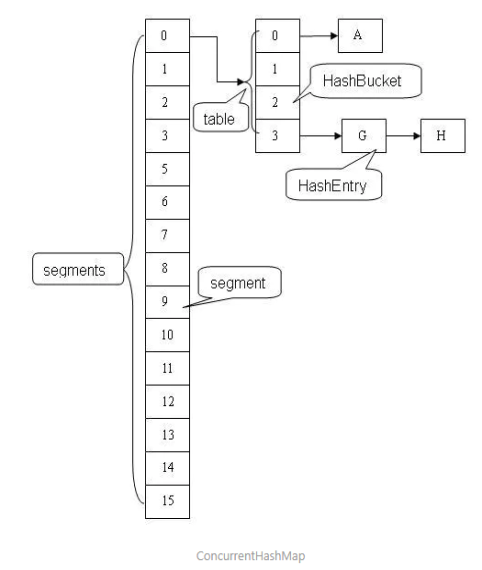
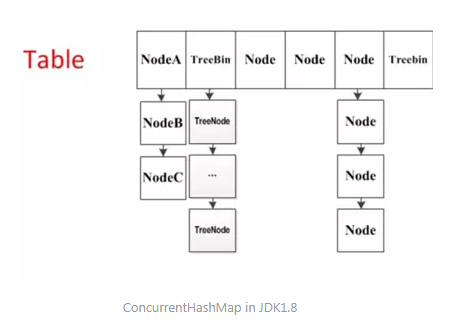
concurrentHashmap就是一个二维数组，加链表，红黑树。



java1.8抛弃了segment分段锁机制。利用CAS + Synchronized



**和HashTable的区别：**

ConcurrentHashMap 是一个并发散列映射表，它允许完全并发的读取，并且支持给定数量的并发更新。  
而HashTable和同步包装器包装的 HashMap，使用一个全局的锁来同步不同线程间的并发访问，同一时间点，只能有一个线程持有锁，也就是说在同一时间点，只能有一个线程能访问容器，这虽然保证多线程间的安全并发访问，但同时也导致对容器的访问变成串行化的了。

这段代码，是resize()中，将就的链表分成两部分，一部分放到低位，一部分放到高位。放高低位这个根据key的hashcode来判断。

do {  
 next = e.next;  
 if ((e.hash & oldCap) == 0) {  
 if (loTail == null)  
 loHead = e;  
 else  
 loTail.next = e;  
 loTail = e;  
 }  
 else {  
 if (hiTail == null)  
 hiHead = e;  
 else  
 hiTail.next = e;  
 hiTail = e;  
 }  
} while ((e = next) != null);

老的链表 a->b->c->d->null

a->c->null

b->d->null

java11 HashMap进行put时，链表采用的尾插，不存在头插的问题。

resize()的时候，链表拆分的时候，会将key分为两部分，一部分放在低位，一部分放在高位。多线程也不会存在循环链。也就不存在get数据的时候，一直循环（死循环）的风险。

下面这篇文章讲1.7和1.8的HashMap和ConcurrentHashMap的对比

<https://blog.csdn.net/weixin_44460333/article/details/86770169>

在ConcurrentHashMap用了volatile保证内存可见性