

Smt. Chandibai Himathmal Mansukhani College

USCSP301 : USCS303 - Operating System (OS)

PRACTICAL - 07 – Synchronization

Table of Contents

Practical Date: 27 Aug 2021	2
Practical Aim:.....	2
Synchronization :	2
Producer:	3
Consumer:	4
Question -01.....	5
Source Code :.....	5
Output :.....	11
Screenshot-1.....	11
Screenshot-2.....	13
Screenshot-3.....	14
Reader Writers Problem	14
Question - 02:.....	15
Source Code :.....	15
Output :.....	19
Sleeping Barber Problem:	19
Question - 03:.....	20
Source Code :.....	20
Output :.....	23
Screenshot-1:	24
Screenshot-2.....	25

Smt. Chandibai Himathmal Mansukhani College

Practical Date: 27 Aug 2021

Practical Aim: a) Bounded Buffer Problem ;
b) Readers - Writer's Problem;
c) Sleeping Barber Problem;

Synchronization :

(Bounded Buffer ,Readers - Writers , Sleeping Barber Problem)

Bounded Buffer Problem :

- Producer-consumer problem, also known as Bounded buffer problem, illustrated the need for synchronisation in systems where many processes share a resource .
- These processes do not take turns accessing the buffer; they both work concurrently. Herein lies the problem .
- What happens if the producer tries to put an item into a full buffer?
- What happens if the consumer tries to take an item from an Empty buffer?
- Order to synchronise these processes, we will block the producer when the buffer is full, and we will block the consumer when the buffer is empty.

So the two processes Producer and Consumer, should work as follows:

(i) The producer must first create a new widget.

Smt. Chandibai Himathmal Mansukhani College

(ii) Then, it checks to see if the buffer is full .If it is, the producer will put it safe to sleep until the consumer wake it up. A “wake up” will come if the consumer finds the buffer empty.

(iii) Next , the producer put the new widget in the buffer. if the producer goes to sleep in step 2 it will not wake up until the buffer is empty, so the buffer will never overflow.

(iv) Then the producer checks to see if the buffer is empty. if it is, the producer assumed that the consumer is sleeping, and so it will make the consumer. keep in mind that between any of these steps, An interrupt might occur allowing the consumer to run .

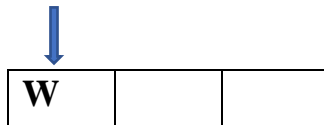
Producer:

1. Make new widget → W

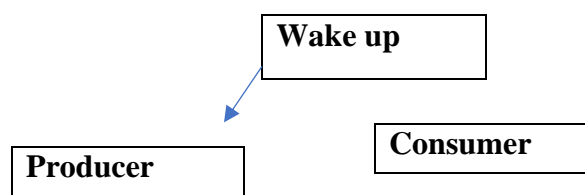
2. IF buffer is full , go to sleep

W	W	W	=	Procedure
---	---	---	---	-----------

3. Put widget i buffer



4. If buffer was empty, wake consumer



Smt. Chandibai Himathmal Mansukhani College

So the two processes Producer and Consumer, should work as follows:

(i) The consumer checks to see if the buffer is empty. If so, the consumer will put it safe to sleep until the producer wakes it up. A “wakeup” the occur is the producer find the buffer empty after food send item into the buffer.

(ii) Then, the consumer will remove a widget from the buffer. the consumer will never try to remove a widget from an empty buffer because it will not wake up until the buffer is full.

(iii) if the buffer was full before it removed the widget, the consumer will wake the producer.

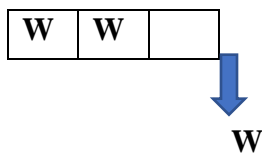
(iv) Finally, the consumer will consume the widget. As was the case with the producer, an interrupt could occur between any of these steps along the producer to run.

Consumer:

1. IF buffer is empty , Go to sleep

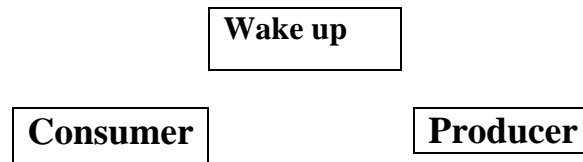


2. Take widget from Buffer



3. IF buffer was Full , wake Producer

Smt. Chandibai Himathmal Mansukhani College



4. Consume the widget

<W

Question -01

Write a Java Program for bounded buffer problem using synchronisation .

