ConcurrentLinkedQueue

ConcurrentLinkedQueue是一个基于链表的无界非阻塞队列，并且是线程安全的，它采用的是先进先出的规则，内部采用的CAS算法实

方法摘要	
boolean	**add**(E e) 将指定元素插入此队列的尾部。
boolean	**contains**(Object o) 如果此队列包含指定元素，则返回 true。
boolean	**isEmpty**() 如果此队列不包含任何元素，则返回 true。
Iterator<E>	**iterator**() 返回在此队列元素上以恰当顺序进行迭代的迭代器。
boolean	**offer**(E e) 将指定元素插入此队列的尾部。
E	**peek**() 获取但不移除此队列的头；如果此队列为空，则返回 null。
E	**poll**() 获取并移除此队列的头，如果此队列为空，则返回 null。
boolean	**remove**(Object o) 从队列中移除指定元素的单个实例（如果存在）。
int	**size**() 返回此队列中的元素数量。
Object[]	**toArray**() 返回以恰当顺序包含此队列所有元素的数组。
<T> T[]	**toArray**(T[] a) 返回以恰当顺序包含此队列所有元素的数组；返回数组的运行时类型是指定数组的运行时类型。

LinkedBlockingQueue

LinkedBlockingQueue内部由单链表实现的有界阻塞队列，先进先出。添加元素和获取元素都有独立的锁，也就是说LinkedBlockingQ
执行。LinkedBlockingQueue采用可重入锁(ReentrantLock)来保证在并发情况下的线程安全，如果不指定长度，默认长度是 Integer.ma

ArrayBlockingQueue

同LinkedBlockingQueue一样的队列，不过内部是采用的数组实现

PriorityBlockingQueue

PriorityBlockingQueue 是带 优 先级的无界 阻塞队列，每次出队都返回优先级最高或者 最低的元素。其内部是使用平衡二叉树堆实现的
序。默认使用对象的 compareTo 方法提供比较规则，如果你需要自定义比较规则则可以自定义 comparators

DelayQueue

DelayQueue 并发队列是 一个无界阻塞延迟队列，队列中的每个元素都有个过期时间，当从队列获取元素时，只有过期元素才会出队列