### 并发集合

普通集合：List set map，并发环境下，遍历的过程中不容许更新操作（增删改）；

并发集合：Concurrent Collection；

分类：

**非阻塞式集合（Non-Blocking Collection） ConcurrentLinkedDeque**

这类集合也包括添加和移除数据的方法。如果方法不能立即被执行，则返回null或抛出异常，但是调用这个方法的线程不会被阻塞。

实例

添加大量的数据到一个列表中；从同一个列表中移除大量的数据。

**阻塞式集合（Blocking Collection） LinkedBlockingDeque**

阻塞式集合（Blocking Collection）：这类集合包括添加和移除数据的方法。当集合已满或为空时，被调用的添加或者移除方法就不能立即被执行，那么调用这个方法的线程将被阻塞，一直到该方法可以被成功执行。

数据结构：DataStructure

**ConcurrentLinkedDeque**

**[add](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentLinkedQueue.html" \l "add(E))**([E](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentLinkedQueue.html" \o "ConcurrentLinkedQueue 中的类型参数) e)         将指定元素插入此队列的尾部。

**[addAll](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \l "addAll-java.util.Collection-)**(**[Collection](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/Collection.html" \o "interface in java.util)**<? extends **[E](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \o "type parameter in ConcurrentLinkedDeque)**> c) 按指定集合的迭代器返回的顺序将指定集合中的所有元素追加到该deque的末尾。

**[addFirst](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \l "addFirst-E-)**(**[E](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \o "type parameter in ConcurrentLinkedDeque)** e) 在此deque前面插入指定的元素。

**[addLast](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \l "addLast-E-)**(**[E](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \o "type parameter in ConcurrentLinkedDeque)** e) 在此deque的末尾插入指定的元素。

**[clear](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \l "clear--)**() 从这个deque中删除所有的元素。

**[offer](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentLinkedQueue.html" \l "offer(E))**([E](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentLinkedQueue.html" \o "ConcurrentLinkedQueue 中的类型参数) e)     将指定元素插入此队列的尾部。

**[poll](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentLinkedQueue.html" \l "poll())**()      获取并移除此队列的头，如果此队列为空，则返回 null。

peek() 同poll()。

**[getFirst](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \l "getFirst--)**()、**[getLast](https://www.matools.com/file/manual/jdk_api_1.8_google/java/util/concurrent/ConcurrentLinkedDeque.html" \l "getLast--)**()

存储元素Node，有头结点，尾节点，双向链表基于链表的无限双端队列，线程安全（CAS保证）

**LinkedBlockingDeque**

|  |
| --- |
| **[LinkedBlockingDeque](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/LinkedBlockingDeque.html" \l "LinkedBlockingDeque())**()            创建一个容量为 [Integer.MAX\_VALUE](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/lang/Integer.html" \l "MAX_VALUE) 的 LinkedBlockingDeque。 |
| **[LinkedBlockingDeque](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/LinkedBlockingDeque.html" \l "LinkedBlockingDeque(java.util.Collection))**([Collection](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/Collection.html" \o "java.util 中的接口)<? extends [E](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/LinkedBlockingDeque.html" \o "LinkedBlockingDeque 中的类型参数)> c)            创建一个容量为 [Integer.MAX\_VALUE](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/lang/Integer.html" \l "MAX_VALUE) 的 LinkedBlockingDeque，最初包含给定 collection 的元素，以该 collection 迭代器的遍历顺序添加。 |
| **[LinkedBlockingDeque](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/LinkedBlockingDeque.html" \l "LinkedBlockingDeque(int))**(int capacity)            创建一个具有给定（固定）容量的 LinkedBlockingDeque。 |

**[put](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/LinkedBlockingDeque.html" \l "put(E))**([E](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/LinkedBlockingDeque.html" \o "LinkedBlockingDeque 中的类型参数) e)   
          将指定的元素插入此双端队列表示的队列中（即此双端队列的尾部），必要时将一直等待可用空间。

**[take](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/LinkedBlockingDeque.html" \l "take())**()   
          获取并移除此双端队列表示的队列的头部（即此双端队列的第一个元素），必要时将一直等待可用元素。

