतर हे feature add केलं आहे java ने. using following two keywords

default or static

modifier

Q. interface मध्ये method ला body देण्याची concept का आणली?

→ वापरला class ने जर multiple children असतील आणि त्या child classes मध्ये एखादी common method असेल तर. code ची redundancy वाढेल म्हणजे interface मध्ये method ला body देण्याचे feature आणले.

```
interface Demo{
        void fun();
        default void gun(){

        }
        static void sun(){

}       }
```

```
interface Demo1{
        void m1();
}
interface Demo2{
        void m2();
}
interface Demo3 extends Demo1, Demo2{
        void m3();
}
```

o/p: NO error

interface मध्ये multiple inheritance शक्य आहे.

```
interface Demo1{
        static void m1(){
                sop("Demo1-m1");
        }
}
Interface Demo2{
        static void m1(){
                sop("Demo2-m1");
        }
interface Demo3 extends Demo1, Demo2{

        }
```

O/P :⇒ No error

⇒ इथे दोन्ही method मध्ये body आहे तरी error नाही. (ambiguity ची error नाही येत इथे).

नोट
⇒ static हे override मध्ये part च घेत नाही.
फक्त inherit होतात.

Try this code at home
~~inherit~~

```
interface Demo1{
        static void m1(){
                sop("Demo1-m1");
        }
interface Demo2{
        static void m1(){
                sop("Demo2-m1");
        }
```

trial 4 error हा code काय

```
interface Demo3 extends Demo1, Demo2{

    }
class DemoChild implements Demo3{

    public static void main(String[] args){

        Demo1 obj = new DemoChild();
        obj.m1();
```

↳ हा call class स्टा नाबाने केला पाहिजे

```
    }
}
```

lect 67
seen on 10-8-202_

* Interface 02 *

```
interface Demo{
        static void fun(){
                sop("In fun-Demo");
        }
        default void gun(){
                sop("In gun-Demo");
        }
```

internally call → public static void fun()

internally in byte code → public default void gun(){}

default: ही access specifier नाही तो modifier आहे.

overriding scenario

```
class Demo{
        static void fun(){
                sop("In fun-Demo");
        }
}
class DemoChild extends Demo{
        // void fun(){
        //         sop("In fun-DemoChild");
        // }
        static void fun(){
                sop("In fun-DemoChild");
        }
}
class Client{
        p s v main(String[] args){
                Demo obj1 = new DemoChild();
                obj1.fun();
        }
```

⇒ No error

↳ त्याच्या reference कडे पाहिल्या method ला call जाणार कारण static ला object नाही लागत नसते.

```
class Demo{
        static void fun(){
                sop(" In fun-Demo");
        }
class DemoChild extends Demo{
}
class client{
        P S V main(String [] args){
        Demo obj1 = new DemoChild();
        obj1.fun();
}
```

o/p :- In fun-Demo.

```
interface Demo{
        void gun();
        default void fun(){
                sop("In fun-Demo");
        }
}
class DemoChild implements Demo{
        Public void gun(){
                sop(" In gun-Demochild");
        }
class client
        P S V main(String[] args){
        Demo obj = new DemoChild();
        obj.gun();
        obj.fun();
}
```

o/p No error :→ In gunDemo child
In fun Demo.

Escape ↑↑↑ for copy lines

```
interface Demo{
        default void fun(){
                sop("In fun-Demo");
        }
}
class DemoChild implements Demo{
        public void fun(){
                sop("In fun Demo");
        }
}
```

default हा overriding साठी allow करतो

for example: ~~[illegible]~~ surname change करायच त्यासाठी हा scenario मांडला आहे.

movie: ~~मी~~ Shivaji Raj Bhosale Boltoy.

