# Multithreading in java

mainly there are 4 multi-term are their

~~(*) multithreading~~

(A) Multiprogramming

multiple programme are getting executed at same time

(B) Multitasking

A same resource is being shared between multiple programme

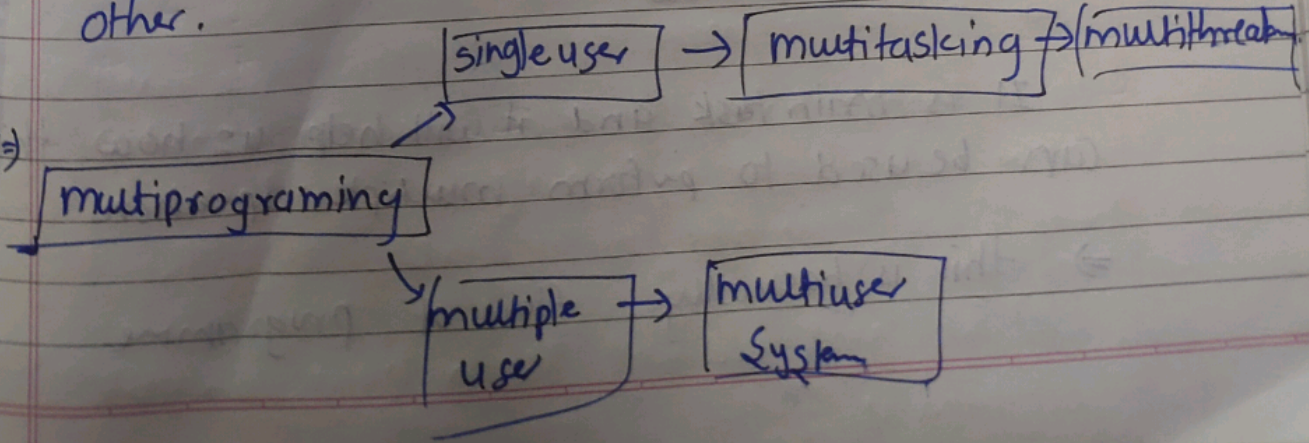
(C) Multithreading

In this process more than one thread is processing the programme

(D) Multiprocessing

In such case multiple processes exist on same single unit and process is switching betn each other.

chart =>



*)
multiprogramming concept arrived in order to use the CPU for maximize performance and generate optimal usage from it.

CPU-time ⇒ Amount of time cpu gives in order to do some task or execute a instruction

# Multitasking

for single user we are running multiple tasks at same time, this process is called multitasking.

⇒ All programm are present in memory and they are being executed simultaneously alternatly.

windows, macos ⇒ Multitasking enviroment for single user

Ⓐ Multithreading

Single tasks is being performed by multiple threads.

Ex:- youare using chrome and you're using youtube so you can see recommendations, comments, videos all at one place so at such time there exist multiple threads in system and they are handling multiple tasks from same programm..

# Control flow

It is main task and it will help us how threads can be used to perform multiple operations

⇒ this is basically flow of programme

Page No.
Date

Programme always start from main then it will traverse from code inside main and at end return at last.

Ex:-)
class Test {

③ → static void display() {

④ → S.o.p("Hello world");

}

① → static public void main() {

② → display();
⑤ → S.O.P("bye world");

}

}

main is entry point

this happens in single thread system

you can learn control flow using help of flow of programe

⇒ But inside multithread system we want multiple threads to execute same programme
* multithreading in java provides multiple flow of control to execute a single programme

Ex:-) in main you want a different thread to handle display() function and some other thread will handle main function at such case we need multiple flow of control & for that

we use multithreading.

target:

Ex: class Test {

```
static void display() {
        int i=0;
        while(true) {
        S.o.p("Hello"+ i);
        }
    }
```

→ one thread → we want

```
public static void main() {
        int j=0
        display();
        while(true) {
        S.o.p(j+" world");
        j=j+1;
        }
    }
}
```

→ another thread.

