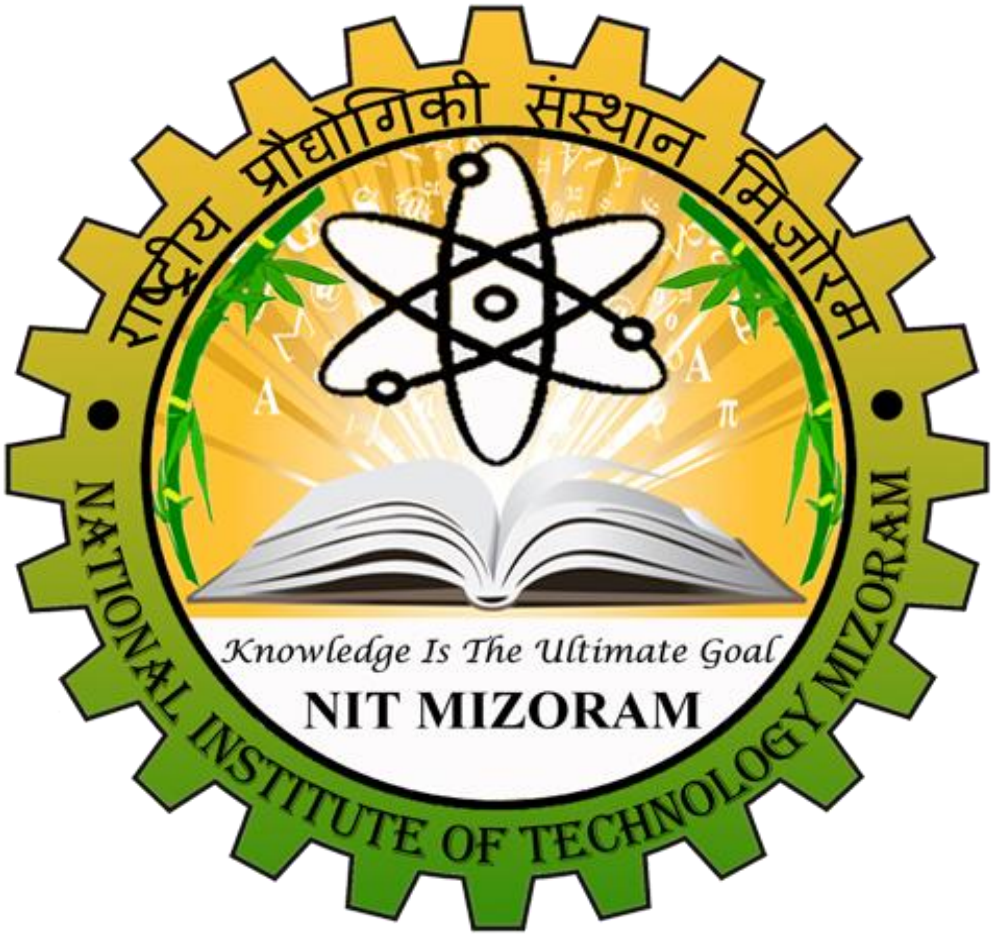
NATIONAL INSTITUTE OF TECHNOLOGY MIZORAM



OS LAB 4 ASSIGNMENT

Name: NIRAJ KUMAR

Department: CSE

Enrollment No. : BT19CS031

Given 2 process A and B executing simultaneously, both the process A and B

share a common, fixed-size array. Process A place an element in the array. If

array is already full then the process A will have to wait for an empty slot in

the array. On the other hand, process B removes an item from the array. If array

is already empty, then the process B will have to wait for an item in array.

Implement Peterson's Algorithm for the two processes using shared memory

such that there is mutual exclusion between them.

