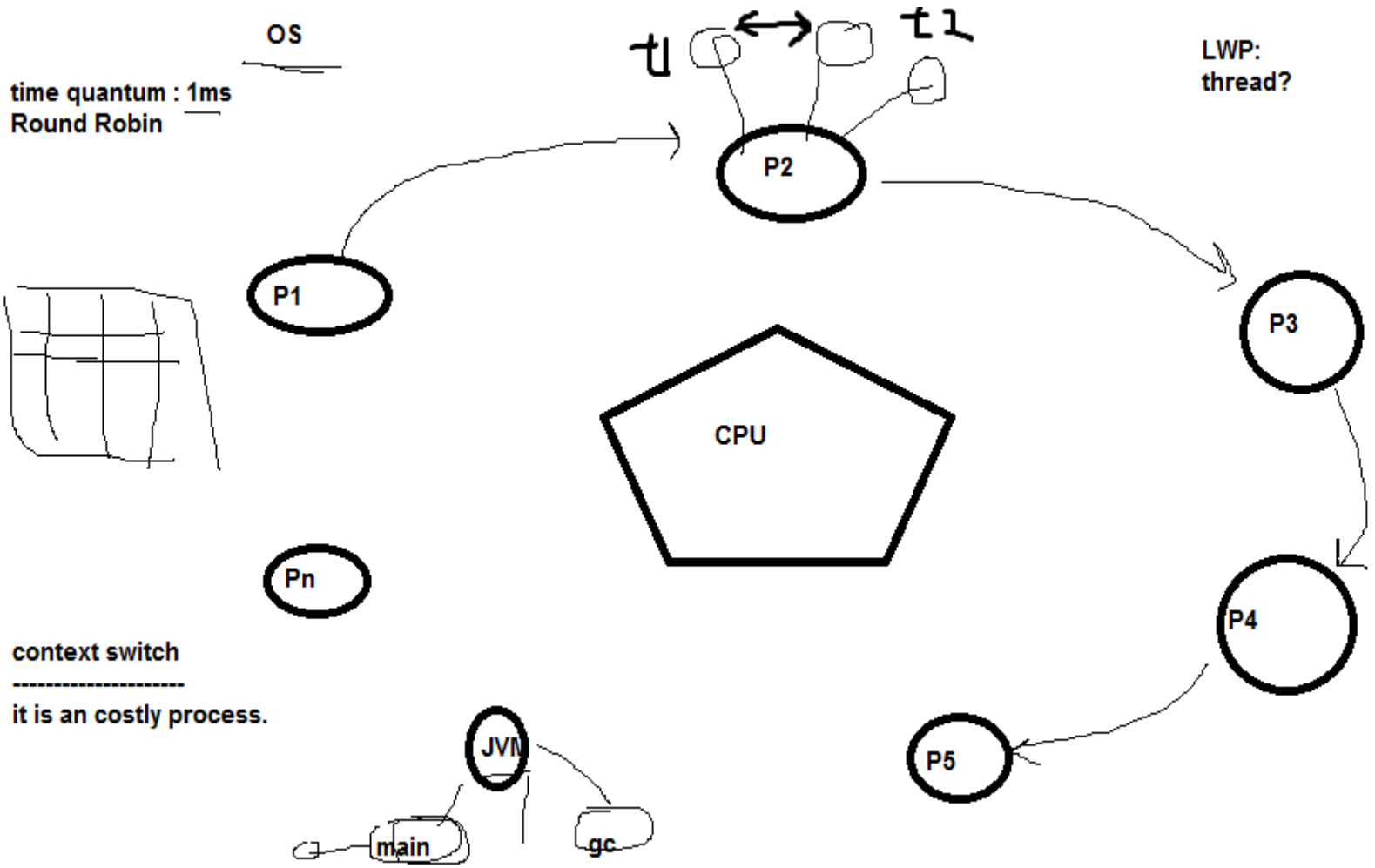
DAY -7

Day 5: Introduction to Java threads, thread life cycle, synchronization, dead lock

- Program vs process
- Thread as LWP
- Creating and running thread
- Thread life cycle
- Need of synchronization
- Producer consumer problem
- dead lock
- Hands on & Lab

What is threads? LWP



rgupta.mtech@gmail.com

Basic fundamentals

- Thread: class in java.lang
- thread: separate thread of execution

What happens during multithreading?