In these code we only get Hello infinit no. of time

So inorder to make that happens we will use two ~~functio~~ threads one for main(); another for ~~test()~~ display(); and with these help we can able to achieve multithreading

④ So next we will learn how can we achieve it

Thread class is so much important inorder to achive multithreading

#Multithreading in java

Ⓐ Thread class
Ⓑ Runnable interface → (It also need thread class)

Use of above both we can use multithreading inside java programme.

In order to achieve multithreading you need a class extending Thread class

(*) In java all things happens with help of class you want exception extend Exception class, you want inheritance do extend with parent class, same for multithreading

Ex:- class BlahBlahBlah extends Thread { code ..... };

thus how we can create multithreading

→ Each class extending thread has a function called run an whatever we want to run throught that thread write inside it.

⇒ run is void function

```
Class MyThread extends Thread {
                    → r
    Public void run() {

        int i=1;
        while(true){
            S.O.P( i +"Hello");
            i++;
        }
    }
}
```

⇒ Code inside run will be function to be completed by that thread

run is starting point of thread

run function() is important in case of multithreading

```
// main class
Class Test {

    p.s.v.main(){

        Mythread a= new MyThread();
        ↑
        // creating object of thread class's child

        // in order to start that thread which
        give rise to ext additional flow of control
```

special case ⇒ we call start function but run function will be get executed.

a.start(); → start method used to run the run function from it's class

```
        int j=1;
        while(true){

        S.o.p (j + " world");
        j=j+1;
        }

}
```

thread.start()
↑
⇒ will create additional Controll flow

Such like that code will propagate & executed

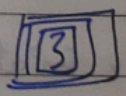
④ try to run this code

⇒ now you will seed "Hello" sometime & "world" sometime as we create additional Controll flow using ~~java~~ java multithreading

* run function is important here
* we use .start() function in order to start exploration of code

you can do all in same class also
# declare main class extending thread

```
class Test extends Thread {

P.S. Void run() {}  →  run function, it is function in thread class and we will override

    int i=0;
    while (true) {
        S.o.p. (" i + " Hello");
        i=i+1;
    }

P.S. v. main() {

    Test a= new Test()
    // start new thread
        a.start(); → this will ~~run~~ execute run() function from Test class.

    int j=0;
    while (true) {

    S.o.p( j + " world");
    }
```



⇓ thus how we can be able to execute the code

## ⊕ Runnable interface

We can use Predefined interface called runnable. We implement it with (Runnable) interface and from that we can do multithread

⇒ we need thread class object inorde to start() execution of run function

④ class mythread implement · Runnable {

Ex:

pseudo code

```
void run () {
    —11 —
}

main () {
// create object of mythread class,
mythread a = new Mythread();

// now create object of Thread class and assign value to a refern and pass a to new object

Thread m = new Thread (a);

m.start()
```

Such like that you can use interface, create a class implementing that interface and in main create object of that class and threan

Page No.
Date

pass it to a new Thread object and with thread object we can use start() func & run multithreaded code

