A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

Multi Threading

Multitasking :-

=> Executing multiple task simultaneously where each task is independent program.

Ex. => 1) ide prog
2) listing songs
3) Downloading

① process based (OS level) ② Thread based (Programmer level)

~~Multitasking~~ :-

Thread based :-

=> Executing several task simultaneous where each task is seperate independent task of same program.

We use multithreading to increase the performance of System by reducing the idol time of CPU.

For developing web Application, use multitasking

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

☆ Thread => it is a light weight process
(it a seprate process and doing some jo

☆ Define thread :-

- 2 ways

1] By Extending thread class
2] By Implementing Runnable Interface

1] Extending thread class

```
class MyThread extends Thread
{
    Public void run() {
        for(int i=0;i<10;i++)
        { S.O.p("child thread");
        }
    }
```

Defining thread

Job of thr

```
Class ThreadDemo
{ PSVm(String...args)
{ MyThread mt=new MyThread();//Thread Instans
  mt.start();
}
}
```

// main thread excutes the Child thread

Multithreading (2)

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

Case (1) :- Thread Scheduler

⇒ part of JVM (which order they are executed)
• may vary from System to System
if t.start() is called → different flow created

Case (2) :- if we call t.start() & t.run()
1] Diff betn. t.start() & t.run()

if we call t.start() new flow is created (thread)., & ↓ Thread Scheduler is responsible for the exec
if we call t.run() then main thread only call on thread.
1st t.run() 10 times & then main 10 times executed by main method only normal method call
O/P :- child thr X10
Main thread X10

in Start() there are various o/perd
1] Register the Thread with thread scheduler
2] Perform activities
3] invoke run();

Case 3 :- if we overload run() : run(int i);
⇒ the only non-arg run() is called when we create the object of thread class.

⇒ we have to call arg method Explicitly.

```
Class mythread extends Thread
{ public start()
  { super.start();
    sop("start method");
  }
}
```

```
psvm(){
  t.start()
  sop("main
}
```



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

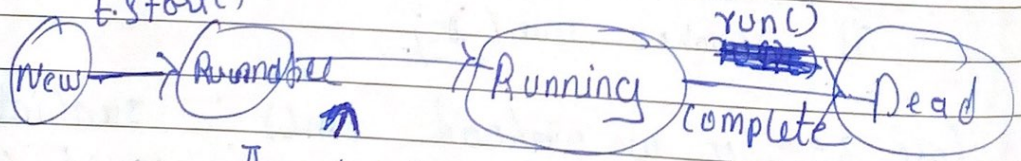
Case 4:- if we not write run() method.
⇒ we will not the o/p, as the parent class Thread has empty implementation.
⇒ we should also override run() always.

Case 5:- Overridding Start() method.

⇒ it will give priority to object class method ⇒ So new thread will not created
⇒ So only start method is executed as normal method

Case 6:-
if we write super.start() then the start() method is executed by main method & the super.start() create a thread job.

Case 7:-
Thread
t.start()



Case 9:- if we call the same t.start() again the IllegalThreadStateException;

Multithreading (2)

* if we implement Runnable then the interface has only run method, so we should pass the Runnable implemented class object to Thread to start the run();

Secenio 2 :-

when class implements Runnable interface

```
class MyThread implements Runnable
{
  public void run(){
    Sop("child Thread");}
}
```

* Case Study

```
MyThread mt=new MyThread();
Thread t1=new Thread();
Thread t2=new Thread(mt); // passing object to Thread
```

Case 1: t1.start() // no o/p, empty imple. in thread class
=> new thread will be created by no o/p as it has empty implementation

case 2:- t1.run(); // no o/p -> as it has empty imple in the class
=> No new thread will be created, and direct run method of Thread class

Case 3:- t2.start();
=> new thread will created and and child Thred run method is called, Job is executed.

t2.run();

Case 4: t2.run() :- t2.run();
No new thread created direct method call of MyThread class.

Case 5: r.start() :-
Compiler Error No symbol in Runnable cla.

case 6: r.run();
=> direct method call of MyThread class.

★ Which is best & recommed to extend or use multithreading

★ Implements Runnable is Recommended

=> Because extending Thread we cannot use Inheritance
=> In Implements we can use Inheritance.