1. JVM calls main() method
2. main() starts a new thread. Main thread is tempory frozen while new thread start running so JVM switch between created thread and main thread till both complets

rgupta.mtech@gmail.com

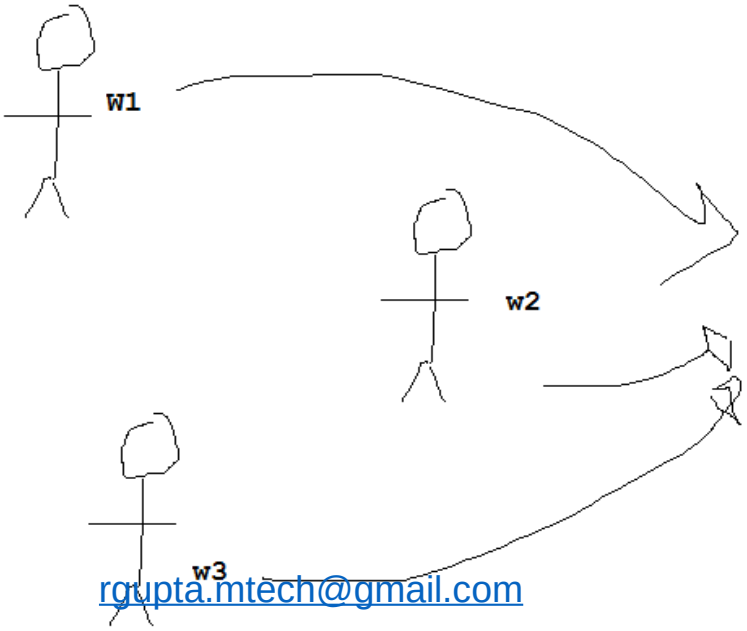
Creating threads in Java?

- Implements Runnable interface
- Extending Thread class……..

- Job and Worker analogy…

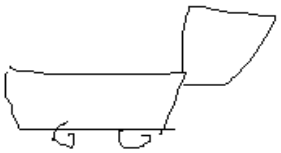
```
class Job implements Runnable{


    public void run() {
        // TODO Auto-generated method stub
    }

}
```

```
class MyThread extends Thread{
    public void run() {
        // TODO Auto-generated method stub
    }
}
```



Thread
----
consider object of threads as worker

Implementation of Runnable as Job



simulation

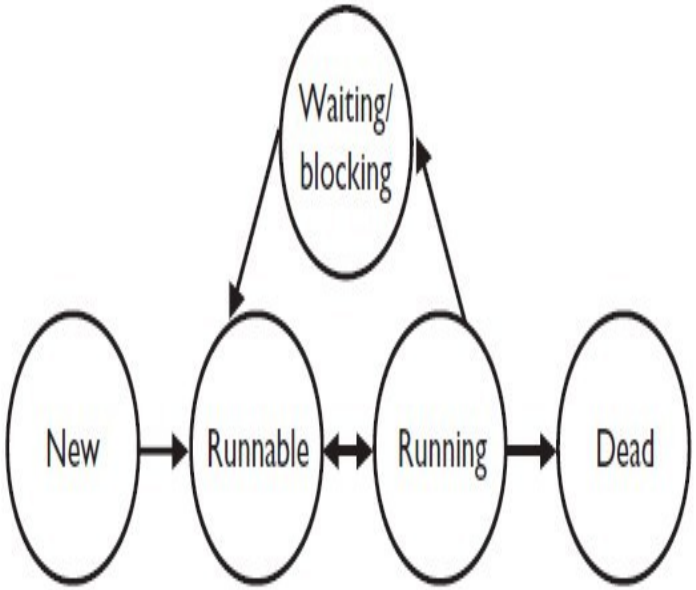
Sleep()