# state diagram for threads

sleep → shutting a thread by own

① Possible states of threads

Ⓐ ready
Ⓑ running
Ⓒ waiting.
Ⓓ timed wait.
Ⓔ Blocked.
Ⓕ Terminated.
Ⓖ New

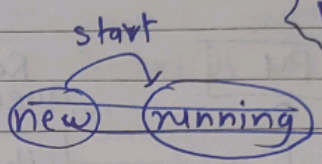
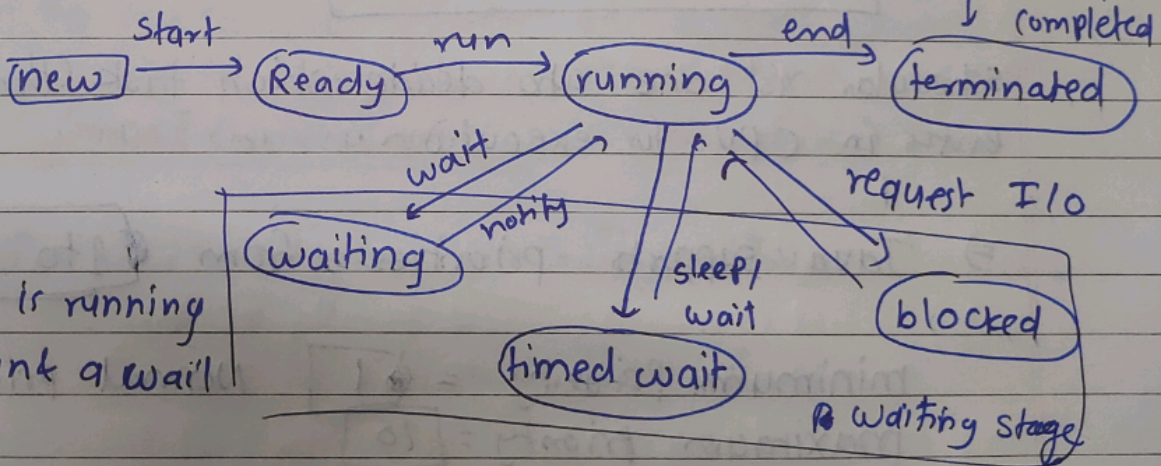



Diagram will show all possible states for a thread & how it will get executed

* state diagram for thread




If thread it is running it ~~wait~~ runt a wait

⇓

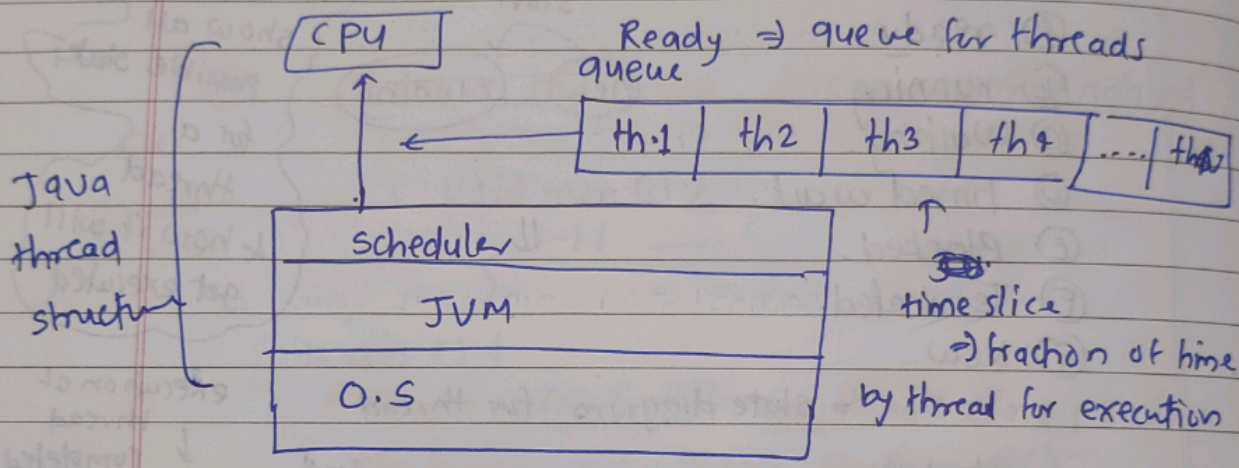
If a thread is started & ended it's ~~existined~~ does exist now you have restart create new thread. ~~&~~ execute

at a time CPU can perform one task

CPU time ⇒ time take by CPU to solve a instruction.

# Thread priorities

Say a programme have 1...n threads. and each thread has own tasks to perform but say e.some tasks have more priority than other tasks so at such case we are able to assign thread priorities




scheduler is one who decide which task (thread) should enter in CPU for execution.