★ Thread class Constructor :-

1] Thread t=new Thread();
2] Thread t=new Thread(Runnable r);
3] Thread t=new Thread(Runnable r, String name);
4] Thread t=new Thread(ThreadGroup g, String name);
5] Thread t=new Thread(ThreadGroup g, Runnable r);
6] Thread t=new Thread(ThreadGroup g, Runnable r, String name);
7] Thread t=new Thread(ThreadGroup g, Runnable r, long Stacksize)
8]

Multithreading (3)

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

How to get & set current thread name

1) To get thread name?

⇒ S.o.p(Thread.currentThread().getName());

2) To set thread name?

⇒ S.o.p(Thread.currentThread().setName("mayur");

F.o. main thread

S.o.p(Thread.currentThread().getName());

⇒ MyThread mt = new MyThread();
S.o.p(mt.getName()); // Thread-0 (and so-on)

⇒ S.o.p(mt.setName()); //"mayur"

Method to set Thread Names:-

⇒ Public final String getName()
⇒ public final void setName(String);

We can get current executing thread obj by Thread.currentThread() ⇒ Static method in Thread.

Thread priorities (1 to 10) Not from (0-1

1) default Provided by JVM
2) or provided by Programmer

* Every thread has some priority, it may default by JVM or by programmer will provided

Valid Range :- (1 to 10)
Thread.MIN_PRIORITY => 1
Thread.NORM_PRIORITY => 5
Thread.MAX_PRIORITY => 10

* Thread class provide constants, to represent

* Thread scheduler will use priority while allocating processor
=> The thread having high priority will get Priority

=> if two Thread having same priority then the order may depend on Thread Scheduler.

=> Following Methods to get & set priority

1) Public final int getPriority()

2) Public final int setPriority()
only range 1 to 10 => otherwise runtime exception
IllegalArgumentException.

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

Thread
Parent class
Mythread
Main thread
Parent thread

Default priority => (5)

Only main thread default priority is (5) and for all other thread is inherited from parent to child.

=> if we set main thread to valid range then all child will inherit that priority

Some OS does not support thread priorities

To prevent Thread execution:-

1] yield()
2] join()
3] sleep()

1] yield() → to pause current executing thread to give chance to waiting thread of same priority.

→ if there no thread waiting then the method may continue.

-> if all thread have same priority, then curr will (cont'n)

The yielded thread when it will get chance depends on scheduler, we cannot predict

```
public static native void yield()
```

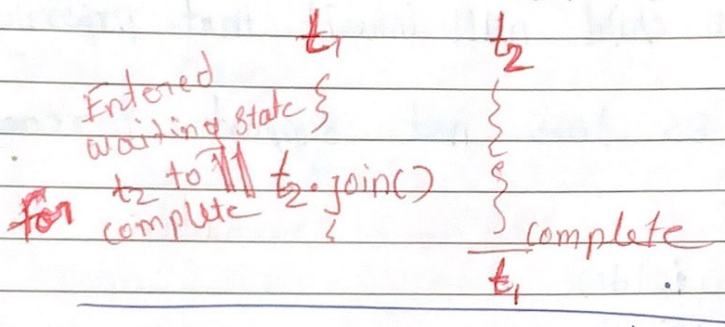
underline OS support must be required for native()

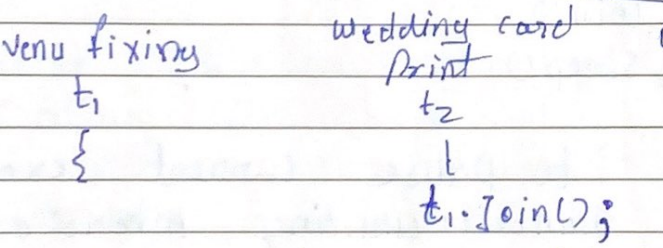
when a yield() is called the thread goes from running state to runnable/Ready state

Join() :-

If a thread wants to wait until complet other thread then Join() is called

⇒ if T_1, thread has to wait until thread t_2, completion, the (T1) thread will call Join method on T_2, object



