Date: 12-8-22
Lect 68

Inner classes 01

1. Normal Inner Class
2. method Local Inner Class
3. Static Inner Class (static Nested class)
4. Annonymous Inner Class
   - Normal
   - Parameterized anonymous inner class (in Android, function onclick)

```
void fun (class X() { ) {
}
```

Normal Inner Class

```
class outer{
    class Inner{
        sop("In m1-Inner");
    }
}
void m2() {
    sop("In m2-outer");
}
```

```
class college{
    class department{
    }
}
```

college नसेल तर भाग department नसेल.

```
class building{
    class flat{
    }
}
```

building नसेल Flats पण picture नाही येत. मग याला आपल्याला complete म्हणतो.

जेव्हा बाहेरच्या class वरती आतला class dependent असतो तेव्हा inner class use करतात.

sun & planets

Internet & websites.

Gallary & planets

RBI & local banks.

sunrays & solar planets

University & college

Real time example dependancy असेल तेव्हा वापरा.

atmosphere & war & UNO

school & class (division).

defence & Army (navy.).

OS & virtual machine technical example

example interview technical प्रश्नांसाठी

company & employee

example static nested class वापरा

company बंद पडली तर employee बंद पडत नाही.

```
class Outer{
    class Inner{
        void m1(){
            Sop("In m1-Inner");
    }   }
    void m2(){
        Sop("In m2-outer");
    }   }
class Client{
    Public static void main(String[] args){
        Outer obj = new outer();
        obj.m2();
        Outer.Inner obj1 = obj.new Inner();
        //outer.Inner obj1 = new Outer().new Inner();
        obj1.m1();
}
```

in multithreading

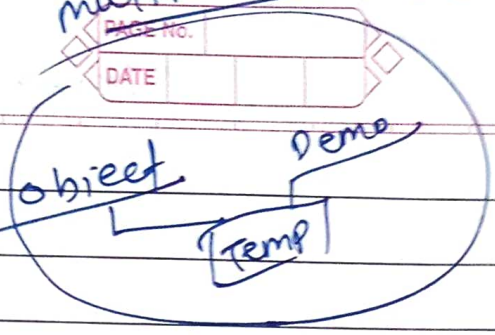
multiple inheritance

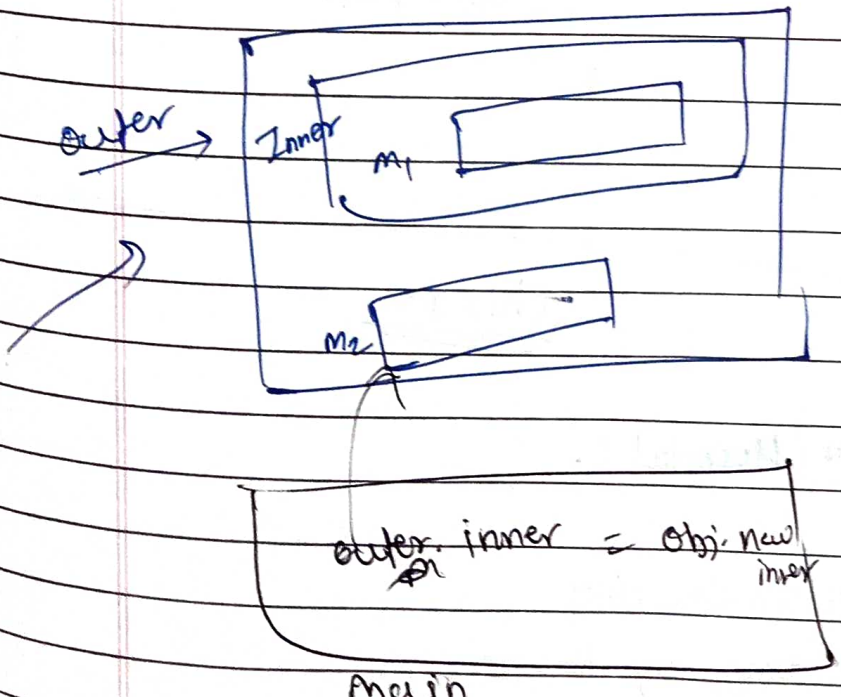
PAGE No.
DATE

interface Demo {
    void m1();
}

class Temp implements Demo {
    p s v main() {
    }
}

extends





lect 69:

Inner Classes - 02

PAGE No. 4
DATE

NVDIA

Glass door ⇒ company review.

पुढची company review साठी मागच्या company ला विचारले.

Companya skype वरच chat चेनिन.

केलेली parent object class proof ⇒ constructor

```
class outer {
        class Inner {

        }
}
```

⇒ 1s
outer.class, 'outer$Inner.class'

```
class Outer$out {
        class Inner {

        }
}
```

'outeroutInner.class'
'outer$out.class'

```
class Hospital {
        class Doctor {

        }
        class medical {

        }
}
```

see the ByteCode

this $0

→ constructor

Hospital $Doctor (Hospital);

javap -c Hospital\$Doctor.class

escape seq. character else error came.

```
class Hospital{
        class Doctor{
                class MBBS{
                        }
                }
        class medical{
                }
}
```

→ see byte code of ~~this~~ each class

```
outer.Inner obj2= obj.new Inner();
```

बाहेरच्या class चा object बनवल्याशिवाय आतला object बनवता येत नाही.

```
class outer{
        class Inner{
        void m1(){
                sop("Inner-m1");
        }   }
        void m2(){
                sop("outer-m2");
        }
        p s v main (String[] args){
        Inner obj = new Outer().new Inner();
            obj.m1();
        }
```

→ जर main त्याच class मध्ये असेल तर.

Method local Inner Class

समजा एखादी class जर एखादी method call केल्यानंतरच picture मध्ये येत असेल तर त्यांना method local inner class म्हणतात.

```
class Bank{
    void createAccount(){
        class Current{
        }
        class Savings{
        }
    }
}
```

```
class Outer {
    void m1(){
        sop("In m1-Outer");
        class Inner{
            void m1(){
                sop("In m1-Inner");
            }
        }
    }
    void m2(){
        sop("In m2-Outer");
    }
    p s v main(String [] args){
        Outer obj = new Outer().
        obj.m1();
        obj.m2();
    }
}
```

=> Class 'Outer$1Inner.class'
↳ level.

```
class outer{
    void m1(){
        Sop("In m1-outer");
        class Inner{
            void m1(){
                Sop("In m1-Inner");
            }
        }
        Inner obj=new Inner();
        obj.m1();
    }
    m2(){
        sop("In outer-m2");
    }
public static void main(){
    outer.obj=new outer();
    obj.m1()
    obj.m2();
```

Compulsory object

```
* static nested inner class *

class outer{
    void m1(){
    }
}
static class Inner{
}
Public static void main(String[] args){
    Inner obj=new Inner();
    obj.m1();
} }
```

```
class outer{
    void m1(){
        sop("In m1-outer");
    }
    static class Inner{
        void m1(){
            sop("In m1-Inner");
        }
    }
}

class client{
    public static void main(String[] args){
        outer.Inner obj = new outer.Inner();
        obj.m1();
    }
}
```

static inner ची obj अशा घेतली जाते.

* Anonymous Inner class *

```
class Demo{

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

        };
    }
}
```

Inner one time use

'Client $1.class' → anonymous Inner class
Client.class →
Demo.class.

lect 70:
seen:- 19-8-23

Inner classes 03

revise केले point.

Hashset दोन्ही internally same आहेत.

* key value pair मध्ये पाहिजे तर → Hashmap वापरा.
  ↳ जास्त वापरला जातो.
* Time utilize करा

revision

→ constructor चा पहिला parameter हा hidden this reference असतो.
→ constructor मध्ये recursive call allowed नाही.
→ constructor overload करता येतो.
→ constructor ला access specifier जो class चा असतो तोच by default असतो.
→ instance block ला पण constructor मध्ये merge केला जातो.
→ constructor ला static keyword चालत नाही
  कारण, constructor object तयार होण्याच्या picture मध्ये येत नाही
→ interface मध्ये constructor नसतो.

* Polymorphism
①

```
class Parent{
    int x=10
    void m1(){
        Sop("In Parent-m1");
    }

}
class child extends Parent{
    int a=20;
    void m1(){
        Sop("In child-m1");
    }

class Demo{
    Demo(Parent p){
        Sop("constructor-Parent");
        p.m1();
    }
    Demo(child c){
        Sop("In constructor-child");
        c.m1();
    }
    Ps void main(String[] arg){
        Demo obj1=new Demo(new Parent());
        Demo obj2=new Demo(new child());
    }
```

*****

boilerplate code In company

```
class Demo {
    Demo(){
        sop("In Constructor-Demo");
    }
}
class DemoChild extends Demo{
    DemoChild(){
        sop("In Constructor-DemoChild");
    }
}

class Parent{
    Parent(){
        sop("In Parent-Constructor");
    }
    Demo m1(){
        sop("In m1-Parent");
    }
}

class child extends Parent{
    child(){
        sop("In constructor-child");
    }
    DemoChild m1(){
        sop("In m1-child");
    }
    return new DemoChild();
}
class client{
    ? S void main(String[] args){
        Parent p = new child();
        p.m1();
    }
}
```

covariant return type overriding

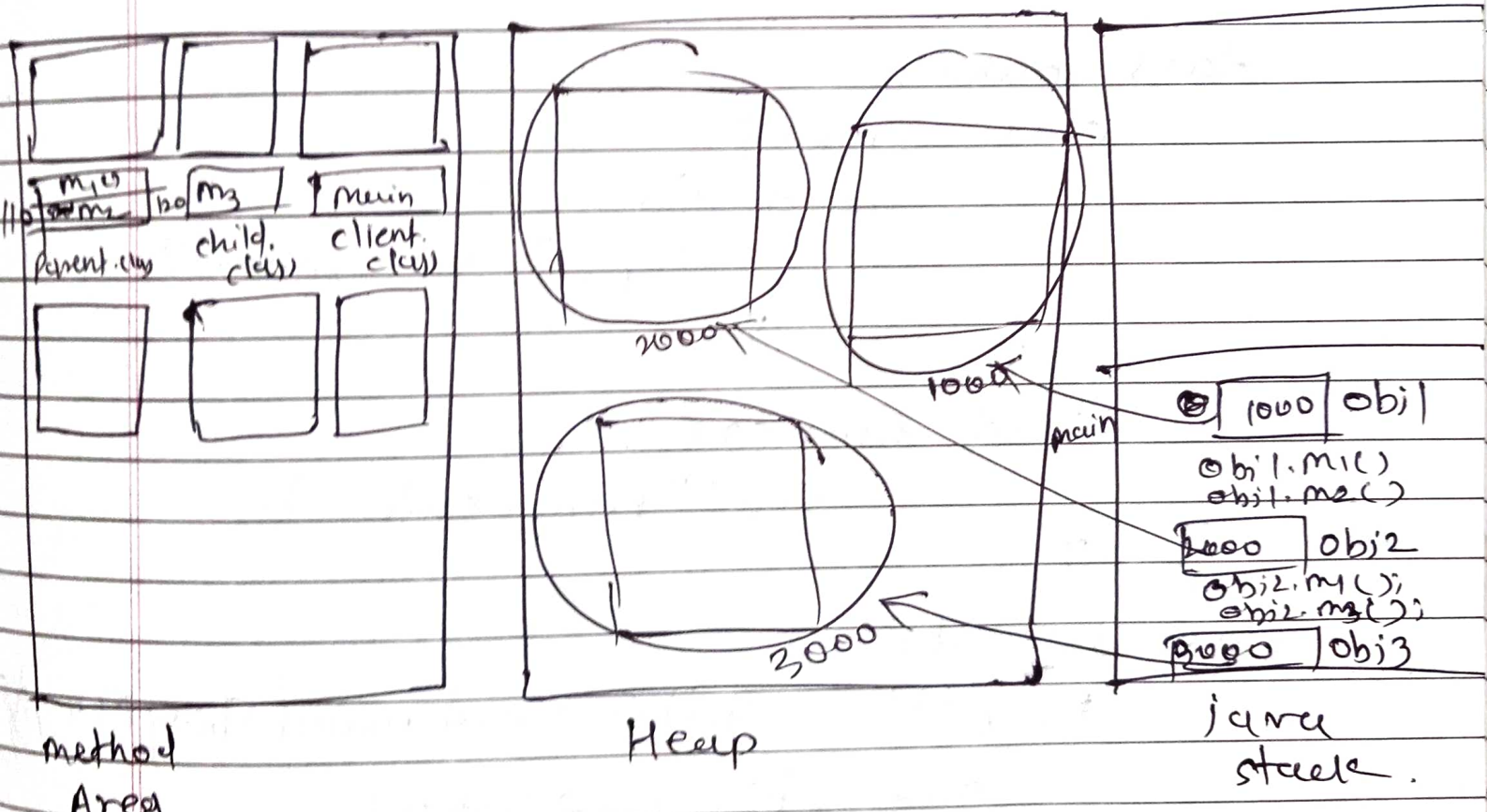
DATE

सरांनी

```
Clas Parent ~~class~~ {
    int x = 10
    static int y = 20;
    void m1() {
        sop("In my parent");
    }
    static void m2() {
        sop("In m2 parent");
    }
    Parent() {
        sop("In Constructor Parent");
    }
class child extends parent {
    int a = 50;
    static int b = 60;
    child() {
        sop("In constructor child");
    }
    void m1() {
        sop(In m1 child");
    }
    static void m3() {
        sop("In m3 child");
    }
}
class client {
    p s v main (String[] args) {
        Parent obj = new Parent()
        obj.m1();
        obj.m2();
```

```
child obj2 = new child();
obj2.m1();
obj2.m3();

Parent obj3 = new child();
obj3.m1();
obj3.m2();
// obj3.m3(); ~~can~~ error cannot find symbol.
```

* after compilation कोई add होने से ~~से~~ diagram of this code.



method Area

Heap

java stack.

Lect 7 | seen: 19-8-23

PAGE No. DATE — Pen Quest (PQ)

Inner Classes - 04 :

* Normal inner class

-> inner class ही dependency वाला class आहे.

PG मध्ये काय strict असतं कुठे बोट लावायचं म्हटलं की त्याला विचारल्या शिवाय लावायला जरी ते Parent असले तरी.

```
obj.Inner obj1 = obj.new Inner();
```

↓

error

Compile time ला काहीच reference होत नाही.

```
class outer{
    class Inner{
        void fun2(){
            sop("fun2 - Inner");
        }
    }
    void fun1(){
        sop("fun1 - outer");
    }
}
class client{
    public static void main(String[] args){
        outer obj = new outer();
        obj.fun();
        Outer.Inner obj1 = obj.new Inner();
        obj1.fun2();
```

O/P: fun1 - outer

fun2 - Inner

bytecode of ⇒ class Outer$Inner {

final outer this$0

Outer$Inner (Outer);

Code:

0: aload_0

1: aload_1

2: putfield #1

5: aload_0

6: invokespecial #2

~~ldc~~ →

Outer$Inner (Outer$Inner this, Outer this$0)

Constructor

Field this$0: Louter

outer class ka object copy aayega

Outer.Inner obj1 = new Outer().new Inner();

this this

Outer$Inner(obj1, new outer())

this$0

getstatic ⇒ out ⇒ static in printstream class

ldc ⇒ load constant

↳ string load ~~creat~~

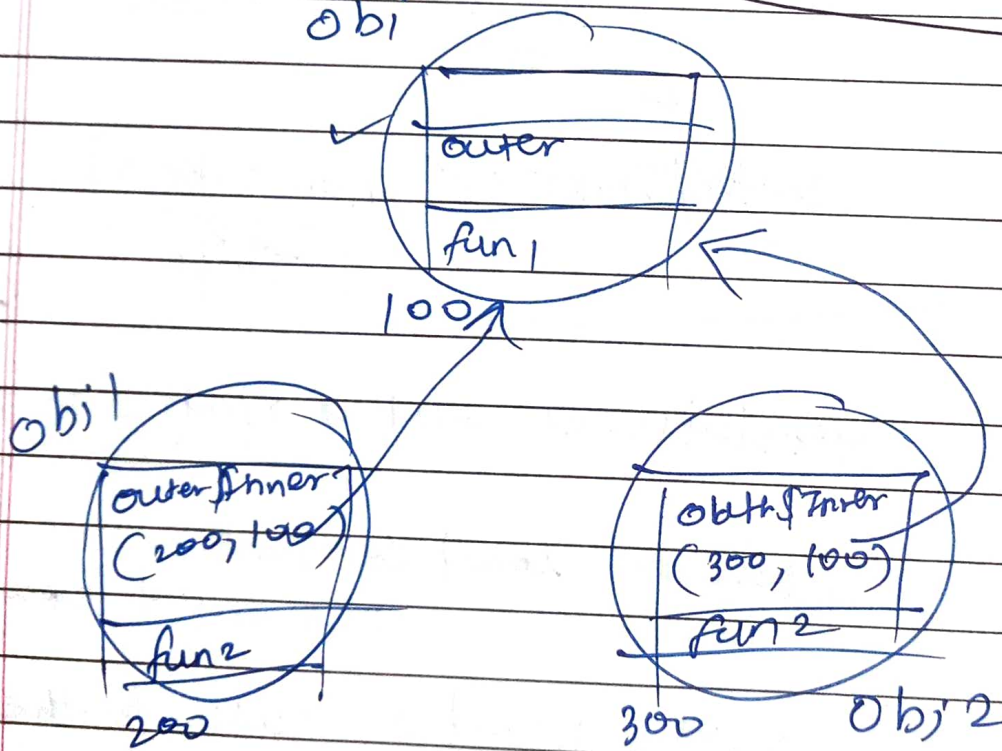
invoke virtual ⇒ call to the function

```
class client {
    public static void main(String[] args){
        outer obj = new outer();          // 100
        outer.Inner obj1 = obj.new Inner();   // 200, outer(obj)
        obj1.fun2();
```

callinternally → outer$Inner(obj1, obj) (200, 100)

```
        outer.Inner obj2 = obj.new Inner();   // 300
        obj2.fun2();
```

outer$Inner(obj2, obj) (300, 100)



⇒ this$0 हा प्रत्येक Inner class मध्ये अदृश्य रूपात तो का आहे ?

⇒ तर कोणत्या outer च्या object वरती हा Inner चा object बनवला है त्याला कळण्यासाठी this$0.

```
class Outer{
        int x = 10;
        int y = 20;
        class Inner{
                void fun2(){
                        sop("fun2-Inner");
                        sop(x);
                        sop(y);
                }       fun();
        }

    void fun1(){
            sop("fun1-outer");

    }
```

```
class client {
        p s void main(string [] args){

            outer obj = new outer();

            outer Inner obj1 = obj. new outer();
                obj1.fun2();

    }
```

o/p:⇒ fun2- inner
10
20
fun1-outer

⇒ Inner मधून सगळं काही outer चा access होतं.

Ref 72: seen: 19-8-23

Inner classes -or

```
class Outer {
    int x = 10;
    static int y = 20;
    void m1() {
        int a = 30;
        static int b = 30; // error
    }
    public static void main(String[] args){
        int p = 50;
        static int q = 60; // error.
```

*
```
class Outer {
    int x = 10;
    static int y = 20;
    class Inner {
        int a = 10;
    } } static int b = 40;
```

error: modifier 'static' is only allowed in constant variable declarations.

⇒ inner class चा object च अनविल्याशिवाय त्यातल्या गोष्टींचा access मेळत नाही.

⇒ जर आपण sop(outer.inner.b); ~~चालवले~~ * तर होकतो पण inner class static पाहिजे होतो ना.

⇒ कारण inner class हा instance आणि instance गोष्टी access करायची म्हटल की object बनवावा लागतो.

child ची reference parent ची ~~reference~~ obj ला घेता येते?
→ parent class ला आपली मेथड implement करायची complete करावे

PAGE No. DATE

```
class outer{
        int x = 10;
        static int y=20;
        class Inner(){
                final static int a = 30;
        }       ↳ final ~~मेथड~~ की static चालेल.
}
class client{
    public static void main(String[] args){
        sop(outer.y);
        outer obj = new outer();
        System.out.println(obj.x);
        System.out.println(obj.new Inner().a);
}
```

In Android onclick

Anonymous Inner class **

```
~~class client~~ class Demo{

        void many(){
            sop(" kriti Sanon");
        }
}

class client{
    public static void main(String[] args){
        Demo obj = new Demo(){
                  (new client$1())
            void many(){
                sop("Disha Patni");
            }
        };
        obj.many();
}   }
```

दोन्ही क्लासमध्ये parent-child relation.

```
class Parent{
        void marry(){
                sop(" Kriti sanon");
        }
}
class Client {
    p s v main( String [] args){

        Demo obj = new Demo(){

            void marry(){
                sop("Disha Patni");
                //fun();
            }
            void fun(){
                sop("In fun");
            }

        };   //.fun(); → error
        obj1.marry();
        //obj1.fun();
    } }
```

but error can resolved using type of ह्यात Demo for method instead of void

शिकणे →

* Anonymous Class आपण जेव्हा parent मधल्या ज्या method आहे त्यांना override करायलाच वापरल्या पाहिजेत.

→ Anonymous class चा constructor लिहता येत नाही. कारण त्याचा object आपल्याला बनवता येत नाही. object आपल्याला Compiler च बनवून देतो anonymous ~~आपल्या~~ class चा

→ सगळी कामं compiler करतो.

normal Inner class example

↳ country & currency

Anonymous Inner class

↳ one time use examples.

↳ msg & OTP.

↳ injection & ~~डर~~ (middle)

↳ flight & boarding pass

↳ company & opening ceremony.

↳ railway station & tickets

↳ gun & bullets.

lect 73   seen 20-8-23

Inner Classes - 06

QT → for GUI in ... → for CPP

Javafx ⇒ चे video ⇒ for eqtech
↳ है Java पासून फार वेगळं आहे.

→ java latest version ⇒ 21
(void main()) आणि.

```
level 0 —— class Outer{
               void m1(){
                   sop("In m1 outer");
level 1 ——         class Inner{
                       void m1(){
                           sop("In m1-Inner");
                       }
                   }
               }   → Inner obj = new Inner();
                     obj.m1();
               void m2(){
                   sop("In m2-outer");
               }
           }
class client{
    public static void main(String[] args){
        Outer obj = new outer();
        obj.m1();
        obj.m2();
    }
}
```

outer.class

outer$1Inner.class
client.class

```
class outer{

    void m1(){

        void m2(){
            class ~~Inner~~ xyz{

            }

    }
```

~~$2 xyz.class~~ हे होऊ शकतं का?

जर एखादा class एखाद्या functionality वर depend असेल तर त्यावेळेस method local inner class वापरला जातो.

```
class outer{

object <- (void) m1(){
        SOP("In m1-outer");
        class Inner{
            void m1(){
                SOP("In m1-Inner");
            }
        }
        return new Inner;   -> internally
                               Object obj = new Inner;
    }
}

class client{
    p s v main(String [] args){
        outer obj = new outer();
        object obj1 = obj.m1();
        obj1.m1();   // -> error -> cannot find symbol
    }   obj.m1().m1();   // -> cannot find symbol
```

* method local inner class के example:

```
class Phonepe{
    transaction function(){
        class Bankaccount{ }
    }
}
```

* class Himachal{
    Fedding{
        refugecamp{

    }

* class playstore{
    update{
        class apps{

    }

* static Inner class → tightly bind करना outer class को।

```
class outer{
    static class Inner{
        void m1(){
            sop("In m1");
    }
}
class client{
    Public static void main (String[] args){
        Outer.Inner obj1 = new outer.Inner();
        obj1.m1();
    }
```

Exception Handling In Java.