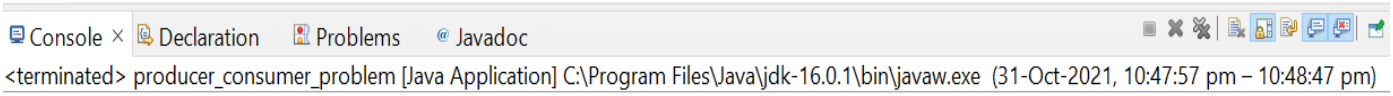
```
import java.util.*;
import java.lang.*;

// Java program to implement solution of producer
//consumer problem.

public class producer_consumer_problem //Main class
{

      public static void main(String[] args)throws Exception
      {
            //This is the shared array variable between producer and
consumer
            ArrayList<Integer> sharedarray = new
ArrayList<Integer>();

            //creating objects of thread class and passing the
objects of producer
            //and consumer class and along with that we set the
proper name of both threads
```

```
            Thread thread1 = new Thread(new
Producer(sharedarray),"Producer Thread"); // producer thread
            Thread thread2 = new Thread(new
Consumer(sharedarray),"Consumer Thread"); // consumer thread

            //starting both the threads parallely
            thread1.start();
            thread2.start();

      }

}

//Producer class

class Producer implements Runnable
{
      ArrayList<Integer> sharedarray = null;
      int max_size=5; // max limit for the item to be produced at
once is set to 5
      int i=0;

      //Constructor
      public Producer(ArrayList<Integer> sharedarray)
      {
            this.sharedarray = sharedarray;
      }

       //Override run method from Runnable interface
       public void run()
       {
             while(true)
                   try
                   {
                         produce(i++);
                     }
                   catch (Exception e)
                   {
                     System.out.println("Exception Arised");
                   }
       }

       //produce() function which produces items one by one
       //and store inside the sharedarray

       public void produce(int i) throws Exception
       {
        //Synchronizing the common resource i.e shared array
       synchronized(sharedarray)
```

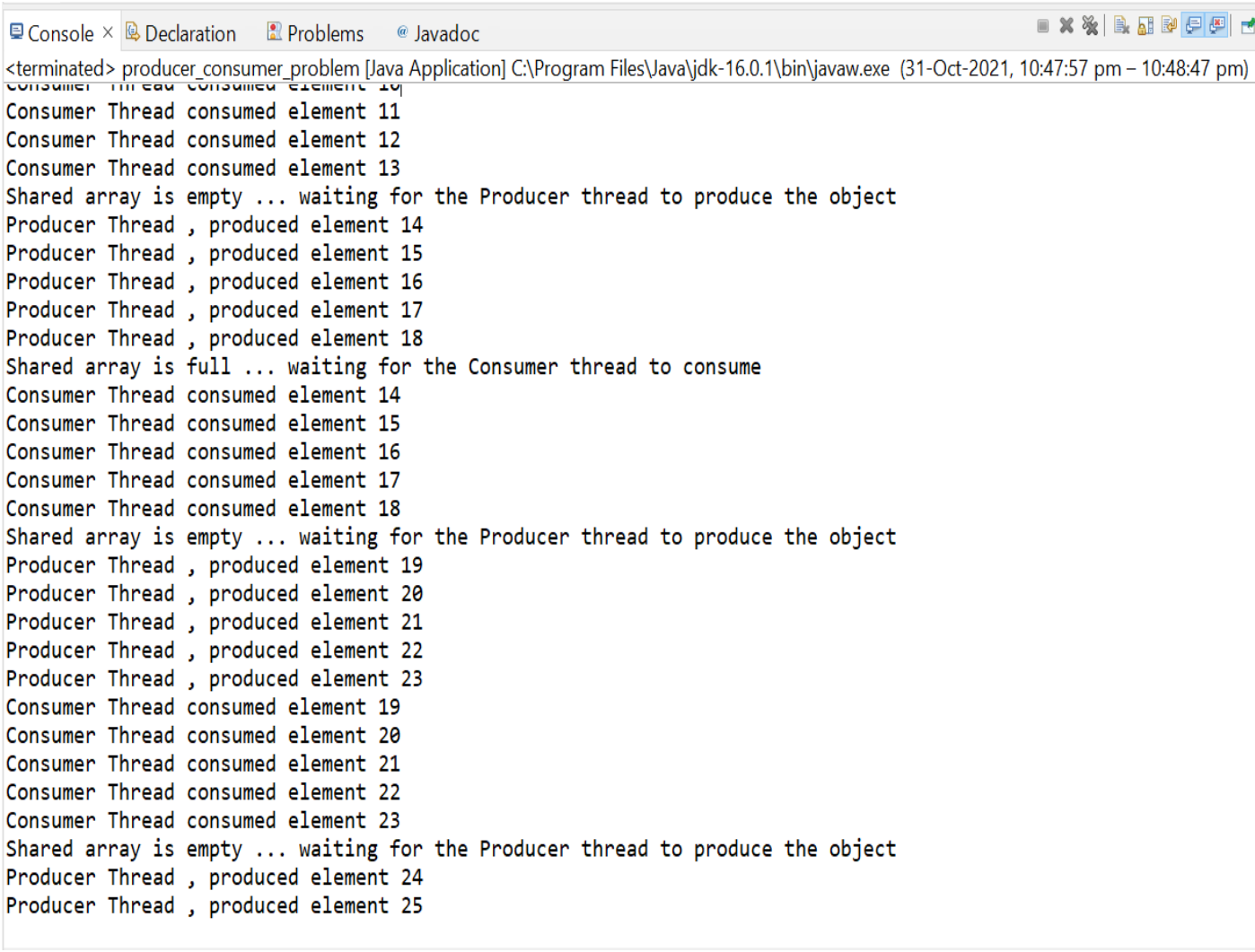
```
      {
            while(sharedarray.size()==max_size) //no. of items
produced , reached max. capacity
            {
                  System.out.println("Shared array is full ...
waiting for the Consumer thread to consume");

                  //we put Producer thread in waiting state and allow
Consumer thread to execute first
                  sharedarray.wait();
            }

            //Producer ready to produce, array contains at least one
free space
            sharedarray.add(i);
            System.out.println("Producer Thread , produced element
"+i);

            // We put Producer thread to sleep for 1 sec. so that
all operations are done smoothly
            Thread.sleep(1000);

            // notifies the Consumer thread that now it can start
consuming
            sharedarray.notify();
       }
    }
}

//Consumer class

class Consumer implements Runnable
{
      ArrayList<Integer> sharedarray=null;

      //constructor
      public Consumer(ArrayList<Integer> sharedarray)
      {
            this.sharedarray = sharedarray;
      }

       //Override run method from Runnable interface
       public void run()
       {
            while(true)
                try
               {
                  consume();
                }
```

```
                catch (Exception e)
                {
                   System.out.println("Exception arised");
                }
             }

       // consume() function which consumes item one by one
       // and gradually creates free space for new items inside
sharedarray
       public void consume()throws Exception
       {
            // Synchronizing the common resource i.e sharedarray
      synchronized(sharedarray)
      {
           while(sharedarray.isEmpty()) // array is empty and
nothing to consume
             {
                  System.out.println("Shared array is empty ...
waiting for the Producer thread to produce the object");

                  //we put Consumer thread in waiting state and allow
Producer thread to execute first
                  sharedarray.wait();
             }

           //Consumer thread ready to consume , array contains
atleast one item

           System.out.println("Consumer Thread consumed element
"+sharedarray.remove(0));

           //we put Consumer thread to sleep for 1 sec so that all
operations are done smoothly
           Thread.sleep(1000);

           //notifies the Producer thread that it can start
producing
            sharedarray.notify();
       }
      }
 }
```

