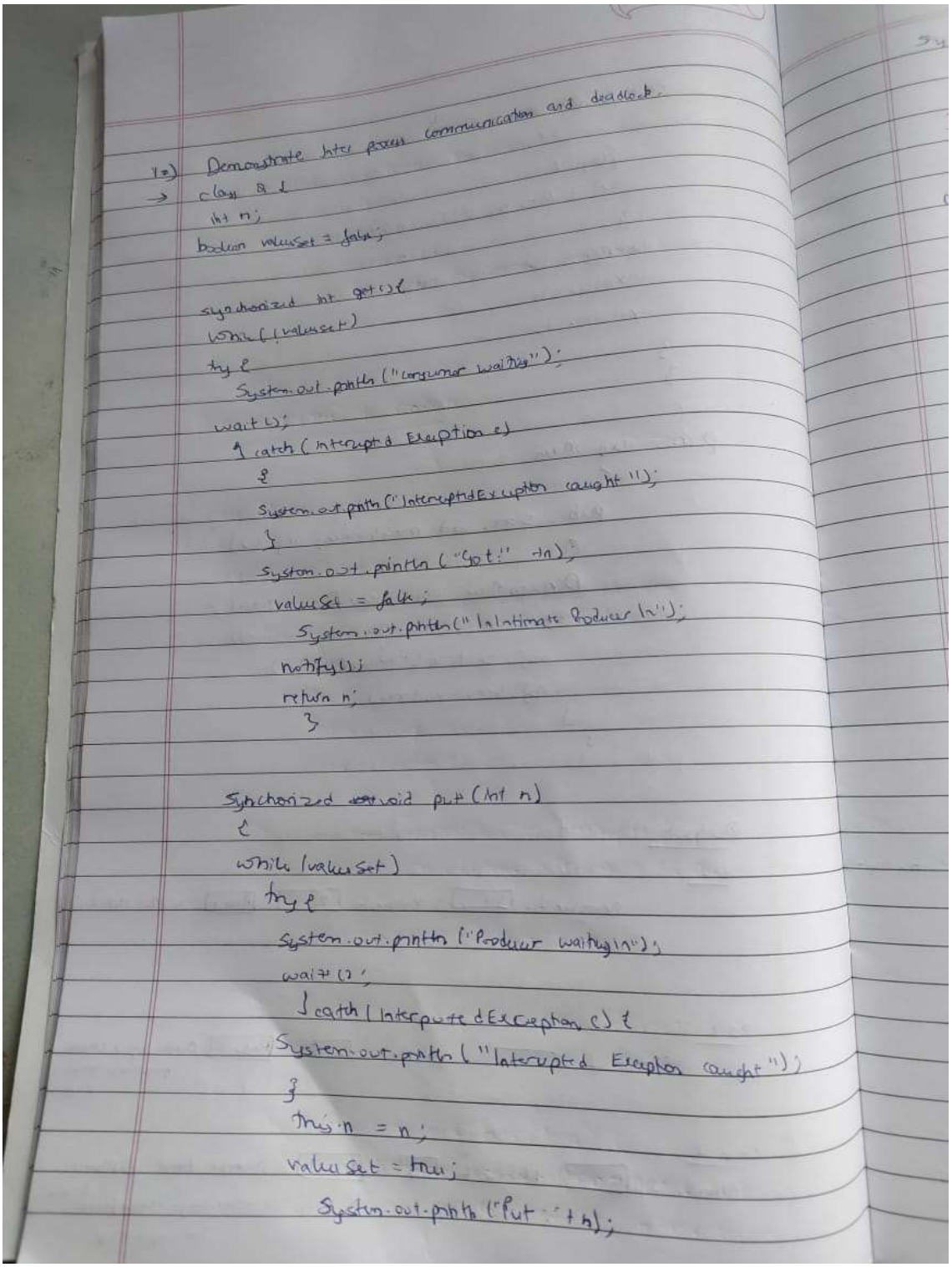
Program 10

Demonstrate Inter process Communication and deadlock



1=) Demonstrate inter process communication and deadlock.