⇒ Java supports priorities from 1 to 10

minimum priority = 1
maximum priority = 10
Normal priority = 5

⇒ Idea of thread priority is tasks / threads with higher priority value should get priority for CPU time

⇒ multithreading mainly provided by OS but even though OS not support JVM will still manages to use multithreading

get & set priority for a thread.

class MyThread extends Thread {

    void run() { }

}

⇒ By default priority of thread is 5

normal

$1 \leq n \leq 10$

main() {

    MyThread m = new MyThread();

    To set priority

    m.setPriority(n);

    m.getPriority() ⇒ In order to get priority for a thread

⇒ You can get priority of a main thread by write following in main function.

Thread.currentThread().getPriority();

⇒ Thread g = Thread.currentThread();

A reference to current thread

(*) g.getName(); → to get name of thread

g.setName(); function to set name of a thread

this function can be useful sometimes

⇒ ThreadGroup ⇒ It is some collection of thread which act together

Thread is predefined class inside java language

ThreadGroup object has some collection of threads

#Thread class in java

① Predefined constructor for Thread class

Ⓐ Thread()
Ⓑ Thread(String name)
Ⓒ Thread(Runnable r)
Ⓓ Thread(Runnable r, String name);
↑
Passing a class created with interface & name will assign name to thread
Ⓔ Thread(ThreadGroup m, String name);
~~Ⓕ Thread(~~
this are few constructors

② Some usefull functions inside Thread class

long getId(); this will give you id of thread and you can't set id

int getPriority(); get priority of thread
void setPriority(); Set priority of thread
↑ Parameter

String getName(); get name of thread
void setName(); set name of thread
↑ argument

Thread.State getState(); ⇒ To get state of thread

thread.Group getThreadGroup(); ⇒ this will return Thread group which related to this thread.

group object
some collection
threads

name

class

thread

read
this

setName()
setDaemon()
setPriority();

Daemon runs in background with least priority.

set fun

Void setDaemon(boolean d);

↑

It is a background thread and it has very low priority

→ you can set a thread as Daemon by setDaemon() function

# Enquiry method

this function help us to find out above as ~~eq~~ enquiry

boolean isAlive();
boolean isDaemon();
boolean isInterrupted();

this will return true on false on condition

→ Interrupt is command which stops propagation of a thread and we can check interrupted or not with isInterrupted function();

# Instance method

Void interrupt(); → It can be used for interrupting a thread

Void join(); → As thread end after execution we use join function to make thread wait after execution

Void join (long millis); ↓ thread is alive until other threads not complete ↓ time

Void run(); main logic

Void start(); function which run start()

# static methods

these are static method that mean they are common for class and directly used via class name

② Thread.activeCount();

this will current achive thread count for program at a instance

→ CurrentThread() →
This will return ~~reference~~ to object of current thread which is getting executed

⊙ Void yield();

this thread can make higher priority threads, to avoid low priority thread to lead to starvation
⇒ Somehime CPU gives time to higher priority thread in order to avoid it

Ⓔ dumpstack()

we can able to see how control flow is going to flow through a thread for such time we can use dumpstack

Thread.sleep();
this function will be used to stop working a thread for a while

⇒ now most of the part from thread is covered so now lets get forward

Imp # Thread.sleep(long time);
↑
this function will stop system processing for time you specified. time it is in milliseconds.
↑
this work same like time.sleep() function in python language.