Job

rgupta.mtech@gmail.com

Thread life cycle…

- If a thread is blocked (put to sleep) specified no of ms should expire
- If thread waiting for IO that must be over..
- If a threadcalls wait() then another thread must call notify() or notifyAll()
- If an thred is suspended() some one must call resume()



rgupta.mtech@gmail.com

Java.lang.Thread

- Thread()
  - construct new thread
- void run()
  - must be overriden
- void start()
  - start thread call run method
- static void sleep(long ms)
  - put currently executing thread to sleep for specified no of millisecond
- boolean isAlive()
  - return true if thread is started but not expired
- void stop()
- void suspend() and void resume()
  - Suspend thread execution….

rgupta.mtech@gmail.com

Java.lang.Thread

- void join(long ms)
  - Main thread Wait for specified thread to complete or till specified time is not over
- void join()
  - Main thread Wait for specified thread to complete
- static boolean interrupted()
- boolean isInterrupted()
- void interrupt()
  - Send interrupt request to a thread

rgupta.mtech@gmail.com

Creating Threads

- By extending Thread class

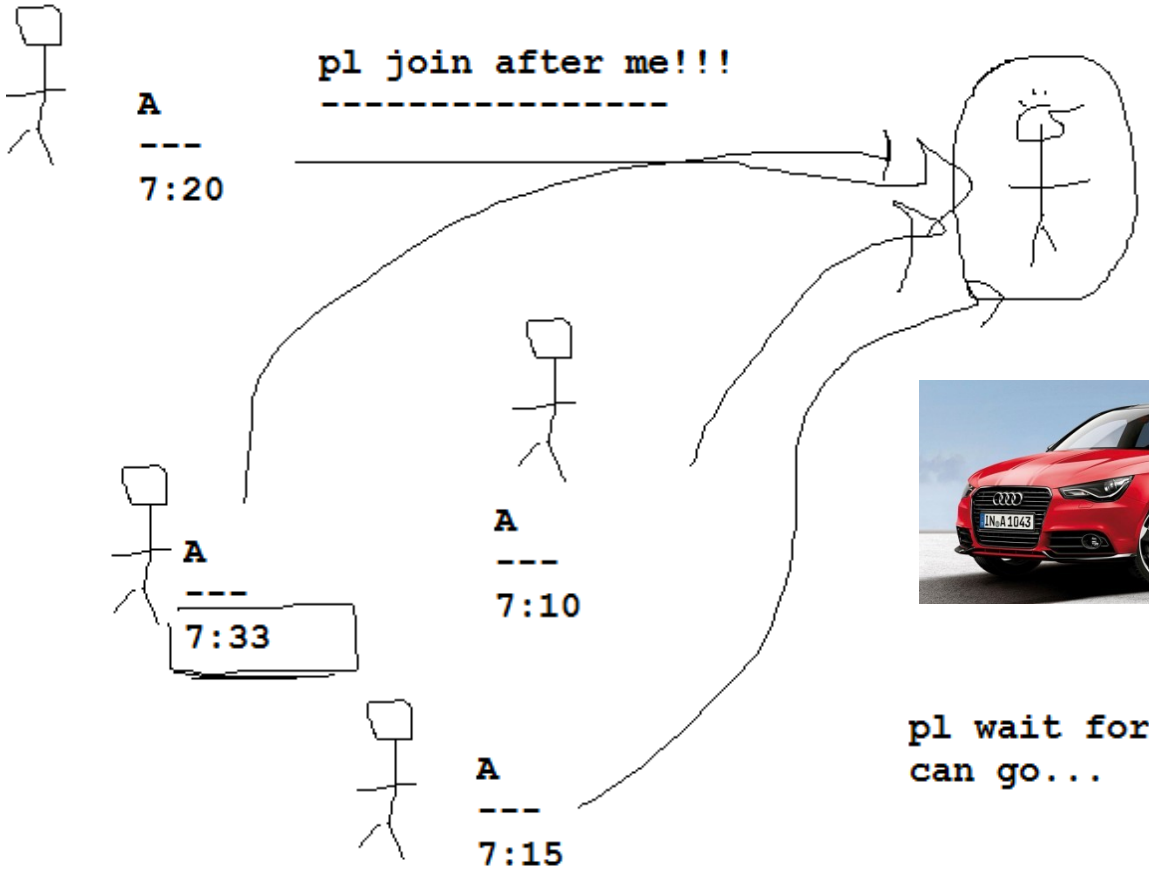
```
class MyThread extends Thread
{
    @Override
    public void run()
    {

    }

}
```

