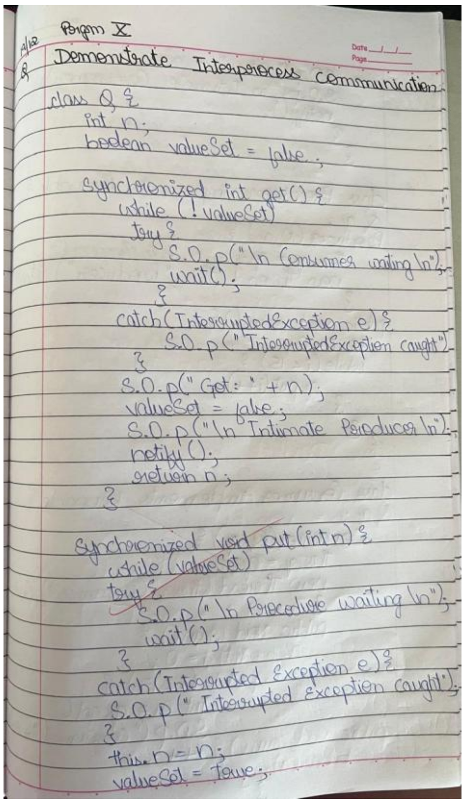
Program 10

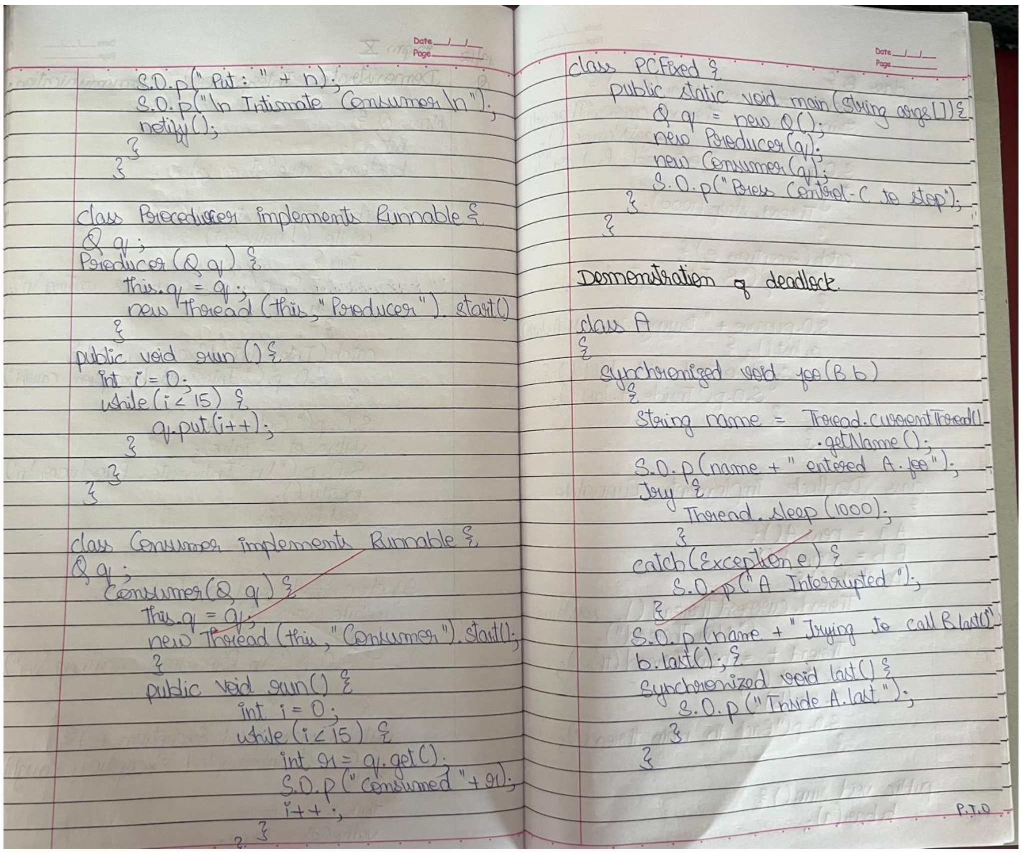
Demonstrate Inter process Communication and deadlock