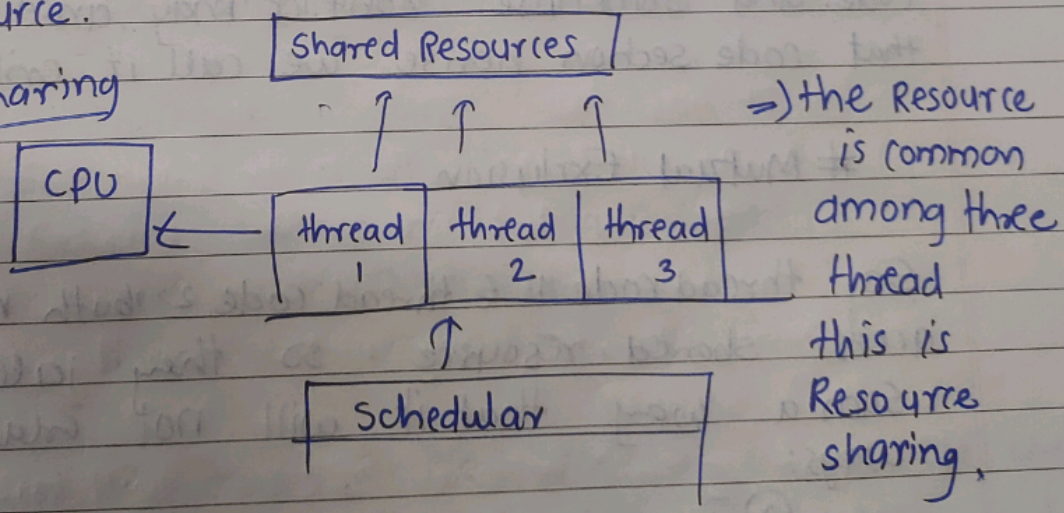
sync ⇒ co-ordination

#Synchronisation

→ It is basically co-ordination betⁿ processes, threads or functions and

→ Threads has own stack but the heap is common among all threads.
⇒ common resource that is heap is a shared resource.

A Resource Sharing



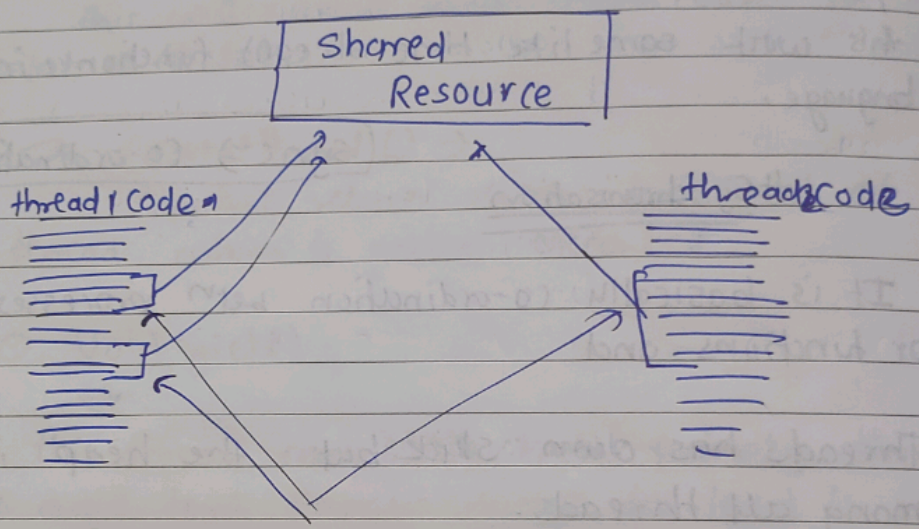

⇒ the Resource is common among three thread this is Resource sharing.

(*) Monitor It is main Resource handling Part In Java

(#) Critical Section

So each thread has own code called thread which run inside it Ex run() function code

=) The section of code in thread code using shared resources is called as critical section




⇒ So this pointing code is called as critical section of code. and sometime conflict may arise due to that code section hense we call it critical section

# Mutual Exclusion

So thread code 1 & thread code 2 both needs to access shared resource so they access in such a way that it will not create a conflict ways:

(A) use resource alternatively
(B) use one after another

⇒ In order to avoid problem threads should access the shared resource one after another making sure that they both not accessing resources simultaneously

⇒ Mutual Exclusion means avoiding one thread when another thread is accessing resource

⊕ ways to have mutual Exclusion

Ⓐ To have some system who allow only one thread at a time to allow access

Systems :

⇒ locking / mutex
⇒ Semaphore
⇒ Monitor

Ⓑ To have system and co-ordination beth the thread themselve

⇒ Race condition .
⇒ Inter-thread Communication.

