BlockingQueue

BlockingQueue is a thread-safe queue that supports operations that wait for the queue

to become non-empty when retrieving an element and wait for space to become available

in the queue when storing an element.

*class* Producer *implements Runnable* {  
 *private BlockingQueue*<String> queue;  
  
 *public* Producer(*BlockingQueue*<String> queue) {  
 *this*.queue = queue;  
 }  
  
 @Override  
 *public void* run() {  
 String[] messages = {  
 "First message",  
 "Second message",  
 "Third message",  
 "Fourth message",  
 "End"  
 };  
  
 *try* {  
 *for* (String message : messages) {  
 queue.put(message);  
 System.*out*.println("Produced: " + message);  
 Thread.*sleep*(500);  
 }  
 } *catch* (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Thread interrupted");  
 }  
 }  
}  
  
*class* Consumer *implements Runnable* {  
 *private BlockingQueue*<String> queue;  
  
 *public* Consumer(*BlockingQueue*<String> queue) {  
 *this*.queue = queue;  
 }  
  
 @Override  
 *public void* run() {  
 *try* {  
 String message;  
 *while* (!"End".equals(message = queue.take())) {  
 System.*out*.println("Consumed: " + message);  
 Thread.*sleep*(1000);  
 }  
 } *catch* (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Thread interrupted");  
 }  
 }  
}  
  
*public class* BlockingQueue2 {  
 *public static void* main(String[] args) {  
 *BlockingQueue*<String> queue = *new* ArrayBlockingQueue<>(10);  
 Thread producer = *new* Thread(*new* Producer(queue));  
 Thread consumer = *new* Thread(*new* Consumer(queue));  
  
 producer.start();  
 consumer.start();  
 }  
}

C:\java\jdk22\bin\java.exe

Produced: First message

Consumed: First message

Produced: Second message

Consumed: Second message

Produced: Third message

Produced: Fourth message

Consumed: Third message

Produced: End

Consumed: Fourth message