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

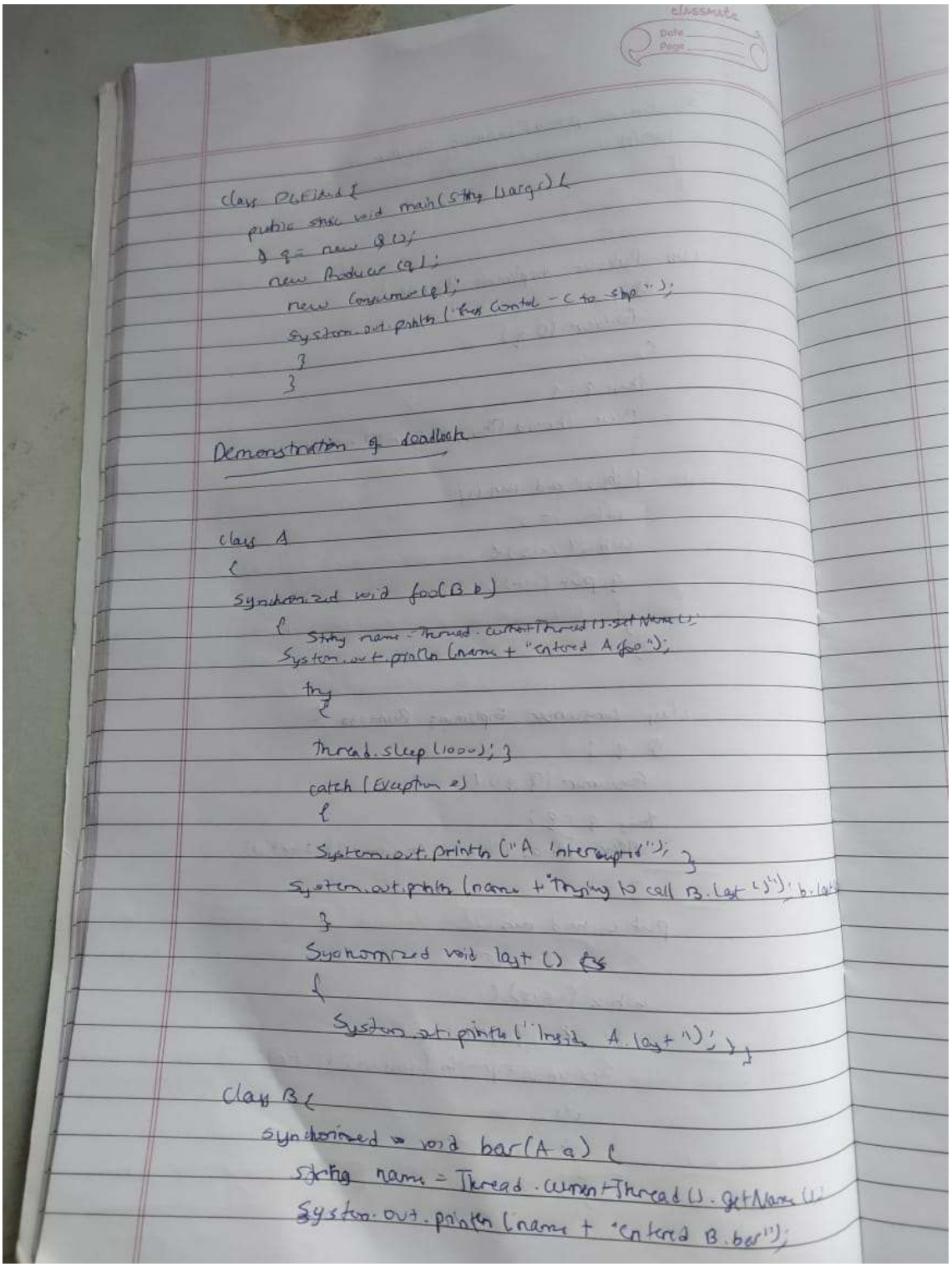
synchronized int get() {
while(!valueSet)
try {
System.out.println("consumer waiting");
wait();
} catch (InterruptedException e)
{
System.out.println("InterruptedException caught");
}
System.out.println("Got:" +n);
valueSet = false;
System.out.println("\nIntimate Producer\n");
notify();
return n;
}

Synchronized void put(int n)
{
while(valueSet)
try {
System.out.println("Producer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("Interrupted Exception caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " +n);

```
System.out.println("Interrupted Consumer");
notify();
}
}

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 r = q.get();
System.out.println("Consumed:"+r);
i++;
}
}
}
```