Exception

- Compiletime exception
- Runtime exception

* IOException — compiletime

```
import java.io.*;
class Demo{
    public static void main (String[] args){

        BufferedReader br= new BufferedReader(
                new InputStreamReader
                (System.in));
        String str = br.readLine();
```

error:- unreported exception IOException; must be caught or declared to be thrown

```
String str = br.readLine();
```

!error

* Array Index out of Bound Exception

```
class ArrayDemo{
    public static void (String[] args){
        int arr[] = new int[]{10,20,30,40,50};
        sop(arr[i]);
    }}}
    for(int i=0; i<=arr.length; i++){
        sop(arr[i]);
    }}}
```

404 → resource not find.

PAGE No.
DATE

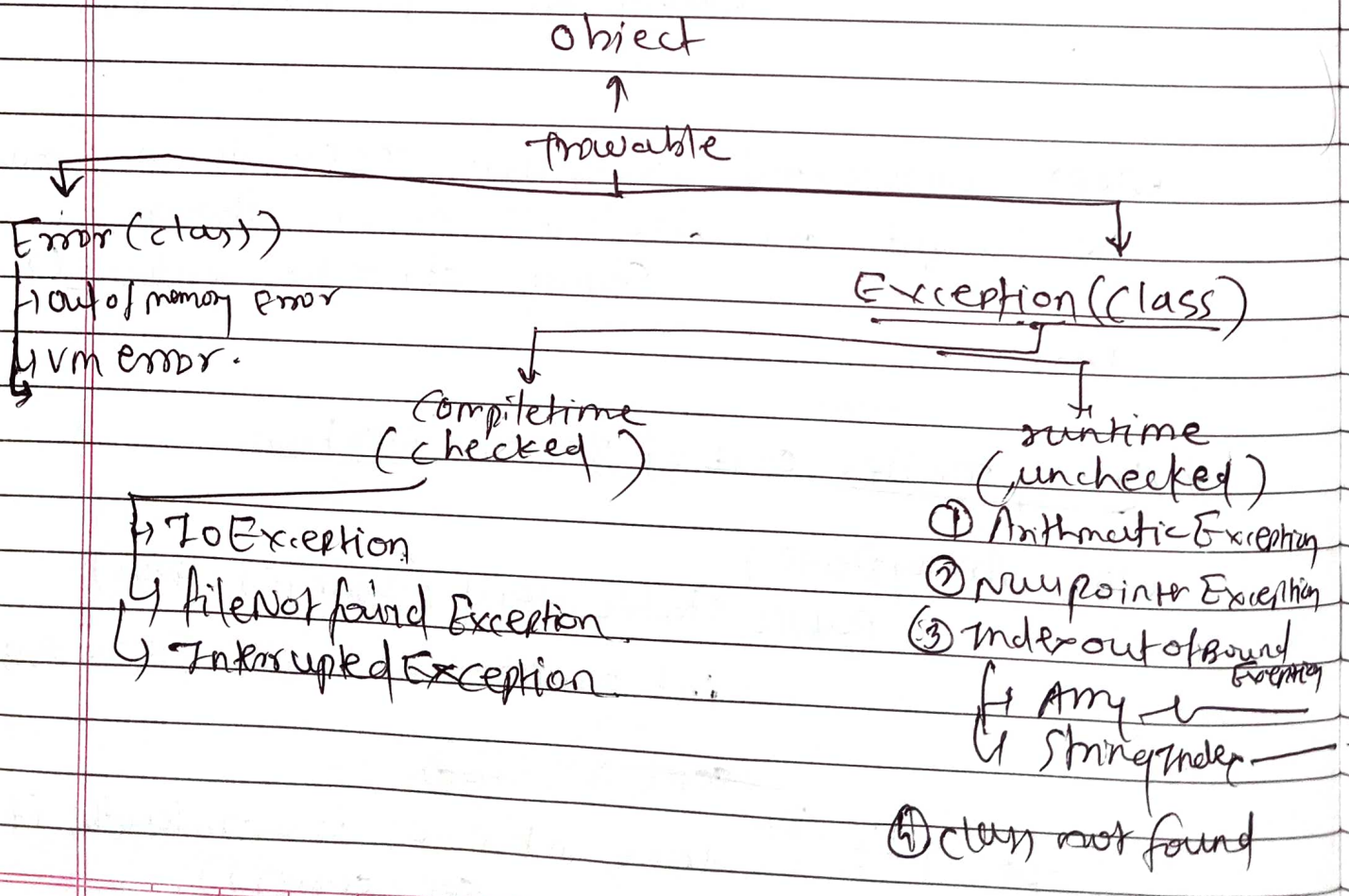
Exception in thread main java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5.

lect 74
seen 20-8-23

Exception Handling

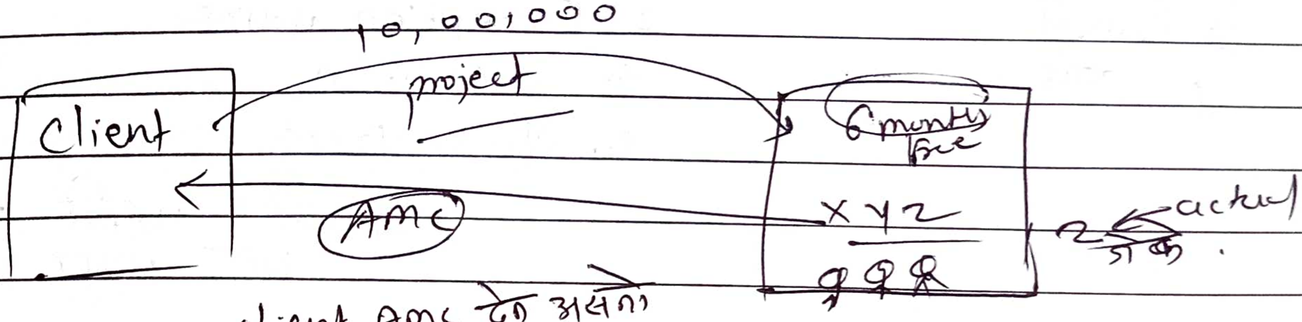
⇒ abnormal termination थांबवण्याची Exception Handling केली जाते.

⇒ code चा flow return 0 पर्यंत चालावा हेसाठी Exception handling केली जाते.



हेल्मेट घातला य रस्त्यावर Accedent होणार नाही असं नाही but जरी accedent झाला तरी Exception पुढचा प्रवास रस्त्यावर Handle होता। हेल्मेट घातला रस्त्यावर exception Handle करते

AMC → Annual maintanance cost



client AMC देत असतो company ला project maintaince ने.

Default Exception Handler.

```
class Demo {
    void m1() {
        sop(10/0);
        m2();
    }
    void m2() {
        sop("In m2");
    }
    p s void main(String[] args) {
        Sop("Start main");
        Demo obj = new Demo();

        obj.m1();
        Sop("End main");
    }
```

→ या line ला exception but m1 ने handle नाही केला म्हणून ते main ला kill करतो.

→ exception इकडे आला आहे but main ला handle नाही केला.

default Exception Handler

हे Exception default Exception Handler Handle करतो

runtime :⇒ Exception in thread main java.lang.ArithmeticException: / by zero
at Demo.m1(program1.java:12)
at Demo.main(program1.java:19).

Check source code
Arithmetic Exception : /by zero
String.

Default Exception Handler format

हे format f मध्ये आहे.

1. Thread Name
2. Exception name
3. Description
4. Stack Trace.
   ↳ कोणत्या पहिल्या line ला error आहे

* null Pointer Exception.

```
class Demo {
        void m1(){
            sop("In m1");
        }
        void m2(){
            sop("In m2");
        }

    public static void main(String[] args){
        sop("start main");
        Demo obj = new Demo();
        obj.m1();
        obj = null;
        obj.m2();
        sop("End main");
    }
```

o/p: start main
In m1
Exception in thread main java lang null pointer exception:
main()

~~class~~

* number format exception. *

```
import java.io.*;
class Demo {
        void getData(){
            BufferReader br = new BufferReader(new
                            InputStreamReader(System.in));

            int data = Integer.parseInt(br.readLine());
            sop(data);
        }
        public static void main(String [] args){
            Demo obj = new Demo();
            Obj.getData();
        }
}
```

update code

after compilation:-

unreported exception IOException; must be caught or declared to be thrown

```
int data = Integer.parseInt(br.readLine());
                                      ^
```

1error.

→ javap java.io.BufferedReader ⇒ for readline

है [illegible] call आला
आहे exceptions [illegible]

```
void getData() throws IOException {
    BufferedReader br = new BufferedReader(
                new InputStreamReader
                (System.in));

    int data = ———
    sop(data)
PSV main( ——— )
    Demo obj = new Demo();
    obj.getData();
}
```

error: unreported exception IOException; must be caught or declared to be thrown

```
obj.getData();
```

^
1 error.

resolved above exception

```
import java.io.*;
class Demo{
    void getData() throws IOException{
    BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
    int data = Integer.parseInt(br.readLine());
      sop(data);
    }
    P S V main(String[] args) throws
                                IOException{
        Demo obj = new Demo();
        obj.getData();
    }
}
```

o/p:- i/p: shashi

Exception in thread main java.lang.NumberFormat Exception: for input string: "shashi"
at java.base/lang.NumberFormatException.forInputString(NumberFormatException.java:65
at java.Integer.parseInt(Integer.java:652)
(Integer.java:770)
at demo.getData()
at demo.main()

→ javap java.lang.NumberformatException,
→ javap java.lang.IllegalArgumentException.
→ java p java.lang.RuntimeExcreption.
→ java p java.lang.Exception
→ javap java.lang.Throwable.
→ javap java.lang.Error

```
int data=0;
try{
    data = Integer.parseInt(br.readline());
}catch(NumberformatException obj;){
    // obj.printStackTrace();
}
sop(data);
}
```

lect75 20-8-23

Exception Handling-02

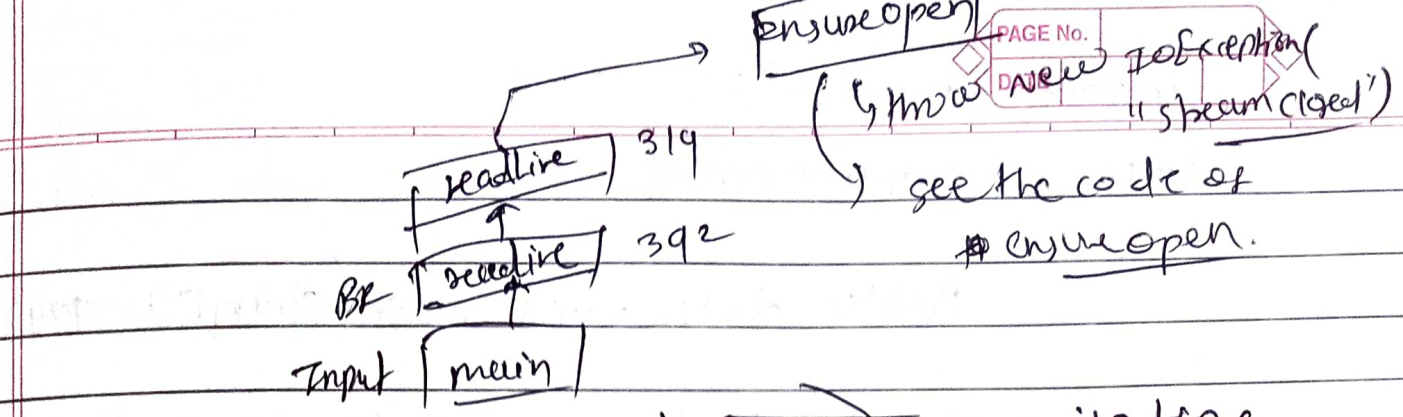
delegat करणे ⇒ throw करे

```
import java.io.*;
class {
    public static void main(String [] args) throws IOException {

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String str = br.readLine();
      sop(str);
      br.close();
      String str2 = br.readLine();
       sop(str2);
} }
```

⇒ instagram कुठला server वापरतय कुठला cloud वापरतंय
⇒ server वरती load आला की load balancer असतात.
   ↓
   server ला automic split करतं.
   ↓
   है काम करणारी माणसं devops असतात.

⇒

o/p: shashi
  shashi
  Exception in main IOException : stream closed
    at ensureopen( )
    at readLine( )
    at readLine ( )
    at main( )



→ throws हा keyword देखील compiletime exception साठी कोणता जातो.

* try, catch and finally

→ Runtime exception throws करून काय फायदा नाही output same च येतं.

⇒ जो riskey code ⇒ try मध्ये
Handling code ⇒ catch मध्ये.

⇒ try मध्ये code मध्ये maximum दोन ते तीन line असाव्या पाहीजेत

⇒

```
class Demo{
    Public static void main(String [] args){
        SOP("start main");
        try {                    throw new ArithmeticException
            SOP(10/0); // RISKY Code
        } Catch(ArithmeticException obj){
                    we can write (Exception → parent class)
            SOP("Exception occured");
                                    // Handling Code
        }
        SOP("End main");
    }
}
```

Exception table मध्ये नोंदी केला.

inmethod Area

Throwable
↑
Exception
↑
RuntimeException
↑
ArithmeticException

RuntimeException
↑
IllegalArgumentException
↑
NumberFormatException

```
import java.io.*;
class ExceptionDemo{
    public static void main(String[] args){
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String str = br.readLine();
    sop(str);
// int data = Integer.parseInt(br.readline());
    // sop(data);
    int data=0;
    try{
        data = Integer.parseInt(br.readLine());
    }catch(NumberFormatException obj){
        sop("Please Enter integer data");
        data = Integer.parseInt(br.readLine());
    }
    sop(data);
}}
```

```
try{
    data = Integer.parseInt(br.readline());
}catch(IllegalArgumentException obj){
    sop("** Please Enter integer data");
    data = Integer.parseInt(br.readline());
}catch(Exception obj){
    sop("Please enter integer data");
    data = Integer.parseInt(br.readline());
}
```

switch case{ default

child → parent

finally block {

```
class Demo{
    p s void main(String [] args){
        for (int i=0; i<10; i++){
            sop( " In loop");
            Thread.sleep( 5000);
        }
    } }
```

error:- unreported exception InterruptedException; must be caught or declared to be thrown
Thread.sleep(5000);
^

handled by my catch →

```
try {
    Thread.sleep( 5000);
} catch( IOException obj){

}
```

error: exception IOException is never thrown in body of corresponding try statement
} catch (IOException obj){
  ^

handled →

```
try {
    Thread.sleep( 500);
} catch (RuntimeException obj){

} catch (InterruptedException obj){
```

Unchecked exception ← वापरला तर compiler error देणार नाही.

आणि checked exception ते विनाकारण catch करू शकत नाही.

Munik (for shipping company (गोदाम)).

finally block => हा connection open/close साठीच आवर्जून वापरतो.

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

    Demo obj = new Demo();
    obj.m1();
    obj = null;

    try{
        obj.m2();
    }catch(NullPointerException obj5){
            sop("Here");
    }finally{
        sop("connection closed");
    }
    sop("end main");
}
```

o/p :=> Here
connection closed
end main

```
class Demo{
    void m1(){

    }
    void m2(){

    }
```

```
public static void main(String[] args){
    Demo obj = new Demo();
    obj.m1();
    obj = null;
    try{
        obj.m2();
    }catch(ArithmeticException obj5){
        sop("Here");
    }finally{
        sop("connection closed");
    }
    sop("end main");
}
```

not match Exception → it go to the default exception handler through the finally block

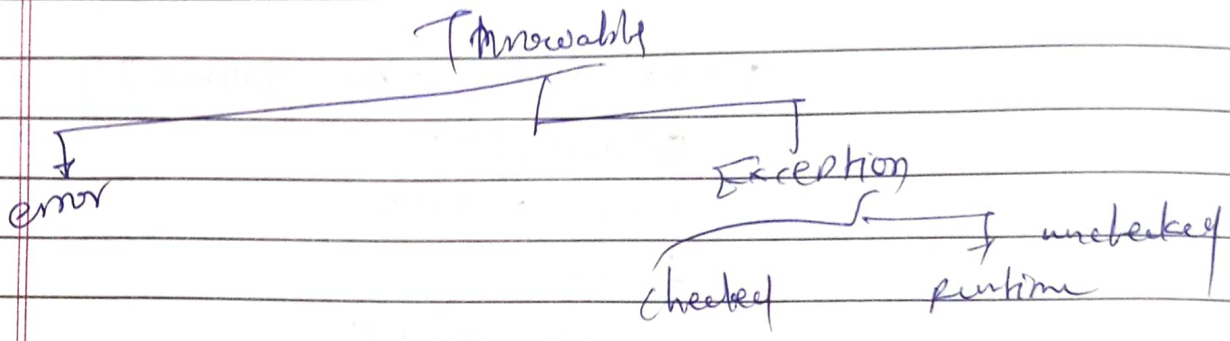
finally Here connection close chance

o/p: connection closed
Exception in thread main java.lang.NullPointerException
at demo.main

there is abnormal termination

Lect 76
21-8-23

Exception Handling - 03




```
class StackDemo{
    static void fun (int x){
        sop(x);
        fun(++x);
    }
    p s v main (String [] args){
        fun(1);
    } }
```

op: 1
10670
Exception in thread main StackOverflowError

→ अगर exception handle नहीं होगा stack की size बढ़ती रहेगी.

→ javap java.lang.StackOverflowError
(child of virtual machine error)

```
try{
fun(1);
} catch(StackOverflow obj){
    sop("stackoverflow");
}
```

User story

Technical work.

PIP
Performance Improvement plan

PAGE No.
DATE

```
try {
    func();
} catch (StackOverflowError obj){

sop("Exception in Thread" + Thread.currentThread().
        obj.printStackTrace();          getName());
}
```

* User Defined exception *

Throw (clause) or keyword.

```
public static void main (String[] args){
    ~~int x = sc~~
    Scanner sc = new Scanner(System.in);
    int x = sc.nextInt();
    try {
        if(x==0){
            throw new ArithmeticException
                    ("Divide by zero");
        }
        sop(10/x);
    }catch(ArithmeticException ae){
        sop("Exception in thread" + Thread.
        ae.printStackTrace();          CurrentThread.
        // ae.getmessage()
        // ae.toString();
```

```java
import java.util.Scanner;
class DataOverflowException extends RuntimeException{
        DataOverflowException(String msg){

                // super(msg);

}       }
class DataUnderflowException extends RuntimeException{
        DataUnderflowException(String msg){

                //super(msg);

        }   }
class ArrayDemo{
        public static void main(String[] args){
                int arr[] = new int[5];
                Scanner sc = new Scanner(System.in);
                sop("Enter integer value");
                sop("Note : 0<element<100);
                for(int i=0; i<arr.length; i++){
                        int data= sc.nextInt();

                        if(data<0)

                            throw new DataUnderflowException
                            ("Mitra Data opeksha lahan ahe");

                        if(data>0)

                        throw new DataOverflowException
                        ("Mitra Data 100 पेक्षा मोठा आहे");

                        arr[i] = data;
        }       for(int i=0; i<arr.length; i++){
                        sop(arr[i]+" ");

        }   }   }
```

महत्वाचे Exception table & आता In one code.

real time exception

example

```
//import java.io.*;
import java.util.Scanner;
class DataOverflowException extends IOException{
    DataOverflowException(String msg){
        super(msg);
    }
}
class DataUnderflowException(String msg){
    super(msg);
}
}
class ArrayDemo{
    public static void main(String[] args) throws
DataUnderflowException, DataOverflowException{
```

same code

examples :- class length of password

=> Unsufficient money Exception

=> Rs in

=> Gmail

@

=> City not found exception

=>

class demo{

psvm

try{
x= Integer.parseInt(br.readline());
}catch(IoException ie){
Sop("Exception-1");
}catch(NumberformatException nf){
Sop("Exception-2");
}catch(NullpointerException np){
Sop(Exception3);
}catch Exception e){
Sop(Exception4);
}

see the bytecode

only for try catch

Exception Table

from	To	Target	
10	13	14	IoException
10	13	17	Numberformat
10	13	20	Nullpointer

etc

lect 77 ⇒
22-8-23 :⇒

5000ms = 5 sec

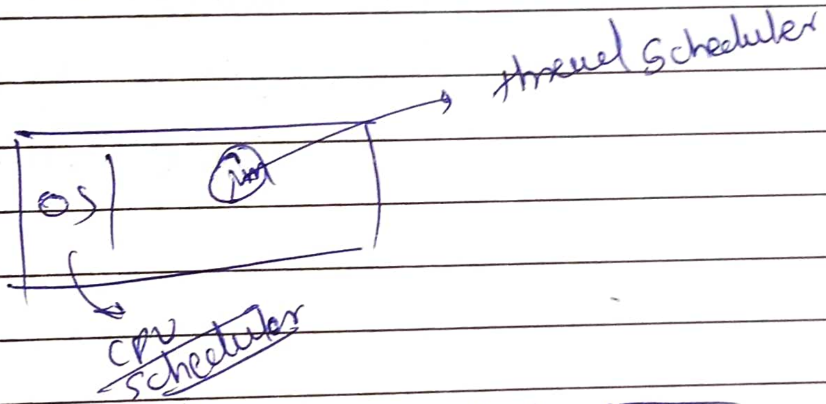
★ MultiThreading-01 ★

वाढवायची होती

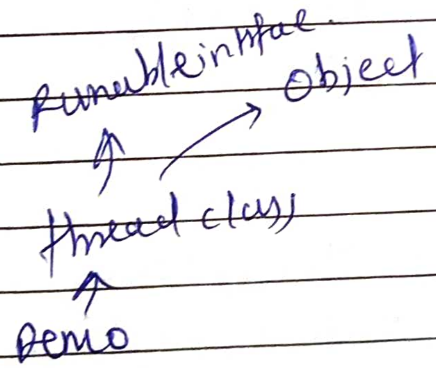
⇒ ज्यावेळेस आपल्याला performance वाढी पाहिजे असते तेव्हा आपण multithreading वापरतो.
ज्यावेळेस आपल्याला sequence matter करत नाही.

⇒ ① Thread class
② Interface → runnable interface.

two types of thread ⇒ Kernal thread & user thread → internal



⇒ प्रत्येक thread साठी java stack असतात.