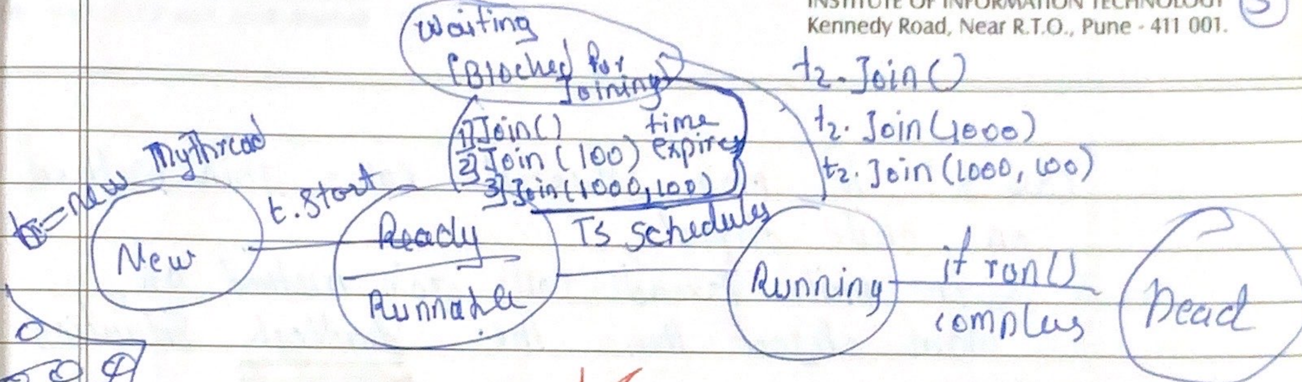


Here, t_2 Has to wait until t_1 is completed, so it call's t_1.join(),
& t_3 Has to wait for t_2 to complete, so it calls t_2.join()

1) public final void Join() throws InterruptedException
2) public final void join(long ms)
3) public final void join(long ms, int ns)

ms is in Long & int is ns because after certain ns, the ms is increment & it throws Interrupted Exception, and it is checked exception, because t_2 is waiting t_1 to complete but

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.



100
~~200~~ 0
20
ms 1sec
100
500 100
~~8000~~
100

Case 2 →

We can allo calle the main method thread, and wait for it to complate its execution, using child thread clas

```
class myThread extends Thread
{
    Static Thread mt;
    Public void run() {
        try {
            mt.Join()
        }
        catch(Interrupted Exception e) {}
        for(int i=0; i<10;i++)
        {sop (child');
        }
class Thread
{ p s v m(String []args) throws IntruptedExc
  { MyThread.mt = Thread.currentThread();
    myThread.mt=new MyThread();
    mt.start();
    for(int i=0; i<=10; i++)
    {
        sop('main');
        Thread.sleep(1000);
    }
  } }
```

In this child called main thread object, child thread wait until main thread complete

Case 3 :- if main thread calles join method on child object, & if child Thread calls join method on main object then the deadlock situation occurs.

Case 5 :- main Thread calls join() on itself

```
Class MyThread
{ p s v m(String[] args)          //will wait int
    Thread.currentThread.join();
}
```

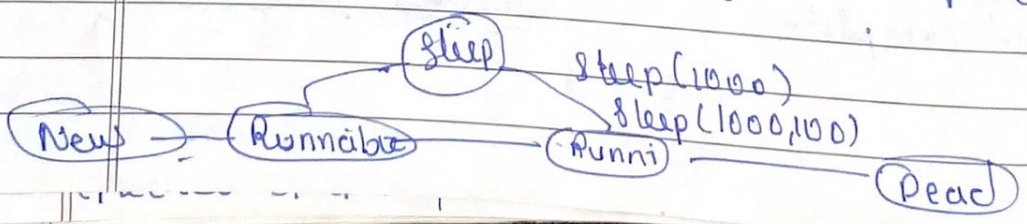
☆ Thread.sleep method,

public static native void sleep(long ms) Int...

public static void sleep(long ms, int ns) Int...

Note → Every sleep meth throws Interrupted Exceptio... Checked Exception, Hence, we are using sleep() we handle interrupted Exception (try, catch, thro...

It will come out of sleep method
1] after time completion.
2] sleeping thread get interrupted.



A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

How a thread interrupts sleep thread?

a thread can interrupt a sleeping thread, or waiting() by using interrupt() of thread class

```
public void interrupt().
```

Note :- if we call interrupt(), the target thread not in sleeping state or waiting then there is no impact of interrupt(), it will only work when target Enters into sleep(),

⇒ if it does not enter into sleep state in lifetime then interrupt has no impact

if we call interrupt on Join() / waiting state the child may - maynot execute for a single time

if we call interrupt on the sleep() it is executed single time and then terminated.

