https://www.cnblogs.com/duanxz/p/3169362.html

一 ：Semaphore信号量

资源有限共享

停车场为例

1. forEach 循环 和 Iterrator 迭代器的区别
2. Iterrator : 在循环元素时， 是根据 hasNext 。跟元素下标有密切的关系

当 remove 或 add 集合时 ，就会导致下标变化 然后报错

1. forEach 在循环元素时 ， 不考虑下标 因此在循环过程中可以 删除或添加元素
2. 并发集合 ：java.util.concurrent包下的集合类

ConcurrentCollection集合分类 ：

**A、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。

[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) :实现了Queue接口。内部 类似AQS 里面也定也给一个静态的 Node内部类 ， 然后 ConcurrentLinkedQueeu依赖 volital 修饰的 Node 定义的 head 和 tail （队首 、队尾） ，由于操作 队列就是存数据（入队） 取数据 （出队） 。所以基本都是对 head 和 tail的出操作

[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)   
 [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)   
 [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)   
 [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)

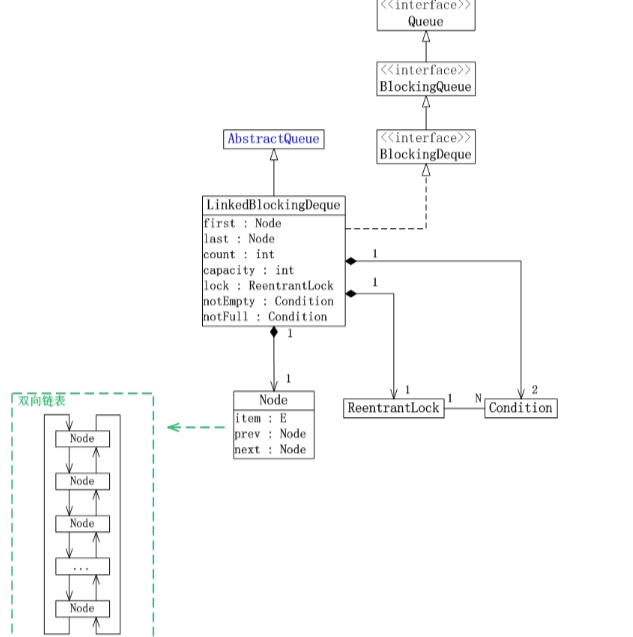
**B、阻塞式集合（Blocking Collection） LinkedBlockingDeque**

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。

Deque是一个双端队列，deque(双端队列) 是 "Double Ended Queue" 的缩写。因此，双端队列是一个可以从任意一端插入或者抽取元素的队列。实现了在队列头和队列尾的高效插入和移除。  
　　BlockingDeque 类是一个双端队列，在不能够插入元素时，它将阻塞住试图插入元素的线程；在不能够抽取元素时，它将阻塞住试图抽取的线程。

[LinkedBlockingDeque](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) : 双向并发阻塞队列 ，该阻塞队列同时支持FIFO(先进先出) FILO(后进先出)

支持从队列同时从队首 、队尾 出队或入队 。并且该线程是线程安全的。在创建时可以执行队列大小，默认大小Integer.MAX\_VALUE



[LinkedBlockingQueue：](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)

分析ConcurrentHashMap

HsahMap

**四、队列的分类 (集合)**

**1、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。

**[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。

Node

**2、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。

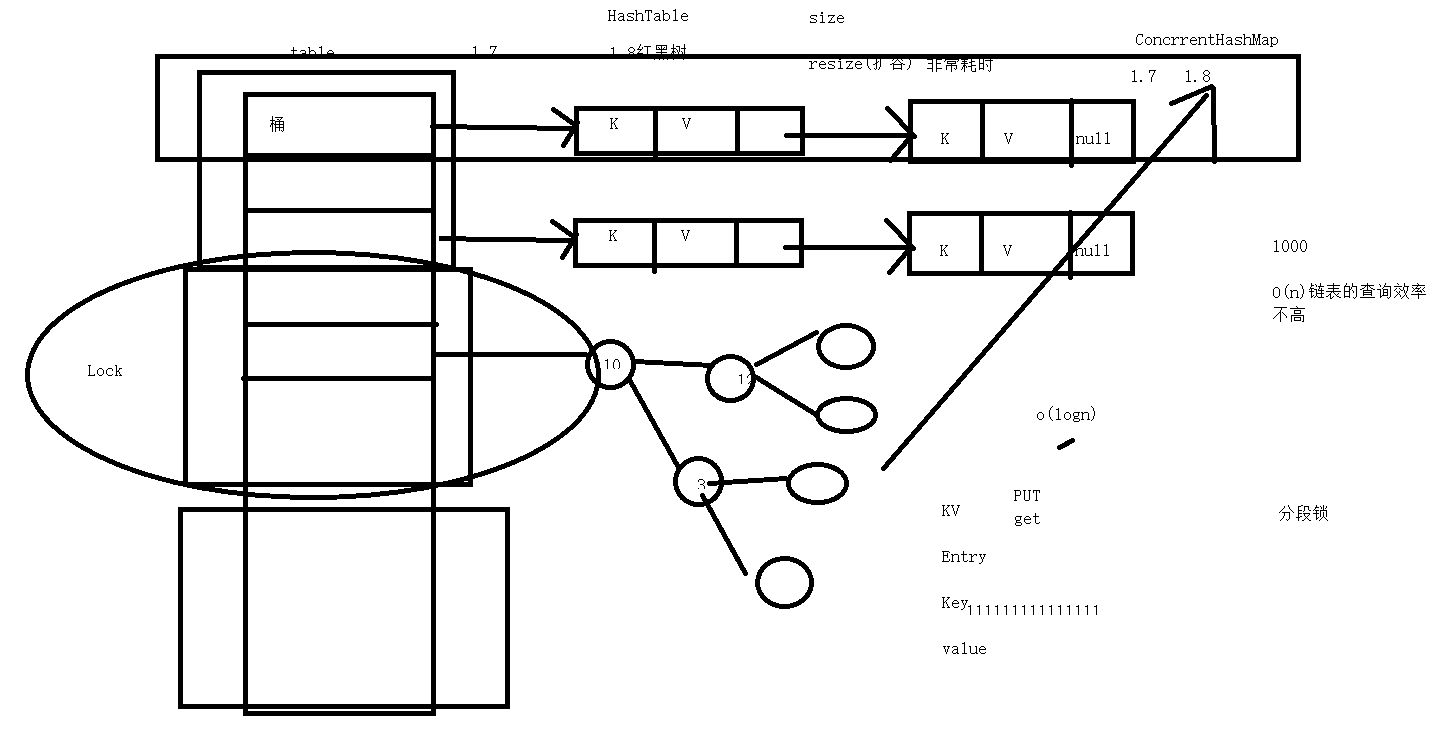
**HashMap entry<k,v>**

**3、LinkedTransferQueue 生产-消费**

**4、PriorityBlockingQueue 优先级**

**HashMap原理：**

**数组+链表**



原子操作

CAS

AtomicInteger

[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)

Long

64

32 32

CAS

CompareAndSet

 此引用所引用的对象类型。

ABA问题

1. -》120--》100
2. 》120

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

100 1---->120 2----->100----3

100 1------120