```
class threadDemo{
    for(int i=10; i<20; i++){
        sop(i);
    }
    p s v main(String args[]){
        for(int i=0; i<10; i++){
            sop(i);
            thread.sleep(2000);
        }
}
```

op: unreported exception InterruptedException; must be caught or declared to be thrown.
thread.sleep(2000);
^
1error

Creating thread using thread class (Part 1)

PAGE No.
DATE

```
class MyThread extends Thread{
    public void run(){
        for(int i=0; i<10; i++){
            sop("In run");
        }
    }
}

class ThreadDemo{
    Public static void main(String[] args){
        MyThread obj = new MyThread();
        obj.start();
        for(int i=0; i<0; i++){
            sop("In main");
        }
    }
}
```

```
class MyThread extends Thread{
    public void run(){
        for(int i=0; i<10; i++){
            Sop("In run");
            try{
                Thread.sleep(1000);
            }catch(InterruptedException ie){
            }
        }
    }
}
```

run method Thread class ची आहे. तिथे या method ने exception throws केले नाहीत म्हणून इथ override केल्यावर आपण throws exception नाही करू शकत. यासाठी आपल्याला दुसरा option => try catch

class ThreadDemo

public static void main(String[] args) throws InterruptedException {

MyThread obj = new MyThread();

obj.start();

for(int i=0; i<0; i++){

sop("In main");

Thread.sleep(100);

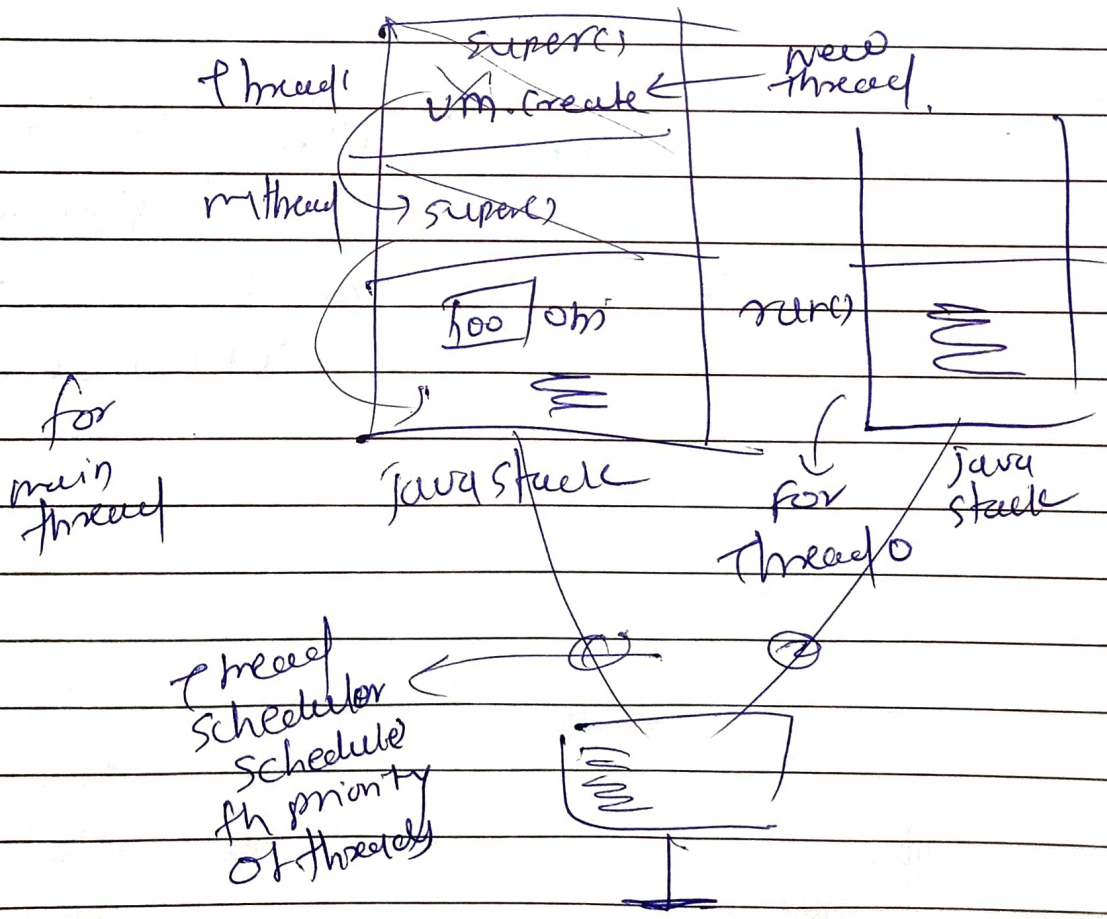
}

}

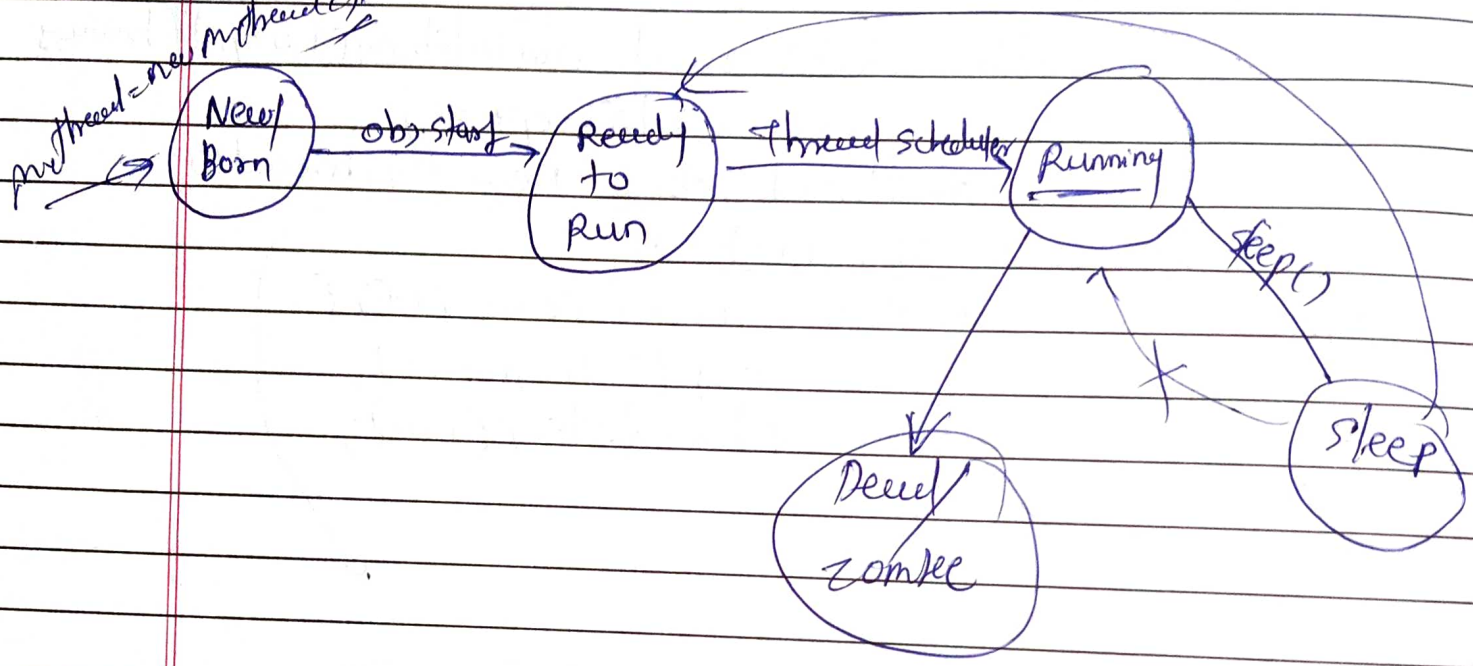
}

start() see source code

O/P: In main
In run



Thread life Cycle :=>




```
SOP("Child Thread = " + Thread.currentThread().
                        getName());
SOP("Main Thread = " + Thread.currentThread().
                        getName());
```

lect 78
seen :- 24-8-23

PAGE No.
DATE

multithreading - 2

पुढील गोष्टी multithreading मध्ये Compulsory करायला पाहिजेत.

1. Thread class extend करायला पाहिजे
2. Run method ला override करायला पाहिजे
3. start method call करायला पाहिजे

→ Thread class च्या constructor मधून दुसऱ्या constructor ला call आहे.

No argument constructor मधून
This(null, null)
↳ null runnable ने typecast केला
↙
दोन parameter वाल्या constructor मध्ये call जातो.

→ Thread मधल्या start method मधून start0 ला call आहे.

```
class mythread extends Thread{
    public void run(){
        sop("In run");
        sop(Thread.currentThread().getName());
//  public void start(){
        sop("In mythread start");
        run();
    }
class ThreadDemo{
    public static void main(String[] args){
        mythread obj = new mythread();
        obj.start();
        sop(Thread.currentThread().getName());
```

o/p:

```
class myThread extends Thread{
        public void run(){
                sop("In run");
            sop(Thread.currentThread().getName());
    }

    public void start(){
            sop("In mythread start");
        run();
    }
}
class ThreadDemo{
        MyThread obj = new MyThread();
            obj.start();
            sop("Thread.currentThread.getName());
```

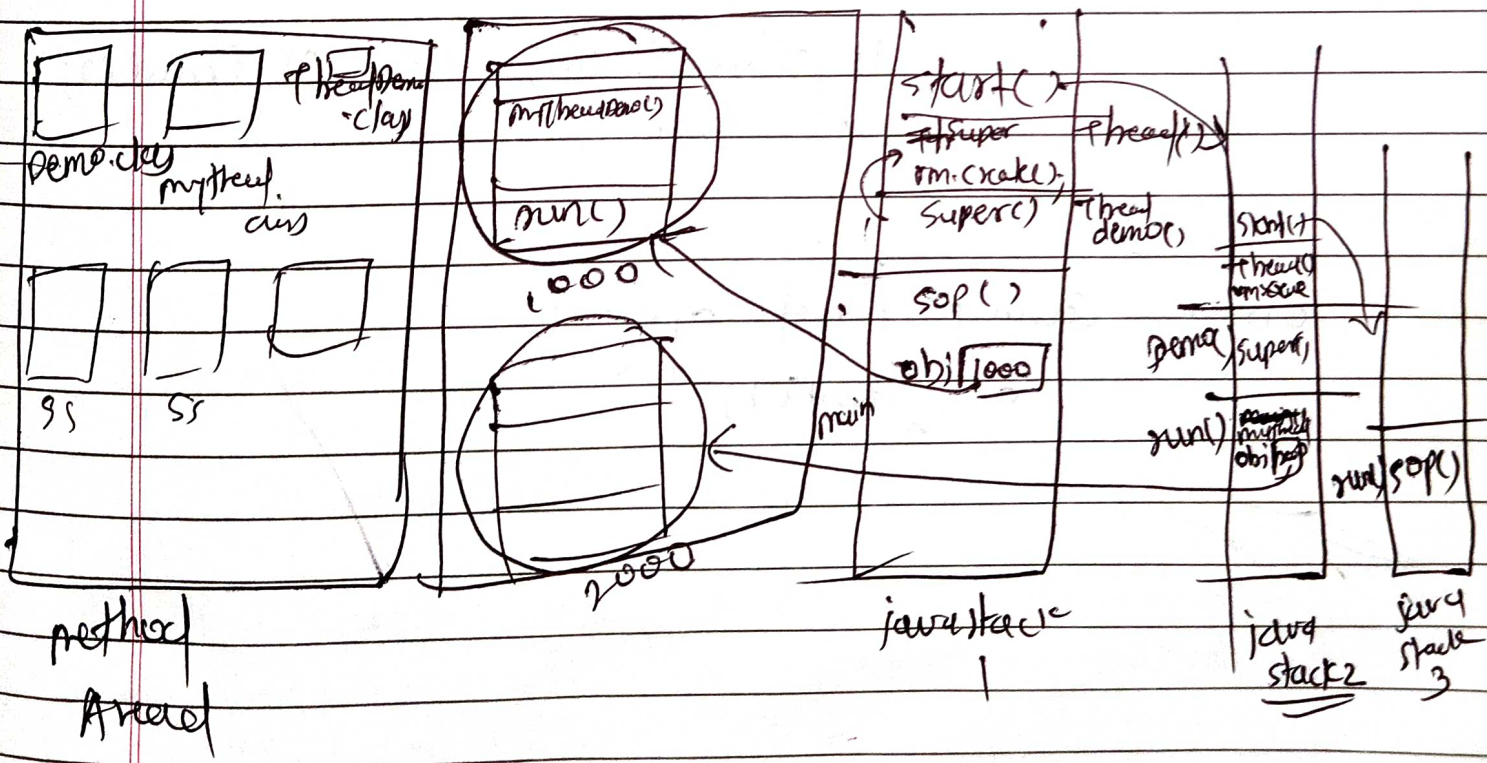
start()ला override केला तर multithread use करायला [illegible] नाही

O/P:=) In mythread start
In run
main
main

```
class Demo extends Thread{
        public void run(){
                System.out.println("Demo:"+
            Thread.currentThread().getName());
} }
class MyThread extends Thread{
        public void run(){
            sop("MyThread:"+Thread.currentThread().
                getName());
```

```
        Demo obj = new Demo();
        obj.start();
    }
}
class ThreadDemo{
    public static void main(String[] args){
        sop("ThreadDemo:" + Thread.currentThread().
            getName());

        MyThread obj = new MyThread();
        obj.start();
    }
}
```

op:- ThreadDemo : main
MyThread : Thread-0
Demo : Thread-1



parent method | child process & parent process in OS

PAGE No.
DATE

Creating childThread Using Runnable Interface

```
Runnable{
    public abstract void run();
}
```

```
class myThread implements Runnable{
    public void run(){
        sop("In run");
    }
}
class threadDemo{
    public static void main(String[] arg){
        myThread obj = new myThread();
        obj.start();
    }
}
```

error: cannot find symbol.
obj.start();
^

```
class MyThread implements Runnable {
    public void run() {
        sop("In run");
        sop(Thread.currentThread().getName());
    }
}

class ThreadDemo {
    p s v main(String[] args) {
        MyThread obj = new MyThread();

        Thread t = new Thread(obj);
        t.start();
        sop(Thread.currentThread().getName());
    }
}
```

call to constructor of thread class as
thread(Runnable)

MyThread ही Runnable ची child आहे म्हणून parent-child relation
MyThread type ची obj thread(runnable) accept होणार.

O/P: main
In run
Thread-0

⇒ child Thread तयार करताना Runnable ची use करणार करणार येणार.

```
class Demo {
}
class MyThread extends Demo implements Runnable { }
```

Thread {

min priority
norm priority
max priority
~~thread~~ currentThread() ⇒ object return thread

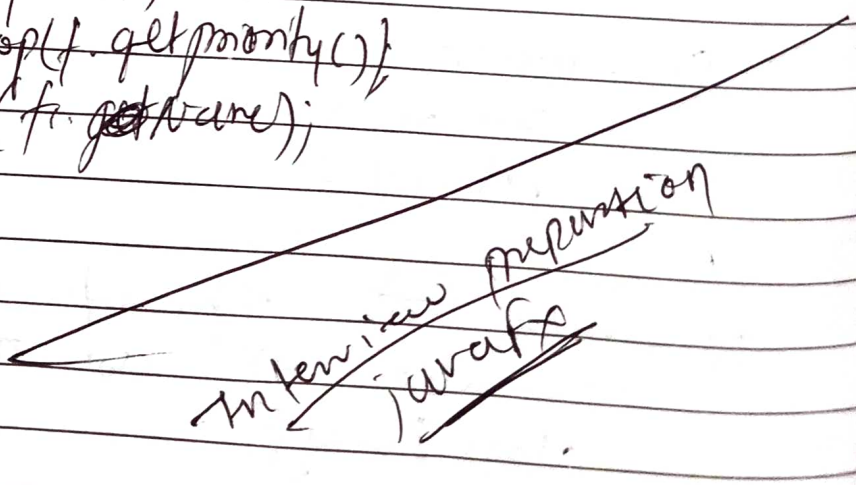
native yield(); → in cpp

constructor ⇒
Thread()
Thread(Runnable)
Thread(ThreadGroup).

synchronized void start();

Priority of Threads

class MyThread extends Thread{
}
class ThreadDemo {
public static void main(String[] args){
Thread t = Thread.currentThread();
sop(t);
//sop(t.getPriority());
//sop(t.getName());
} }



```
class MyThread extends Thread{
    public void run(){
        Thread t = Thread.currentThread();
        sop(t.getPriority());
    } // t.setPriority(-2);
}

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

        Thread t = Thread.currentThread();
        sop(t.getPriority());

        MyThread obj1 = new MyThread();
        obj1.start();
        t.setPriority(7); // setPriority(11)
                          // IllegalArgumentException

        MyThread obj2 = new MyThread();
        obj2.start();
    }
}
```

```
class MyThread extends Thread{
    public void run(){

    }
}

class ThreadDemo{
    Public static void main(String[] args){
        Thread t = Thread.currentThread();
        sop(t.getName());
        MyThread obj1 = new MyThread();
        obj1.start();
        obj1.start(); => IllegalThreadStateException
```

lect 79
25-8-23

* multiThreading - 03 *

(79) ← lect No

⇒ In cpp ⇒ inline function
↳ request to compiler to donot push the stackframe for execution for fast performance.

⇒ Back end developer ला सगळ्या आणि concurrency शिकावं लाग.

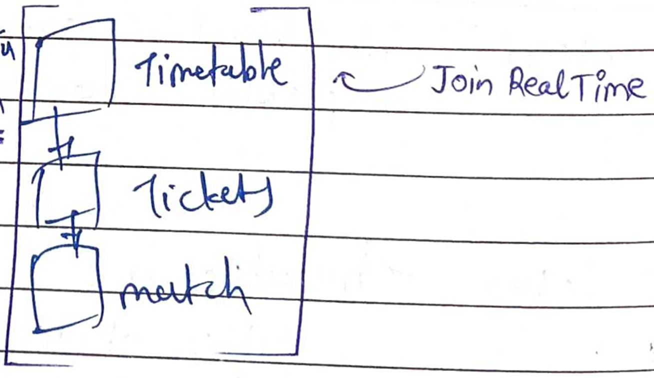
⇒ start method ला version wise बघा. (atleast 8,11)
↳ आतमध्ये call बघा ⇒ start0()
↳ native method.

⇒ even
↳ jvm आपण स्वतःला thread create करता येत नाही.
↳ तो posix thread वापरतो
↳ pthread_create() ⇒ call
3rd party library

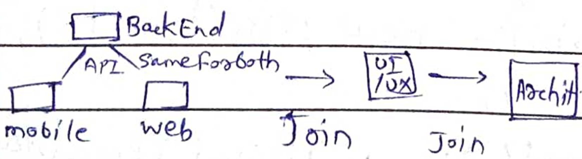
⇒ version 19 ⇒ sop(Thread.currentThread());
[#value, ,]
↓
ThreadId

by default ⇒ [ThreadName, priority, ThreadGroup]

self sleep ↳ ① sleep
② join ⇒ पूर्ण thread stop होईना run झाल्याशिवाय priority दुसऱ्याला नाही देत.
nice value ③ yield



what is API?



DATE

concurrency methods in thread class

1. sleep
2. join
3. yield.

```
class MyThread extends Thread{
    public void run(){
        sop(Thread.currentThread());
    }
}
class ThreadDemo{
    public static void main(String[] args)
        throws InterruptedException{
        sop(Thread.currentThread());
        MyThread obj = new MyThread();
        obj.start();
        Thread.sleep(100);
        Thread.currentThread().setName("Core2Web");
        sop(Thread.currentThread());
    }
}
```

error - handled by InterExcep

o/p: => Thread[main, 5, main]
Thread[Thread-0, 5, main]
Thread[Core2Web, 5, main];

```
public void run(){
    sop(Thread.currentThread());
    try{
        Thread.sleep(1000);
    }catch(IOException ie){
    }catch(InterruptedException){
    }
}
```

error
IOException cannot be write in catch.

garbage collector एक Thread Demon thread

real time example — App UI Thread, web UI Thread, Thread

backend Thread

→ आपण पाहिले, main backend code हाय... उदाहरण...

Join method

```
class Mythread extends Thread{
    public void run(){
        for(int i=0; i<10; i++){
            sop("In thread-0");
        }
    }
}

class ThreadDemo{
    public static void main(String[] args) throws
                    InterruptedException{
        MyThread obj = new MyThread();

        obj.start();
        obj.join();    → main thread थांबतो; thread-0 चे काम पूर्ण होत नाही तो पर्यंत
        for( int i=0; i<10; i++){
            sop( "In main Thread");
```

see join method code — code पाहिले

```
class MyThread extends Thread{
    static Thread nmMain = null;
    public void run(){
        try{
            nmMain.join();
        } catch(InterruptedException obj){
        } for(int i=0; i<10; i++){
            sop( "In Thread-0");
        }
    }
}
```