- Implementing the Runnable interface

```
class MyRunnable implements Runnable
{

    @Override
    public void run()
    {

    }

}
```

rgupta.mtech@gmail.com

Understanding join() method



pl wait for me for 5 min else u can go...

rgupta.mtech@gmail.com

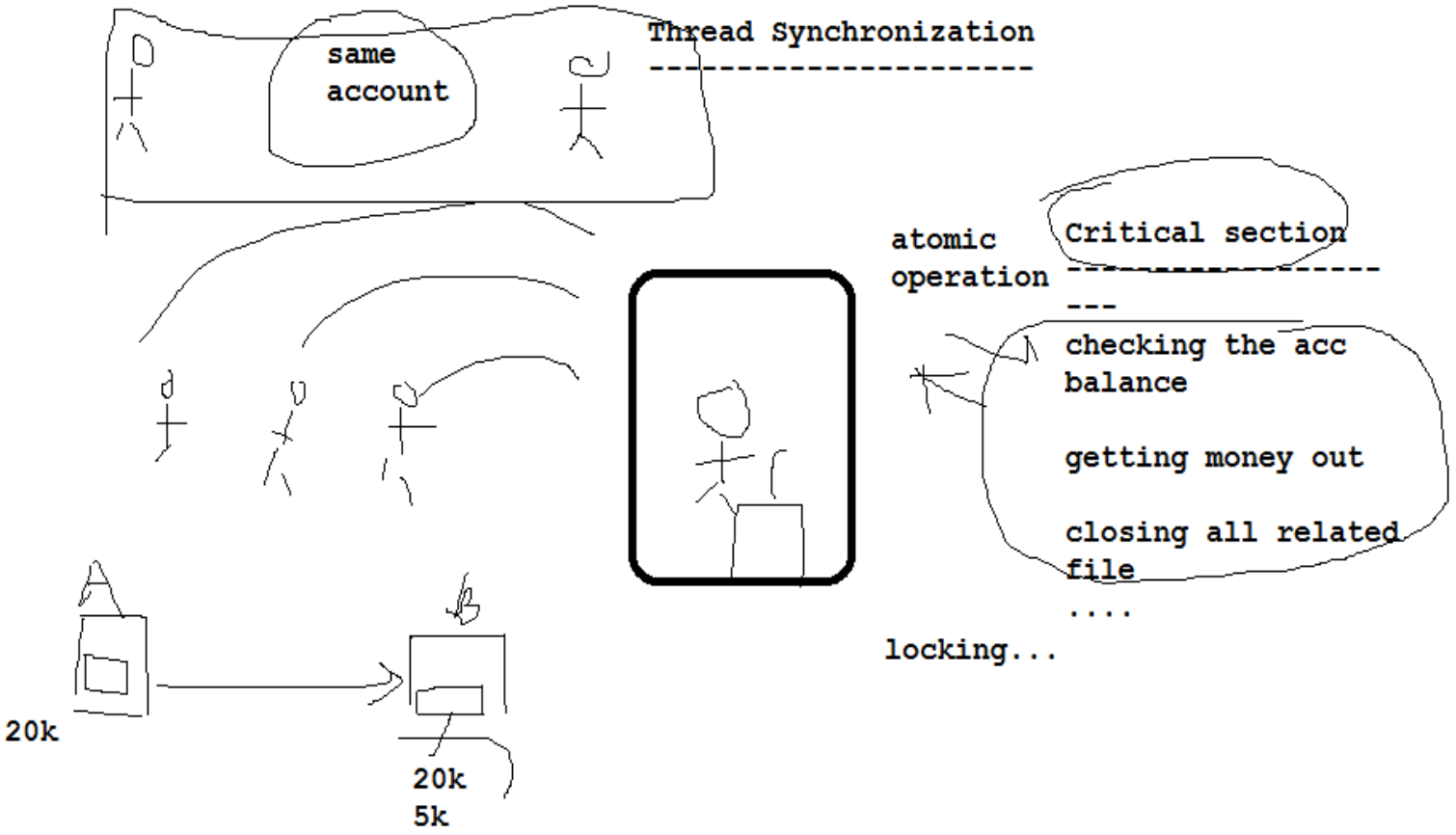
Checking thread priorities

```
class Clicker implements Runnable
{
     int click=0;
     Thread t;
     private volatile boolean running=true;
     public  Clicker(int p)
     {
          t=new Thread(this);
          t.setPriority(p);
     }
     public void run()
     {
          while(running)
               click++;
     }
     public void stop()
     {
          running=false;
     }
     public void start()
     {
          t.start();
     }
}
```