Property	yield()	Join()	sleep()
Purpose	1] If a thread wants to pass it execution to give chance, remaining thread of some priority	1] If thread waits to wait until completing other thread then go for join	2] If thread do Want to perf any operation
It is overloaded	No	Yess	Yess
Is final	No	Yess	No
throws Interrupted	No	Yess	Yess
It is native	Yess	No	sleep (Long ms) N sleep (Long ms, int → No
Is static	Yes	No	Yes

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

★ Syncronization :-

⇒ only Syncronized keyword is used, and we have method & block for Syncronized.

★ To avoid ~~for~~ data incursitency we used Syncronized.

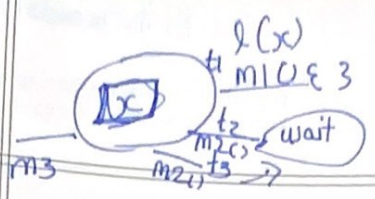
→ The advantage is we can resolve data inconsitency prb,
→ but disadvantage is it increases waiting time of thread, & create performance problem.
→ if no specific requirement then it is not recommed to use Syncronized keyword.

⇒ Syncronization concept is implimented by using Lock, Every obj in java has unique lock, whenever we are using Syncronized keyword, lock concept came in picture.

⇒ if the thread wants to ~~get~~ execute Syncronize on the obj then it has to use the lock of that object.

⇒ Requiring and relasing lock is taken care by JVM, not programmer.

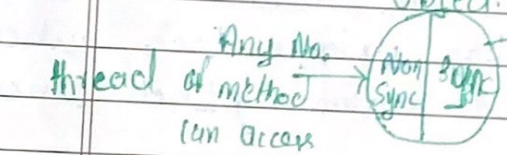
```
Class m
{
Syn m1()
Sync m2()
  m3()
}
```



⇒ In above example, if a th. m1() is executing then it has lock of l(x) object, i.e if Thread 2, comes in for using m1(), it will enter waiting state.
⇒ if t2, comes & asks for m2(), it will also go into waiting state. as the (x) obj is locked by th.m1().

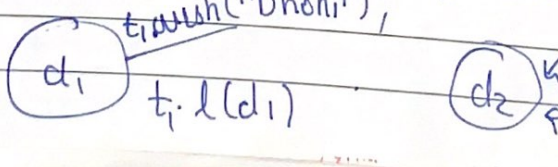
⇒ the Non, syncronized method comes for execution it can execute the m3(),

Note :- only the nonsyncronized method can directly execute otherwise
⇒ for the Syncronized method it has to wait for other thread to relase the locked object.
⇒ the Jvm is responsible to lock & relase the lock after execution.



if we are changing the state of object then we use Syncronize Method i.e State of obj changing [add, update, delte, repla

⇒ if we are doing read only operation then non-syncroni is used.
⇒ if multiple thread are accessing same obj then synchroni is
★ Case Study :- if there are 2 Objects then 2 Seprate thread can access it simultaneously and we get mis output. (even if we access syndronized method)



A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

☆ if their are two diff objects, and the diff Threads are accessing the method, synchronized() then their is irregular o/p

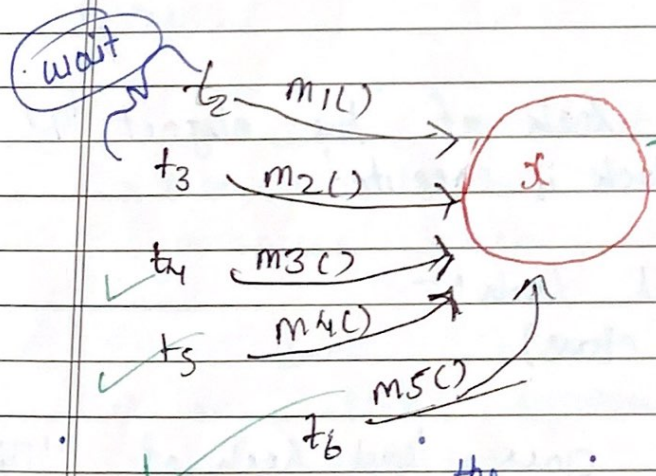
⇒ But if we declare the method static synhronized then their is class level Lock, we will get regular output.

☆ Class level Lock :-

⇒ Every class in java has unique Lock i.e class level Lock.

→ if thread wants to execute Static Synichronized method, then thread require class level Lock.

