[*https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/com/saravanjs/java22/console/multithreading/producerconsumer/ProducerConsumerExample.java*](https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/com/saravanjs/java22/console/multithreading/producerconsumer/ProducerConsumerExample.java)

*public class* SharedBuffer {

*private* LinkedList<Integer> buffer = *new* LinkedList<>();  
 *private int* capacity;  
  
 *public* SharedBuffer(*int* capacity) {  
 *this*.capacity = capacity;  
 }  
  
 *public void* produce() *throws* InterruptedException {  
 *synchronized* (*this*) {  
 *while* (buffer.size() == capacity) {  
 *// Buffer is full, wait for a consumer to consume* System.*out*.println("Buffer is full, waiting for a consumer to consume");  
 wait();  
 }  
  
 *// Produce an item and add it to the buffer  
 int* item = (*int*) (Math.*random*() \* 100);  
 buffer.add(item);  
 System.*out*.println("Produced: " + item);  
  
 *// Notify consumers that an item is available* notify();  
 }  
 }

}  
  
*public void* consume() *throws* InterruptedException {  
 *synchronized* (*this*) {  
 *while* (buffer.size() == 0) {  
 *// Buffer is empty, wait for a producer to produce* System.*out*.println("Buffer is empty, waiting for a producer to produce");  
 wait();  
 }  
  
 *// Consume an item from the buffer  
 int* item = buffer.removeFirst();  
 System.*out*.println("Consumed: " + item);  
  
 *// Notify producers that space is available in the buffer* notify();  
 }  
 }  
}  
  
  
*public class* Consumer *implements Runnable* {  
 *private* SharedBuffer buffer;  
  
 *public* Consumer(SharedBuffer buffer) {  
 *this*.buffer = buffer;  
 }  
  
 @Override  
 *public void* run() {  
 *try* {  
 *while* (*true*) {  
 buffer.consume();  
 Thread.*sleep*(1000); *// Simulate some work* }  
 } *catch* (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

*public class* Producer *implements Runnable* {  
 *private* SharedBuffer buffer;  
  
 *public* Producer(SharedBuffer buffer) {  
 *this*.buffer = buffer;  
 }  
  
 @Override  
 *public void* run() {  
 *try* {  
 *while* (*true*) {  
 buffer.produce();  
 Thread.*sleep*(10000); *// Simulate some work* }  
 } *catch* (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

*public class* ProducerConsumerExample {  
  
 *public static void* main(String[] args) {  
 SharedBuffer buffer = *new* SharedBuffer(5);  
  
 Thread producerThread = *new* Thread(*new* Producer(buffer));  
 Thread consumerThread = *new* Thread(*new* Consumer(buffer));  
  
 producerThread.start();  
 consumerThread.start();  
 }  
}

Produced: 51

Consumed: 51

Buffer is empty, waiting for a producer to produce

Produced: 67

Consumed: 67

Buffer is empty, waiting for a producer to produce

...