rgupta.mtech@gmail.com

```
.....
Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
Clicker hi=new Clicker(Thread.NORM_PRIORITY+2);
Clicker lo=new Clicker(Thread.NORM_PRIORITY-2);
lo.start();
hi.start();
try
{
     Thread.sleep(10000);
}
catch(InterruptedException ex){}
lo.stop();
hi.stop();
//wait for child to terminate
try
{
     hi.t.join();
     lo.t.join();
}
catch(InterruptedException ex)
{

}
System.out.println("Low priority thread:"+lo.click);
System.out.println("High priority thread:"+hi.click);
....
....
```

Understanding thread synchronization



rgupta.mtech@gmail.com

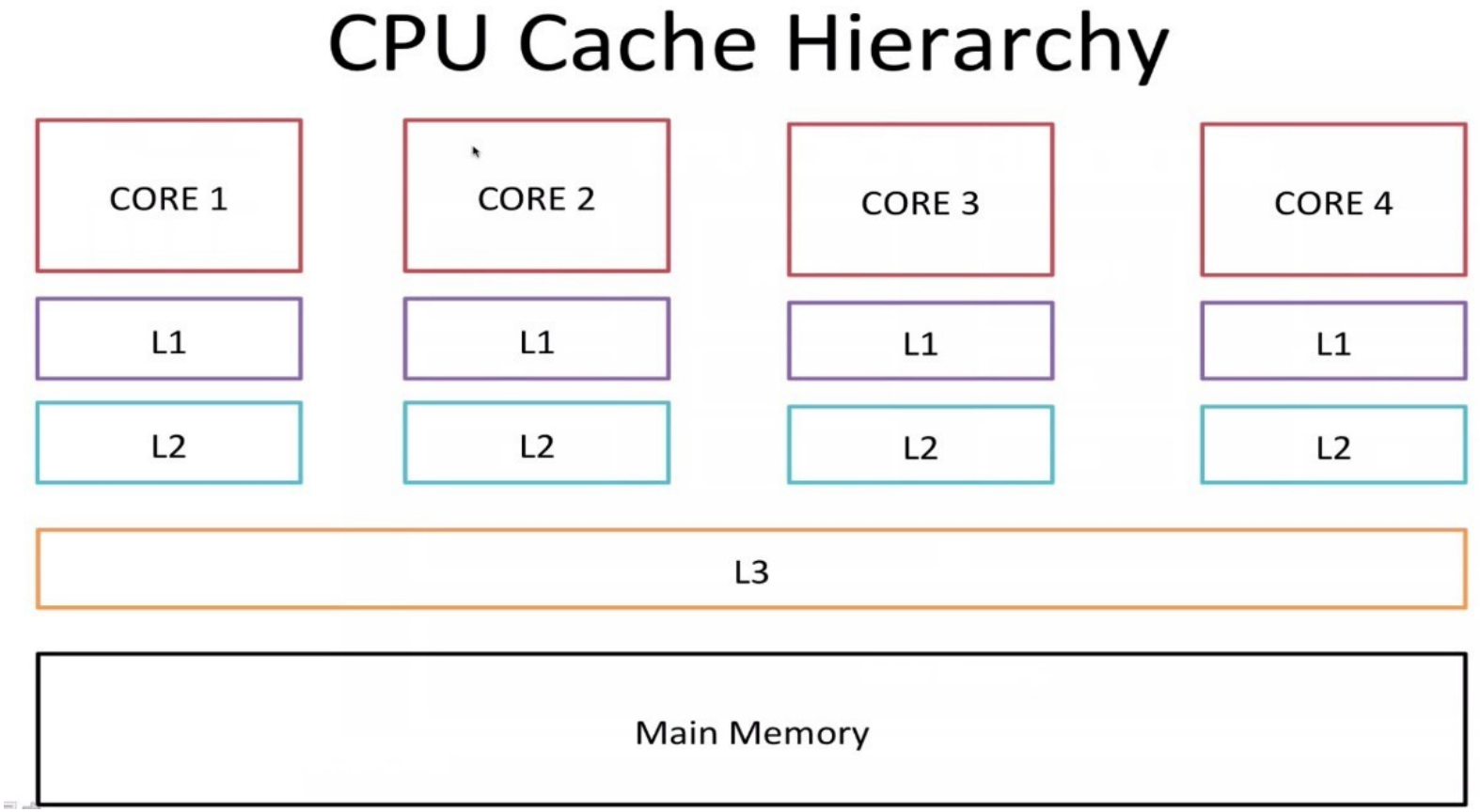
Synchronization

- Synchronization
  - Mechanism to controls the order in which threads execute
  - Competition vs. cooperative synchronization

- Mutual exclusion of threads
  - Each synchronized method or statement is guarded by an object.
  - When entering a synchronized method or statement, the object will be locked until the method is finished.
  - When the object is locked by another thread, the current thread must wait.

rgupta.mtech@gmail.com

volatile



rgupta.mtech@gmail.com

Using thread synchronization

```
class CallMe
{
     synchronized void call(String msg)
     {
          System.out.print("["+msg);
          try
          {
               Thread.sleep(500);
          }
          catch(InterruptedException ex){}
          System.out.println("]");
     }
}
```

```
class Caller implements Runnable
{
     String msg;
     CallMe target;
     Thread t;
     public Caller(CallMe targ,String s)
     {
          target=targ;
          msg=s;
          t=new Thread(this);
          t.start();
     }
     public void run()
     {
          target.call(msg);
     }
}
```

```
Caller ob1=new Caller(target, "Hello");
Caller ob2=new Caller(target,"Synchronized");
Caller ob3=new Caller(target,"Java");
```

rgupta.mtech@gmail.com

Inter thread communication

- Java have elegant Interprocess communication using wait() notify() and notifyAll() methods
- All these method defined final in the Object class
- Can be only called from a synchronized context

rgupta.mtech@gmail.com