B Kalg

Advance code

```
interface Demo1 {
        default void fun(){
                sop("In fun -Demo1");
        }
}
interface Demo2{
        default void fun(){
                sop("In fun-Demo2");
        }
}
class DemoChild implements Demo1, Demo2{

}
class Client{
        public static void main(String[] args){
                DemoChild obj = new DemoChild();
                obj.fun();
        }
}
```

error: class DemoChild inherits unrelated defaults for fun() from types Demo1 & Demo2.

```
class DemoChild implements Demo1, Demo2{
^
1 error
```

→ तरच override करायचय नाही तरच default वापरा.

```
interface Demo1{
        default void fun(){
                sop("In fun-Demo1");
    }   }

interface Demo2{
        default void fun(){
                sop("In fun-Demo2");
            }
    }

class DemoChild implements Demo1, Demo2{
    public void fun(){
            sop("In fun-DemoChild");
    }   }

class Client{
        P S V main(String[] args){
        DemoChild obj = new DemoChild();
        obj.fun();
        Demo1 obj1 = new DemoChild();
        obj1.fun();
        Demo2 obj2 = new DemoChild();
        obj2.fun();
```

o/p:

In fun Demo child
In fun Demo child
In fun Demo child.

```
interface Demo {
        default void fun() {
            Sop("In fun");
        }
}
class DemoChild implements Demo {
}
class Client {
        public static void main(String[] args) {
            Demo child obj = new DemoChild();
            obj.fun();
        }
}
```

o/p: In fun.

याचा अर्थ default असल्यावर
① override पण करता येते
② आणि ती child class मध्ये पण येते.

default & main feature
static & sub feature } in 1.8

Page No.
Date

static ⇒

```
interface Demo {
        static void fun() {
            sop("In fun");
        }
}
class Demo child implements Demo {
}
class client {
        Public s v main(String[] arg) {
            Demochild obj = new Demochild();
            obj.fun();
        }
}
```

error: cannot find symbol.

जर आपण interface मध्ये static method write केली तर ती child class कडे येत नाही.

```
class Demo {
        static void fun() {
            sop("In fun");
        }
}
class Demo child extends Demo {
}
class client {
        P S V main(String[] arg) {
            Demochild obj = new Demochild();
            obj.fun();
        }
}
```

o/p :- In fun.

Q. interface मध्ये static चा वापर का केला? आणि ती method कुणी override नाही केली पाहिजे.

→

```
interface Demo{
        Static void fun(){
                Sop("In fun");
        }
}
class DemoChild implements Demo{

}
class Client{
        P S V main (String [] args){
                Demo obj = new DemoChild();
                obj.fun();
        }
}
```

error: illigal start of expression.

→ इथे main interface च्या आतमध्ये असे static method call करावी लागते.

```
interface Demo{
        static void fun(){
                sop("In fun");
        }
}
class DemoChild implements Demo{
        void fun(){
        }
}
class Client{
        P S V main (String [] args){
```

}

```
interface Demo{
        static void fun(){
                sop("In fun");
        }
}
class DemoChild implements Demo{
        void fun(){
                sop("In fun-child");
                Demo.fun();
        }
}
class Client{
        p s v main(){
                DemoChild obj = new DemoChild();
                obj.fun();
        }
}
```

o/p:-

```
interface Demo1{
        static void fun(){
                sop("In fun-Demo1");
        }
}
interface Demo2{
        static void fun(){
                sop("In fun Demo2");
        }
}
class DemoChild implements Demo1, Demo2{
        void fun(){
                sop("In fun-child");
        }
}
class Client{
        p s v main(){
                Demo obj = new ChildDemo();
                obj.fun();
```

Try this codes of defaults & statics.

prep insta

error: illegal start of expression

⇒ जेव्हा इंटे तेव्हा त्याचा static method ला call करायचा तो interface च्या नावाने करावं →

⇒ तो interface च्या reference ने पण call नाही होत

interface. method name ()

lect 68: InnerClasses-01

11-8-23

```
interface Demo{
        static void fun(){
                sop("In fun-static");
        }
        default void gun(){
                sop(" In gun-default");
        }
}

class DemoChild implements Demo{

}

class Client{
        ps v main (String[] args){
                DemoChild obj= new DemoChild();
                obj.fun(); // error cannot find symbol.
                obj.gun();
        }
```

Demo obj = new DemoChild();

interface ← Demo; child ← DemoChild

जर आपल्याला असं करायचं होत आपण override करायचं कशाला करायचं?