Source Code :

```
// Name: Gaurang Sanyasi  
// Batch: B2  
// PRN: 2020016400785461  
// Date: 27 August, 2021  
// Prac-07: Synchronization
```

```
public interface P7_Q1_Buffer_GS  
{  
    public void set(int value) throws InterruptedException;  
    public int get() throws InterruptedException;  
}
```

Smt. Chandibai Himathmal Mansukhani College

Source code 2:

// Name: Gaurang Sanyasi

// Batch: B2

// PRN: 2020016400785461

// Date: 27 August, 2021

// Prac-07: Synchronization

```
public class P7_Q1_CircularBuffer_GS implements P7_Q1_Buffer_GS
{
    private final int[] buffer={-1,-1,-1}; //shared buffer
    private int occupiedCells=0; //count number of buffer used
    private int writeIndex=0; //Index of next element to write to
    private int readIndex=0; //Index of next element to read
    public synchronized void set(int value) throws InterruptedException
    {
        while(occupiedCells==buffer.length)
        {
            System.out.println("Buffer is full.Producer waits.");
            wait();
        }
        buffer[writeIndex]=value;
        writeIndex=(writeIndex + 1) % buffer.length;
        ++occupiedCells;
        displayState("Producer write "+value);
        notifyAll();
    } //set() ends
    public synchronized int get() throws InterruptedException
```

Smt. Chandibai Himathmal Mansukhani College

```
{
while(occupiedCells==0)
{
System.out.println("Buffer is empty.Consumer waits.");
wait();
}
int readvalue=buffer[readIndex];
readIndex=(readIndex + 1) % buffer.length;
--occupiedCells;
displayState("Consumer reads "+readvalue);
notifyAll();
return readvalue;
} //get() ends

public void displayState(String operation)
{
System.out.printf("%s%s%d)\n%s",operation," (buffer Cells occupied: ",occupiedCells,
"buffer Cells: ");
for(int value:buffer)
System.out.printf(" %2d ",value);
System.out.print("\n ");
for(int i=0;i<buffer.length;i++)
System.out.print(" .... ");
System.out.print("\n ");
for (int i=0;i<buffer.length;i++)
{
if(i==writeIndex && i==readIndex)
System.out.print(" WR");
else if(i==writeIndex)
```

Smt. Chandibai Himathmal Mansukhani College

```
System.out.print(" W ");
else if(i==readIndex)
System.out.print(" R ");
else
System.out.print(" ");

}
System.out.println("\n");
} //displayState() ends
} //CircularBuffer class ends
```

Source code 3:

```
// Name: Gaurang Sanyasi
// Batch: B2
// PRN: 2020016400785461
// Date: 27 August, 2021
// Prac-07: Synchronization
```

```
import java.util.Random;
public class P7_Q1_Consumer_GS implements Runnable
{
    private final static Random generator=new Random();
    private final P7_Q1_Buffer_GS sharedLocation;
    public P7_Q1_Consumer_GS(P7_Q1_Buffer_GS shared)
    {
        sharedLocation=shared;
    }
    public void run()
    {
```


Smt. Chandibai Himathmal Mansukhani College

```
int sum=0;
for(int count=1;count<=10;count++)
{
try{
Thread.sleep(generator.nextInt(3000));
sum+=sharedLocation.get();
}catch(InterruptedExpection e){
e.printStackTrace();
}
}
System.out.printf("\n%s %d\n%s\n", "Consumer read values totaling",
sum, "Terminating Consumer");
} //run() ends
} //Consumer class ends
```

Source code 4:

```
// Name: Gaurang Sanyasi
// Batch: B2
// PRN: 2020016400785461
// Date: 27 August, 2021
// Prac-07: Synchronization
```

```
import java.util.Random;
public class P7_Q1_Producer_GS implements Runnable
{
private final static Random generator=new Random();
private final P7_Q1_Buffer_GS sharedLocation;
public P7_Q1_Producer_GS(P7_Q1_Buffer_GS shared)
{
sharedLocation=shared;
}
public void run()
{
for(int count=1;count<=10;count++)
```

Smt. Chandibai Himathmal Mansukhani College

```
{
try{
Thread.sleep(generator.nextInt(3000));
sharedLocation.set(count);
}catch(InterruptedException e){
e.printStackTrace();
}
}
System.out.println("Producer done producing. Terminating Producer");
} //run() ends
} //Producer class ends
```

Source code 5:

```
// Name: Gaurang Sanyasi
// Batch: B2
// PRN: 2020016400785461
// Date: 27 August, 2021
// Prac-07: Synchronization
```

```
import java.util.concurrent.*;
public class P7_Q1_Test_GS
{
public static void main(String args[])
{
ExecutorService application=Executors.newCachedThreadPool();
P7_Q1_CircularBuffer_GS sharedLocation=new P7_Q1_CircularBuffer_GS();
sharedLocation.displayState("Initial State");
application.execute(new P7_Q1_Producer_GS(sharedLocation));
application.execute(new P7_Q1_Consumer_GS(sharedLocation));
application.shutdown();
}
```

