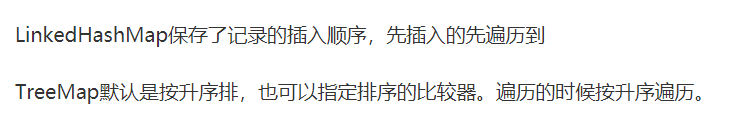
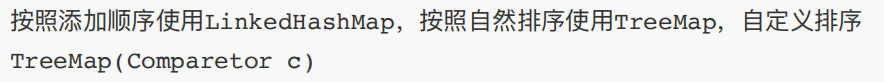
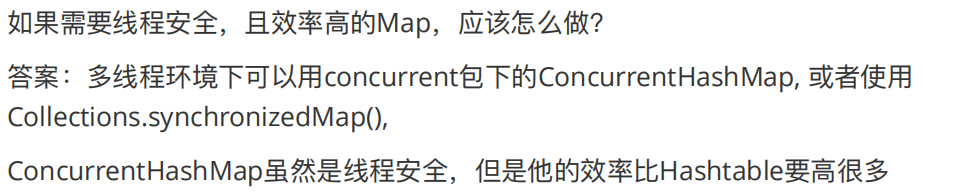


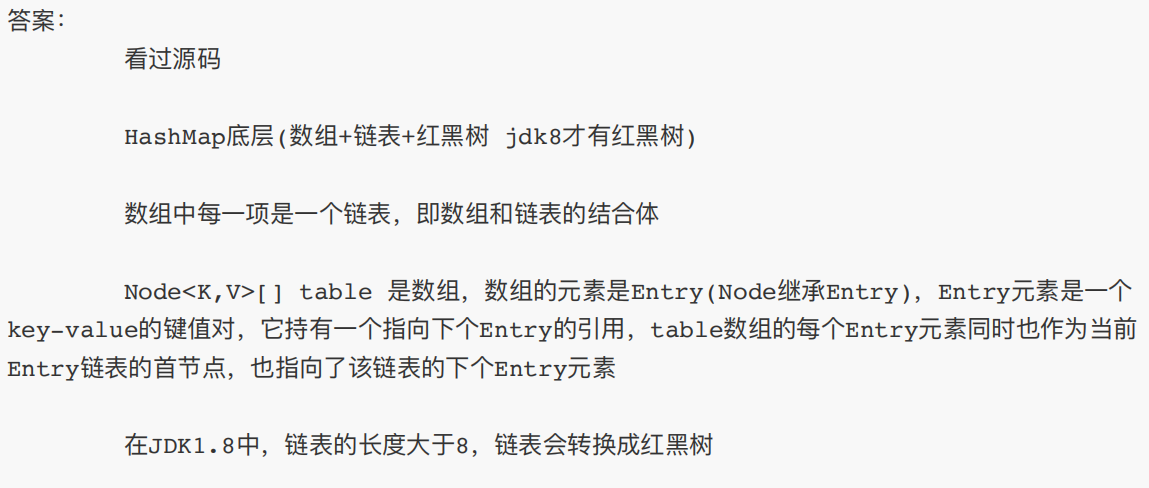
LinkedHashMap默认的构造参数是默认  插入顺序的，就是说你插入的是什么顺序，读出来的就是什么顺序，但是也有访问顺序，就是说你访问了一个key，这个key就跑到了最后面