```
class ThreadDemo {
    public static void main(String[] args) throws
    InterruptedException {
        myThread.nmMain = Thread.currentThread();
        MyThread obj = new MyThread();
        obj.start();
        obj.join();
        for(int i=0; i<10; i++){
            sop("In main Thread");
        }
    }
```

o/p : 1 ↑ cursor → deadlock scenario ⇒ पहले आप
पहले आप.

Thread.currentThread.join());

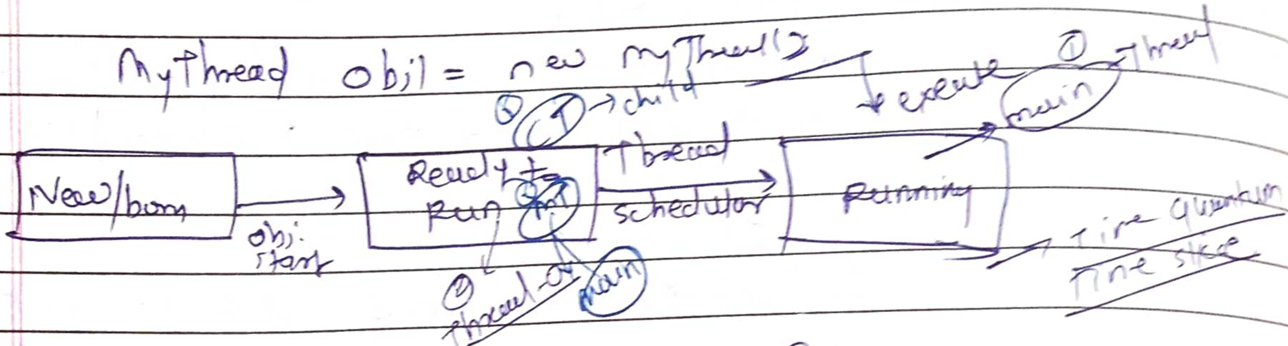
~~join~~

lect 80
seen:⇒ 27-8-23

multithreading-04

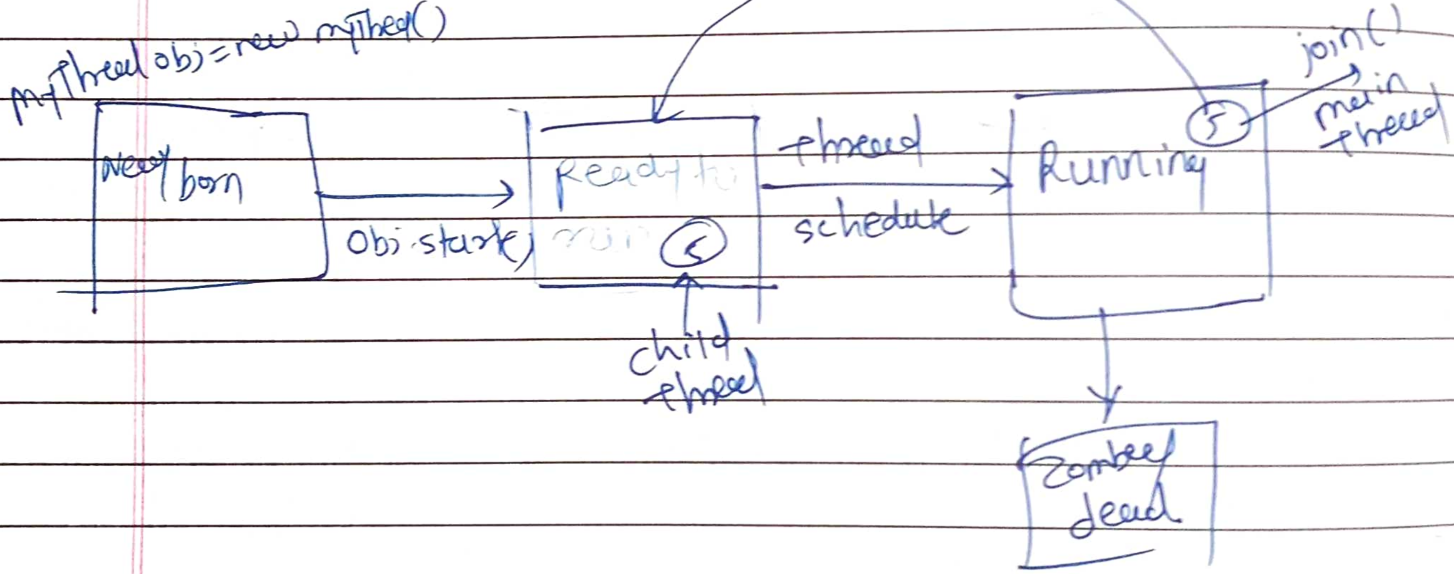
join ⇒ native

MyThread obj1 = new myThread()

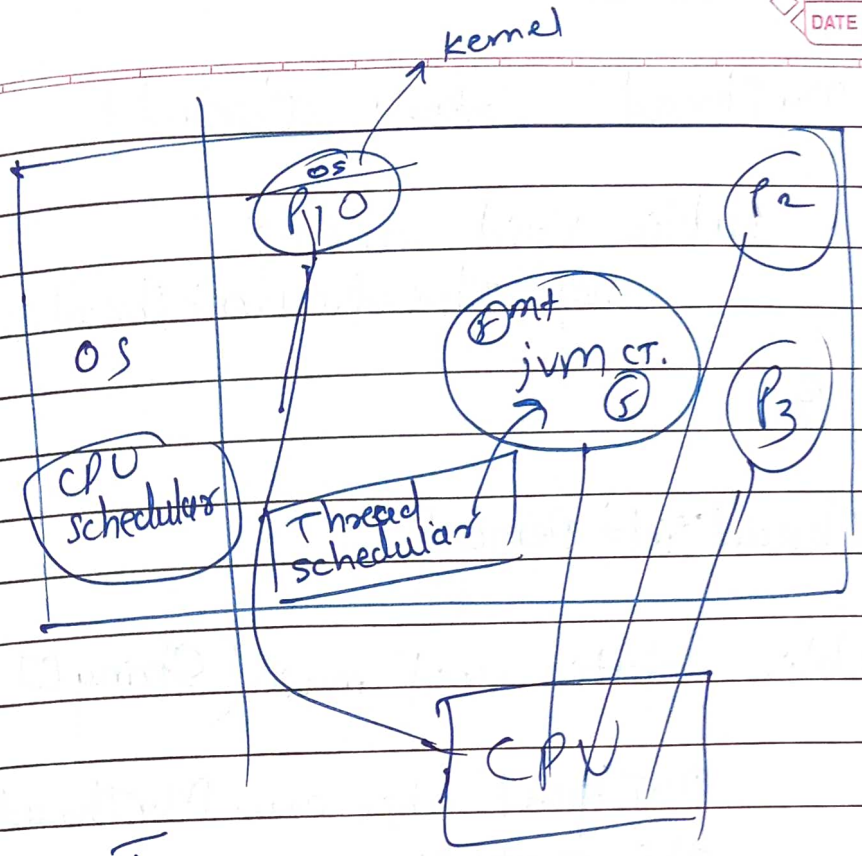


⇒ server वरती pool बनवलेले असतात ⇒ आणि नंतर user च्या request प्रमाणे thread assign होतात.

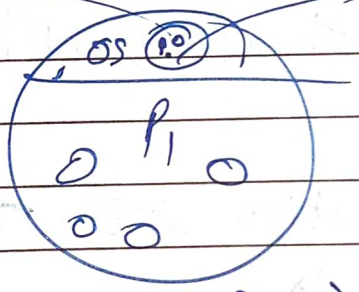
⇒ java मध्ये spring boot शिकताना कधी start method ला call करता.



⇒ Thread ला light weight process असे म्हणतात



actually kernel-level thread ला CPU मिळतो



kernel → हा process साठी काम करत असतो.

⇒ NVIDIA सारख्या कंपनी चे interview या OS level पर्यंत होतात.

⇒ CPU पर्यंत फक्त kernel thread जातो, कुठलाही process thread CPU पर्यंत नाही जात.

✱ Yield method ✱

```
class MyThread extends Thread {

    public void run() {
        sop(Thread.currentThread().getName());
    }
}

class ThreadYieldDemo {

    Public static void main(String[] args) {

        MyThread obj = new MyThread();
        obj.start();
        obj.yield();
        sop(Thread.currentThread().getName());
    }
}
```

write for loop in here →

⇒ yield method native ~~नाही~~. आहे.

↳ आधी non native होती

⇒ yield ही request आहे. yield ~~होईलच~~ याची गॅरंटी नाही.

→ sleep() ला call ज़रूर वापरावा.
→ join() ला call without parameter नसावा.

KT → Knowledge transfer

* Thread Group *

In Thread class ⇒ ThreadGrp.getname

In Thread Group ⇒ getname

जेव्हा आपण file search करतो तेव्हा एक thread काम करत असेल तर खूप time लागतो. त्यासाठी multiple thread सोडले जातात मग जर त्यातल्या एखाद्या thread ला जर file सापडली तर बाकीच्यांना थांबवायला पाहिजे मग ते एका group मध्ये असतील तर ते एकत्र kill होतील.

→ destroy() नावाची मेथड → bytecode

↳ javap java.lang.ThreadGroup

* ThreadGroup ची by default priority 10 असते

```
class myThread extends Thread{
        public void run(){

            sop(Thread.currentThread().getThreadGroup());

        }
}

class ThreadGroupDemo{
        p s v main(String[] args){
        myThread obj = new MyThread();
            obj.start();
            sop(obj.getName());
}   }   //sop( Thread.currentThread().getName());
```

(protocols: आपण new thread create करत असताना नाव दिले पाहिजे, thread() String parameter constructor ला call करावे)

```
class MyThread extends Thread{
    public void run(){
        Sop(getName());
        Sop(Thread.currentThread().getThreadGroup());
    }
}
class ThreadGroupDemo{
    public static void main(String[] args)
                    throws InterruptedException{
        MyThread obj = new MyThread();
        obj.start();
        Thread.sleep(1000);
        obj.setName("xyz");
    }
}
```

o/p: Thread-0
java.lang.ThreadGroup[name=main, maxpri=10]

recommended way for assign name to thread

```
class MyThread extends Thread{
    MyThread(String str){      <- IMP
        super(str);
    }
    public void run(){
        Sop(getName());
        Sop(Thread.currentThread().getThreadGroup());
    }
}
class ThreadGroupDemo{
    p s v main(String[] args){
        MyThread obj = new MyThread("xyz");
        obj.start();
    }
}
```

Asynchronous thread name

o/p:⇒ XYZ
java. lang. ThreadGroup[name=main, maxpri=10].

```
class myThread extends thread{
    myThread (String str){
        Super(str);
    }
    public void run(){
        sop(getName());
        sop(thread.currentThread().getThreadGroup());
    }

class ThreadGroupDemo{
    p s v main (String [] args){
        myThread obj = new myThread("XYZ");
        obj.start();
        myThread obj1 = new myThread("pqr");
        obj1.start();
        myThread obj2 = new myThread();
        obj2.start();
    }
```

कोणता run thread या नाव नाय असल तर

O/P: error: constructor is parameterized.

→ check if we give same name / priority

(Thread t "counter") → check This code in version 11 & 19
↓
getname method in Thread class

→ to solve this error we have to make a no argument Constructor.

```
class MyThread extends Thread{
    MyThread(ThreadGroup tg, String str){
        super(tg, str);
    }
    public void run(){
        sop(Thread.currentThread());
    }
}
class ThreadGroupDemo{
    public static void main(String[] args){
        ThreadGroup pthreadGp = new ThreadGroup
                                ("Core2Web");
        MyThread obj1 = new MyThread(pthreadGp, "C");
        MyThread obj2 = new MyThread(pthreadGp, "java");
        MyThread obj3 = new MyThread(pthreadGp, "Python");
        obj1.start();
        obj2.start();
        obj3.start();
```

child thread group

```
        ThreadGroup cthreadGp = new ThreadGroup(
                                pthreadGp, "Incubator");
        MyThread obj4 = new MyThread(cthreadGp, "Flutter");
        MyThread obj5 = new MyThread(cthreadGp, "ReactJS");
        MyThread obj6 = new MyThread(cthreadGp, "SpringBoot");
        obj4.start();
        obj5.start();
        obj6.start();
    }
}
```

Lect 81 Multithreading-05

seen 21-8-23

PAGE No.
DATE

Active Count →

Runnable आणि thread यांच्यात फरक → ① एक extra class extends करता येतो

② thread pool चा concept मध्ये ते threadpool execute करायची method आहे ना → execute या method च्या आत आणि तिला parameter runnable आहे.

server वगैरे create करताना ही multithreading ची concept वापरली जाते तिथे आपल्याला फक्त runnable वापरता येतो.

```
class MyThread extends Thread{
    MyThread(ThreadGroup, tg, String str){
        super(tg, str);
    }
    public void run(){
        sop(Thread.currentThread());
    }
}

class ThreadGroupDemo{
    public static void main(String[] args){
        ThreadGroup pThreadGp = new ThreadGroup("India");   // Parent ThreadGroup
        MyThread t1 = new MyThread(pThreadGp, "Maha");
        MyThread t2 = new MyThread(pThreadGp, "Goa");
        t1.start();
        t2.start();
        ThreadGroup cThreadGp = new ThreadGroup(pThreadGp, "Pakistan");   // child ThreadGroup
        MyThread t3 = new MyThread(cThreadGp, "Karachi");
        MyThread t4 = new MyThread(cThreadGp, "Lahore");
        t3.start();
        t4.start();
    }
```

methods

int activeGroupCount()
int activeCount()

} java.lang.ThreadGroup

⇒ thread grp संबंधी सगळ्या method try करा.

```
sop(pThreadGp.activeCount());
sop(pThreadGp.activeGroupCount());
```

o/p :- ⇒ 4

1

```
class myThread extends Thread{
        myThread(ThreadGroup tg, String str){
                super(tg, str);
        }
    public void run(){
            sop(Thread.currentThread());
        try{
            Thread.sleep(5000);
        }catch(InterruptedException ie){
                    sop( ie.toString());
        }
    }
class ThreadGroupDemo{
    public static void main(String[] args) throws
                InterruptedException{
            ThreadGroup pThreadGp= new ThreadGroup
                            ("India");
```

copy cp program1.java program2.java.

PAGE No.
DATE

```
MyThread t1 = new MyThread(pThreadGP, "Maha");
MyThread t2 = new MyThread(pThreadGP, "Goa");
t1.start();
t2.start();
ThreadGroup cThreadGP = new ThreadGroup(pThreadGP,
                                       "Pakistan");
MyThread t3 = new MyThread(cThreadGP, "karachi");
MyThread t4 = new MyThread(cThreadGP, "Lahore");
t3.start();
t4.start();
ThreadGroup cThreadGP2 = new ThreadGroup
                          (pThreadGP, "Bangladesh");
MyThread t5 = new MyThread(cThreadGP2, "Dhaka");
MyThread t6 = new MyThread(cThreadGP2, "mirpur");

t5.start();
t6.start();
cThreadGP.interrupt();
Sop(pThreadGP.activeCount());
Sop(pThreadGP.activeGroupCount());
} }
```

मराठी उदाहरण java ची concept आहे.

→ interrupt() thread class मध्ये पण आहे.

⁂ ⁂ ⁂ ⁂ ⁂ ⁂ ⇒ for backend

PAGE No.
DATE

```
class MyThread implements Runnable{
    public void run(){
        sop(Thread.currentThread());
        try{
            Thread.sleep(5000);
        }catch(InterruptedException ie){
            sop(ie.toString());
        }
    }
}
class ThreadGroupDemo{
    p s v main(String[] args) throws InterruptedException{
    ThreadGroup pThreadGp = new ThreadGroup("India");
    MyThread obj1 = new MyThread();
    MyThread obj2 = new MyThread();
    Thread t1 = new Thread(pThreadGp, obj1, "Mehdi");
                  ↓ Constructor (ThreadGroup, runnable, String)
    Thread t2 = new thread(pThreadGp, obj2, "Gou");
    t1.start();
    t2.start();
    ThreadGroup cThreadGp = new ThreadGroup(
                                pThreadGp, "Pakistan");
    MyThread obj3 = new MyThread();
    MyThread obj4 = new MyThread();

    Thread t3 = new Thread(cThreadGp, obj3,
                                "Karchi");
    thread t4 = new thread(cThreadGp, obj4, "Lahor
```

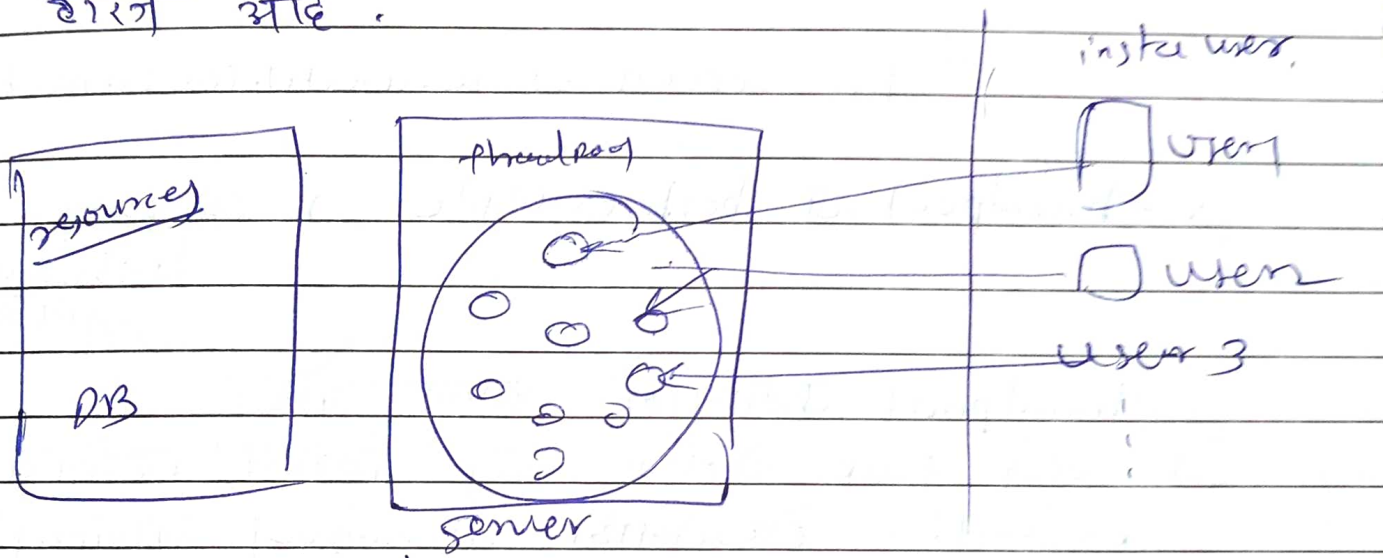
interface parent कहीही class बसतो.

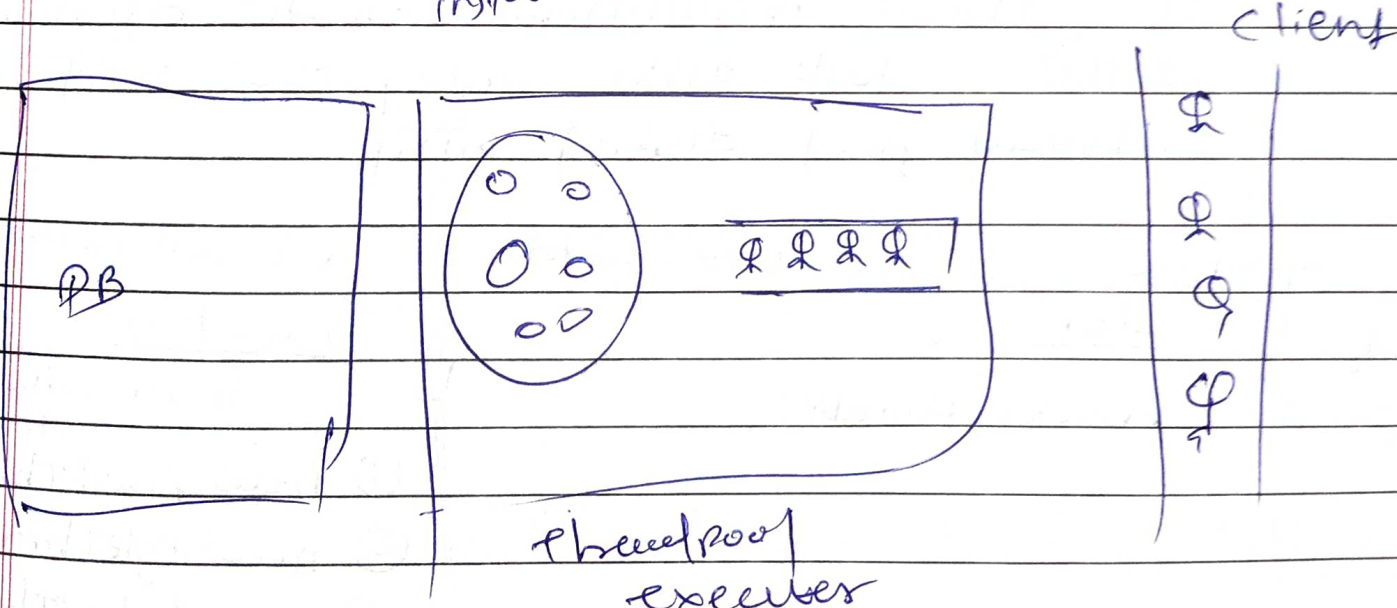
PAGE No.
DATE

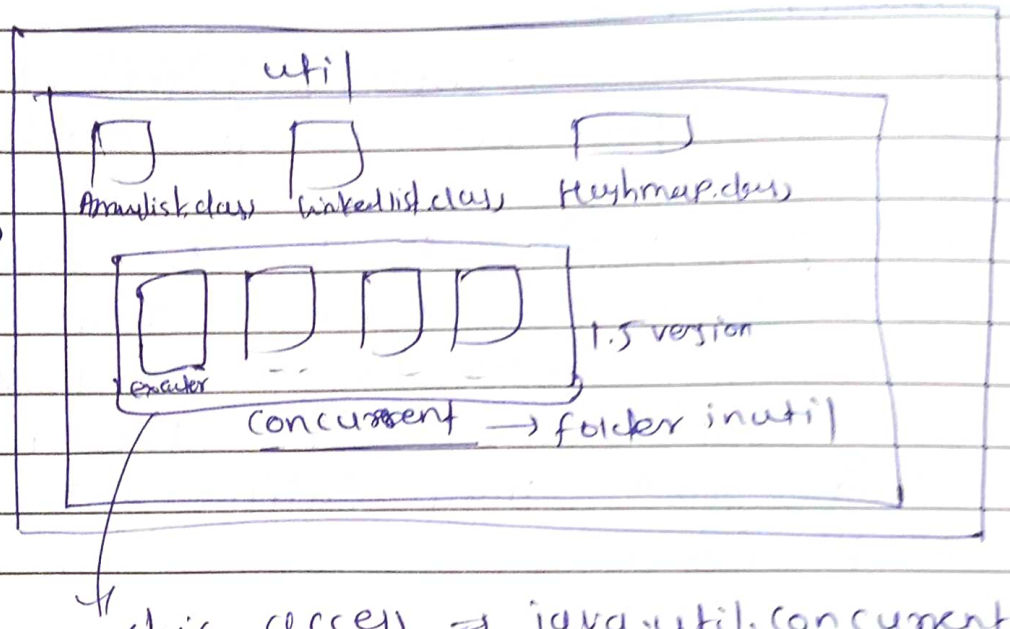
Thread pool :-

→ स्वतः चं server बनवायचं असेल तर ही Thread pool ची concept माहीत

→ Thread pool मध्ये threads ready to run मध्ये असल्या सारख्ये असतात त्याना Thread class चा constructor or start() करण्याची गरज नाही कारण ते already बनवून ठेवले आहेत. फक्त त्यांना run करण्याची गरज आहे.







java.util.*;

this access ⇒ java.util.concurrent;

⇒ threadpool च best example ⇒ company
↳ threads म्हणजे employee

⇒ threadpool कशासाठी आला आहे?
⇒ जर्से task येतील तर लगेच त्यांचे fast execution होण्यासाठी threadpool आणला.
जर thread पहिल्यापासून create करून ठेवले तर त्यामुळे task साठी delay होऊ शकतो म्हणून threadpool वापरला जातो.

Interface Executor — java.util.concurrent.Executor.

↳ execute(Runnable);

java.util.concurrent.Executors

class Executors
↳ all methods are static

① newFixedThreadPool(int);
② newSingleThreadExecutor();
③ newCachedThreadPool();

return type:

ExecutorService

server तैयार running अवस्था.

Concurrent. Executors

PAGE No.
DATE (Day)

interface Executor

interface ExecutorService

class AbstractExecutorService → java.util.concurrent.AbstractExecutorService

class → ThreadPoolExecutor() → java.util.concurrent.ThreadPoolExecutor

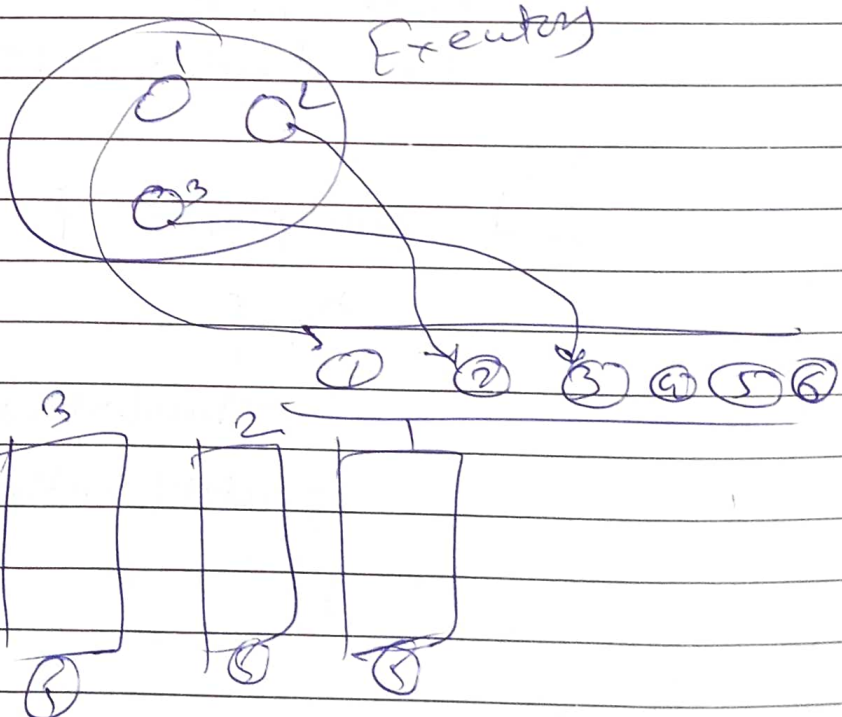
Executors → object parent.

run() {

Threadpool

```
import java.util.concurrent.*;
class MyThread implements Runnable{
    int num;
    MyThread (int num){
        this.num = num;
    }
Public void run() {
    sop(thread.currentThread() + "start thread:" + num);
    dailyTask();
    sop(thread.currentThread() + " End thread:" +num)
}
void dailyTask() {
    try {
        Thread.sleep (5000);
    } catch( InterruptedErephon ie ){
    }
}
}
```

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

// ExecutorService ser = Executors.newFixedThreadPool(5);
ExecutorService ser = Executors.newCachedThreadPool();
        for(int i= 1; i<=6 ; i++){

Runnable  MyThread obj = new MyThread(i);

                ser.execute(obj)
                            (Runnable

        }
        ser.shutdown();

}   }
```



thread ने काम संपन्न की ते परत pool मध्ये जाऊन थांबतात.

java.util.concurrent.ExecutorService
+ void shutdown();
shutdownNow();

→ ही method नसते task येणीस तोपर्यंत thread तयार ठेवते.