OUTPUT –




```
Producer Thread , produced element 0
Producer Thread , produced element 1
Producer Thread , produced element 2
Producer Thread , produced element 3
Consumer Thread consumed element 0
Consumer Thread consumed element 1
Consumer Thread consumed element 2
Consumer Thread consumed element 3
Shared array is empty ... waiting for the Producer thread to produce the object
Producer Thread , produced element 4
Producer Thread , produced element 5
Producer Thread , produced element 6
Producer Thread , produced element 7
Producer Thread , produced element 8
Shared array is full ... waiting for the Consumer thread to consume
Consumer Thread consumed element 4
Consumer Thread consumed element 5
Consumer Thread consumed element 6
Consumer Thread consumed element 7
Consumer Thread consumed element 8
Shared array is empty ... waiting for the Producer thread to produce the object
Producer Thread , produced element 9
Producer Thread , produced element 10
Producer Thread , produced element 11
Producer Thread , produced element 12
Producer Thread , produced element 13
Shared array is full ... waiting for the Consumer thread to consume
Consumer Thread consumed element 9
Consumer Thread consumed element 10
```


Console
Declaration
Problems
Javadoc
<terminated> producer_consumer_problem [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (31-Oct-2021, 10:47:57 pm – 10:48:47 pm)
Consumer Thread consumed element 11
Consumer Thread consumed element 12
Consumer Thread consumed element 13
Shared array is empty ... waiting for the Producer thread to produce the object
Producer Thread , produced element 14
Producer Thread , produced element 15
Producer Thread , produced element 16
Producer Thread , produced element 17
Producer Thread , produced element 18
Shared array is full ... waiting for the Consumer thread to consume
Consumer Thread consumed element 14
Consumer Thread consumed element 15
Consumer Thread consumed element 16
Consumer Thread consumed element 17
Consumer Thread consumed element 18
Shared array is empty ... waiting for the Producer thread to produce the object
Producer Thread , produced element 19
Producer Thread , produced element 20
Producer Thread , produced element 21
Producer Thread , produced element 22
Producer Thread , produced element 23
Consumer Thread consumed element 19
Consumer Thread consumed element 20
Consumer Thread consumed element 21
Consumer Thread consumed element 22
Consumer Thread consumed element 23
Shared array is empty ... waiting for the Producer thread to produce the object
Producer Thread , produced element 24
Producer Thread , produced element 25