Smt. Chandibai Himathmal Mansukhani College

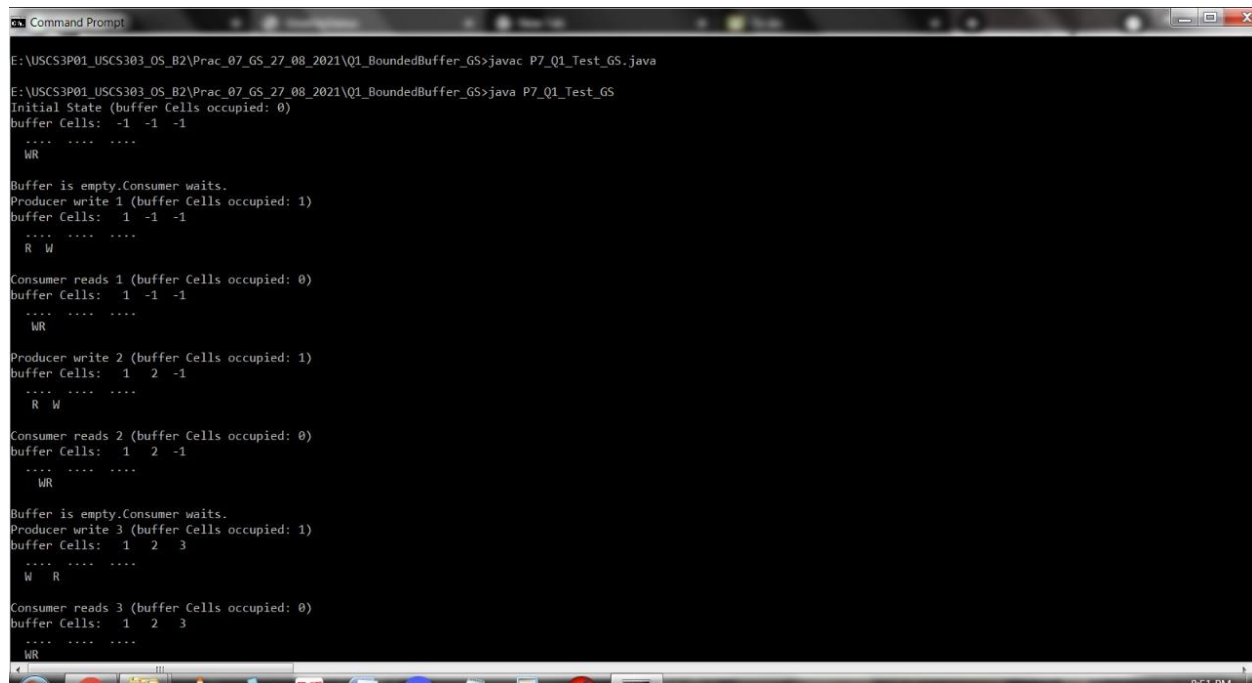
}

}

Output :

Screenshot-1

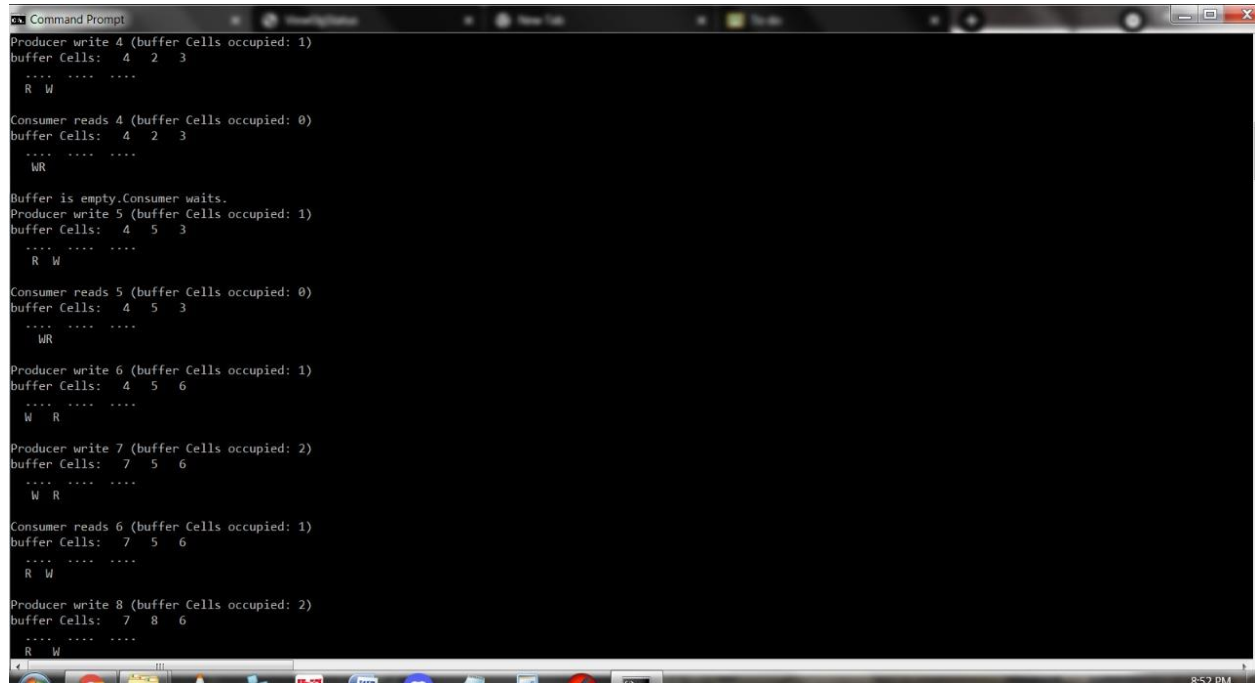
Smt. Chandibai Himathmal Mansukhani College



```
Command Prompt
E:\USCS3P01_USCS303_05_B2\Prac_07_GS_27_08_2021\Q1_BoundedBuffer_GS>javac P7_Q1_Test_GS.java
E:\USCS3P01_USCS303_05_B2\Prac_07_GS_27_08_2021\Q1_BoundedBuffer_GS>java P7_Q1_Test_GS
Initial State (buffer Cells occupied: 0)
buffer Cells: -1 -1 -1
.....
WR
Buffer is empty.Consumer waits.
Producer write 1 (buffer Cells occupied: 1)
buffer Cells:  1 -1 -1
.....
R W
Consumer reads 1 (buffer Cells occupied: 0)
buffer Cells:  1 -1 -1
.....
WR
Producer write 2 (buffer Cells occupied: 1)
buffer Cells:  1  2 -1
.....
R W
Consumer reads 2 (buffer Cells occupied: 0)
buffer Cells:  1  2 -1
.....
WR
Buffer is empty.Consumer waits.
Producer write 3 (buffer Cells occupied: 1)
buffer Cells:  1  2  3
.....
W R
Consumer reads 3 (buffer Cells occupied: 0)
buffer Cells:  1  2  3
.....
WR
```