→ याच्यातल्या methods वापरा.
स्वतः शोधा गोष्टी.

→ Synchronize concept collection मध्ये शिकवणार आहेत.
जसे BufferReader
StringBuilder & StringBuffer

lect 82
seen: 1-9-23

Major

PAGE No.
DATE

: Packages :

→ ThreadPoolExecutor ⇒ हा most derived class → use करायचा
राहतो सगळ्या latest गोष्टी असतात.

```
import java.util.concurrent.*;
```

RMI ⇒ Remote method invokation.

```
class MyThread implements Runnable{
    int num=0;
    MyThread (int num){
        this.num =num;
    }
    public void run(){
        sop("start:" + num + Thread.currentThread());
        dailyTask();
        sop("End :" + num + Thread.currentThread())
    }
    Void dailyTask(){
        try{
            Thread.sleep(40000);
        }catch(InterruptedException ie){
        }
    }
}
class ThreadPoolDemo{
    Public static Void main(String[]args){
        ThreadPoolExecutor tpe=(ThreadPoolExecutor)
            Executors.newFixedThreadPool(2);
```

jconsole

>jshell> int x = 10;
x ==> 10

PAGE No.
DATE

```
ThreadPoolExecutor tpe2 = (ThreadPoolExecutor)
                    Executors.newFixedThread
                    Pool(2);

for(int i=1; i<=4; i++){
    MyThread obj = new MyThread(i);
    tpe1.execute(obj);

}

for(int i=1; i<=4; i++){
    MyThread obj = new MyThread(i);
    tpe2.execute(obj);

}
tpe1.shutdown();
tpe2.shutdown();

}
```

⇒ install jconsole ⇒ for version.

⇒ see performance in UI.

Devops → checks
Team performance

Packages

Packages

- predefined (10 %)
  - lang
  - io
  - util
  - sql
  - javax
  - awt
  - swing
  - .net
- User Defined (90%)

→ package चे नाव small letter मध्ये लिहायचे कारण त्यांचे package स्थानात small letter मध्ये असतात.

Addition.java

```
Package arithfun;
Public class Addition {
        int num1=0;
        int num2=0;
    public Addition( int num1, int num2){
        this.num1 = num1;
        this.num2 = num2;
    }
    Public int add(){
        return num1+num2;
    }
}
```

(Public → protected)

javac -d . Program2.java

(. → directory)

> cat addition.java client.java.

PAGE No.
DATE

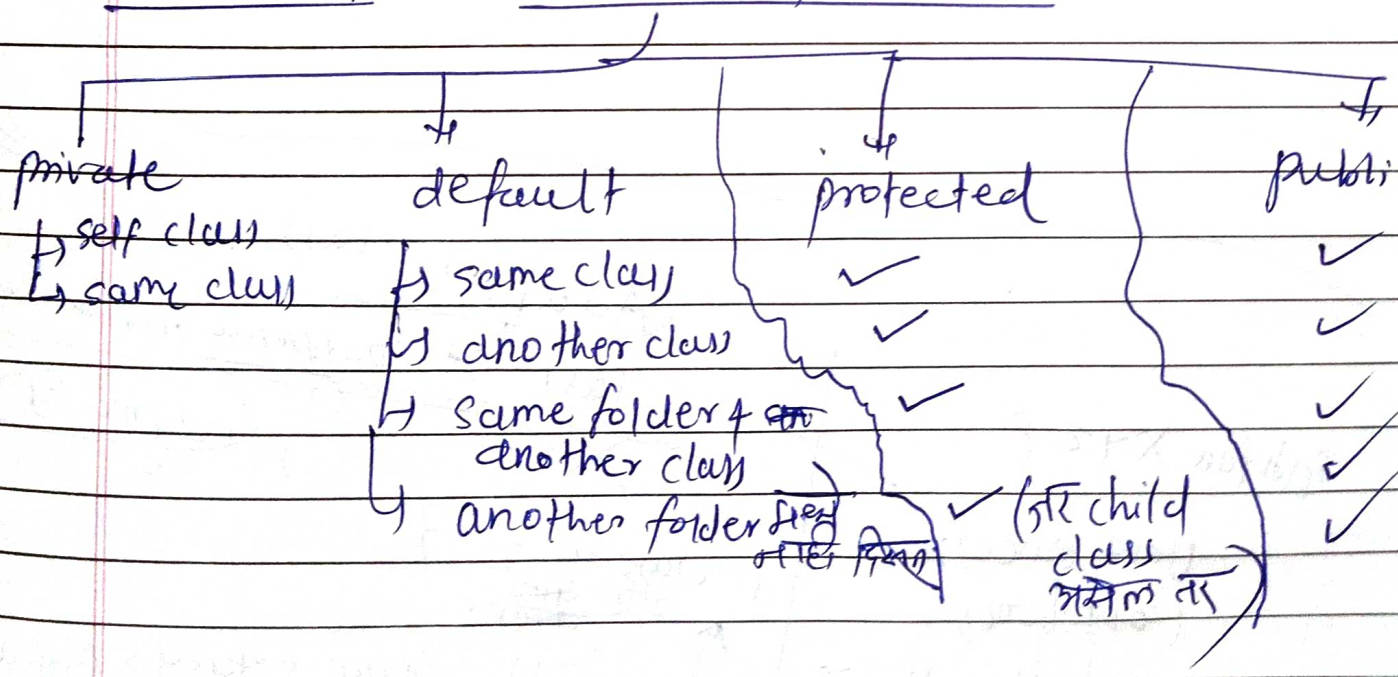
> client.java.

> javac -d . Addition.java
(-d : directory ; . : current folder)

```
import java.util.Scanner;
import arithfun.Addition;
class Client{
    public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
      int x = sc.nextInt();
      int y = sc.nextInt();

      Addition obj = new Addition(x,y);
      sop( obj.add());

} }
```

→ sub directory
package core2web.Incubator

* Access Specifier in packages *



```
import java.util.Scanner;
import Dummy.Addition;

class Client extends Addition{
    Client(int x, int y){
        super(x,y);
    }

Public static void main (String[] arg) {
    Scanner sc = new Scanner(System.in);
    int x = sc.nextInt();
    int y = sc.nextInt();
    Client obj = new Client(x,y);
    sop( obj.add());
}
```

add() is protected in Addition.java.

Adapter class

class AbstractDemo implements XYZ{
() { empty blocks }
30

interface XYZ{
void one();
void two();
void three();
{ 30 methods }

class Demo extends AbstractDemo

lect 83 seen: 1-9-23

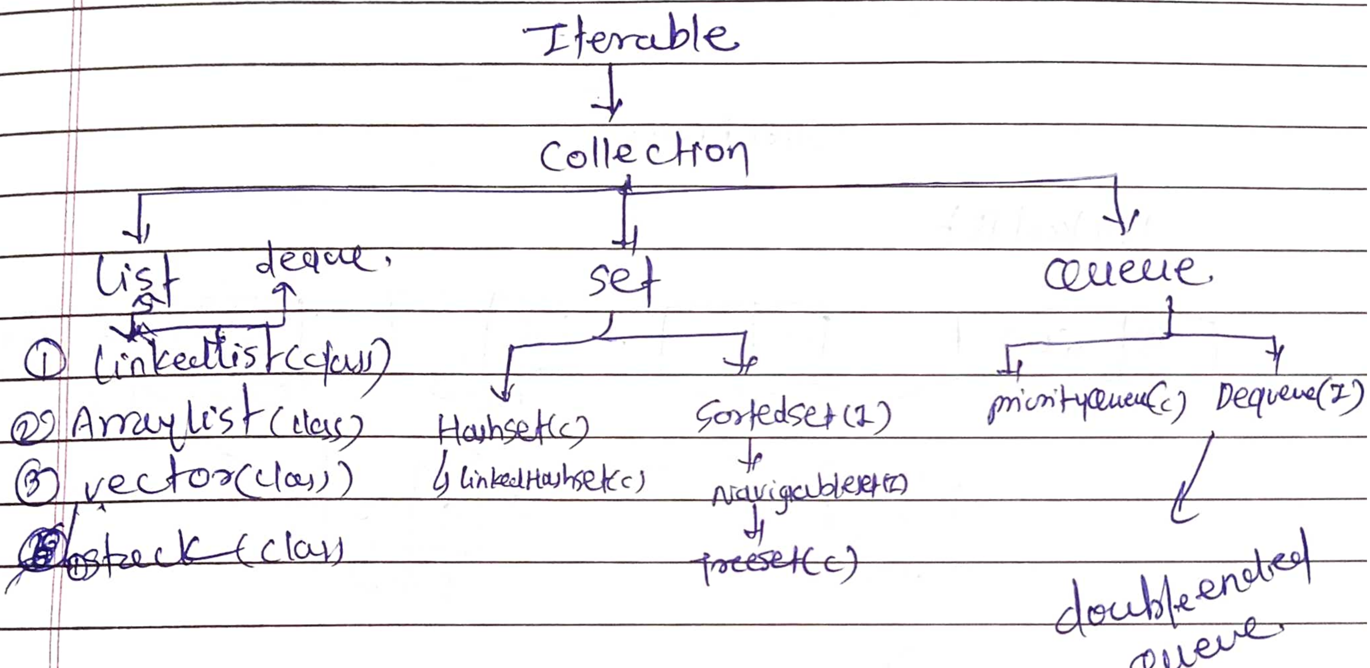
॥ श्री ॥

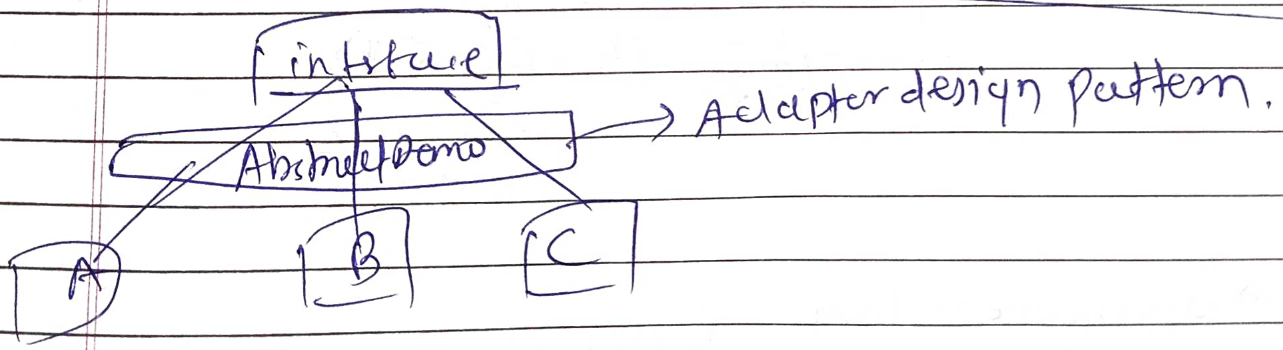
* Collection *

9-Interface

1. Collection
2. List
3. Set
4. Sorted set
5. Navigable set
6. Queue
7. Map
8. Sorted map
9. Navigable map

→ hashmap → सगळ्यात जास्त use केला जातो





जर एका class ने xyz ला implement केला तर त्याला 30 च्या 30 method ला body द्यावी लागेल म्हणून एक class आपण बनवायचा त्याला आपण adapter class म्हणतो.

f^n मध्ये call केला तरी original address च असतो.

(k, v)

map(I) → HashMap(C) → LinkedHashMap

↓↓

SortedMap

↓

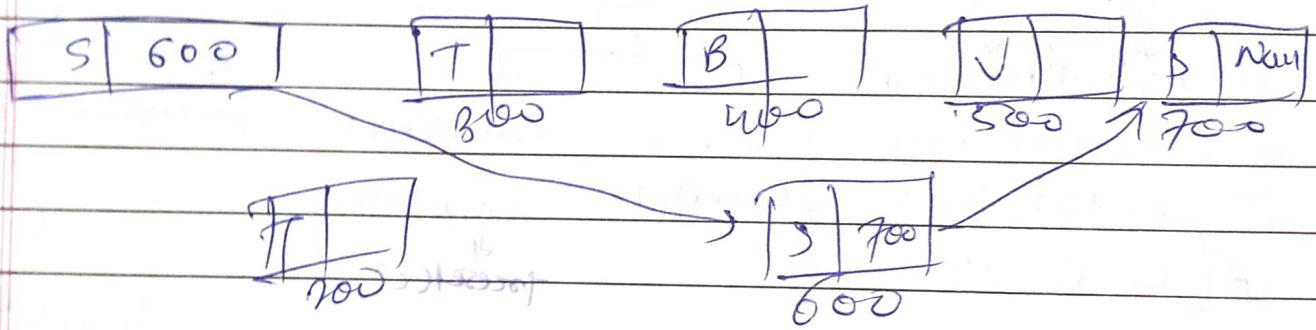
NavigableMap

↓

TreeMap(C)

Collection

linked list



⇒ 1.2 ⇒ मध्ये collection आलंय

⇒ Collection हे सगळं utility package मध्ये आहे.

⇒ ~~java.util.~~ java.util.Collection<E>

~~java.lang.~~ Iterable<T>

↳ forEach

↳ spliterator

⇒ Username ⇒ key } hashmap in instagram.

⇒ id ⇒ value. }

PAGE No.
DATE

Username ⇒ sort करायचा है ⇒ SortedHashMap

⇒ universal iterator ⇒ in Iterable

↳ एका direction मध्ये जायचा

⇒ ListIterator ⇒ पुढेपण जाता मागेपण जाता.

⇒ ↳

⇒ Enumerable ⇒

⇒ vector legacy ⇒ cpp मध्ये vector हा HEAVY class आहे.

↳ याची iterator वापरता नाही त्यापेक्षा Enumerable use होतो (Enumeration).

⇒ data ठेवण्यासाठी collection वापरतात.

⇒ iterator ही data access करण्यासाठी फिरता collection वरती.

⇒ javap java.util.List.

↳ default void replaceAll();

↳ default void sort();

↳ static <E> of(E, E...);

⇒ javap java.util.LinkedList.

↳ List, Deque.

⇒ java.util.ArrayList (unlimited size)

↳ AbstractList ⇒ Adaptor पाहिजे.

↳ RandomAccess ⇒ in only arrays of arraylist

> java.util.Vector
↳ AbstractList → adaptor
↳ RandomAccess

> javap java.util.stack
↳ vector.

> javap java.util.set
↳ collection parent.

> javap java.util.Hashset
↳ Abstractset.
↳ set parent

> javap java.util.LinkedHashset
↳ parent Hashset.

> javap java.util.Sortedset.
↳ set parent.

> javap java.util.NavigableSet
↳ parent Sortedset

> javap java.util.Treeset
↳ Navigableset → parent

> javap java.util.Queue
↳ parent Collection

> javap java.util.priorityQueue
↳ ~~parent~~ AbstractQueue.

> javap java.util.ConcurrentBlockingQueue.

util.Deque.
↳ queue.

> util.Map
↳ no parent.
↳ root interface

> util.SortedMap
↳
> util.NavigableMap
↳
> util.TreeMap.

lect 84:
seen: 22-9-23

Collection-02

javafx project → instalu करा.

(capgemini होणार कंपनी.)

annotation → @override

→ off campus होणार And आपल्याला knowledge पाहिजे.

Iterable ← revision

↓

Collection → root

- List
  - LinkedList ✓
  - ArrayList ✓
  - Vector ✓
    - Stack ✓
- Set
  - Hashset ✓
    - ↓ linkedhashset
  - sortedset
    - ↓ Navigableset
    - ↓ Treeset
- Queue
  - priority queue
  - deque
  - Blocking queue

✓ → class

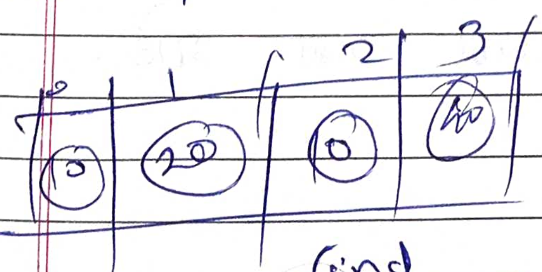
Map (I) — root

- ✓ Hashmap (c)
  - Linkedhashmap ✓
- ↓ sortedmap
  - ↓ Navigablemap ~~Treemap~~
  - ↓ (c) Treemap

1.2 आधीच्या version मध्ये ज्याच्या तीनच माहित होत्या.

① Arrays ② Vector ③ Hashtable

put(K,V)



(10) → find (at) index of value 10 → array मध्ये असल direct function नाही हा आपल्याला code लिहावा लागेल

90% लोक असे असतात त्यांना DSA येत नाही फक्त collection Basic concept माहीत असतात.

Generics => < >

linkedlist<string> => याचा अर्थ असा collection मध्ये वेगवेगळा data पण सगळ्या string च राहणार

XYZ => | Edu | love | socialmedia | --- |

=> collection framework => एक Box आहे त्यामध्ये सगळ्या गोष्टी (methods) readymade आहेत.

=> json file => K, V, मध्ये ठेवली जाते.

=> सगळं हेच collection framework वापरला जातो याच्या व्यतिरिक्त काहीच वापरला जात नाही.

=> सगळ्यांत अवघड scenario => nested keyvalue pair

<hashmap(K,V)>

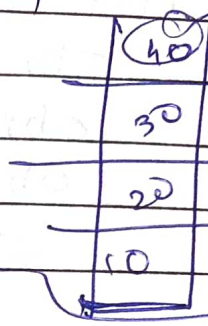
< K , V >

=> CoreWeb चा backend java मध्ये springboot मध्ये त्याच्यामध्ये बदल करून (K,V) pair मध्ये hashmap चा वापर केला आहे.

PAGE No.
DATE

10 | 20 | 30 | 40

* queue → समजात आहे आपण data पहीला बाहेर पडला पाहीजे.
* stack → समजात शेवटी आलेला data पहीला बाहेर पडला पाहीजे.



* List / LinkedList → याला आपण random access नाही करू शकत म्हणजेच index नी access करू शकत नाही.
→ duplicate data ठेवू शकतो.

* Set →
→ duplicate data is not allowed.

* map → key value

Collection <E> → Element type
↳ याचा अर्थ collection मध्ये कुठलाही interface or class वापरायचा तर Bracket मध्ये सांगायचं की तुम्ही collection मध्ये कुठल्या type चा data घेणार आहे.

Collection मध्ये आपल्याला object च दिलेले आहेत.

```
import java.util.*;
class CollectionDemo{
    public static void main(String[] args){
        List obj = new ArrayList();
        obj.add(10);
        obj.add(20);
        obj.add(10); // add("Shaikh");
        sop(obj);
    }
```

→ javac program1.java.

Note: program1.java uses unchecked or unsafe operations.

Note:- Recompile with -Xlint : unchecked for details.

→ solution

List<Integer> obj = new ArrayList<Integer>();

→ see all methods in collection & List interface compare them & check

→ add(int, E)
index, object add करेल.

→ ① springboot शिकण्यासाठी dependency → ① java
② jsp servlet
③ Networking
④ sql connectivity.

② जर React js शिकायचं तर javascript आणि पाहिजे.

→ collection म्हणजे कोणतेगळे object bind करून ठेवणे होय.

PAGE No.
DATE

* ~~Linked~~ List: Interface *

① It preserve insertion order.
② It stores dublicate data also allowed.

child

① Arraylist
② Linkedlist
③ vector
   ↳ stack

① ArrayList ⇒ javap java.util.ArrayList.
   ↳ check all methods

→ comparator

Array → ① Homogenous data structure
② Contigeous memory allocation
③ Randomely data access.
④ fixed size.
⑤ Array index start from zero (0)
⑥ सगळे element एकाच नावाने ओळखले जातात.

Arraylist → ① याची size fix नाही. हा dynamic array आहे.
② dynamic memory
③ Arraylist मध्ये कोणत्याही type चा object ठेवू शकतो.

→ पर्याय in array ⇒ object class चा array बनवायचा. पण त्यात पण drawback आहेत.

```
import java.util.*;
class ArrayListDemo{
    public static void main(String[] args){
        ArrayList al = new ArrayList();
① // add(Element)
        al.add(10);
        al.add(20.5f);
        al.add("shashi");
        al.add(10);
        al.add(20.5f);
        sop(al);
② //~~size();~~ int size()
        sop(al.size());

③   boolean isEmpty();

④   boolean contains(java.lang.Object);
        sop(al.contains("shashi")); // true

⑤   int indexOf(java.lang.Object);
        sop(al.indexOf(20.5f)); // 1
        sop(al.lastIndexOf(20.5f)); // 5

⑥   E get(int)
        sop(al.get(3)); // core2web

⑦   E set(int, E);
        sop(al.set(3, "Incubator"));
        sop(al);

⑧   void add(int, E)
        al.add(3, "core2web");
        sop(al);
```

PAGE No.
DATE

⑨ E remove(int)
sop(dl.remove(2));

⑩ boolean (object));

⑪ boolean addAll(collection);
~~Arraylist dl2 =n~~

⑫ boolean addAll(int, collection);
dl.addAll(3, dl2);

③ void removeRange(int, int);
dl.removeRange(3,5);
sop(dl);

protected
extends ArrayList

⑭ java.lang.object[] toArray();

ArrylistDemo dl= new ArrylistDemo();

Object arr[] = dl.toArray();

for(Object data : arr){
sop(data + " ");

}
Sop ln( );

Arraylist dl2=new Arraylist
dl2.add("salman")
dl2.add("shahrukh
~~dl2.add("Amir")~~
dl.addAll(dl2);
sop(dl);

⑮ clear
~~sop~~ dl.clear();
sop(dl);

kc/85
seen: 7-9-23

collection-03

⇒ source code वेळी सगळ्या classess चे
⇒ 5 few experience असलेल्यासाठी opening आहेत.