# Locking / mutex

@ Resource has a mutex (a variable)

mutex = 0 Not locked
mutex = 1 locked

thread which need to have access over Resources check mutex value if it's 0 lock resources

In Java we only have Monitor In order to handle Resource sharing

and same time another want to access it will show mutex = 0 so it wait for resources to be unlocked so that

→ After time gone thread will unlock resources, change mutex value & go back to wait queue

Sometime, I say one thread checked for mutex at it came 0 but it's time is over and at that time another thread come see mutex = 0 & locked and first thread also locked so both thread using same resource and this is conflict

(*) mutex is not a great way as it is only value we need a system to handle it

(#) Semaphore (It is a system

It is a os system which will manages Resource Related system, it mainly seen in unix system

(*) It has two signals:
- (A) wait() (To make thread wait)
- (B) Signal() (to give into to thread that system is busy)

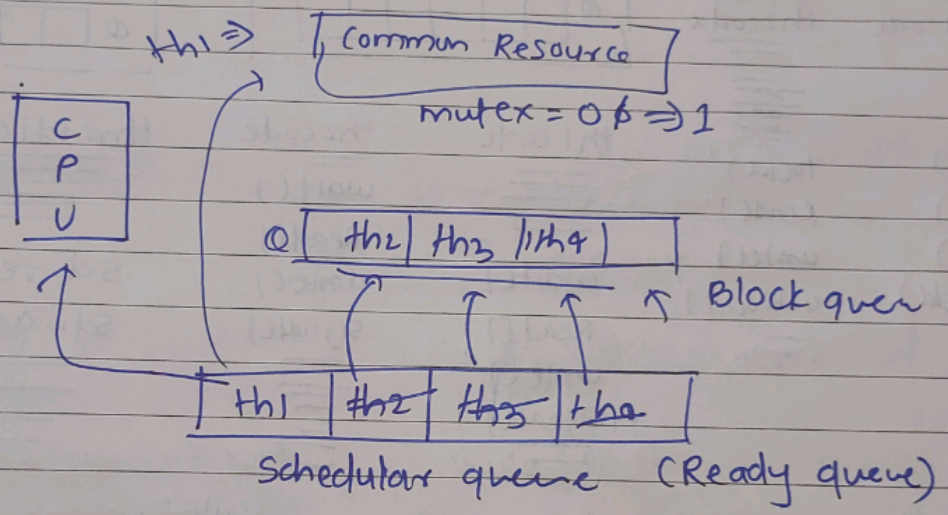
(*) When system is being used currently it will give wait() signal

(*) When system become free it will give signal() sign and next thread in queue will get access over Resource

⇒ After it take access it's following threads will wait for receive wait() signal

⇒ semaphore also has mutex value and on basis of locking and mutex value it will give signal

⇒ Semaphore has a queue (Block queue) for thread so when one is using others are in block queue & receive wait signal



