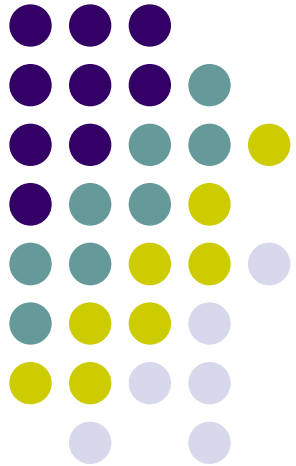
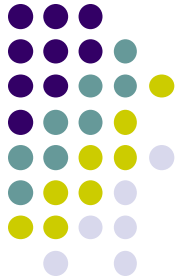
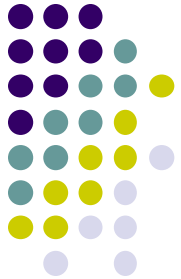
ITMC403 Parallel and Distributed Computing

Consumer Producer

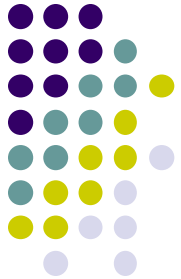




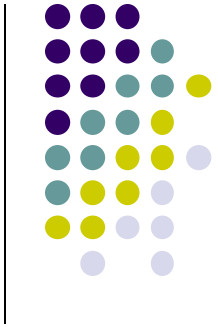
```
class PrintJava
{
        public static void main(String args[])
        {
                        Q q = new Q();

                        new Producer( q );

                        new Consumer( q );

                        System.out.println("Press Control-C to  stop.");
        }
}
```



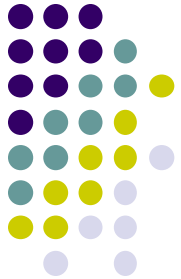
```
class Producer implements Runnable
{
        Q q;
        Producer(Q q)
        {
                this.q = q;
                new Thread(this, "Producer").start();
        }

        public void run()
        {
                int i = 0;
                while(true)
                {
                q.put(i++);
                }
        }
}
```



```
class Consumer implements Runnable
{
        Q q;

        Consumer(Q q)
        {
                this.q = q;
                new Thread(this, "Consumer").start();
        }

        public void run()
        {
                while(true)
                {
                        q.get();
                }
        }
}
```

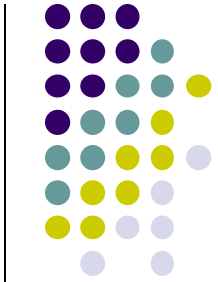


```
class Q
{
        int n;
        boolean valueSet = false;

        synchronized int get()
        {
            if(!valueSet)
                    try
                    {
                            wait();
                    }
                    catch(InterruptedException e)
                    {
                       System.out.println(" InterruptedException caught");
                    }
                    System.out.println("Got: " + n);
                    valueSet = false;
                    notify();
                    return n;
    }
```



```
        synchronized void put(int n)
        {


                if(valueSet)
                try
                {
                        wait();
                }
                catch(InterruptedException e)
                {
                        System.out.println("InterruptedException caught");
                }
                this.n = n;
                valueSet = true;
                System.out.println("Put: " + n);
                notify();
        }
}
```

Do we need a Thread "manager"? Class Discussion



- If your code is responsible for creating a several tasks, linking them with Threads, and starting them all, then you have things to worry about:
  - What if you start too many threads? Can you manage the number of running threads?
  - Can you shutdown all the threads?
  - If one fails, can you restart it?