→ Once it gets class lvel Lock, then it can execute any static synchronized(), and on compe the execution, it realses the Lock.



t1 → cl(x)
t1 m1()

```
m1()
{
}
```

```
class x
{
Static Syn m1()
Static Syn m2()
  Syncr m3()
  Static m4()
  m5()
```

not

In above, ~~as~~ ~~two~~ the Static synchronized(), can't accessed ~~same~~ by the other thread, if the thread is locked by class Lock m1 & m2(), m1 is locked, if the Syn(), then the other thread can access as it in a object level acces, Jum thedas

m(){ ==
{ 10 lines
}
}

☆ Synchronized block :-

⇒ In the above example if there are 10k lines of code and only 10 line require synchronization then it is bad practice to declare entire block as synchronize.
⇒ We can use synchronize block in such case.

☆ Different methods to declare synchronized block

1] To get lock of current object
```
Synchronized (this)
  { ... }
```
If a thread gets the lock of current object then only allowed to execute this area.

2] To get lock of particular object 'b' :-
```
Synchorized (b)
  { ... }
```
If a thread gets lock of the object 'b' then only the block is executed

3] To get class level lock :-
```
Sychronized (object.class)
  { ... }
```
If a thread gets class level lock of "Display" class, then only it is allowed to execute this Area

☆ We do not have primitive lock on the synchronized block, i.e
```
int x=10;
Synchronized(x) // Error :- Compile time error found : int required reference
```

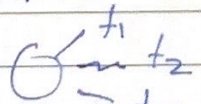
A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

★ Synchronized most common questions?

1] What is Synchronized keyword & where we can Apply?
⇒ Synchronized is modifier, applicable for class, method, object, not for keyword

2] Advantages of Synchronized?
→ To avoid data inconstency.

3] disadvantage → increasing the waiting time
& Performance prblm

4] ⇒ Race condition? → 

⇒ If multiple thread access the same object then we use Synchronized keyword. (this is called race condition) use Synchroized keyword

5] what is object Lock & when it is required?
⇒ whereever a thread wants to execution Synchronized method.

6] Class Lock & when it is required?
⇒ whenever thread wants to execute static Synchronized block,

7] Class Levels & object level Lock?
⇒ for static Synchronized block class level Lock is required
⇒ for Synchronized method, block object level Lock is required.

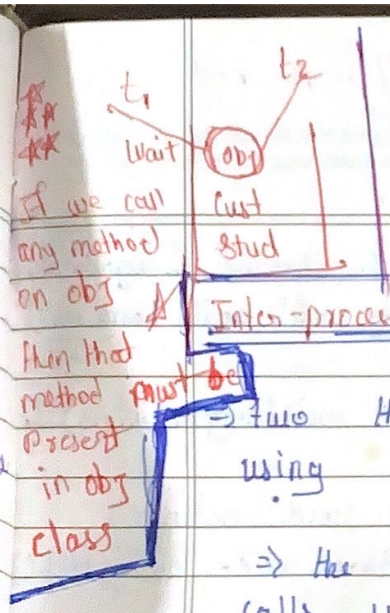
☆ Using Synchronized block vs Syncronized() is to increase the performance.

☆ thread cannot accquire multiple Lock simultaneously

```
Class X
{
Public Syncronized void m1()  Lock of X
  {Y y = new Y();              Lock of X,Y
   Synchronized(y)
   {
    Z z=new Z();               Lock of X,Y,Z
    Synchronized(z);
    {
    }
   }
  }
```

☆ Synchronized Statement :- (Interview people term) the Synchronized block (method, that how lines those are present in it called Synchronized statement

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.



t1.start()
t1.join()
They are in thread class.

obj.o
o.m1()
Student
Customer

To call the method it should be in that class

Thread can call wait mthd on any obj so it is present in the obj class

Inter-process communication :-

⇒ Two thread can communicate with each other using wait(), notify(), notifyAll().

⇒ The thread which is waiting for updation calls wait() & the thread which updates calls notify(), notifyall(), & waiting thread will get notification and continue its execution with those updated method.

⇒ wait(), notify(), notifyall(), are present in object class, but not in thread class, thread can call these method on any java object.

Note :- 3 pts to call wait(), all mthds

To call wait(), notify(), notifyall() on any obj
(1) thread should be owner of the object (2) i.e thread must have lock of that obj, i.e (3) thread should be in synchronized area,
Hence we call all method only from synchronized area
⇒ otherwise will get R.E IllegalMonitorStateException

⇒ if thread calls wait() on any obj it immediately realeses the lock of the particular obj and enter in waiting state.

Except wait(), notify(), notifyall(), thread dont realse lock, but for this it immediatly realeses the lock.