⇒ Arraylist कुठलाही data object च असतो.
- ती capacity check करता येत नाही.
- collection मध्ये कुठलाही primitive datatype नसतो.

```
import java.util.*;
class ArraylistDem{
        public static void main(String[] args){
    Arraylist al = new Arraylist();
        al.add(10);
        al.add(30);
        al.add(new Integer(40));
        al.add(30);
          // sop(al);
        for (Integer obj : al){

            System.out.println(obj);

        }
}
}
```

error: Incompatible type.

(Integer obj : al)
→ Object

for each object चा type identify करता नाही म्हणून for each मध्ये वापरत नाही

try करा error solve करा.

solution:=

```
for( var obj : al){
    sop(obj);
}
```

class

```
for(object obj : al){
    sop(obj);
}
```

error: cannot find symbol

```
for(int i=0; i<al.size(); i++){
    Sop(al.get(i));
}
```

→ return object

→ Company मध्ये code मध्ये collection वरती for loop किंवा for each loop वापरत नाही त्याऐवजी cursors चा वापर करतात.

→ जेव्हा आपल्याला सर्व objects वरती operations करायचे असतील तेव्हा cursor वापरला जातो.

```
import java.util.*;
class ITCompany{
    String compName=null;
    int empCount=0;
    ITCompany(String compName, int empCount){
        this.compName = compName;
        this.empCount = empCount;
    }
```

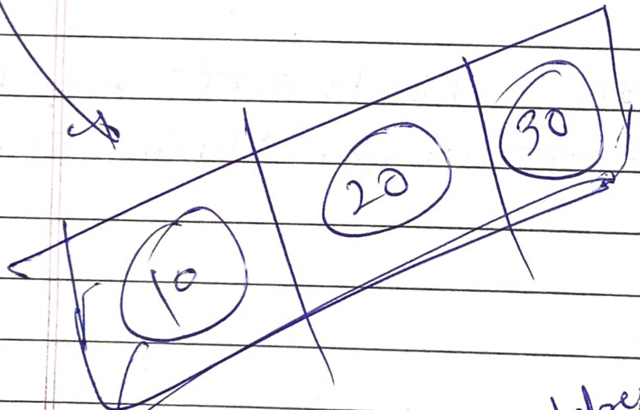
```
class ArrayListDemo{
    p s v main(String [] args){

        ArrayList obj = new ArrayList();

        obj.add(10);
        obj.add(10.5);
        obj.add("Core2Web");
        obj.add(new ITCompany("Biencaps",20));

        sop(obj);

    }
}
```

print करताना याना toString मधनय

o/p: [10, 10.5, Core2Web, ITCompany@78308db1]

address

आपल्याला इथे Biencaps print करायच होता की पण येत नाहि. त्यासाठी compar ची concept आली आहे.



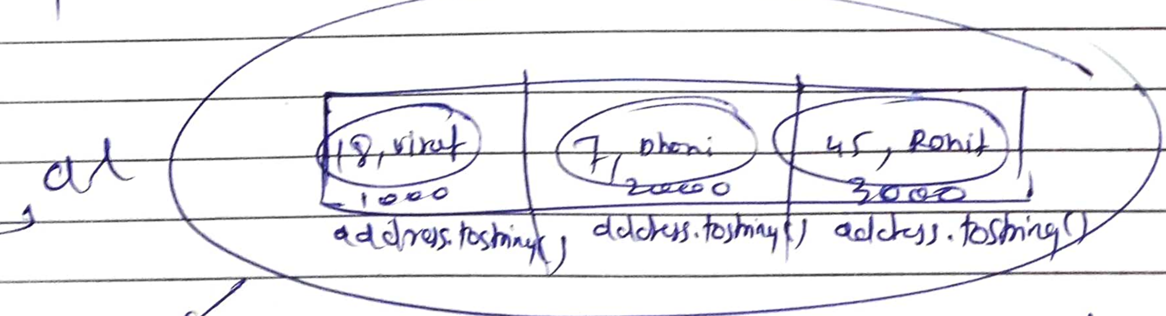
जेव्हा गेल्हा address print करायला गेला पण त्याच्याकडे address.toString() करता नंतर दिसतो त्यामुळे address नाही दिसत.

मग मी toString() override केली आहे ← comparision and difference

program3.java

```
import java.util.*;

class CricPlayer{
        int jerNo=0;
        String name=null;
        CricPlayer (int jerNo, String name){
        this. jerNo= jerNo;
        this. name= name;

        }
    public String toString(){
            return name;
    }
                //return jerNo + ":" + name;

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

ArrayList al= new ArrayList();

    al. add( new CricPlayer (18, "virat"));
    al. add (new CricPlayer ( 7, "Dhoni"));
    al. add (new CricPlayer( 45, "Rohit"));
                                    ↳ user defined data.
    SOP( al);

}
```

→ 4- cursor
1. Iterator
2. enumeration
3. ListIterator
4. splitIterator



इथपण .toString() call होते पण Object class ची आणि तिच्यामध्ये काहीच print नाही केलं.

java.lang.Object
↳ ~~ustm~~ public toString();

~~al.~~ al. add (1, "Hardik");
sop(al);

o/p: => [18 : Virat, 7 : Dhoni, 45 : Rohit]
[18 : Virat, Hardik, 7 : Dhoni, 45 : Rohit]

शक्यतो collection मध्ये Data देणार similar type चा द्या. ~~त्यां~~ कारण त्यांनी त्यासाठी generic ची concept आणली आहे.

↳ त्यांच काय म्हण आहे सुरवातीला दिसलं पाहीजे की कुठल्या type चं collection आहे.

overside
a) → al. (1, new Cricplayer(33, "Hardik"));

al.clear()
sop(al);

train ला मुख्य

LinkedList

- class
- add()
- addfirst()
- addlast()

फक्त दोनच constructor आहेत.

- No fixSize.
- index use करू शकतो in java

java मध्ये Linked List आहे ना ती आहे List interface चा child आणि List interface index ला support करतो. यामुळे indexing चालत in java

⇒ पण हे C/C++ मध्ये नाही चालत तिथे linkedlist मध्ये head पासून नाव लागत.

⇒ पण linked list ला index असता कामा नये
→ this is the basic dsa concept.

* javap java.util.LinkedList ⇒

① E getFirst();

② getLast();

E removeFirst();

E removeLast();

void addFirst(E);

void addLast(E);

LinkedList$Node<E> node(int);

```
import java.util.*;
class LinkedListDemo{
    public static void main(String[] args){
        LinkedList ll = new LinkedList();
        ll.add(20);
        ll.addFirst(10);
        ll.addLast(30);
        SOP(ll);
        ll.add(2,25);
        SOP(ll);
```

op:⇒ [10, 20, 30]

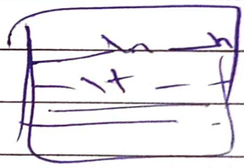
*. Cursor *

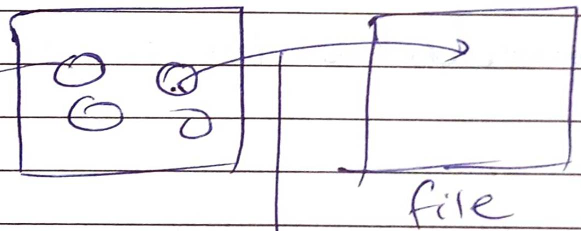
1. Iterator
2. List Iterator
   ↳ only for list (I)

→ universal
  ↳ collection
    ↳ ① set ② list ③ queue

3. Enumeration → vector → stack साठी वापरला
4. spliterator
   ↳ stream साठी वापरला
     ↳ ① serialization
       ② deserialization

in file ⇒ uniform stream (→ flow) of byte
↳ file मध्ये एकच line असते





object ला लिहून file मध्ये ठेवणे याला serialization म्हणतात.
आणि file मधून return काढणे त्याला deserialization म्हणतात.

Iterator (Universal Cursor)

↳ is Interface

boolean
public abstract hasNext();
public abstract next();
public default void remove();

Springboot मध्ये cursor ची सगळ्या लोकप्रिय आहे

⇒ javap java.util.Iterator

~~Interface~~

⇒ javap java.util.ListIterator

Iterator Parent.

ही सगळ्या List मध्ये वापरली जाऊ शकते.

public abstract boolean hasPrevious()

bidirectional चालतो.

⇒ hashmap वरती कुठलाही cursor चालत नाही.

⇒ Iterator ही data वरती sequential फिरवण्याचे काम करतो.

⇒ Iterator/ही collection मध्ये जो data आहे त्याचा access देतो individual level ला.

itr → income tax return

Iterator<E> iterator() method Arraylist मध्ये आहे.
↓ returntype
↳ सगळ्या collection मध्ये आहे.

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

    Arraylist al = new ArrayList();

        al.add("Kanha");
        al.add("Rahul");
        al.add("Ashish");

        Iterator iter = al.iterator()
        ↙ interface
        while (iter.hasNext()){
                          ↳ next data आहे का
                            ↳ boolean return type
        Sop(iter.next());
                 ↳ पुढच्या box मधला data
        }
}
```

(hasNext) पहिल्यांदा हा cursor नसता डायरेक्ट उभा असतो.

↓ हाच code for loop ला use करून करा.

⇒ iterator data add करू शकत नाही कारण त्यांच्याकडे ती मेथड च नाही म्हणून यांनी List iterator आणला.

see source code of ArrayList for iterator

PAGE No.
DATE 31 En

```
ArrayList al = new ArrayList();
    al.add("kanha");
    al.add("Rahul");
    al.add("Ashish");
Iterator itr = al.iterator();
while(itr.hasNext()){
```


```
    if("Rahul".equals(itr.next()))
        iter.remove();

    sop(itr.next());
}
SOP(al);
```

दोन वेळा next ला call आहे.

O/P :
Rahul
Exception in thread "main" java.util.NoSuchElementException

⇒ ~~next आणि~~ next आणि hasnext चा code अपूर्ण ठेवलेला आहे

lect 86
Seen:- 12-9-23

Collection - 04

Arraylist	linked list
① सुरवातीला data add करण्यासाठी सुद्धा method नाही	① सुरवातीला data add करण्यासाठी method आहे addfirst()

cpp मध्ये vector हा crucial role play करतो.

① कारण Arraylist and linkedlist स्वतः implement करावं लागतं.

② आणि vector हा STL मध्ये bydefault आहे.

↳ vector चा size fix नसतो.

→ जे vector आणि stack वापरण्यास सांगितले आहे ते सुद्धा Arraylist and linked list वापरून करू शकतो.

① पण ते सुद्धा dynamic array पद्धतीने असल्याने vector वापरावा.

② stack → आपला data एकावर एक ठेवत आणि शेवटी आलेला पहिल्यांदा delete करायचा असल्याने stack वापरावा आहे.

PAGE No.
DATE

> javap java.util.Vector => for see methods

vector(int, int);
↳ fill ratio सांगते

Enumeration <E> element();
↳ cursor type.

Synchronized methods in the vector
↳ ह्या method वरती multiple thread येऊन लागेल तरच synchronized. एकच thread काम करतो.

Dangling pointer
↳ दुसऱ्याच्या pointer ने data पकडून ठेवला असतो.

आपण आपली method synchronized करू शकतो.

```
import java.util.*;
class VectorDemo {
    p s v main (String[] args) {

    Vector v = new Vector();
      v.addElement(10);
      v.addElement(20);
      v.addElement(30);
      v.addElement(40);
      sop(v);
}
```

op: => [10, 20, 40].

sop(v.capacity());
↳ current capacity x 2 in vector =>

v.removeElement(30);

Try 2 also

```
Arraylist al = new Arraylist();
    al.add("Shesh");
    al.add("Ashish");
    al.add("kanha");

Vector v = new Vector(al);
        v.addElement(10)
                    (20)
                    (30)
        v.addElement(40);
        sop(v);
    }
```

Stack

→ javap java.util.stack.

stack();
one constructor

```
import java.util.*;
class Stackprog
    psvm(String[] args){
    Stack void main(String[], args){
    stack s= new stack();
    s.push(10)
    s.push(20)
    s.push(30)
    sop(s)
```

→ [10, 20, 30, 40]

→ insertion order preserve
FIFO

removed & return करना

pop-का element निकालेगा
last element pop होगा

```
s.pop();
sop(s);
    o/p:- [10, 20, 30]
```

```
sop(s.peek());
sop(s)
```
return करना value

```
sop(s.search(30));
    o/p: 2
```

3	40
2	30
1	20
0	10

⇒ multiple thread वापर आणि vector try करे.

Cursor :-

1. iterator
2. ListIterator
3. Enumeration.

```
Arraylist al = new Arraylist
    al.add(10);
    al.add(20.5);
    al.add('2ω');
    for (var x : al){
        sop(x.getClass().getName());
    }
}
```
method in object class

⇒ class नावाच्या एक class आहे | In python metaclass

> javap java.lang.Class
most important

→ या class कडे बघून java मधील सगळे class बनवले जातात

> आपण class हा या class च्या type चा असतो.

> book ⇒ java ⇒ class नावाचा topic आहे त्याच्या मध्ये.

```
ArrayList al = new ArrayList();
al.add(10);
al.add(20.5);
al.add("ZW");
for(var x : al){
    sop(x.getClass().getName());
}
```

// Iterator

```
Iterator cursor = al.iterator();
sop(cursor.getClass().getName());
```

O/P: java.util.ArrayList$Itr

```
sop(cursor.next());
sop(cursor.next());
sop(cursor.next());
sop(cursor.next());
sop(cursor.next());
```

NoSuchElement Exception.

Iterator ET interface आहे next या method मध्ये body नाही आणि ET cursor आता ArrayList$Itr type चा आहे. या next method ही ArrayList मध्ये body दिली आहे तो call होतो.

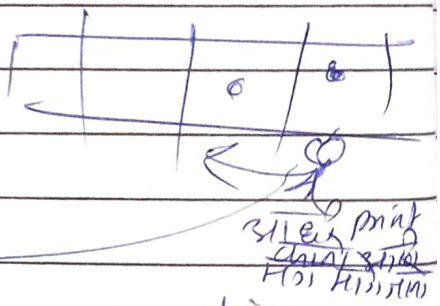
al.add("Ashish");
al.add("Kanha");
al.add("...");

PAGE No.
DATE

```
while (cursor.hasNext()){
    if ("kanha".equals(cursor.next()))
        cursor.remove();
}
sop(al);
}
```

// ListIterator

```
ListIterator litr = al.listIterator();
  ||
//sop(litr.getClass().getName());

while (litr.hasNext()){
    sop(litr.next());
}
While (litr.hasPrevious()){
    sop(litr.previous());
    litr.next() ; exception.
}
```



→ तोहा हा पहील्या घटक येतो मागे मागे

येत येत तेहा तो hasPrevious check करतो आणि मग त्या false होते आणि तरी पण तो data print करतो this is twist
→ Exam check करा previous method जेमकी काय करते.

// Enumeration → legacy

```
Vector dd = new Vector();
Enumeration cursor
        dd.add("Ashish");
        dd.add("Kanha");
        dd.add("Badhe");

Enumeration cursor = dd.elements();
```

Anonymous inner class

```
Sop(cursor.getClass().getName());
```

O/P:- java.util.Vector$1

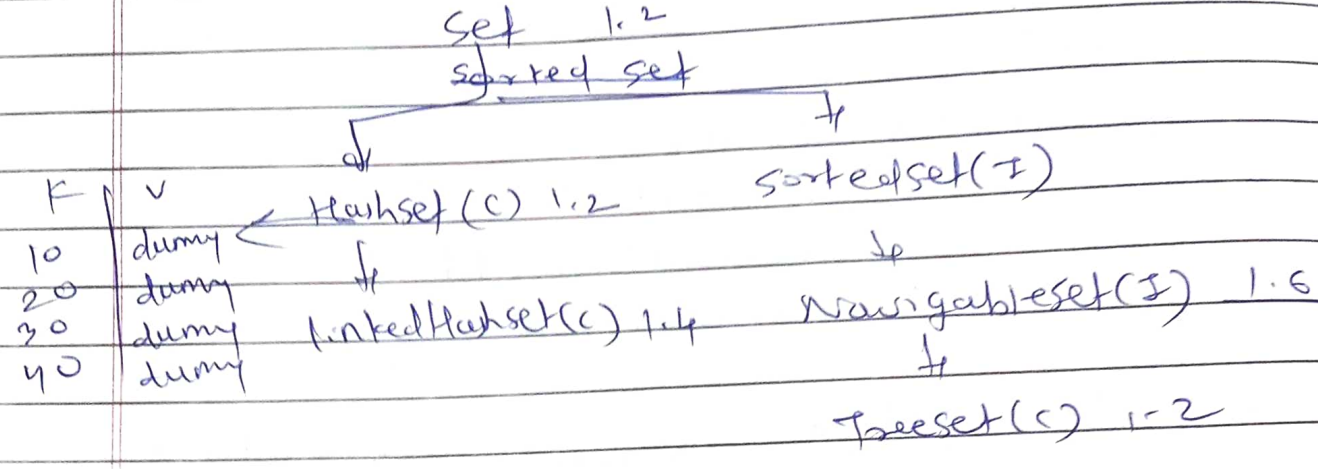
> javap java.util.Enumeration;

```
While(cursor.hasMoreElements()){
        Sop(cursor.nextElement());
}
}
}
```

Lect 87:

16.9.23

collection-05



⇒ set duplicate data chalat nhi.

⇒ Hashset internally Hashmap hota.

① Hashset ⇒ javap java.util.Hashset.

```
import java.util.*;
class HashsetDemo{
    public static void main(String[] args){
        Hashset hs = new Hashset();
        hs.add(10);
        hs.add(20);
        hs.add(10);
        hs.add(10);
        sop(hs);
        hs.add("Katrina");
        hs.add("Rahul");
        hs.add("Ashish");
        hs.add("Badhe");
        hs.add("Rahul");
        hs.add("Ashish");
```

see source code of Hashset.

(constructor of Hashset)

7 java.util.Hashtable

O/P: [Rahul, Ashish, Badhe, kahha]

Hashing function use केलेलं ET असतं sequence print होतो.

Sop(hs.add("Ashish"));

O/P: true or false

LinkedHashset hs = new LinkedHashset();

hs.add("kanha");
hs.add("Rahul");
hs.add("Ashish");
hs.add("Badhe");
hs.add("Rahul");
hs.add("Ashish");

Sop(hs);

?

O/P: [kanha, Rahul, Ashish, Badhe];

Source code of LinkedHashset

```
class CricPlayer{
    int jerNO = 0;
    String pName = null;
    CricPlayer(int jerNO, String pName){
        this.jerNO = jerNO;
        this.pName = pName;
    }
}

class HashsetDemo{
    public static void main(String[] args){
        LinkedHashset hs = new LinkedHashset();

        hs.add(new CricPlayer(18, "virat"));
        hs.add(new CricPlayer(7, "msDhoni"));
        hs.add(new CricPlayer(45, "Rohit"));
        hs.add(new CricPlayer(7, "msDhoni"));
        Sop(hs);
    }
}
```

O/P: 4 addresses.

⇒ User defined classes मध्ये असतील तर त्यामध्ये content जरी same असतील तरीपण त्या-त्या object चे गोळे वेगवेगळे असतात. (Unique)

```
import java.util.*;
class HashDemo{

    public static void main(String[] args){

        Hashset hs= new Hashset();
        hs.add(10);
        hs.add(20);
        hs.add(new Integer(10));
        hs.add(new Integer(20));
        Sop(hs);

    } }
```

O/p:⇒ [20, 10]

> javap util.sortedset.
  ↳ हा interface.

याच्या method वापरायच्या असतील तर आपल्याला याच्या child class पर्यंत जावं लागेल म्हणजेच treeset पर्यंत.

① subset(E,E) ② first(), ③ last();

> hashset मध्ये index नाही.

> javap java.util.Navigableset.

lower(E)

floor(E)

ceiling(E);

String class हा comparable आहे
आणि StringBuffer हा comparable नाही.

PAGE No. DATE

```
>javap java.util.TreeSet

        TreeSet();
        TreeSet(Comparator)
        TreeSet(collection);

import java.util.*;

class TreeSet Demo{
    p s v main(String[] args){
        TreeSet ts = new TreeSet();

        ts.add(30);          |  ts.add("Kanha");
        ts.add(10);          |  ts.add("Ashish");
        ts.add(40);          |  ts.add("Rahul");
        ts.add(20);          |  ts.add("Badhe");
        sop(ts);             |  sop(ts);
    }
}
                                o/p: [Ashish, Badhe, Kanha, Rahul]
```

TreeSet हा sortedset चा child आहे. त्यामुळे त्याला already sort करता येत.