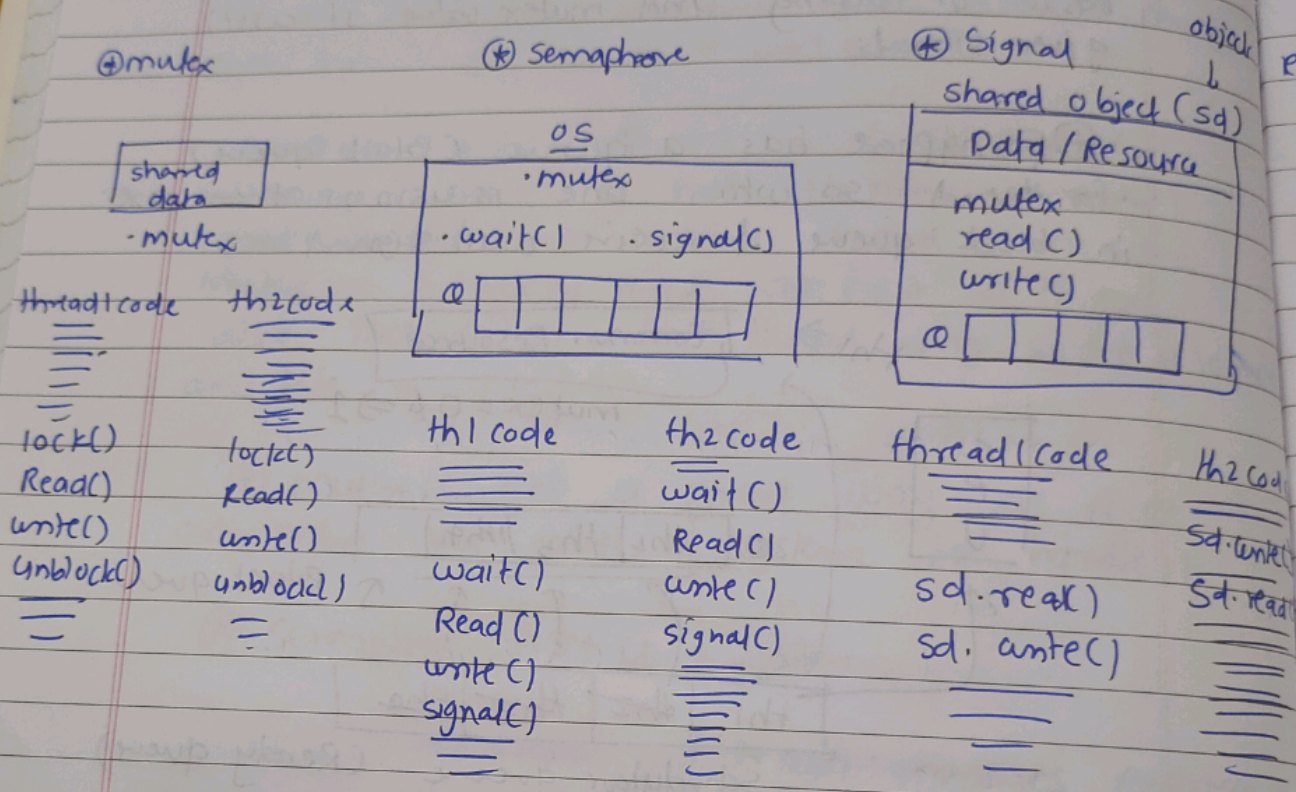

⇒ Semaphore make Ready queue thread to block queue

⇒ when wait() is ended thread moved from block queue to ready queue & then it can grab resources

⇒ In semaphore locking unlocking is done by Operating System

# #Monitor

this is main part of OOPs where we handle multithreading in java.




See diagram an you will get overall idea about monitor and then read following theory.

monitor

object in java contain function and qua it do not have locking/unlocking, they don't have features

⇒ objedr of monitor itself provide locking, unlock and many more

Page No.
Date

⇒ To have these we just have to use keyword synchronise.

# How to attempt synchronisation in java

Refer

A monitor block or monitor funct

synchronised block

```
synchronised ( this) {
    Code ;
}
```

→ at a time only one thread can go inside block

So in synchronised block only one code and at a time ~~thread~~ only one thread can evaluated.

⇒ 1 or you can create function synchronised so at a time only one thread can use that fun

synchronised function

```
Synchronised static void display(){
    Code
}
```

# ATM

⊙ only one ~~tuts~~ customer at a time

solve using class

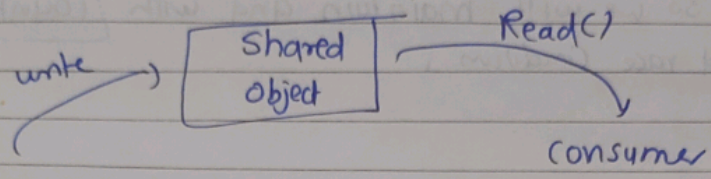
```
class ATM {
  CheckBalance(Cla name);  → create thread with na
  withdraw (name, amount);  ⇒ also name an
                               amount
}

Class customer {

  ATM atm;  → single atm for multiple customy
  String name;
  int amount;

method  use ATM() {

      CheckBalance();   Check Balance + thn
      withdraw();       withdraw many

  }
```

⇒ Done with this

#Synchronisation via inter-Thread Communication

In such case we want synchronisation Provided that there exists inter-thread communication.

