Q Producer Consumer problem

Source Code:

```
class Q
{
    int n;
    boolean valueSet = false;
    synchronized int get()
    {
        while(!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)
    {
        while(valueSet)
        try
        {
            wait();
        }
        catch(InterruptedException e)
        {
            System.out.println("Interrupted
```

```
            Exception caught");
        }
        this.n = n;
        valueSet = true;
        System.out.println("Put: " + n);
        notifyAll();
    }
}
class Producer implements Runnable
{
    Q q;
    Producer(Q q)
    {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run()
    {
        int i = 0;
        while(i < 15)
        {
            q.put(i++);
        }
    }
}
class Consumer implements Runnable
{
    Q q;
    Consumer(Q q)
    {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
}
```

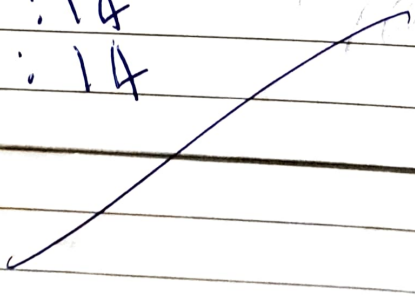
```
public void run()
{
    int i=0;
    while (i<15)
    {
        int n= i.get();
        i++;
    }
}
class TestSynchronization1
{
    public static void main(String args[])
    {
        Q q= new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

Output:

```
Put: 0
Got: 0
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4
```

Port: 5
Prot: 5
Port: 6
Prot: 6
Port: 7
Prot: 7
Port: 8
Prot: 8
Port: 9
Prot: 1
Port: 10
Prot: 10
Port: 11
Prot: 11
Port: 12
Prot: 12
Port: 23
Prot: 23
Port: 14
Prot: 14

06/2/2024



Q. Write a java program to demonstrate the deadlock.

Source Code:

```
class A
{
    synchronized void foo(B b)
    {
        System.out.println(Thread.currentThread().getName()
        + " is accessing B.foo()");
        try
        {
            Thread.sleep(1000);
        }
        catch(InterruptedException e)
        {
            System.out.println("A interrupted");
        }
        System.out.println("Trying to access B.last()");
        b.last();
    }
    synchronized void last()
    {
        System.out.println("Inside A.last()");
    }
}
class B
{
    synchronized void bar(A a)
    {
        System.out.println(Thread.currentThread().getName()
        + " is accessing B.foo()");
```

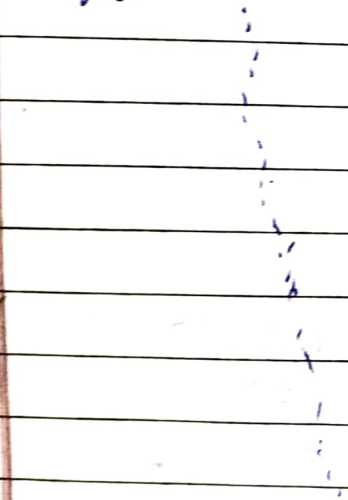
```
try
{
    Thread.sleep(1000);
}
catch(InterruptedException e)
{
    System.out.println("B interrupted");
}
System.out.println("Trying to access
    A.last()");
A.last();
}
synchronized void last()
{
    System.out.println("Inside B.last()");
}
}
class DeadLock_Main_Program implements Runnable
{
    A a = new A();
    B b = new B();
    DeadLock_Main_Program()
    {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in mainThread");
    }
    public void run()
    {
        b.bar(a);
        System.out.println("Back in otherThread");
    }
```

```
public static void main(String args[])
{
    new Deadlock_main_Program());
}
}
```

Output:

~~Running Thread is calling B.last()~~

mainThread is calling A.foo()
Running Thread is calling B.last()
Trying to call A.last
Trying to call B.last



20/12/2024