**Blocking Queues**

阻塞队列是一种队列，当尝试出列而队列为空时、或者尝试入列而队列已满时会阻塞。尝试从一个空队列出列的线程被阻塞直到某个其他线程插入一条到队列中。尝试从一个满队列入列的线程被阻塞直到某个其它线程在队列中创造出空位，或者通过出列一项或者通过完全清空整个队列。

这是一个表格展示两个线程在一个阻塞队列上合作：

|  |
| --- |
| A BlockingQueue with one thread putting into it, and another thread taking from it. |
| **BlockingQueue带有一个线程往里放，另一个线程往外取** |

Java 5在java.util.concurrent包中带来的阻塞队列实现。你可以读到关于那个类在我的[**java.util.concurrent.BlockingQueue**](http://tutorials.jenkov.com/java-util-concurrent/blockingqueue.html)教程中。即使Java 5带来阻塞队列的实现，了解实现背后的原理也是有用的。

**Blocking Queue Implementation**

阻塞队列的实现看起来类似于一个有界信号量([**Bounded Semaphore**](http://tutorials.jenkov.com/java-concurrency/semaphores.html#bounded))。这是一个阻塞队列的简单实现：

public class BlockingQueue {

private List queue = new LinkedList();

private int limit = 10;

public BlockingQueue(int limit){

this.limit = limit;

}

public synchronized void enqueue(Object item)

throws InterruptedException {

while(this.queue.size() == this.limit) {

wait();

}

if(this.queue.size() == 0) {

notifyAll();

}

this.queue.add(item);

}

public synchronized Object dequeue()

throws InterruptedException{

while(this.queue.size() == 0){

wait();

}

if(this.queue.size() == this.limit){

notifyAll();

}

return this.queue.remove(0);

}

}

注意notifyAll()只有在队列大小等于大小边界(0或者极限)时才被enqueue()和dequeue()调用。如果队列大小不等于或者任一种边界当enqueue()或dequeue()被调用，不存在有线程等待进队或出队项目。