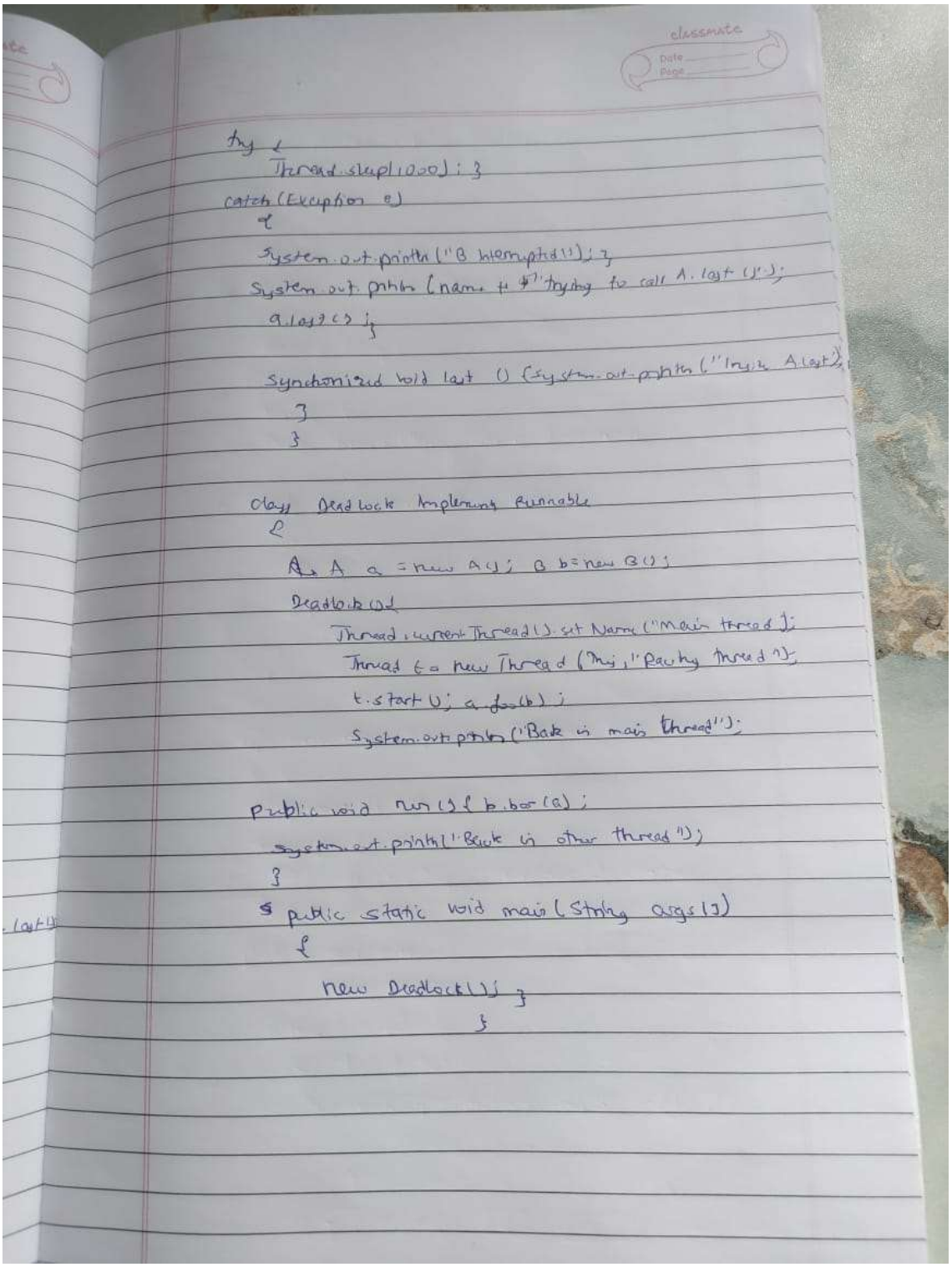


```
class PCFixed {
public static void main(String[] args) {
Q q = new Q();
new Producer(q);
new Consumer(q);
System.out.println("Press Control - C to stop");
}
}
```

Demonstration of deadlock

```
class A
{
Synchronized void foo(B b)
{
String name = Thread.currentThread().getName();
System.out.println(name + "entered A.foo");
try
{
Thread.sleep(1000);}
catch (Exception e)
{
System.out.println("A interrupted");}
System.out.println(name + "Trying to call B.last()");b.last();
}
Synchronized void last() {
{
System.out.println("Inside A.last");}}

class B{
synchronized void bar(A a) {
String name = Thread.currentThread().getName();
System.out.println(name + "entered B.bar");
```

