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

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

```
synchronized void put(int n) {
   while(valueSet){
     try {
       System.out.println(&quot;\nProducer waiting\n&quot;);
       wait();
     }
     catch(InterruptedException e) {
       System.out.println(&quot;InterruptedException caught&quot;);
     }
     this.n = n;
     valueSet = true;
     System.out.println(&quot;Put: &quot; + n);
     System.out.println(&quot;\nIntimate Consumer\n&quot;);
     notify();
  }
}
class Producer implements Runnable {
  Q q;
  Producer(Q q) {
    this.q = q;
    new Thread(this, &quot;Producer&quot;).start();
  }
  public void run() {
    int i = 0;
    while(i&lt;15) {
      q.put(i++);
    }
  }
}
class Consumer implements Runnable {
   Q q;
   Consumer(Q q) {
     this.q = q;
     new Thread(this, &quot;Consumer&quot;).start();
```

```
  Producer(Q q) {
    this.q = q;
    new Thread(this, &quot;Producer&quot;).start();
  }
  public void run() {
    int i = 0;
    while(i&lt;15) {
      q.put(i++);
    }
  }
}
class Consumer implements Runnable {
   Q q;
   Consumer(Q q) {
     this.q = q;
     new Thread(this, &quot;Consumer&quot;).start();
   }
   public void run() {
     int i=0;
     while(i&lt;15) {
       int r=q.get();
       System.out.println(&quot;consumed:&quot;+r);
       i++;
     }
   }
}
class PCFixed {
   public static void main(String args[]) {
     Q q = new Q();
     new Producer(q);
     new Consumer(q);
     System.out.println(&quot;Press Control-C to stop.&quot;);
   }
}
```

```
D:\24BMSCE>javac PCFixed.java

D:\24BMSCE>java PCFixed
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
```

Command Prompt

```
Intimate Producer
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
```

78°F
Partly cloudy

Search

Command Prompt

```
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

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

D:\24BMSCE>
```

78°F
Partly cloudy

Search

LAB - 10

Demonstrate Interprocess Communication & deadlock.

```
class Q {
 int n;
 boolean valueSet = false;
 synchronized int get(){
   while (!valueset)
    try {
     System.Out.println("\n. Consumer waiting\n");
     wait();
    }
    catch (InterruptedException e){
     System.Out.Println("Interrupted Exception
                 caught");
    }
    System.out.println("Got: " +n);
    ValueSet = false;
    System.out.println("\n Intimate producer\n");
    notify();
    return n;
 }
 synchronized void put(int n){
   while (ValueSet){
    try {
     S.o.pln("\n. producer waiting \n");
     wait();
    }
    catch (InterruptedException e){
     S.o.pln(" Interrupted Exception caught");
    }
    this.n = n;
    ValueSet = true;
    S.o.pln("put: " + n);
```

```
        S.o.pln ("\n Intimate 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();
      S.o.pln ("Consumed:"+r);
      i++;
    }
  }
}
```

```
class PC
Public
  Q q =
  new
  new C
  S.o.
 }
}

O/P:
  Press
  Put :

  Intim

  Prod
  Got :

  Intim

  prod
  Cons
  Got

  Intim
  Consu
  Got

  Intim

  prod
  Got
```

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

O/P:

```
Press control - c to stop
Put : 0

Intimate Consumer

Producer waiting
Got : 0

Intimate ~~Con~~ producer

producer waiting
Consumed : 0
Got : 1

Intimated producer
Consumed : 1
Got : 2

Intimiate consumer

producer waiting
Got : 2
```

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

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

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

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

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

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) {
    S.o.pln (" A Interrupted ");
  }
  S.o.pln (name + " trying to call B.last()");
  b.last();
}
synchronized void last () {
  S.o.pln ("Inside A.last");
}
}

class B {
  synchronized void bar (A a) {
  String name = Thread.currentThread().
                getName();
  S.o.pln (name + " enteread B.bar ");
  try {
    Thread.sleep (1000);
  }
  catch (Exception e) {
    S.o.pln ("B Interrupted ");
  }
```

```
~~S.opt~~ S.o.pln (name + " trying to call A.last()");
        a.last();
    }

    synchronized void last(){
        S.o.pln ("Inside 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);
    S.o.pln ("Back in main thread");
    }
    public void run(){
        b.bar(a);
        S.o.pln ("Back in other thread");
    }
    public static void main (String [] args){
        new Deadlock();
    }
}
```

o/p.

```
MainThread entered A.foo
RacingThread entered B.fobar
MainThread trying to call B.last()
RacingThread trying to call A.last()
```