Program X

Q. Demonstrate Interprocess communication.

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

    synchronized int get() {
        while (!valueSet)
          try {
              S.O.p("\n Consumer waiting \n");
              wait();
            }
          catch(InterruptedException e) {
              S.O.p("InterruptedException caught")
          }
          S.O.p("Got: " + n);
          valueSet = false;
          S.O.p("\n Intimate Producer \n");
          notify();
          return n;
      }

    synchronized void put(int n) {
      while (valueSet)
      try {
        S.O.p("\n Procedure waiting \n");
         wait();
      }
    catch(Interrupted Exception e){
     S.O.p(" Interrupted exception caught");
     }
     this.n = n;
     valueSet = True;
```



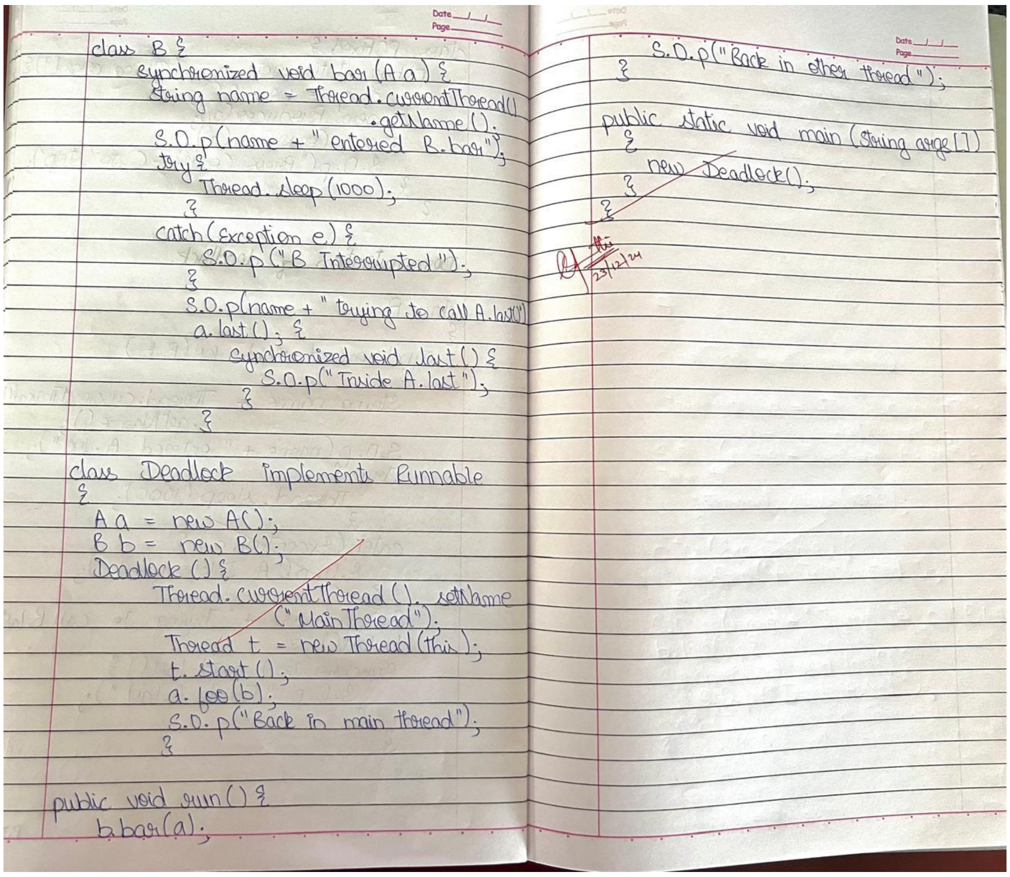
```
        S.O.p(" Put : " + n);
        S.O.p("\n Intimate Consumer \n");
        notify();
      }
    }

class Proceducer 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().
               S.O.p("Consumed " + r);
               i++;
         }
       }
```

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

Demonstration of deadlock.

```
class A
{
  synchronized void foo(B b)
   {
     String name = Thread.currentThread()
                          .getName();
     S.O.p(name + " entered A.foo");
     Try {
        Thread.sleep(1000);
       }
     catch(Exception e) {
        S.O.p("A Interrupted");
      }
     S.O.p(name + "Trying to call B.last()")
     b.last(); }
     Synchronized void last() {
        S.O.p("Inside A.last");
      }
    }
```

P.T.O



```
class B {
    synchronized void bar (A a) {
     String name = Thread.currentThread()
                          .getName();
      S.O.p(name + " entered B.bar");
      try {
          Thread.sleep (1000);
        }
      catch(Exception e) {
          S.O.p("B Interrupted");
        }
        S.O.p(name + " trying to call A.last()")
        a.last(); {
            synchronized void last() {
                S.O.p("Inside A.last");
            }
        }

class Deadlock implements Runnable
{
  A a = new A();
  B b = new B();
  Deadlock () {
        Thread.currentThread().setName
                    ("MainThread");
         Thread t = new Thread(this);
         t.start();
         a.foo(b);
         S.O.p("Back in main thread");
        }

public void run() {
     b.bar(a);
```

```
        S.O.p("Back in other thread");
   }

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

23/12/24

CODE:

```
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 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.");

}

}
```

ii. 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("Inside A.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); // get lock on a in this thread.

     System.out.println("Back in main thread");

   }

 public void run() { b.bar(a); // get lock on b in other thread.

  System.out.println("Back in other thread");

  }

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