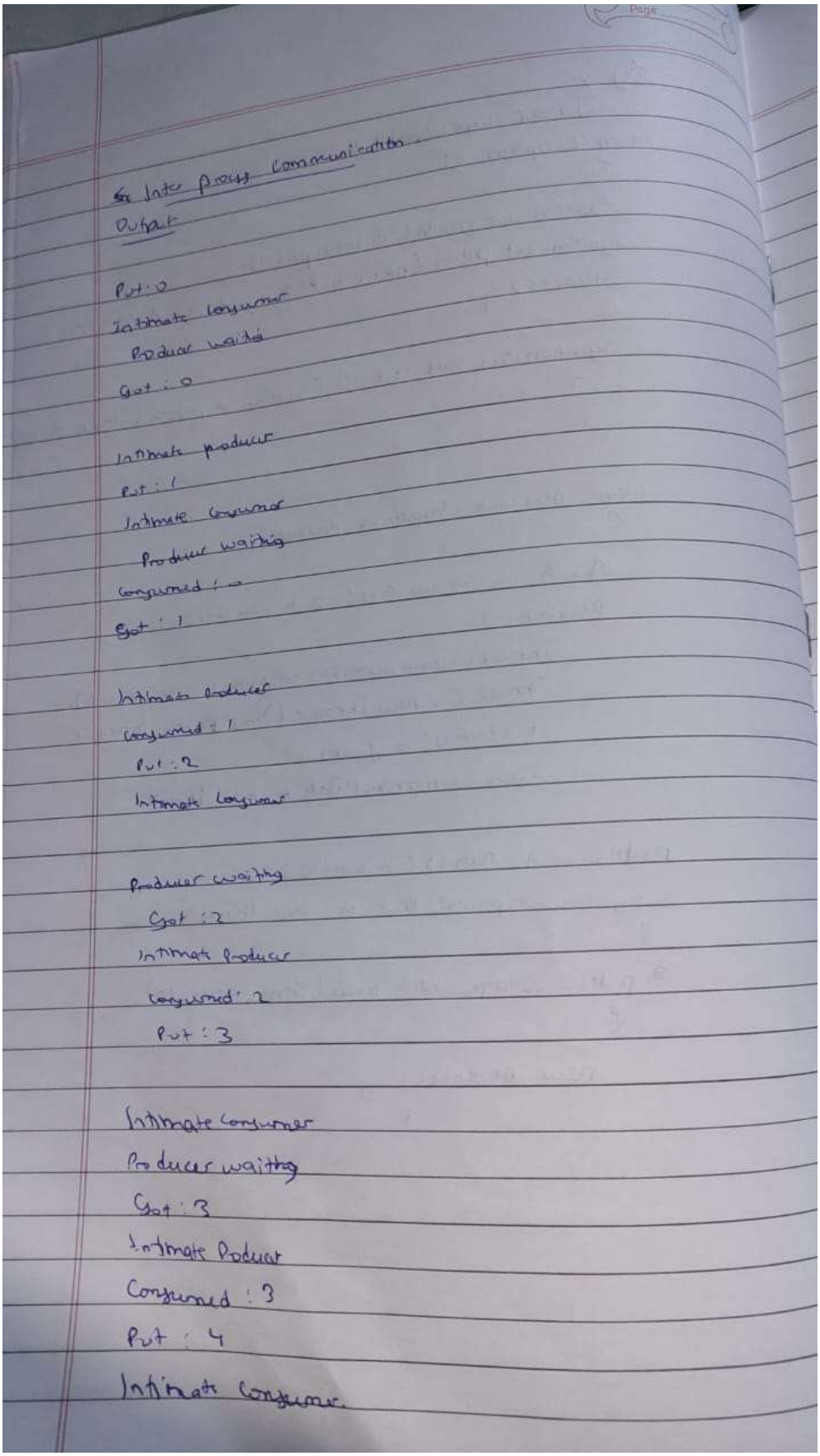


try {
Thread.sleep(1000); }
catch (Exception e)
{
System.out.println("B interrupted"); }
System.out.println(name + "trying to call A.last()");
a.last();
}

synchronized void last() {System.out.println("In a.last");
}
}

class Deadlock implements Runnable
{
A a = new A(); B b = new B();
Deadlock()
Thread.currentThread().setName("Main thread");
Thread t = new Thread(this, "Racing thread");
t.start(); a.foo(b);
System.out.println("Back in main thread");

public void run() { b.bar(a);
System.out.println("Back in other thread");
}
public static void main(String args[])
{
new Deadlock(); }
}