Smt. Chandibai Himathmal Mansukhani College

Screenshot-2



```
Command Prompt
Producer write 4 (buffer Cells occupied: 1)
buffer Cells:  4  2  3
.....
R  W

Consumer reads 4 (buffer Cells occupied: 0)
buffer Cells:  4  2  3
.....
W R

Buffer is empty. Consumer waits.
Producer write 5 (buffer Cells occupied: 1)
buffer Cells:  4  5  3
.....
R  W

Consumer reads 5 (buffer Cells occupied: 0)
buffer Cells:  4  5  3
.....
W R

Producer write 6 (buffer Cells occupied: 1)
buffer Cells:  4  5  6
.....
W  R

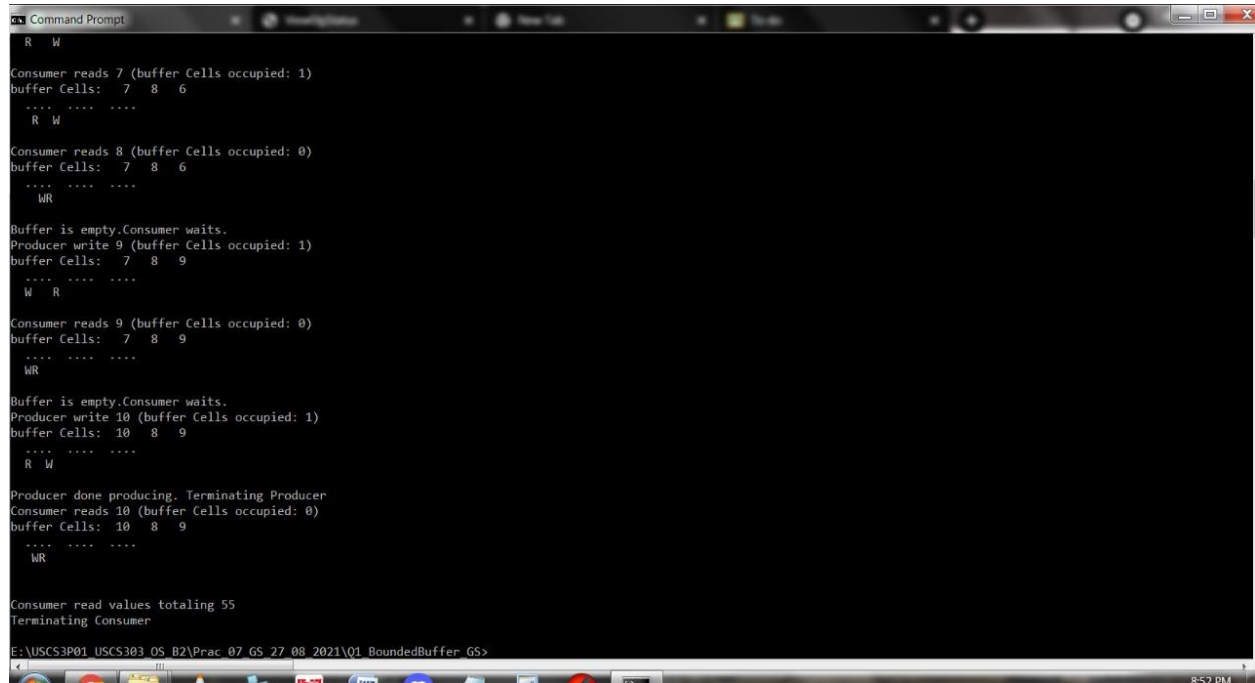
Producer write 7 (buffer Cells occupied: 2)
buffer Cells:  7  5  6
.....
W  R

Consumer reads 6 (buffer Cells occupied: 1)
buffer Cells:  7  5  6
.....
R  W

Producer write 8 (buffer Cells occupied: 2)
buffer Cells:  7  8  6
.....
R  W
```

