A synchronized shared buffer implementation

```
using System;
using System.Threading;

public class SynchronizedBuffer
{
private int buffer = -1;

private int occupiedBufferCount = 0;

public int Buffer
{
get
{
Monitor.Enter( this );

if ( occupiedBufferCount == 0 )
{
Console.WriteLine(Thread.CurrentThread.Name + ” tries to read.” );
DisplayState( “Buffer empty. ” +Thread.CurrentThread.Name + ” waits.” );
Monitor.Wait( this );
}
–occupiedBufferCount;

DisplayState( Thread.CurrentThread.Name + ” reads ” + buffer );

Monitor.Pulse( this );
int bufferCopy = buffer;

Monitor.Exit( this );
```

```
return bufferCopy;
}
set
{
Monitor.Enter( this );
if ( occupiedBufferCount == 1 )
{
Console.WriteLine(Thread.CurrentThread.Name + ” tries to write.” );
DisplayState( “Buffer full. ” + Thread.CurrentThread.Name + ” waits.” );
Monitor.Wait( this );
}
buffer = value;

++occupiedBufferCount;

DisplayState( Thread.CurrentThread.Name + ” writes ” + buffer );
Monitor.Pulse( this );

Monitor.Exit( this );
}
}

public void DisplayState( string operation )
{
Console.WriteLine( “{0,-35}{1,-9}{2}
“,operation, buffer, occupiedBufferCount );
}

static void Main( string[] args )
{
SynchronizedBuffer shared = new SynchronizedBuffer();
Random random = new Random();

Console.WriteLine( “{0,-35}{1,-9}{2}
“,”Operation”, “Buffer”, “Occupied Count” );
shared.DisplayState( “Initial state” );
```

```
Producer producer = new Producer( shared, random );
Consumer consumer = new Consumer( shared, random );

Thread producerThread = new Thread( new ThreadStart(
producer.Produce ) );
producerThread.Name = “Producer”;

Thread consumerThread = new Thread( new ThreadStart(
consumer.Consume ) );
consumerThread.Name = “Consumer”;

producerThread.Start();
consumerThread.Start();
}
}

public class Consumer
{
private SynchronizedBuffer sharedLocation;
private Random randomSleepTime;

public Consumer( SynchronizedBuffer shared, Random random )
{
sharedLocation = shared;
randomSleepTime = random;
}

public void Consume()
{
int sum = 0;

for ( int count = 1; count <= 10; count++ ) { Thread.Sleep(
randomSleepTime.Next( 1, 1001 ) ); sum +=
sharedLocation.Buffer; } Console.WriteLine("{0} read values
totaling: {1}. Terminating {0}.",Thread.CurrentThread.Name,
sum ); } } public class Producer { private SynchronizedBuffer
sharedLocation; private Random randomSleepTime; public
Producer( SynchronizedBuffer shared, Random random ) {
sharedLocation = shared; randomSleepTime = random; } public
```

```
void Produce() { for ( int count = 1; count <= 10; count++ ) { Thread.Sleep( randomSleepTime.Next( 1, 1001 ) ); sharedLocation.Buffer = count; } Console.WriteLine( "{0} done producing. Terminating {0}.",Thread.CurrentThread.Name ); } }
```
[/csharp]