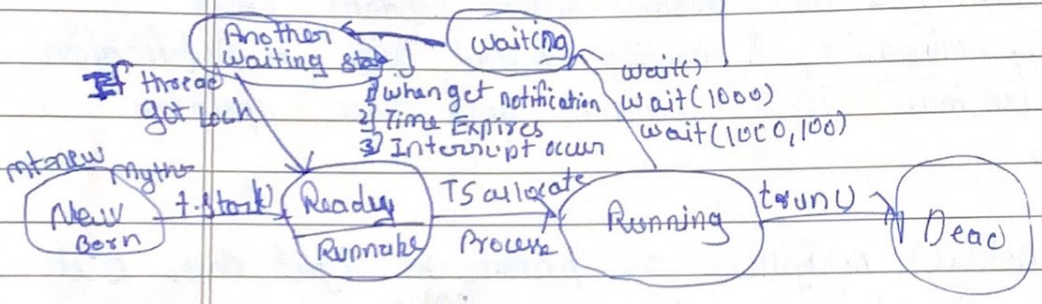
for all multithreading methods

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001

* Notify() may not release the lock not immediately.

Method	is Relase Lock.
yield()	NO
Join()	No
Sleep()	No
wait()	Yes
notify()	Yes
notifyAll()	Yes

final
Public void wait() throws InterruptedException
public final native void wait(Long ms) throws IE
Public final void wait(Long ms, int ns)
public final native void notify()
public final native void notifyAll()



* if main thread goes into sleep state then the child will get the Procession for execution and execute but after, the notify, it is dead, so no one to give notification
main { Thread.sleep(100); } // child has { Sync(this) this.notify(); }
=> so main thread will wait infinetly :-
So to avoid, we do in main method
t.wait(10000); // to if it does not get notified it will still execute.

Producer - consumer prb

class Produce Thre {

Producer Thread
produce()
{ Synchronized(q)
{ produce items to queue
q.notify()
} }

object
DRP DD
queue

consumer Thread
class ConsumerThread
{ consume()
{ Synchronized(q)
{ if(q is empty)
q.wait();
else
consume item
} } }

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

Difference between notify() and notifyAll()

- we can use notify() only to notify for single waiting thread, suppose there are n threads waiting then we have to use notifyAll(); in notify() we cannot tell which order they will be notified, if multiple thread are waiting for it.
- To give notification to multiple thread we use notifyAll(). of particular object of all waiting threads

obj.notifyAll(); → obj.wait() = 60 thread
obj2.wait() => 40 thread
So only obj. all thread will be called

=> Even if we notify all thread at time but execution will begun one by one. (thread).

=> if we call wait() on another object then it is invalid, because we have to call wait() on same object we have lock on

Sat s1=new sat();
Sat s2=new sat();

Invalid: synchronized(s1) { s2.wait() } // IllegalMonitorStateException

HasLock of s1

synchronized(s1) { s1.wait() } => valid

Deadlock & Starvation

=> Deadlock means two threads waiting for each other permanently,

=> Starvation means it would get chance after very long time.

10th thread - t1.setPriority(1)
~~t1.setPriority(10)~~ => after all the above will high priority

Scholar id
550420@ss
OS

A.I.S.S.M.S
INSTITUTE OF INFORMATION TECHNOLOGY
Kennedy Road, Near R.T.O., Pune - 411 001.

★ ~~Daemon~~ Daemon threads (low priority thread)
→ the thread i.e executing in background,
Ex. GC, Signal dispatcher, Action
→ main objective is to provide support for Non-daemon threads,
→ if Main thread runs with low memory then JVM run garbage collector to destory useless objects

① public boolean isDaemon()
② public void setDaemon(boolean b)
if thread started and we set the daemon true then IllegalThreadStateException.

⇒ It is impossible to change the daemon nature of main, it is already started by JVM, so we cannot change
⇒ whenever Last Non Daemon thread terminates all Daemon thread will be terminated irrespective of position.

★ Green thread :~ — Completely managed by JVM without using OS support

Two models
- Green thread → Sun Solaris
- native OS thread → window based OS → JVM with OS support

t.start(), t.suspend(), t.resume(); ⇒ get warning
⇒ javac -Xlint:deprecation API TestJav
We can stop the current executing state into Dead, & the stop() is depricated method.
The multithreading is used using 2 modle.
1) Green thred, 2) native OS thread