Smt. Chandibai Himathmal Mansukhani College

Screenshot-3



```

R W
Consumer reads 7 (buffer Cells occupied: 1)
buffer Cells: 7 8 6
....
R W
Consumer reads 8 (buffer Cells occupied: 0)
buffer Cells: 7 8 6
....
WR
Buffer is empty.Consumer waits.
Producer write 9 (buffer Cells occupied: 1)
buffer Cells: 7 8 9
....
W R
Consumer reads 9 (buffer Cells occupied: 0)
buffer Cells: 7 8 9
....
WR
Buffer is empty.Consumer waits.
Producer write 10 (buffer Cells occupied: 1)
buffer Cells: 10 8 9
....
R W
Producer done producing. Terminating Producer
Consumer reads 10 (buffer Cells occupied: 0)
buffer Cells: 10 8 9
....
WR
Consumer read values totaling 55
Terminating Consumer
E:\USCS3P01 USCS303 05 B2\Prac 07 GS 27 08 2021\01 BoundedBuffer GS>

```

Reader Writers Problem

- In Computer Science, the readers-writers problem are examples of a common computing problem in concurrency .
- Here many Threads (small processes which share data) try to access the same shared resource at one time.

Smt. Chandibai Himathmal Mansukhani College

- Some Trends married and some may write, with the constraint that no process may access the shared resource for other reading or writing while another process is in the act of writing to it.
- (In particular we want to prevent more than one thread modify the shared resource simultaneously and allowed for two or more readers to access the shared resource at the same time).
- A readers-writer lock is a data structure that solves one or more of the readers writers problems.

Question - 02:

Write a Java Program for Readers Writers problem using semaphore.

(Implement readers writers problem using semaphore)

Source Code :

// Name: Gaurang Sanyasi

// Batch: B2

// PRN: 2020016400785461

// Date: 27 August, 2021

// Prac-07: Synchronization

```
import java.util.concurrent.Semaphore;
```

```
class P7_Q2_ReaderWriter_GS
```

```
{
```

```
static Semaphore readLock = new Semaphore(1,true);
```

Smt. Chandibai Himathmal Mansukhani College

```
static Semaphore writeLock = new Semaphore(1,true);
static int readCount = 0;
static class Read implements Runnable {
    @Override
    public void run() {
        try {
            // Acquire Section
            readLock.acquire();
            readCount++;
            if(readCount == 1){
                writeLock.release();
            }
            readLock.release();
            //Reading section();
            System.out.println("Thread"+ Thread.currentThread().getName() + " is READING");
            Thread.sleep(1500);
            System.out.println("Thread"+ Thread.currentThread().getName() + "has FINISHED READING");
            //Releasing section
            readLock.acquire();
            readCount--;
            if(readCount == 0){
                writeLock.release();
            }
            readLock.release();
        } //try ends
        catch (InterruptedException e){
            System.out.println(e.getMessage());
        }
    }
}
```


Smt. Chandibai Himathmal Mansukhani College

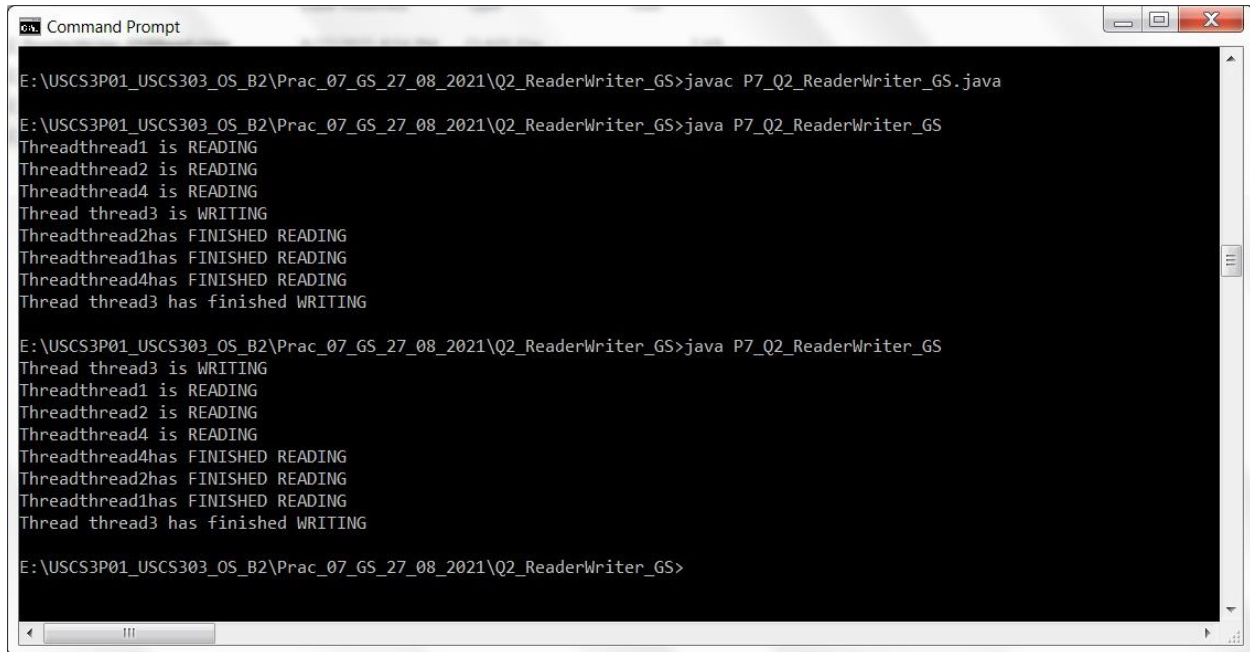
```
}  
}  
} //run() ends  
}  
} //static class Read ends  
static class Write implements Runnable {  
    @Override  
    public void run() {  
        try {  
            writeLock.acquire();  
            System.out.println("Thread "+Thread.currentThread().getName() + " is WRITING ");  
            Thread.sleep(2500);  
            System.out.println("Thread "+Thread.currentThread().getName() + " has finished WRITING ");  
            writeLock.release();  
        } catch (InterruptedException e) {  
  
            System.out.println(e.getMessage());  
        }  
    }  
} // run() ends  
}  
} //class Write ends  
public static void main(String[] args)  
    throws Exception {  
    Read read = new Read();  
    Write write = new Write();  
    Thread t1 = new Thread(read);  
    t1.setName("thread1");  
    Thread t2 = new Thread(read);  
    t2.setName("thread2");  
    Thread t3 = new Thread(write);  
    t3.setName("thread3");
```

