简单的队列的层次结构：最简单的两个对列直接实现Queue，继承AbstractQueue方法。



对列提供了额外的插入、提取、分析操作。每一种方法提供了两个版本的方法：如果失败，一个直接抛出异常，另外一个返回特殊的值（null或者false,根据操作不同而不同）。

|  |  |  |
| --- | --- | --- |
|  | 抛出异常 | 返回特殊的值 |
| Insert | add(**e**) | offer(e) |
| Remove | Remove() | Poll() |
| Examine | Element() | Peek() |

AbstractQueue的实现，add(),remove,element，调用对应的offer,poll,peek,在返回null或者false直接抛出异常。

**非阻塞算法：**

实现对列使用单链表，包含一个头指针，一个尾指针。在Valois’s算法中，头指针永远指向一个哨兵节点（链表的第一个节点）。尾指针指向最后一个或者倒数第二节点。这个算法使用cas,加上一个修改计数器避免ABA问题。为了允许出列过程自由出列节点，出列操作确保尾节点尾指针既不会指向出列的节点，也不会指向它的前驱。这意味着出队的节点可以安全的被重复使用。

为了在多个读取序列上获取一致的值，需要重新检测之前的值以确保他们没有被改变。

**两把锁队列结构算法：**

这个算法使用两个分离的锁head和tail锁，为了实现并发的出队、入队，在链表头部引入了一个哨兵节点。由于有这个哨兵节点，入队永远不会访问头指针，出队永远不会访问尾指针，这样就避免了可能的死锁问题（进程已不同的序列访问锁）。

ConcurrentLinkedQueue的实现：

1. offer(E e):入队操作：如果有其他节点正在添加操作，轮询检测等待其他节点添加完成。没有其他节点正在入队操作，执行入队操作。
2. poll():出队操作：如果其他节点正在进行出队操作，帮助其他节点完成操作，如果没有，直接进行出队操作

PriorityQueue的实现：

使用平衡二叉树实现

阻塞对列：

支持在取数据的时候等待对列非空，在存储数据的时候等待对列有空间。阻塞对列的方法都有四个版本，支持不能立即满足但是可以在未来满足的操作：第一版本直接抛出异常，第二个版本返回特殊的值，第三个版本无限阻塞当前的操作，第四个版本阻塞给定的时间后放弃。阻塞对列的实现是使用生产者-消费者模式的队列，但是额外支持Collection接口的操作。阻塞对列的实现是线程安全的。所有对列方法都是原子的操作，都是用内部锁或者其他形式的并发控制。但是，批量的集合操作没必要保证原子性。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 抛出异常 | 返回特殊值 | 阻塞 | 超时 |
| Insert | Add(e) | Offer(e) | Put(e) | Offer(e,time,unit) |
| Remove | Rmove() | Poll() | Take() | Poll(time,unit) |
| Examine | Element() | Peek() | Not support | Not support |

1. 接口实现视角：



2. 继承视角



1. ArrayBlockingQueue内部结构使用数组，记录takeIndex,putIndex减少数组移动操作，当takenIndex或putindex达到length的时候，将其置为0实现循环链表的效果。使用条件锁wait和signal来实现加锁和释放锁。
2. DelayQueue内部使用PriorityQueue保存每一个元素。在取元素的时，如果未达到延迟时间会等待，使用条件锁来实现。
3. LinkedBlockingQueue:依赖于链表实现，对列的元素采用先进先出顺序。对列的头元素是在对列中最长的元素。尾是在对列中时间最短的元素。新元素插入对尾，从对头获取元素。链表一般比依赖数组的队列有更高的吞吐量，但是在大多数并发应用中，不能预测性能。使用方法fullylock,fullyunlock内部保证顺序一致，解决锁竞争问题。
4. PriorityBlockingQueue无限的阻塞对列，与PriorityQueue使用同样的原则，提供了阻塞获取的操作。这个对列是逻辑的无限。可能会由于资源错误导致失败（引起outofMemoryError）
5. SynchronousQueue,每一次插入操作必须等待其他线程的并大删除操作，反之亦然。没有任何内部容量，甚至一个容量也没有。不能从SynchronousQueue中peek，因为元素仅仅在尝试删除它的时候，除非其他线程在尝试删除一个元素，否则不能插入。不能进行迭代操作。对列的头是尝试添添加的线程的第一个线程的第一个元素。如果没有这样的对列线程，没有任何元素是可靠的。