wait() and notify(), notifyAll()

wait()

- Tells the calling thread to give up the monitor and go to the sleep until some other thread enter the same monitor and call notify()

notify()

- Wakes up the first thread that called wait() on same object

notifyAll()

- Wakes up all the thread that called wait() on same object, highest priority thread is going to run first

rgupta.mtech@gmail.com

Incorrect implementation of produce consumer ...

```
class Q
{
        int n;
        synchronized int get()
        {
            System.out.println("got:"+n);
            return n;
        }
        synchronized void put(int n)
        {
            this.n=n;
            System.out.println("Put:"+n);
        }
}
```

```
class Producer implements Runnable
{
    Q q;
    public Producer(Q q) {
        this.q=q;
        new Thread(this,"Producer").start();
    }
    public void run()
    {
        int i=0;
        while(true)
            q.put(i++);
    }
}
```

```
public class PandC {
public static void main(String[] args) {
    Q q=new Q();
    new Producer(q);
    new Consumer(q);
    System.out.println("ctrol C for exit");
}
}
```

```
class Consumer implements Runnable
{
    Q q;
    Consumer(Q q)
    {
        this.q=q;
        new Thread(this,"consumer").start();
    }
    public void run()
    {
        while(true)
            q.get();
    }
}
```

rgupta.mtech@gmail.com

Correct implementation of produce consumer ...

```
class Q
{   int n;
        boolean valueSet=false;
        synchronized int get()
        {
            if(!valueSet)
                try
                {
                    wait();
                }
            catch(InterruptedException ex){}
            System.out.println("got:"+n);
            valueSet=false;
            notify();
            return n;
        }

    synchronized void put(int n)
    {
        if(valueSet)
            try
            {
                wait();
            }
        catch(InterruptedException ex){}
        this.n=n;
        valueSet=true;
        System.out.println("Put:"+n);
        notify();
    }
}
```

rgupta.mtech@gmail.com