```
ts.add(new StringBuffer("Kanha"));
  --"--("Ashish"));
  --"--("Rahul"));
```

१.५ versions पर्यंत StringBuffer comparable नव्हता.

1.6 पासून इथून

```
> javap java.lang.String -> comparable
> --"-- java.lang.Integer.
> jav--"-- java.lang.StringBuffer.
  --"--  --"-- StringBuilder.
```

javap -version version change करने के लिए

> sudo update-alternatives --config javap.

> javap java.lang.Comparable.

① compareTo(T);

class MyClass { → Comparible interface implement करेगा.

String str = null;

MyClass(String str){

this.str = str;

}

}

class TreeSetDemo{

P S v main (String[] args){

TreeSet ts = new TreeSet();

ts.add(new MyClass("Kanha"));

("Ashish"));

("Rahul"));

("Sadhe"));

sop(ts);

error :- class cast Exception.

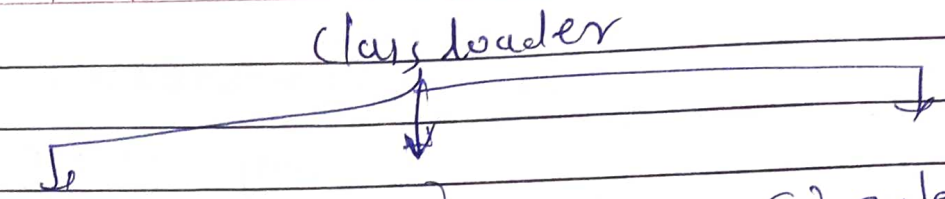
or

ts.add(new String("Kanha"));

("Ashish"));

"Rahul"

"Sadhe"




```
import java.util.*;

class MyClass implements Comparable{

    String str = null;
    MyClass (String str){
        this.str = str;

    }

    public int compareTo(MyClass obj){
                        ~~proxy~~ error
                        object

    // return 1;
    // return 0;
    // return -1; // (obj.str).compareTo(this.str);

}
public String toString(){

    ~~return t;~~
    return str;

}
```

```
class TreeSetDemo{
    p s v main(String [] args){
    TreeSet ts = new TreeSet();
    ts.add(new Myclass("Kanha"));
    ts.add(new myclass("Ashish"));
    ts.add(new myclass("Rahul"));
    ts.add(new myclass("Badhe"));
    Sop(ts);
}
}
```

'ect 88
seen 20-9-23

Collection - 06

काही 1.8 मध्ये StringBuffer ने Comparable implements नाहि केला कारण त्यांच idk काय आहे

```
import java.util.*;
class TreeSetDemo{
        public static void main(String[] args){

        TreeSet t = new TreeSet()
          t.add(new StringBuffer("Kanha"));
          t.add(new StringBuffer("Badhe"));
          t.add(new StringBuffer("Ashish"));
          t.add(new StringBuffer("Badke"));
          t.add(new StringBuffer("Rahul"));
```

```
import java.util.*;
class Movies{
        String movieName = null;
         Float totalColl = 0.0f;
      Movies(String movieName, Float totalColl){
```

y. compareTo(X)

- Treeset मध्ये तोच data देऊ शकतो जो comparable type चा असेल


```
import java.util.*;
class Movies implements Comparable{
    String movieName = null;
    float totColl = 0.0f;

    movies(String movieName = movieName;
    Movies(String movieName, float totColl){

        this.movieName = movieName;
        this.totColl = totColl;

    }

public int compareTo(object obj){

    // return -(movieName.compareTo((Movies)obj).movieName));
    // return movieName.compareTo((movies)obj).movieName);
}
public String toString(){

        return movieName;

    }
}

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

    TreeSet ts = new TreeSet();
    (X) ts.add(new Movies("Gadar2", 150.00f));
    (Y) ts.add(new ---- ("OMG2", 120.00f));
    ---- ("Jailer", 250.00f));
    ---- ("OMG2", 120.00f));

    Sop(ts);
```

write same code of one real time example

```
import java.util.*;
class SortDemo{
    public static void main(String[] args){
        ArrayList al = new ArrayList();

        al.add("Kanha");
        al.add("Ashish");
        al.add("Badhe");
        al.add("Shaghi");
        al.add("Pathur");
        Sop(al);

        // TreeSet ts = new TreeSet(al);
        // Sop(ts);

    }
}
```

sorting

Collections → class → its parent is object.

static sort method

sort(list)

```
Collections.sort(al);
Sop(al);
```

→ sort करने से duplicate data नहीं होता

using this increase performance.

> javap java.util.Comparator

↳ compare(T, T) → एकाच method ला body द्यायची

equals(Object);

↳ equals ही object class ची method आहे म्हणून तिला body द्यायची नाही.

<> Generic

```
import java.util.*;
class Employee{
        String empName = null;
        float sal = 0.0f;
    Employee(String empName, float sal){
        this.empName = empName;
        this.sal = sal;
    }

    public String toString(){
        return "{" + empName + ":" + sal + "}";

    }
}
class SortByName implements Comparator<Employee>{

    public int compare(Employee obj1, Employee obj2){

        return obj1.empName.compareTo(obj2.empName);
    }
}

class SortBySal implements Comparator<Employee>{
    public int compare(Employee obj1, Employee obj2){

        return (int)(obj1.sal - obj2.sal);
}  }
```

PAGE No.
DATE

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

    Arraylist <Employee> al = new Arraylist<Employee>();

    al.add( new Employee("kanha", 200000.00f));
    al.add(new Employee("Ashish", 250000.00f));
    al.add(new Employee("Badhe", 150000.00f));
    al.add(new Employee("Rahul", 175000.00f));

        Sop(al);
    Collections.sort(al, new SortByName());

    Sop(al);

    Collections.sort(al, new SortBySal());
        Sop(al);
} }
```

```
import java.util.*;
class Movies{
        String movieName = null;
        double totColl = 0.0;
        float imdbRating = 0.0f;

Movies(String movieName, double totColl, float imdbRating){
        this.movieName = movieName;
        this.totColl = totColl;
        this.imdbRating = imdbRating;

}
```

```
public String toString(){
    return "{" + movieName + "," + totColl + "," +
    imdbRating + "}";
}
class SortByName implements Comparator{
    public int compare(Object obj1, Object obj2){
        return((Movie)obj1).movieName.compareTo((Movie)obj2.
                movieName);
    }
}
class SortByColl implements Comparator{
    Public int Compare(Object obj1, Object obj2)
    {
        return(int)(((Movie)obj1).totColl) -
                ((Movie)obj2).totColl));
    }
}
class SortByIMDB implements Comparator{
    public int compare(Object obj1, Object obj2){
        return(int)(((Movie) obj1).imdbRating) -
                ((Movie)obj2).imdbRating));
    }
}
```

for descending
-(int x(m...)

```
class userlistsort{

    public static void main (String [] args){

    Arraylist al = new Arraylist();
    al.add(new Movies("RHTDM", 200.00, 8.8f));
    al.add(new movies("ved", 77.00, 7.5f));
    al.add(new movies("Sairat", 100.00, 8.9f));
    al.add(new movies("Bajirao", 500.00, 9.9f));

      Sop(al);

      Collections.sort(al, new SortByName());
      Sop(al);

      Collections.sort(al, new SortBy(011());
      Sop(al);

      collection.Sort(al, new SortByIMDB());
      Sop(al);

}
```

* methods of Sorted Set :

java.util.SortedSet.

PAGE No.
DATE

```
import java.util.*;
class SortedSetDemo{
    public static void main(String[] args){

    SortedSet ss = new TreeSet();
     ss.add("Kanha");
     ss.add("Rajesh");
     ss.add("Rahul");
     ss.add("Ashish");
     ss.add("Budhe");

     sop(ss);

    sop(ss.headSet("Kanha"));
    sop(ss.tailSet("Kanha"));
    sop(ss.subSet("Ashish","Rahul"));
    sop(ss.first());
    sop(ss.last());

}
```

methods
1. Subset(E,E);
2. headSet(E);
3. tailSet(E);
4. First();
5. last();

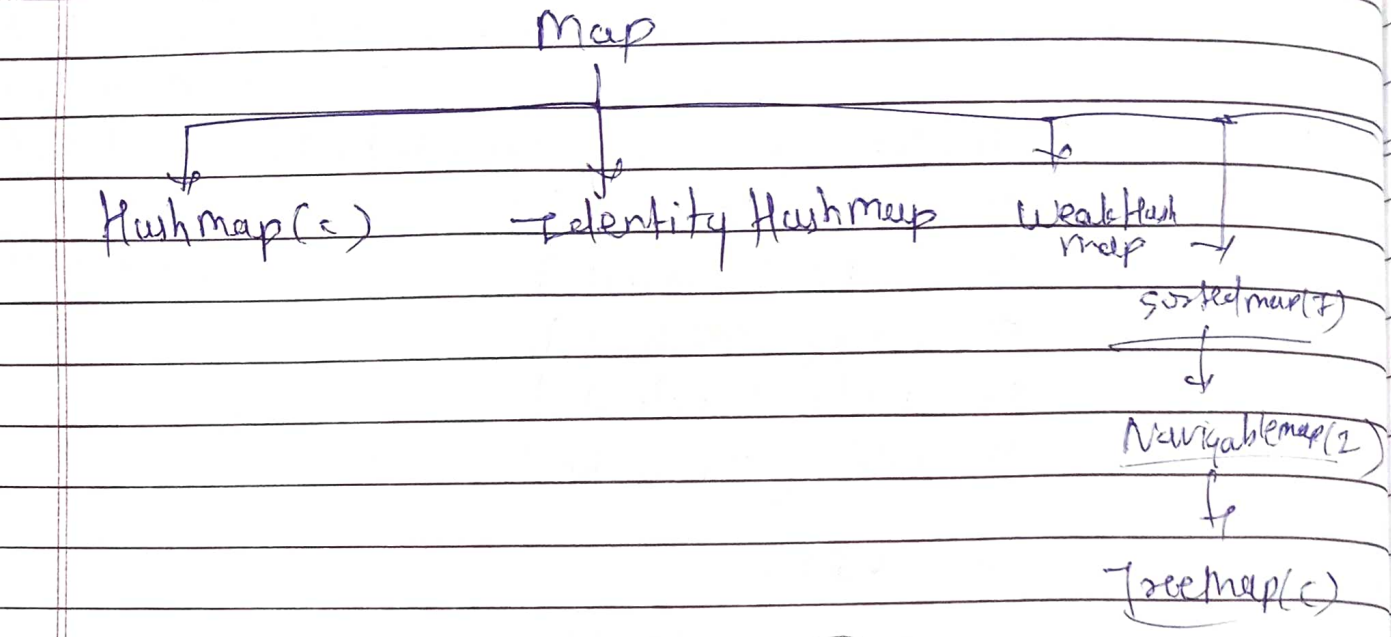
try same to as Navigableset.

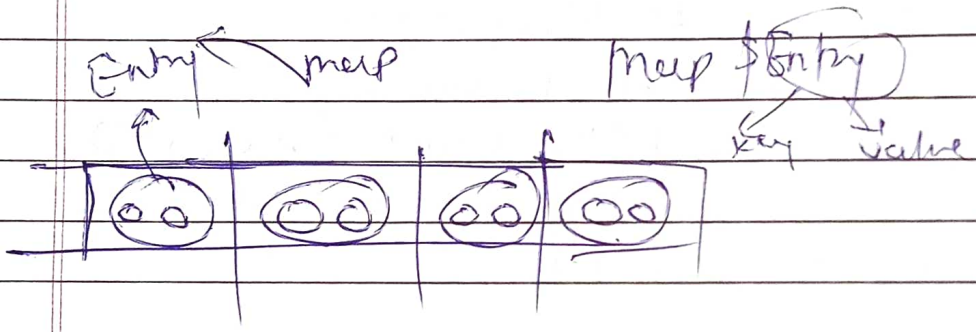
lect 59

Date: 27-9-23

PAGE No.
DATE

Collection - 7





① import java.util.*;
class HashMapDemo{
public static void main(String[] args){
HashMap hm = new HashMap();
hm.put("Kanha", "Infosys");
hm.put("Ashish", "Barclays");
hm.put("Badhe", "Capro");
hm.put("Rahul", "BMC");
SOP(hm);
}
}

HashMap hm = new HashMap();
hm.add("Kanha")
hm.add("Ashish")
hm.add("Badhe")
hm.add("Rahul")
SOP(hm);

Op: {Rahul, Ashish, Badhe, Kanha}

Op: {Rahul=BMC, Ashish=Barclays, Badhe=Capro, Kanha=Infosys}

80894 Str1.hashcode()
81855 Str2.hashcode()
74167 Str3.hashcode()

PAGE No.
DATE

Dictionary(c)
↑
Hashtable (1)
↓
properties(c)

```
Hashmap hm = new Hashmap();
hm.put("Kanha", "Bmc");
hm.put("Achish", "Barclays");
hm.put("Kanha", "Capgo");
hm.put("Rahul", "Bmc");
sop(hm);
}}
```

o/p :⇒ { Rahul = Bmc, Achish = Barclays, ~~Kanha = Capgo~~, Kanha = ~~Bmc~~ }

Hashcode

RAM
2 1 0

```
int hashcode = 0;
hashcode = s[0] * 31^(n-1) + s[1] * 31^1 + s[0]
```

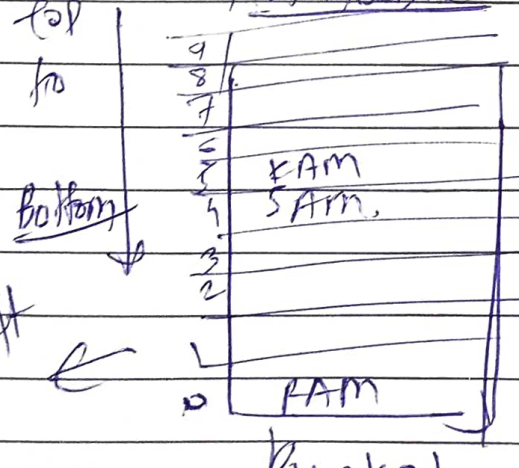
3 2
2

$= 82 * 31 * 31 + 65 * 31 + 77$

⇒ 80894

Hashset ⇒ Hashmap

↓
Hashtable



o/p. Top to Bottom.
& Right to left.
KAM, SAM, RAM

RAM
↓
80894 % 11 ⇒ बाकी 0

SAM
81855 % 11 ⇒ ~~बाकी~~ 1

KAM
74167 % 11 ⇒ बाकी 5

Source code of HashMap

ctrl+f

PAGE No.
DATE

Source code of String hashCode

```
final int hash(Object key)
{
return key == null ? 0 : math.abs(key.hashCode()
                                  % buckets.length);
}
```

classpath.org

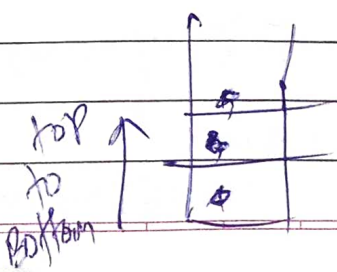
Source code of Integer.

2	101, 112	Top to Bottom like String	hm.put(101, "Info")
1	133		hm.put(150, "Barclay")
0	150		hm.put(133, "Campro")
			hm.put(112, "Bmc")

o/p => {112=Bmc, 101=Infosys, 133=Campro, 150=Barclays}

```
hm.put(5, "Info");
hm.put(3, "Barclay")
(5, "    ")
(7, "    ")
```

o/p =>



java.util.map — javap java.util.map

LinkedHashMap hm = new LinkedHashMap(); → for insertion order.

map वरती cursor ठेवता नाही. पण आपण त्याला convert करून दोन्ही set मध्ये using
entryset()
keyset()

① get();
② remove(object key);
③ set<K> keyset();
④ set<java.util.map$Entry<K,V>> entryset();

```
class HashmapMethods {
    p s v main(String[] args){
        Hashmap hm = new Hashmap();
        hm.put("java", ".java");
        hm.put("Python", ".py");
        sop(hm);

        // get
        sop(hm.get("python")); .py

        // keySet
        sop( hm.keyset()); // [java, Python, Dart]
        // values.
        sop(hm.values()); // [.java, .py, .dart]
        // entryset()
        sop(hm.entryset());
        // [java=.java, Python=.py
        Dart=.dart
```

पु. भी. देशपांडे ① वाऱ्याची चाळ
② व्यक्ती आणि वल्ली .

PAGE No.
DATE

identity Hashmap

```
Hashmap hm = new Hashmap();
  hm.put(new Integer(10), "kanha");
  hm.put( new Integer(10), "Rahul");
   hm.put( new Integer(10), Badhe");
    sop(hm);
   }
}
```

o/p :- 10= Badhe

```
import java.util.*;
class IdentityHashmapDemo{
    p s v main(String[] args){

  IdentityHashmap hm = new IdentityHashmap();

  hm.put( new Integer(10), "kanha");
   hm.put( new Integer(10), "Rahul");
    hm.put( new Integer(10), "Badhe");
     sop( hm);
  }
}
```

try use only 10 10 10 instead of new Integer (10)

Object
① Hash code()
② finalize()

In CPP Destructor
हा memory free करण्यासाठी नाहि वापरला तर
destructor notification करण्यासाठी
तेव्हा object delete होता तेव्हा.

Hashmap GC ला वाव देत नाहि.

→ Hashmap हा GC ला Dominate करतो.
↳ म्हणून त्यासाठी weakHashmap वापरतो
नाही.

```
class Demo{
    String str;
    Demo(String str){
        this.str = str;
    }
    public String toString(){
        return str;
    }
}                          public void finalize()
                           {
                               sop("notify")
                           }

class GCDemo{
    p s v main(String[] args){
        Demo obj = new Demo("Core2Web");
        sop(obj);

        obj = null;  System.gc();
        sop("In main");   ↳ call to finalize method.
```

10/12 tuple ला आहेत GC

```
Demo obj1 = new Demo("Core2Web");
Demo obj2 = new Demo("Biencaps");
Demo obj3 = new Demo("Incubator");

obj1 = null
obj2 = null
System.gc();
```

obj1 → hm.put(obj1.toString())

code same as above.

```
class GCDemo{
    p s v main(String[] args){
    Demo obj1 = new Demo("core2web");
    Demo obj2 = new Demo("Biencaps");
    Demo obj3 = new Demo("Incubator");

    HashMap hm = new HashMap();
    hm.put(obj1, 2016);
    hm.put(obj2, 2019);
    hm.put(obj3, 2023);

    obj1 = null;

    System.gc();
    sop(hm);
```

HashMap remove GC नाही करणार?

O/P :⇒ {core2web=2016, Biencaps = 2019, Incubator = 2023}

~~class GCDemo{ p s v main~~

Weak Hashmap.

```
class GCDemo{
    p s v main(String[] args){
    Demo obj1 = new Demo("core2web");
    ——— 2 = ——— ("Biencaps");
    ——— 3 = ——— ("Incubator");

    WeakHashMap hm = new WeakHashMap();
    hm.put(obj1, 2016);
    hm.put(obj2, 2019);
    hm.put(obj3, 2023);
```

```
abi = null;
System.gc();
Sop(hm);
}
}
```

Lec 90
seen 23-9-23

PAGE No.
DATE

Collection - 08

SortedMap
↓
NavigableMap
↓
TreeMap

① TreeMap

```
import java.util.*;
class TreeMapDemo{
    public static void main(String[] args){

        TreeMap tm = new TreeMap();
        tm.put("Ind", "India");
        tm.put("Pak", "Pakistan");
        tm.put("SL", "SriLanka");
        tm.put("Aus", "Australia");
        tm.put("Ban", "Bangladesh");

        SOP(tm);
    }
}
```

O/P: {Aus=Australia, Ban=Bangladesh, Ind=India, Pak=Pakistan, SL=SriLanka}

```
import java.util.*;
class Platform implements Comparable{
    String str = null;
    int foundYear = 0;
```

Hashmap द्वारा आणि comparator वापरून value sort करायचे assignment

```
platform (String str, int foundYear){
        this.str = str;
        this.foundYear = foundYear;
}
public String toString(){
        return "{" + str + ":" + foundYear + "}";
}
Public int compareTo(Object obj){
        return this.foundYear - ((Platform)obj).
                                foundYear;
}
}
class TreeMapDemo{
    public static void main(String[] args){
    TreeMap tm = new TreeMap();
    tm.put(new platform("Youtube", 2005), "Google");
    tm.put(new Platform("Instagram", 2010), "meta");
    tm.put(new Platform("facebook", 2004), "meta");
    tm.put(new platform("chatGPT", 2022), "OpenAI");
        Sop(tm);
}
}
```

```
import java.util.*;
class Platform{
        String str = null;
        int foundYear = 0;
        Platform(String str, int foundYear){
            this.str = str;
            this.foundYear = foundYear;
        }
       public String toString(){
        return "{" + str + ":" + foundYear + "}";
        }
    }
class SortByName implements Comparator{
        public int compare(Object obj1, Object obj2){

    return ((Platform)obj1).str.compareTo(((Platform)obj2).str);
    }
}
class TreeMapDemo {
    public static void main(String[] args){
     TreeMap tm = new TreeMap(new SortByName());
     tm.put(new Platform("Youtube", 2005), "Google");
     tm.put(new Platform("Instagram", 2010), "Meta");
     tm.put(new Platform("facebook", 2004), "Meta");
     tm.put(new Platform("ChatGPT", 2022), "OpenAI");
      SOP(tm);
```

*. Sorted map * java.util.SortedMap

methods

1. submap(K, K);
2. headMap(K);
3. tailMap(K);
4. firstKey();
5. lastKey();
6. keySet();
7. values();
8. Map.Entry<K,V> entrySet();

```
import java.util.*;
class TreeMapDemo {
    public static void main(String[] args){

SortedMap tm = new TreeMap();
        tm.put("Ind", "India");
        tm.put("Pak", "Pakistan");
        tm.put("SL", "sriLanka");
        tm.put("Aus", "Australia");
        tm.put("Ban", "Bangladesh");
        sop(tm);

    //SubMap
        sop(tm.subMap("Aus", "Pak"));

    //headMap
        sop(tm.headMap("Pak"));
    // tailMap
        sop(tm.tailMap("Pak"));
```

```
// firstkey
    sop(tm.firstKey());
// lastkey
    sop(tm.lastKey());

// keyset
    sop(tm.keySet());

// values
    sop(tm.values());

// entrySet(
    sop(tm.entrySet());
```

↳ cursor वापरता use करायचा map वरती ?
Iterating over map.

```
Set <map.Entry> data = tm.entrySet();
// sop(data);

Iterator <map.Entry> itr = data.iterator();

While (itr.hasNext()){
    // sop(itr.next());
    ~~itr~~.
    map.Entry abc = itr.next();

    Sop(abc.getKey() + ":" + abc.getValue());
}
```

Dictionary

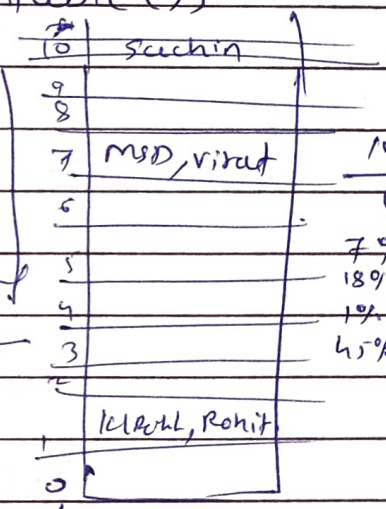
internet पर data json format में होता है।

Hashtable

- Synchronized
  - at a time one thread काम करेगा।
- Same as vector
- legacy class

> javap util.Hashtable.
> javap java.util.Dictionary.

```
import java.util.*;
class HashtableDemo {
    public static void main(String [] args) {
        Hashtable ht = new Hashtable();
        ht.put(10, "sachin");
        ht.put(7, "MSD");
        ht.put(18, "virat");
        ht.put(1, "KLRahul");
        ht.put(45, "Rohit");
        sop(ht);
    }
}
```



o/p:

{10=sachin, 18=virat, 7=MSD, 45=Rohit, 1=KLRahul}

javap java.util.Dictionary

1. keys()
2. elements();
3. get( );
4. put( );
5. remove(Object);

```
import java.util.*;
class HashtableDemo{
    public static void main(String[] args){

    Dictionary ht = new Hashtable();
     ht.put(10, "Sachin");
     ht.put(7, "MSD");
     ht.put(18, "Virat");
     ht.put(1, "KLRahul");
     ht.put(45, "Rohit");
      sop(ht);
      // Keys
      Enumeration itr1 = ht.keys();

     while(itr1.hasMoreElements()){

          sop(itr1.nextElement());

     }

    // Elements
      Enumeration iter2 = ht.elements();
    while(itr2.hasMoreElements()){
          sop(itr2.nextElement());

    }
    sop(ht.get(10));
     ht.remove(1);
      sop(ht);

}   }
```

PAGE No.
DATE

* Properties *

javap java.util.properties.