Smt. Chandibai Himathmal Mansukhani College

```
Thread t4 = new Thread(read);  
t4.setName("thread4");  
t1.start();  
t3.start();  
t2.start();  
t4.start();  
} // main ends  
} //class P7_Q2_ReaderWriter_GS ends
```

Smt. Chandibai Himathmal Mansukhani College

Output :



```
Command Prompt
E:\USCS3P01_USCS303_OS_B2\Prac_07_GS_27_08_2021\Q2_ReaderWriter_GS>javac P7_Q2_ReaderWriter_GS.java
E:\USCS3P01_USCS303_OS_B2\Prac_07_GS_27_08_2021\Q2_ReaderWriter_GS>java P7_Q2_ReaderWriter_GS
Threadthread1 is READING
Threadthread2 is READING
Threadthread4 is READING
Thread thread3 is WRITING
Threadthread2has FINISHED READING
Threadthread1has FINISHED READING
Threadthread4has FINISHED READING
Thread thread3 has finished WRITING
E:\USCS3P01_USCS303_OS_B2\Prac_07_GS_27_08_2021\Q2_ReaderWriter_GS>java P7_Q2_ReaderWriter_GS
Thread thread3 is WRITING
Threadthread1 is READING
Threadthread2 is READING
Threadthread4 is READING
Threadthread4has FINISHED READING
Threadthread2has FINISHED READING
Threadthread1has FINISHED READING
Thread thread3 has finished WRITING
E:\USCS3P01_USCS303_OS_B2\Prac_07_GS_27_08_2021\Q2_ReaderWriter_GS>
```

Sleeping Barber Problem:

- Barber shop consists of a waiting room with and chairs and a barber room with one Barber chair.
- If there are no customers to be served the barber goes to sleep.
- If a customer enters the barber shop and all chairs are occupied then the customer leaves the shop.
- If the barber is busy but chairs are available then the customers its in one of the features if the barber is asleep the customer with sharp the barber .

Smt. Chandibai Himathmal Mansukhani College

Question - 03:

Write a program to co-ordinate the barber and the customers using Java synchronisation.

(Implement Sleeping-Barber problem)

Source Code :

```
// Name: Gaurang Sanyasi
// Batch: B2
// PRN: 2020016400785461
// Date: 27 August, 2021
// Prac-07: Synchronization

import java.util.concurrent.*;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.Random;
public class P7_Q3_Barber_GS implements Runnable
{
    private AtomicInteger spaces;
    private Semaphore bavailable;
    private Semaphore cavailable;
    private Random ran = new Random();
    public P7_Q3_Barber_GS(AtomicInteger spaces, Semaphore bavailable,
        Semaphore cavailable){
        this.spaces = spaces;
        this.bavailable = bavailable;
        this.cavailable = cavailable;
    }
    @Override
    public void run(){
        while(true){
            try{
                cavailable.acquire();
                //Space freed up in waiting area
                System.out.println("Customer getting hair cut");
                Thread.sleep(ThreadLocalRandom.current().nextInt(1000,10000+1000));
```

Smt. Chandibai Himathmal Mansukhani College

```
//Sleep to imitate length of time to cut hair
System.out.println("Customer pays and leaves");
bavailable.release();
} catch (InterruptedException e) { }
} //while ends
} //run ends
} //class ends
```