Ex:



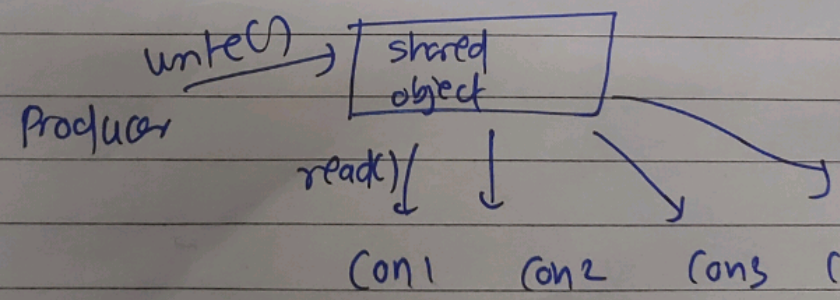

Producer is writing something and that will be read by consumer so there should be sync betn consumer & Producer in order to avoid problems

⇒ such multithreading is one which java do not do, so it wholely responsibility of programme to handle it.

Ex: Add Flag it is True when producer is writing & 0 mean ~~ob~~ shared object is under use by consumer

⇒ i.e while consumer Reads Producer stops, when ~~Consumer~~ Produer reads Consumer Stops

⊛ -multiple consuming




Ex teacher is writing at that time student will wait & on teacher's completion student will start reading

GRS.

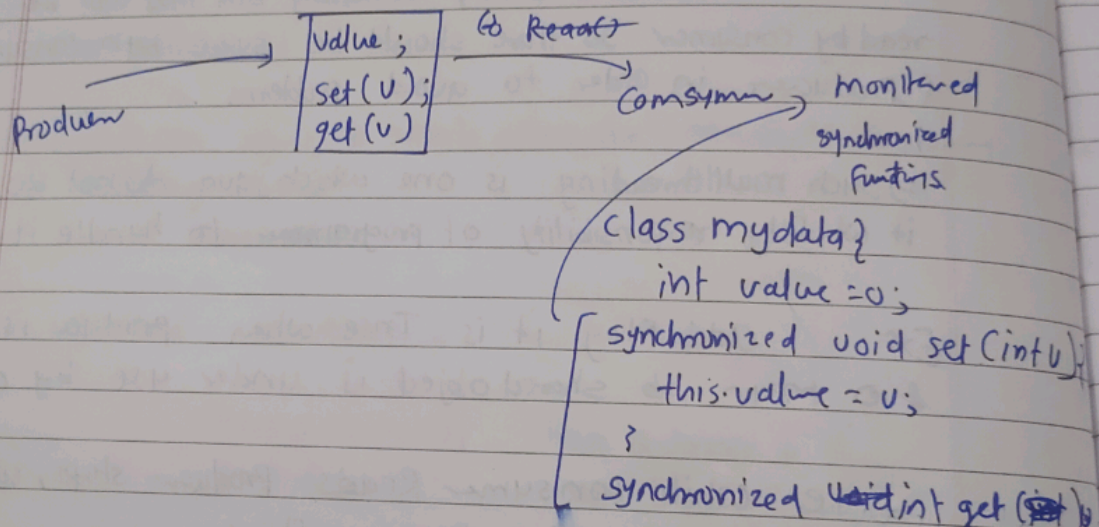
0 => for producer write

So at such case we can't use True or false so we use count and at 1 — n the consumer 1.....n will get time to read.

So At such cases Race is generated in order to get access so we will maintain and with [count] you can avoid race conditions.

EX:

// Programme has to attempt interthread communication




```
class mydata{
    int value=0;
  synchronized void set(int v){
    this.value = v;
  }
  synchronized int get(){
    return this.value;
  }
```