**LinkedTransferQueue**

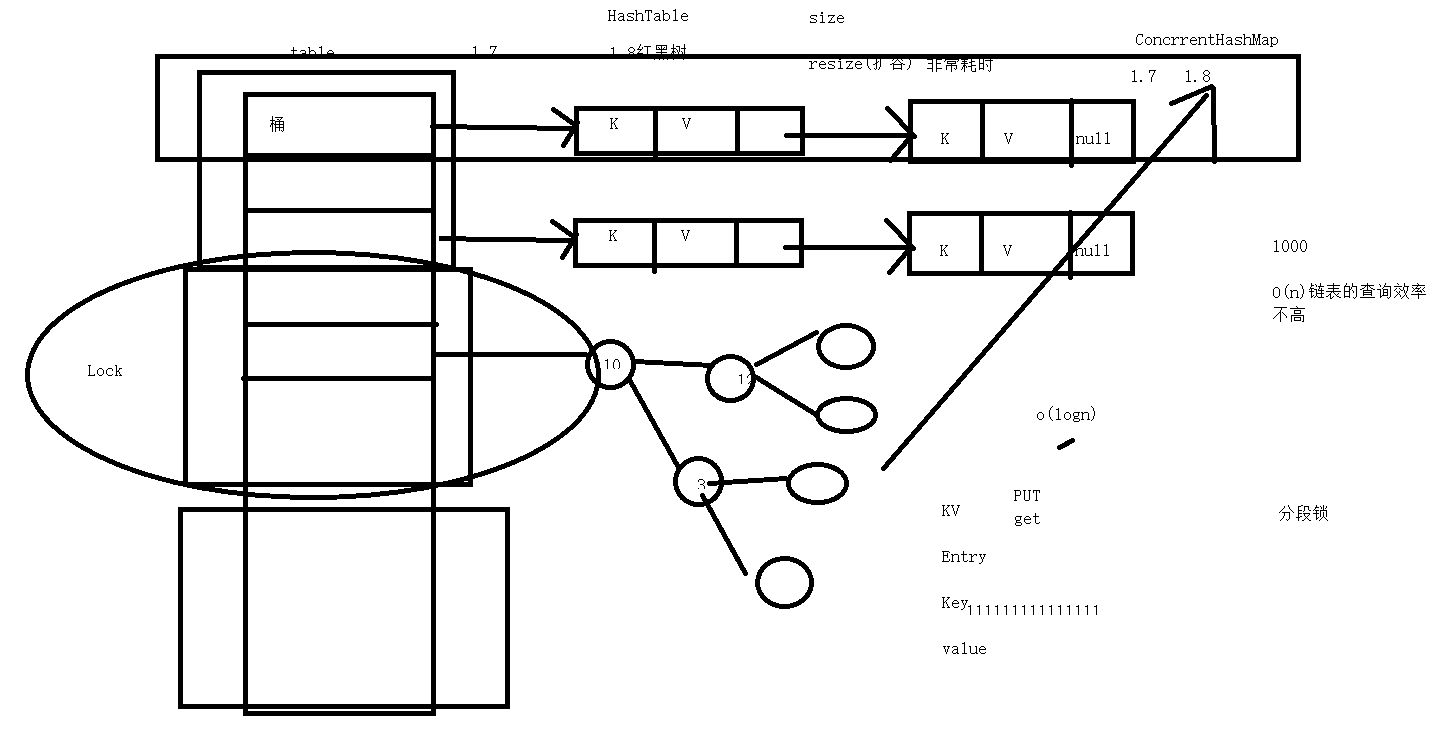
[ArrayBlockingQueue](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ArrayBlockingQueue.html" \o "java.util.concurrent 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/classFrame)**生产-消费**

**PriorityBlockingQueue 优先级**

[ConcurrentHashMap](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentHashMap.html" \o "java.util.concurrent 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/classFrame)   
[ConcurrentLinkedQueue](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentLinkedQueue.html" \o "java.util.concurrent 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/classFrame)   
[ConcurrentSkipListMap](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentSkipListMap.html" \o "java.util.concurrent 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/classFrame)   
[ConcurrentSkipListSet](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/ConcurrentSkipListSet.html" \o "java.util.concurrent 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/classFrame)   
[CopyOnWriteArrayList](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/CopyOnWriteArrayList.html" \o "java.util.concurrent 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/classFrame)  数组拷贝 消耗内存 适合读多写少  
[CopyOnWriteArraySet](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/CopyOnWriteArraySet.html" \o "java.util.concurrent 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/classFrame)  数组拷贝 消耗内存 适合读多写少

分析ConcurrentHashMap与HsahMap

**HashMap数组+链表**



原子操作

CAS CompareAndSet

AtomicInteger

A[tomicBoolean](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/atomic/AtomicBoolean.html" \o "java.util.concurrent.atomic 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/atomic/classFrame)   
[AtomicInteger](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/atomic/AtomicInteger.html" \o "java.util.concurrent.atomic 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/atomic/classFrame)

[AtomicLong](http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/atomic/AtomicLong.html" \o "java.util.concurrent.atomic 中的类" \t "http://tool.oschina.net/uploads/apidocs/jdk-zh/java/util/concurrent/atomic/classFrame)