Source code 2:

```
// Name: Gaurang Sanyasi
// Batch: B2
// PRN: 2020016400785461
// Date: 27 August, 2021
// Prac-07: Synchronization

import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.*;
class P7_Q3_BarberShop_GS{
public static void main(String[] args)
{
AtomicInteger spaces = new AtomicInteger(15);
final Semaphore barbers = new Semaphore(3, true);
final Semaphore customers = new Semaphore(0, true);
ExecutorService openUp = Executors.newFixedThreadPool(3);
P7_Q3_Barber_GS[] employees = new P7_Q3_Barber_GS[3];
System.out.println("Opening up shop");
for(int i = 0; i < 3; i++){
employees[i] = new P7_Q3_Barber_GS(spaces, barbers, customers);
openUp.execute(employees[i]);
}
while(true)
{
try{
Thread.sleep(ThreadLocalRandom.current().nextInt(100, 1000+100));
//Sleep until next person gets in
```

Smt. Chandibai Himathmal Mansukhani College

```
}  
catch(InterruptedException e){}  
System.out.println("Customer walks in");  
if(spaces.get() >= 0){  
new Thread(new P7_Q3_Customer_GS(spaces,  
barbers,customers)).start();  
}  
else{  
System.out.println("Customer walks out, as no seats are available");  
}  
} //while ends  
} //main ends  
} //P7_Q3_BarberShop_GS class ends
```

Source code 3:

```
// Name: Gaurang Sanyasi  
// Batch: B2  
// PRN: 2020016400785461  
// Date: 27 August, 2021  
// Prac-07: Synchronization
```

```
import java.util.concurrent.*;  
import java.util.concurrent.atomic.AtomicInteger;  
import java.util.Random;  
public class P7_Q3_Customer_GS implements Runnable  
{  
private AtomicInteger spaces;  
private Semaphore bavailable;  
private Semaphore cavailable;  
private Random ran = new Random();  
  
public P7_Q3_Customer_GS(AtomicInteger spaces, Semaphore bavailable,  
Semaphore cavailable){  
this.spaces = spaces;  
this.bavailable = bavailable;
```

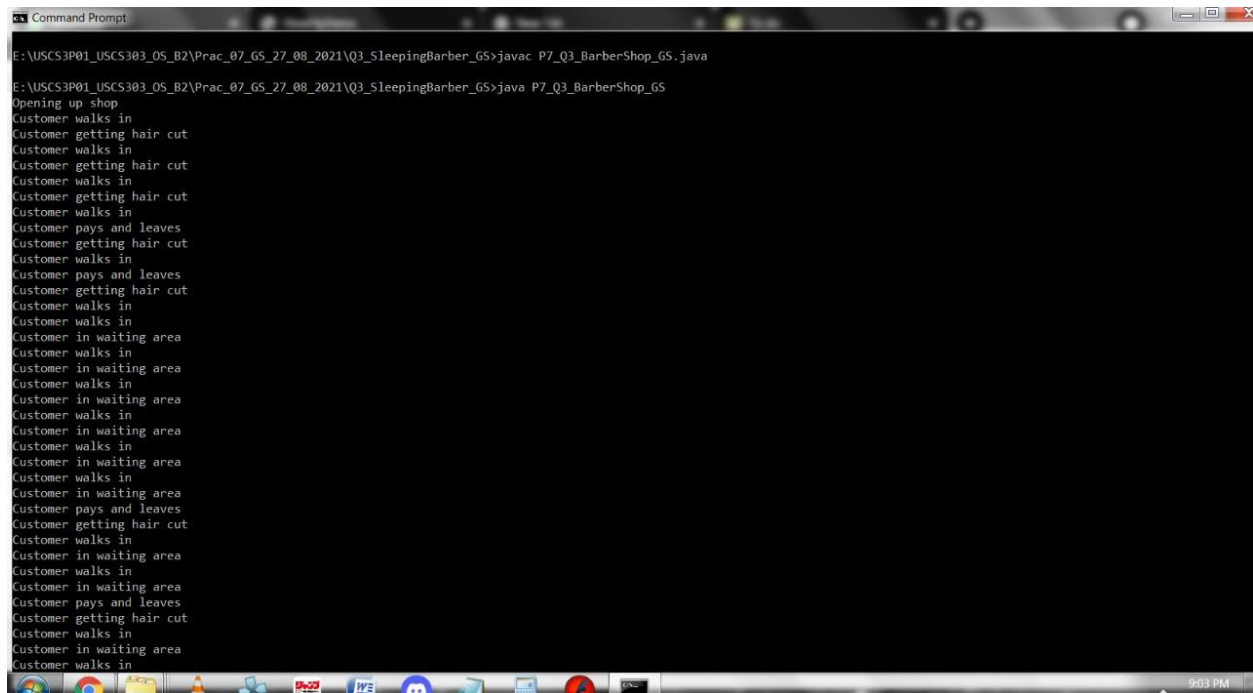
Smt. Chandibai Himathmal Mansukhani College

```
this.cavailable = cavailable;
}
@Override
public void run(){
try{
cavailable.release();
if(bavailable.hasQueuedThreads()){
spaces.decrementAndGet();
System.out.println("Customer in waiting area");
bavailable.acquire();
spaces.incrementAndGet();
}
else
{
bavailable.acquire();
}
}catch(InterruptedException e){ }
} //run ends
} //P7_Q3_Customer_GS class
```