⇒ option is static

we can call parent method if method is static

```
Interface Demo1{
        static void fun(){
                sop(" In fun-Demo");
        }
        default void gun(){
                sop(" In gun-default");
        }
interface Demo2{
        static void fun(){
                sop("In fun-Demo2");
        }
        default void gun(){
                sop( "In gun-default2");
        }
}
~~Client~~ class DemoChild implements Demo1, Demo2{
        void fun(){
                sop(" In fun- DemoChild");
}       }
class Client{
                p s v main(String[] args){
        }       }
```

→ call न करता पण error होते ambiguity

error: class DemoChild inherits unrelated defaults for gun() from types Demo1 and Demo2
Class DemoChild implements Demo1, Demo2{
^
| error

इथे आपण रचना override method केली की तिला म्हणजे parent च्या method ला call नाही करू शकत.

```
Interface Demo1{
        default void gun(){
                sop(" In gun-Demo");
        }}
class DemoChild implements Demo1{
        Public void gun(){
                sop(" In gun-DemoChild");
        }   }
class Client{
        ps v main(String [] args){
                Demo1 obj = new DemoChild();
                obj.gun();
```

Variables In Interface.

```
interface Demo {
        int x = 10;
internally → // public static final int x;
        void fun();
   // internally public abstract void fun();
}
```

(final: त्याला static block नाहीये. तो child class मध्ये पण नाही येत. त्याला initialize करा.)

```
class Aarch {
        static int y = 20;
        static int y;
        static block
   //     b{y=10;}
```

⇒ interface च्या आत मध्ये जे काही variable लिहता ते by default public static final असतं.

- public → scope मोठा
- static → interface च्या नावाने directly access करू शकतो
- final → constant → कुणीही / कोणताही child class change नाही करू शकत.

⇒ interface च्या आतमधल्या variables ना stack frame च्या आतमध्ये जागा मिळते.

```
interface Demo{
        int x = 10;
        void fun();
}
class DemoChild implement Demo {
    public void fun(){
            Sop(X);          //10
            sop(Demo.x);     // 10
    }   }
class Client{
    p s v main(string [] args){
            Demo obj = new DemoChild();
                    obj.fun();
}}
```

इथे पण तो दोन वेळा bipush झाला आहे.

scope वगैरे काही नाही, की त्याला bipush होतो.

```
interface A{
        int x=10;
}
interface B{
        int x= 20;
}
Class Child implements A,B{
        // int x = 30;
        void fun(){
            ① sop(x);        ② sop(A.x); //
                             ③ sop(B.x); //
        }
}
class Client{
    p s v main (string[] args){
            Child obj = new Child();
            obj.fun();
    }. }
```

error:⇒ reference to x is ambiguous sop(x);

both variable x in A and variable x in B match

त्यामध्ये जे जे interfaceचे variable जर print करायचे असतील तर ते interface चा reference वापरूनच print केले पाहिजेत म्हणजे ambiguous error येणार नाहि.

*. Marker Interface *

- serializable
- RandomAccess
- Cloneable → ह्या class चा object बनवून दे.
- EventListner.

→ हे अशा typeचे interface असतात त्यामध्ये एकही method नसते.

→ example शहीद पतीला ते retire झाले तरी त्यांना काही गोष्टी फ्री असतात कारण त्यांना तो tag मिळाला आहे (marker interface)

→ javap java.lang.String;
↳ implements serializable.

javap → java.io.serializable

```
Public interface java.io.serializable{
}
```

javap -c java.lang.Cloneable.

javap -c java.util.RandomAccess.

```
public interface java.util.RandomAccess{
}
```

→ javap java.util.ArrayList.

marker interface jvm ला सांगतो की अशा type चा object आहे तर ह्याला ह्या ह्या functionality दे.

→ javap java.lang.object.

→ linkedlist मध्ये random Access करता येत नाही.
javap.util.LinkedList.

→ marker interface example : अर्जुन तेंडुलकर → tag → sachin Tendulkar
: jay shaha → tag → Amit shaha

Interface imp point

① interface 100% abstraction आहे.
② interface चा object बनवता येत नाही.
③ interface मध्ये variable by default public static final असतात.
④ marker interface
↳ user define पण करता येतात.