① load( ) IOException

② store( ) IOException.

```
import java.io.*;
class PropertiesDemo{
    public static void main(String[] aru)
                throws IOException {
    properties obj = new Properties();

    FileInputStream fobj = new FileInputStream
                ("friends.properties");
```

vim friends.properties

```
    obj.load(fobj);
    Sting name = obj.getProperty("Ashish");
    sop(name);
    obj.setProperty("shudhi","Biencaps");

    FileOutputstream outObj = new fileoutput
                stream("friends.properties");

    obj.store(outObj,"Updated by shudhi");

}   }
```

util.NavigableSet

1. lower(E)
2. floor(E);
3. ceiling(E)
4. higher(E)
5. pollFirst()
6. pollLast()
7. iterator()
8. descendingSet()
9. decendingIterator();
10. SubSet(E, boolean, E, boolean);
11. headSet(E, boolean);
12. tailSet(E, boolean);
13. subset(E, E);
14. headSet(E);
15. tailSet(E);

```
// Navigableset
import java.util.*;
class NavigableSetDemo{
    p s v main(String[] args){

NavigableSet ns = new TreeSet();
  ns.add(10);
  ns.add(30);
  ns.add(14);
  ns.add(27);
  ns.add(23);
  sop(ns);
  // lower();
sop(ns.lower(23));   (<)
  //        lower(25)
            lower(9)
//floor()
 sop(ns.floor(23));   (≤)

// ceiling()
   sop(ns.ceiling(23));   (≥)
          (25)

// higher()
  sop(ns.higher(27));

// pollFirst()
   sop((ns.pollFirst());

// pollLast()
   sop(ns.pollLast());
   sop(ns);
```

// descendingSet()

```
Sop( ns.descendingSet());
```

// Iterator

```
Iterator itr = ns.iterator();
while( itr.hasNext()){
    Sop( itr.next());
}
```

// descending Iterator

```
Iterator itr1 = ns.descendingIterator();
while (itr1.hasNext()){
    Sop(itr1.next());
}
```

// subset()

```
Sop( ns.subset(10, true, 27, false)
```

Try out home → NavigableMap

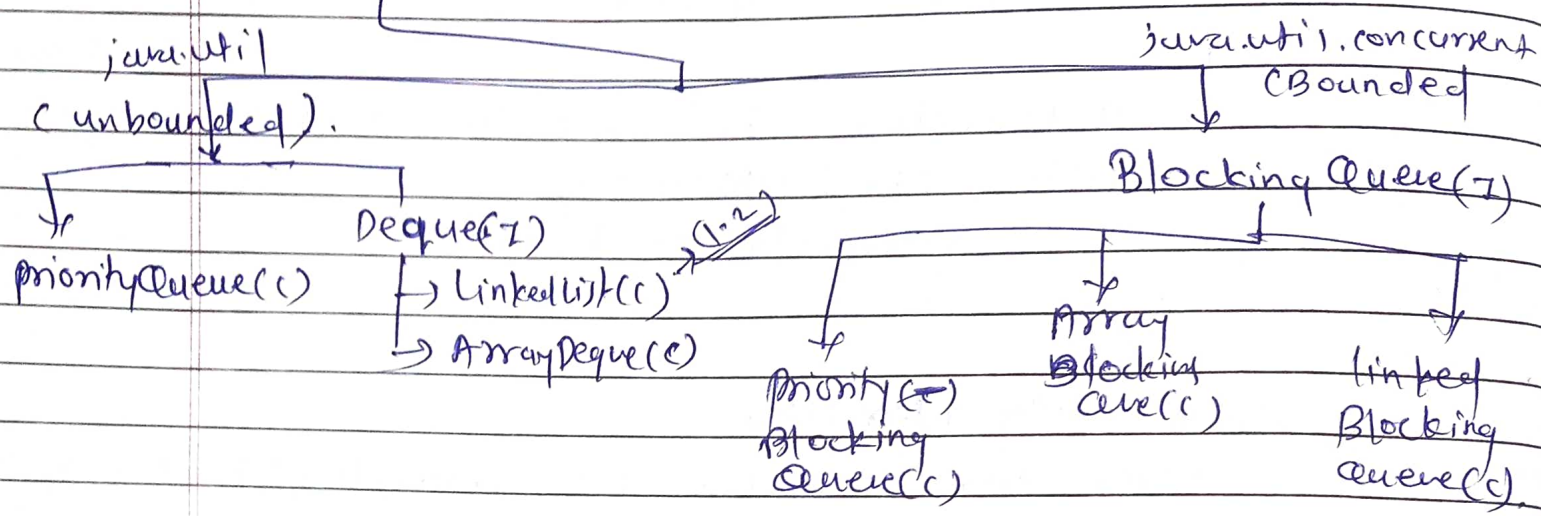
lect 91

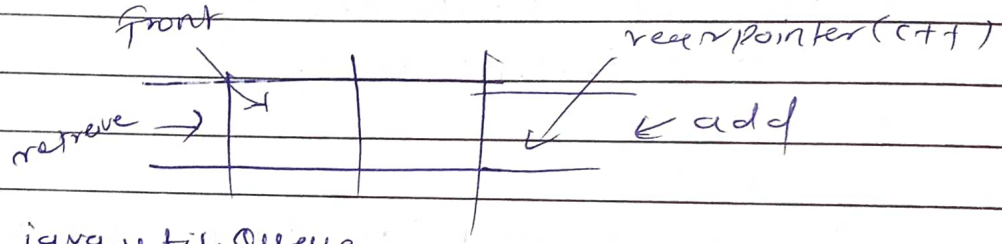
seen: 23-4-23

PAGE No.
DATE

collection - 09

Queue Interface (1.5)





> jswap java.util.Queue

```
import java.util.*;

class QueueDemo{
    Public static void main(String[] args){
    Queue que = new LinkedList();
    que.offer(10);
    que.offer(20);
    que.offer(50);
    que.offer(30);
    que.offer(40);
    sop(que);
    que.poll(); → if exception throw nhi aaega.
    que.remove(); → if queue is empty then remove method throws NoSuchElement exception

    sop(que);
```

```
sop(que.peek());
sop(que.element()); // Throws NoSuchElements
sop(que);
}
}
```

* Priority Queue

```
import java.util.*;
class PQueueDemo{
  public static void main(String[] args){
    PriorityQueue pq = new PriorityQueue();
    pq.offer(20);
    pq.offer(10);
    pq.offer(50);
    pq.offer(30);
    pq.offer(40);
    sop(pq);
  }
}
```

internally use Heap datastructure (min-max Heap)

O/P: [10, 20, 50, 30, 40]

*** => If we use Heap algorithm → explain in heap data structure lecture

```
pq.offer("kanha");
pq.offer("Ashish");
pq.offer("Rahul");
pq.offer("Badhe");
sop(pq);
```

O/P: [Ashish, Badhe, Rahul, Rajesh, Kanha]

=> Why this O/P in this sequence?

java.util.PriorityQueue

PAGE No.
DATE

writecode by using Comparable / comparator for sorting in priority queue

```
class project {
    String projName;
    int teamSize;
    int duration;
```

javap java.util.Deque

Double ended queue.

① offerFirst(E)
② offerLast(E)
③ pollFirst
④ pollFirst
⑤ peekFirst();
⑥ peekLast();

```
import java.util.*;
class DequeueDemo{
    p s v main(String [] args){
Deque obj = new ArrayDeque();
obj.offer(10);
obj.offer(40);
obj.offer(20);
obj.offer(30);
sop(obj); // [10,40,20,30]
```

⑦ offerFirst()
// offerLast();
obj.offerFirst(5);
obj.offerLast(50); [5,10,40,20,30,50]

sop(obj);

```
//pollfirst
//pollLast();
obj.pollFirst();
obj.pollLast();
sop(obj); // [10,40,20,30]
```

PAGE No.
DATE

```
// peekFirst()
// peekLast()
write in sop { obj.peekFirst();
               obj.peekLast();
               sop(obj);

// Iterator
Iterator itr = obj.iterator();
while(itr.hasNext()){
    sop(itr.next());
}
```

o/p: 10
40
20
30

```
//descendingIterator.
Iterator itr2 = obj.descendingIterator();
while(itr2.hasNext()){
    sop(itr2.next());
}
}
```

o/p: 30
20
40
10

---

javap java.util.ArrayDeque

↳ try this methods

not Public / methods / final

1. int inc(int, int);
2. ~~de~~int dec(int, int);
3. sub(int, int, int);
4. elementAt(java.Object[], int);
5. nonNullElementAt(Object[], int);
6. inc(int, int, int);

Multithreading revision. # Thread class & runnable

```
Class ThreadDemo {

    public void run() {

    }

Class client {

    Public Static void main(String [] args) }
```

lect 92

seen 27-9-23

PAGE No.
DATE

Collection -10

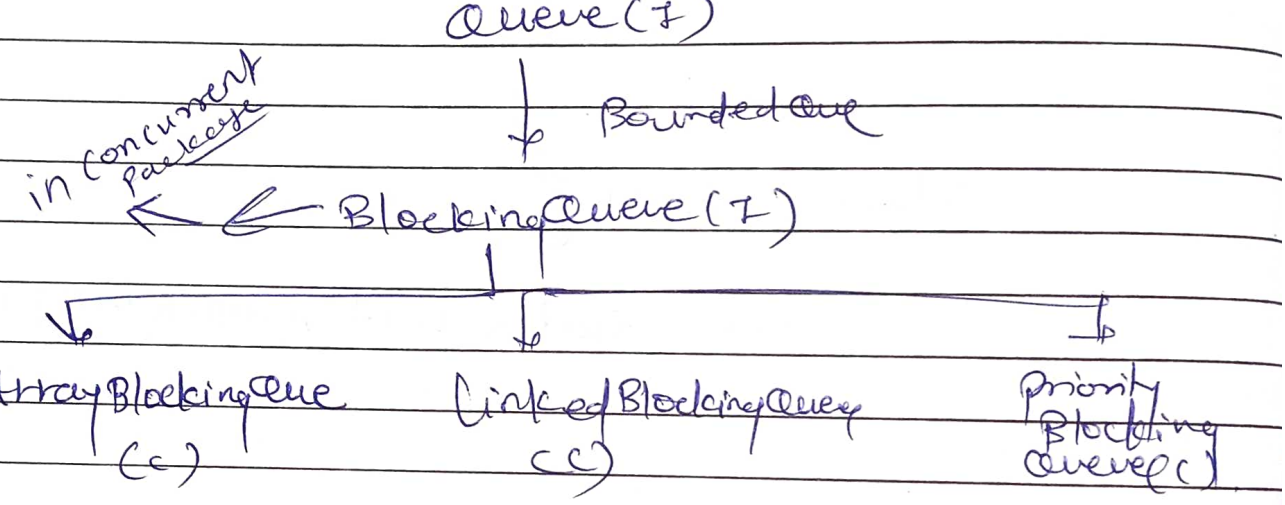
Comparable & Comparator interface यांचा वापर झाला.

Blocking Queue

producer consumer problem

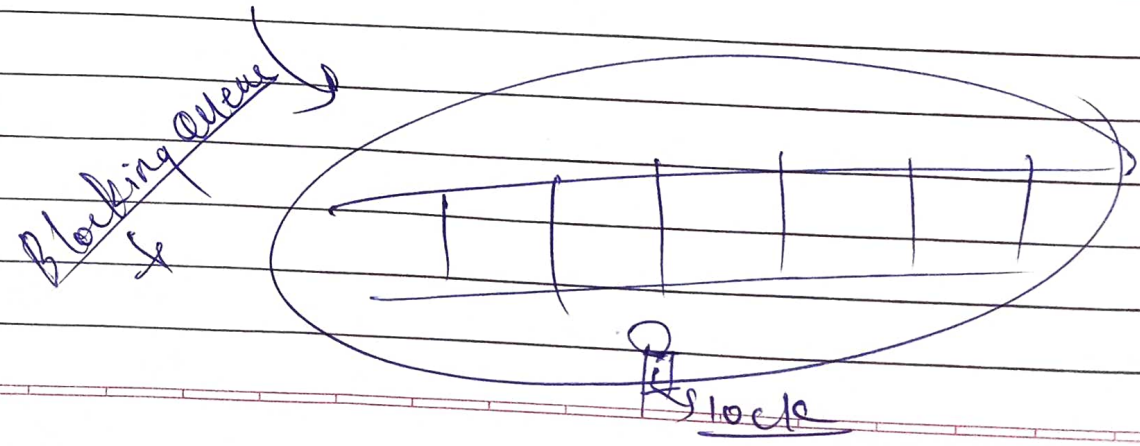
Bounded queue

requirement ↑ price ↑
requirement ↓ price ↓



→ Queue ही Synchronize नाही
→ म्हणून ही Threadsafe नाही

Blocking queue ही Synchronized आहे
म्हणून ही thread safe आहे.
Size वाढवी लागत.



jdkap. java.util.concurrent.BlockingQueue
throws InterruptedException

1. offer(E, long, TimeUnit)
2. take()
3. poll(long, TimeUnit)
4. drainTo(collection, int).

```
import java.util.concurrent.*;

class BlockingQueueDemo{                                 throws InterruptedException.
    public static void main(String[] args){
    BlockingQueue bqueue = new Array
                                BlockingQueue(3);

        bqueue.offer(10);
        bqueue.offer(20);
        bqueue.offer(30);
        // bqueue.offer(40);
        sop(bqueue);
        bqueue.put(40);   ——→ block हो जाएगा.

        sop(bqueue);
```

using try catch

```
sop(queue)
try{
  bqueue.put(40);
}catch(InterruptedException ie){
    sop("BLOCK....");
} sop(bqueue);
```

Queue की size ज्यादा हो तो data add करने पर thread को block करना.

```
import java.util.concurrent.*;
class BlockingQueueDemo{
public static void main(String[] args)
            throws InterruptedException{
    BlockingQueue bqueue = new ArrayBlockingQueue(3);
    bqueue.put(10);
    bqueue.put(20);
    bqueue.put(30);
    sop(bqueue);
```

bqueue.(40, 5, TimeUnit.SECONDS);
data time timeunit)

sop(bqueue);
bqueue.take();
sop(bqueue);

TOP()
peek()
methods

//drainTo//

ArrayList al = new ArrayList();
sop("ArrayList" + al); // []
bqueue.drainTo(al); //
sop("ArrayList" + al); // [10, 20, 30]
sop(bqueue); // []

java.util.concurrent.ArrayBlockingQueue

java.util.concurrent.LinkedBlockingQueue.

LinkedBlockingQueue without (blank) parameter constructor

BlockingQueue bque = new LinkedBlockingQueue();
bque.put(10);
bque.put(20);
bque.put(30);
sop(bque);
bque.offer(40, 5, TimeUnit.SECONDS);
sop(bque);

size limit

java.util.concurrent.PriorityBlockingQueue

concept आहे.

default capacity

PriorityQueue.

```
class BlockinQueueDemo{
    public static void main(String[] args) throws
        InterruptedException{
  BlockingQueue bqueue = new PriorityBlockingQueue(3);
    bqueue.put(10);
    bqueue.put(20);
    bqueue.put(30);
    sop(bqueue);
    bqueue.offer(40, 5, TimeUnit.SECONDS);
    sop(bqueue);
```

source code बघायचा आहे

o/p:⇒ [10, 20, 30]
[10, 20, 30, 40]

} इथं priorityqueue मध्ये size दिली तरी data add होतो याचा इथं internally काहीतरी वेगळ आहे ते शोधायचं आहे.

```
    bqueue.put("Krishna");
    bqueue.put("Ashish");
    bqueue.put("Rahul");
    sop(bqueue);
    bqueue.offer("Badhe", 5, TimeUnit.SECONDS);
    sop(bqueue);
```

⇒ Bubble up function internally for sorting internally

sirache question:

Q1 40 data add कसा होतो?

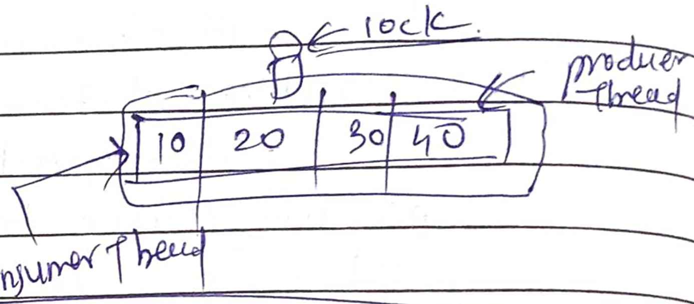
Q2 PriorityBlockingQueue internally heapsort वापरतो का?

user definded class ব্যাপক code করা
→ using comparable interface

PAGE No.
DATE

```
import java.util.Concurrent.*;
import java.util.*;

class Employee{
 String name;
 Employee(String name){
      this.name=name;
 }
 public String toString(){
      return name;
  }
}
class SortByName implements Comparator{
 public int compare(Object obj1,Object obj2){
 return((Employee)Obj1).name.
      CompareTo(((Employee)obj2).name);
 }
}
class BlockingQueueDemo{
 public static void main(String[] args)
 throws InterruptedException{
  BlockingQueue bQueue=
      new PriorityBlockingQueue(
          3, new SortByName());
 bQueue.put(new Employee
                 ("Kanha"));
 bQueue.put(new Employee("Ashish"));
 bQueue.put(new Employee("Rahul"));
 Sop(bQueue);
 bQueue.offer(new Employee("Badle"),
              5, TimeUnit.SECONDS);
 Sop(bQueue);
```

producer consumer problem




```
import java.util.concurrent.*;
class producer implements Runnable{
 BlockingQueue bQueue=null;
 Producer(BlockingQueue bQueue){
     this.bQueue=bQueue;
 }
 Public void run(){
   for(int i=1; i<=10; i++){
     try{
        bQueue.put(i);
        Sop("produce"+i);
     }catch(InterruptedException ie){
     }
     try{
        Thread.sleep(1000);
     }catch(InterruptedException ie){
     }
   }
 }
```

```
class Consumer implements Runnable{
    BlockingQueue bqueue = null;
    Consumer(BlockingQueue bqueue){
        this.bqueue = bqueue;
    }
    public void run(){
        for(int i=1; i<=10; i++){
            try{
                bqueue.take();
                sop("consume"+i);
            }catch(InterruptedException ie){
            }
        }
        try{
            Thread.sleep(3000);
        }catch(InterruptedException ie){}
    }
}
class ProducerConsumer{
    public static void main(String[] args){
        BlockingQueue bqueue = new ArrayBlockingQueue(3);

        Producer produce = new Producer(bqueue);
        Consumer consume = new Consumer(bqueue);
        Thread producerThread = new Thread(produce);
        Thread consumerThread = new Thread(consume);
        producerThread.start();
        consumerThread.start();
```

1cf93
29-9-23

collection-11

* Lambda Expressions. *

1) Interface
   1) Normal
   2) marker
   3) functional.

1) functional Interface
   ↳ फक्त एकच method असते.
   1) Runnable Interface
   2) comparable Interface

→ lambda expression हे फक्त functional interface वरच वापरता येतं.

```
class Core2web{
void lang(){
    sop("BootCamp, Java");
}
}
class Year2022{
public static void main(String[] args){
Core2web c2w = new Core2web(){
    void lang(){
        sop("BootCamp/Java/Py/OS");
    }
};
c2w.lang();
}
```

Anonymous Inner class

```
interface core2web{
        void lang();
}
class Year2022{
    p s v main(String[] args){
Core2Web c2w = new Core2web(){
public void lang(){
        sop("Bootcamp/Java/Py/OS");
}
};
c2w.lang();
}
```

lambda expression हे मधील class लपवायचा काढतो. म्हणून शॉर्टकट मध्ये company lambda expression वापरतात.

// 1.8 → lambda fn
// functional interface
// code efficient length कमी करतो.

CPP // मध्ये
PAGE No. / DATE / lambda fn आले.

```
interface Core2web{
        void lang();
}       // String or lang();
class Year2022{
        P S V main(String [] args){

            Core2web c2w = ()-> {
                    sop("Bootcamp/Java/Python/os");
            };
            c2w.lang();
        }}
```

→ तर for statement असेल तर

```
core2web c2w = ()-> sop("Bootcamp/Java/Py/os");
c2w.lang();
```

```
Core2web c2w = ()-> {
        return "Bootcamp/Java/Py/os";
};
sop(c2w.lang());
```

```
Core2web c2w = ()-> "Bootcamp/Java/Py/os";
sop(c2w.lang());
```

lambda function with one parameter.

```
interface core2web{
    String lang(int numCourse);
}
class Year2022{
    P S v main(String[] args){
        Core2Web c2w = (xyz) -> "Bootcamp/Java/Py/os"+":"+xyz;
        sop(c2w.lang(8));
    }
}
```

```
Corewweb c2w = xyz -> "Bootcamp/Java/Py/os"
                      +":"+xyz;
```

```
interface core2Web{
    String lang(String company1, String company2);
}
class Year2022{
    public static void main(String[] args){
        CoreWeb c2w = (name1, name2) -> {
            return name1 + "&" + name2 + "By Shashi";
        };
        Sop(c2w.lang("Biencaps", "Incubator"));
    }
}
```

PAGE No.
DATE

Runnable interface with lambda function.

```
class Demo implements Runnable{
    Public void run(){
        sop(Thread.currentThread.getName());
    }
class ThreadDemo{
    p.s v main(String[] args){
        Demo obj1 = new Demo();
        Thread t1 = new Thread(obj1);
        Demo obj2 = new Demo();
        Thread t2 = new Thread(obj2);
        t1.start();
        t2.start();
    }}
```

Runnable with Anonymous Inner class.

```
class ThreadDemo{
    public static void main(String[] args){
        Runnable obj1 = new Runnable(){
            public void run(){
                sop(Thread.currentThread().getName());
            }
        };
        Runnable obj2 = new Runnable(){
            public void run(){
                sop("Thread.currentThread().getName());
            }
        };
    }
```

```
Thread t1 = new Thread(obj1);
t1.start();
Thread t2 = new Thread(obj2);
t2.start();
}}
```

o/p: Thread-0
Thread-1

Runnable with lambda function

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

Runnable obj1 = () ->{
    sop(Thread.currentThread().
        getName());
  };
Thread t1 = new Thread(obj1);
t1.start();
```

Lambda function with Comparable or Comparator.

```
import java.util.*;
class Employee{
  int empId = 0;
  String name = null;
Employee(int empId, String
                    name){
  this.empId = empId
  this.name = name;
}
public String toString(){
  return empId + ":" + name;
}
}
class Demo {
  public static void main(
      String[] args){
ArrayList al = new ArrayList();
al.add(new Employee(25,"Kanha"));
al.add(new Employee(15,"Ashish"));
al.add(new Employee(22,"Rahul"));
Sop(al);

collections.sort(al,(obj1,obj2)->{
return((Employee)obj1).name.
        compareTo(((Employee)obj2).
            name);
  }
);
SOP(al);
}
}
```

PAGE No. DATE

* for Each Iterator *

① Iterable

javap java.lang.Iterable

```
class ArraylistDemo {

    public static void main(String [] args){

    Arraylist al = new Arraylist();
        al.add(10);
        al.add(30);
        al.add(20);
        al.add(40);
        sop(al);
    // for( var data : al){
    //         sop(data);
    // }

    * al.forEach((data) -> sop(data));
} }
```

in version 17 → var (new feature)

↳ lambda function