Output :

Smt. Chandibai Himathmal Mansukhani College

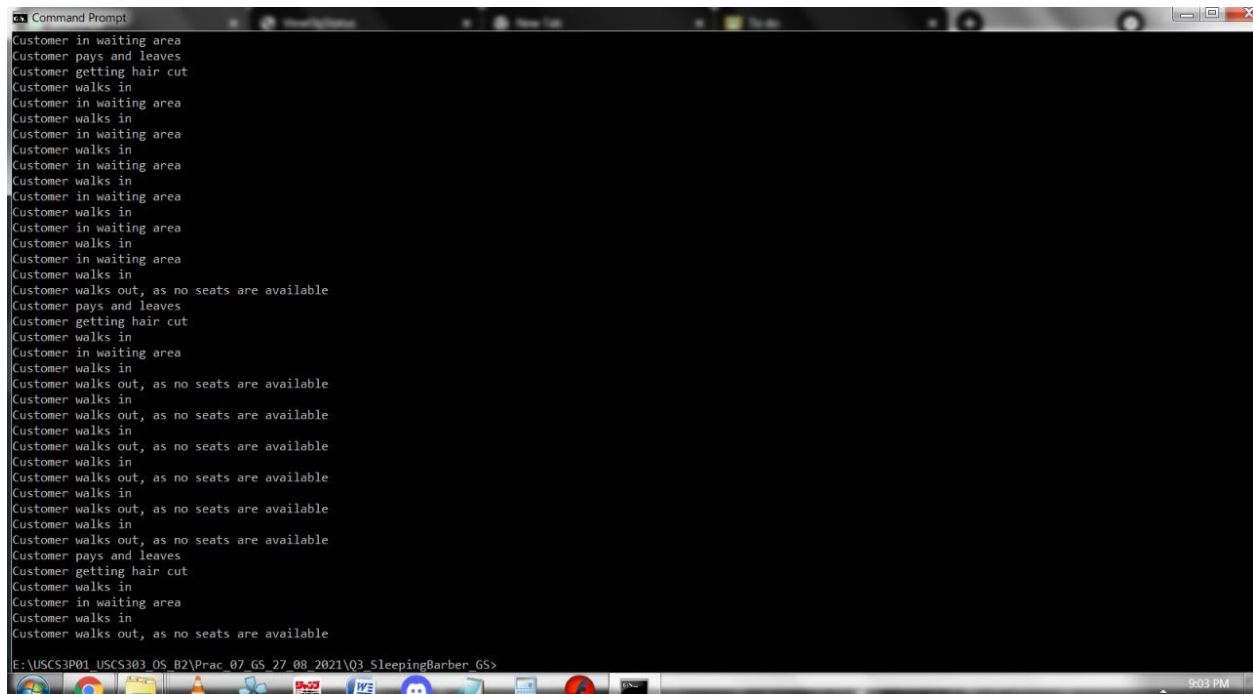
Screenshot-1:



```
Command Prompt
E:\USCS3P01_USCS303_05_B2\Prac_07_GS_27_08_2021\Q3_SleepingBarber_GS>javac P7_Q3_BarberShop_GS.java
E:\USCS3P01_USCS303_05_B2\Prac_07_GS_27_08_2021\Q3_SleepingBarber_GS>java P7_Q3_BarberShop_GS
Opening up shop
Customer walks in
Customer getting hair cut
Customer walks in
Customer getting hair cut
Customer walks in
Customer getting hair cut
Customer walks in
Customer pays and leaves
Customer getting hair cut
Customer walks in
Customer pays and leaves
Customer getting hair cut
Customer walks in
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer pays and leaves
Customer getting hair cut
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer pays and leaves
Customer getting hair cut
Customer walks in
Customer in waiting area
Customer walks in
```


Smt. Chandibai Himathmal Mansukhani College

Screenshot-2



```
Command Prompt
Customer in waiting area
Customer pays and leaves
Customer getting hair cut
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer in waiting area
Customer walks in
Customer walks out, as no seats are available
Customer pays and leaves
Customer getting hair cut
Customer walks in
Customer in waiting area
Customer walks in
Customer walks out, as no seats are available
Customer walks in
Customer walks out, as no seats are available
Customer walks in
Customer walks out, as no seats are available
Customer walks in
Customer walks out, as no seats are available
Customer walks in
Customer walks out, as no seats are available
Customer walks in
Customer walks out, as no seats are available
Customer pays and leaves
Customer getting hair cut
Customer walks in
Customer in waiting area
Customer walks in
Customer walks out, as no seats are available
E:\USCS3P01 USCS303 OS B2\Prac 07 GS 27 08 2021\Q3 SleepingBarber GS>
```