Inter process Communication

Output

Put: 0

Intimate Consumer

Producer waiting

Got: 0

Intimate producer

Put: 1

Intimate Consumer

Producer waiting

Consumed: 0

Got: 1

Intimate Producer

Consumed: 1

Put: 2

Intimate Consumer

Producer waiting

Got: 2

Intimate Producer

Consumed: 2

Put: 3

Intimate Consumer

Producer waiting

Got: 3

Intimate Producer

Consumed: 3

Put: 4

Intimate Consumer.

Deadlock program

output-

Main Thread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()

//inter process communication

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

   synchronized int get() {
      while (!valueSet)
         try {
            System.out.println("\nConsumer waiting\n");
            wait();
         } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
         }
      System.out.println("Got: " + n);
      valueSet = false;
      System.out.println("\nIntimate Producer\n");
      notify();
      return n;
   }

   synchronized void put(int n) {
      while (valueSet)
         try {
            System.out.println("\nProducer waiting\n");
            wait();
         } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
         }
      this.n = n;
      valueSet = true;
      System.out.println("Put: " + n);
      System.out.println("\nIntimate Consumer\n");
      notify();
   }
}

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 r = q.get();
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}

class sync {
    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

// deadlock

```
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.foo");
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("A Interrupted");
        }
        System.out.println(name + " trying to call B.last()");
        b.last();
    }

    synchronized void last() {
        System.out.println("Inside A.last");
    }
}

class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");
        try {
            Thread.sleep(1000);
```

```
        } catch (Exception e) {
            System.out.println("B Interrupted");
        }
        System.out.println(name + " trying to call A.last()");
        a.last();
    }

    synchronized void last() {
        System.out.println("Inside B.last");
    }
}

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a);
        System.out.println("Back in other thread");
    }

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

```
C:\Users\shett\OneDrive\Documents\javaclasslab>javac sync.java

C:\Users\shett\OneDrive\Documents\javaclasslab>java sync
Press Control-C to stop.
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer


Producer waiting

Consumed: 0
Got: 1

Intimate Producer

Consumed: 1
Put: 2

Intimate Consumer


Producer waiting

Got: 2

Intimate Producer

Consumed: 2
Put: 3

Intimate Consumer


Producer waiting

Got: 3

Intimate Producer

Consumed: 3
Put: 4

Intimate Consumer
```

```
Consumed: 3
Put: 4

Intimate Consumer


Producer waiting

Got: 4

Intimate Producer

Consumed: 4
Put: 5

Intimate Consumer


Producer waiting

Got: 5

Intimate Producer

Consumed: 5
Put: 6

Intimate Consumer


Producer waiting

Got: 6

Intimate Producer

Consumed: 6
Put: 7

Intimate Consumer


Producer waiting

Got: 7

Intimate Producer

Consumed: 7
Put: 8

Intimate Consumer


Producer waiting
```

```
Intimate Producer

Consumed: 7
Put: 8

Intimate Consumer


Producer waiting

Got: 8

Intimate Producer

Consumed: 8
Put: 9

Intimate Consumer


Producer waiting

Got: 9

Intimate Producer

Consumed: 9
Put: 10

Intimate Consumer


Producer waiting

Got: 10

Intimate Producer

Consumed: 10
Put: 11

Intimate Consumer


Producer waiting

Got: 11

Intimate Producer

Consumed: 11
Put: 12

Intimate Consumer
```

```
Intimate Consumer


Producer waiting

Got: 11

Intimate Producer

Consumed: 11
Put: 12

Intimate Consumer


Producer waiting

Got: 12

Intimate Producer

Consumed: 12
Put: 13

Intimate Consumer


Producer waiting

Got: 13

Intimate Producer

Consumed: 13
Put: 14

Intimate Consumer

Got: 14

Intimate Producer

Consumed: 14

C:\Users\shett\OneDrive\Documents\javaclasslab>
```

```
C:\Users\shett\OneDrive\Documents\javaclasslab>javac Deadlock.java

C:\Users\shett\OneDrive\Documents\javaclasslab>java Deadlock
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()
```