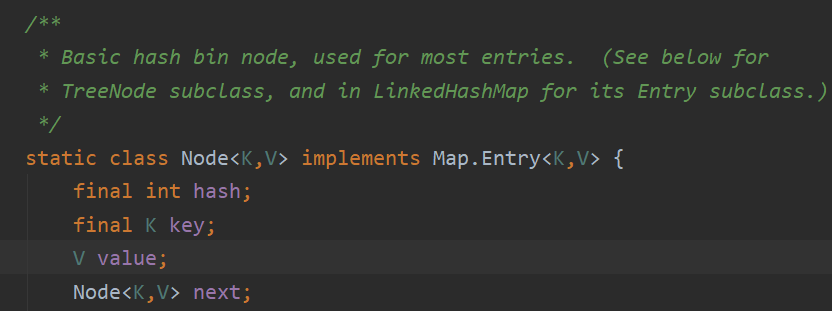


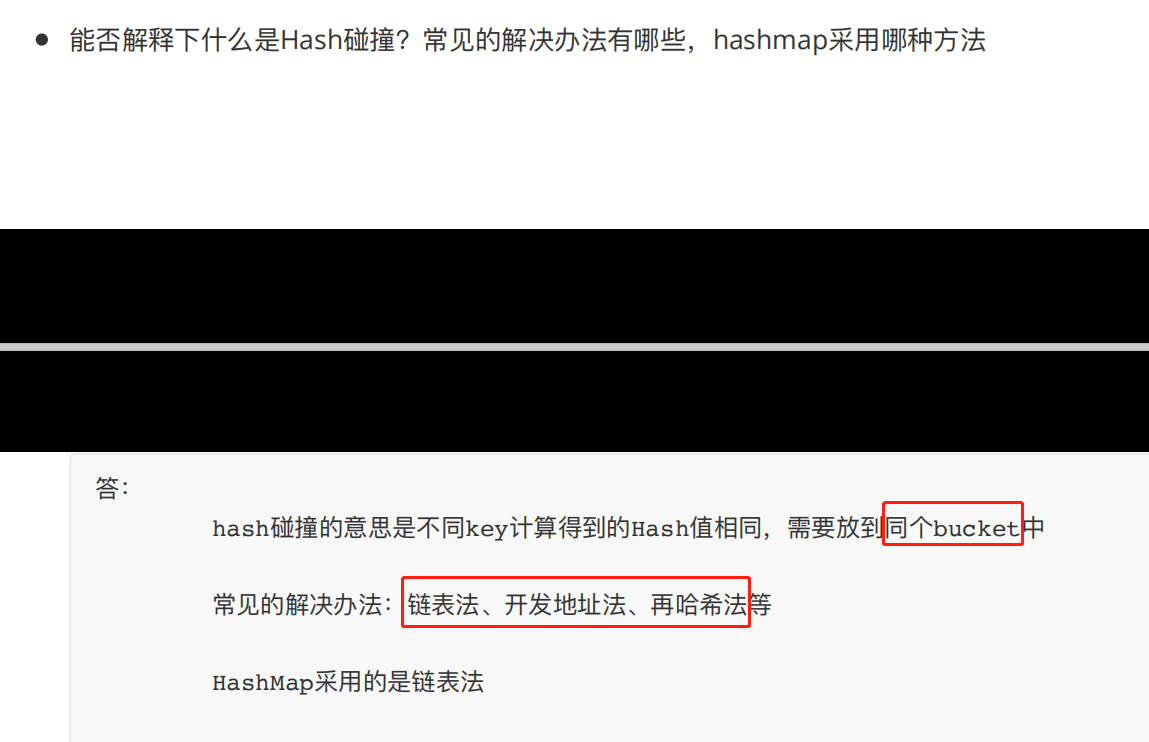






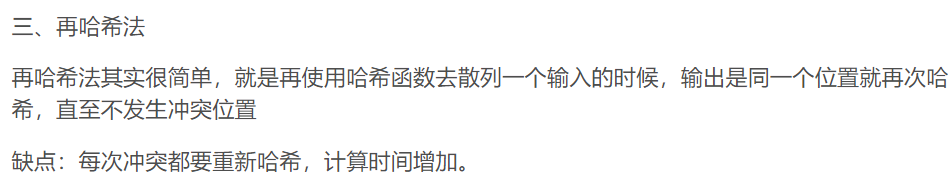
元素是一个node对象，node对象是一个链表

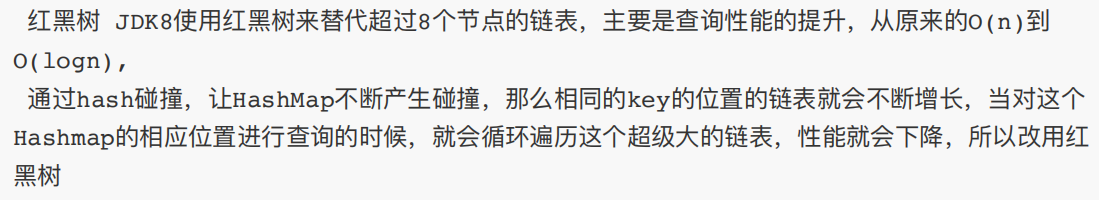


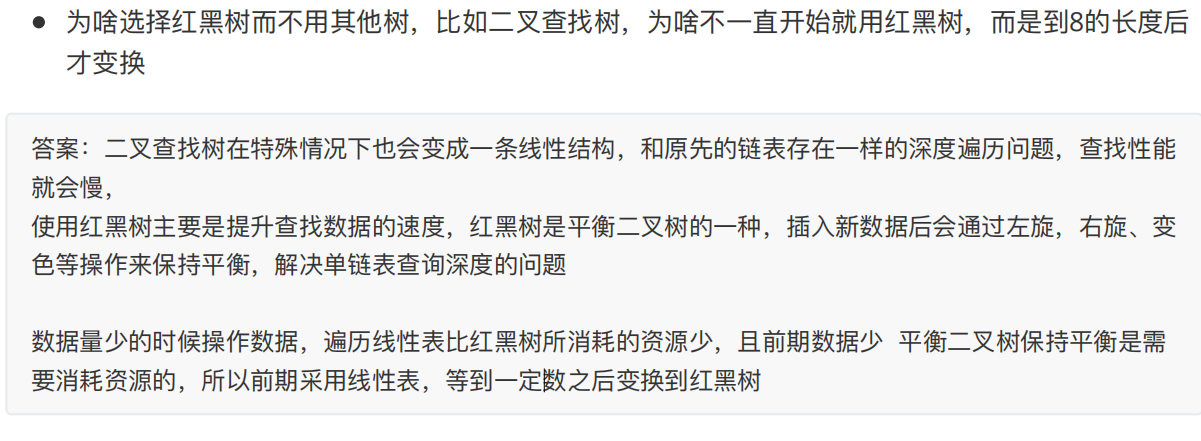


一个数组元素就是一个bucket

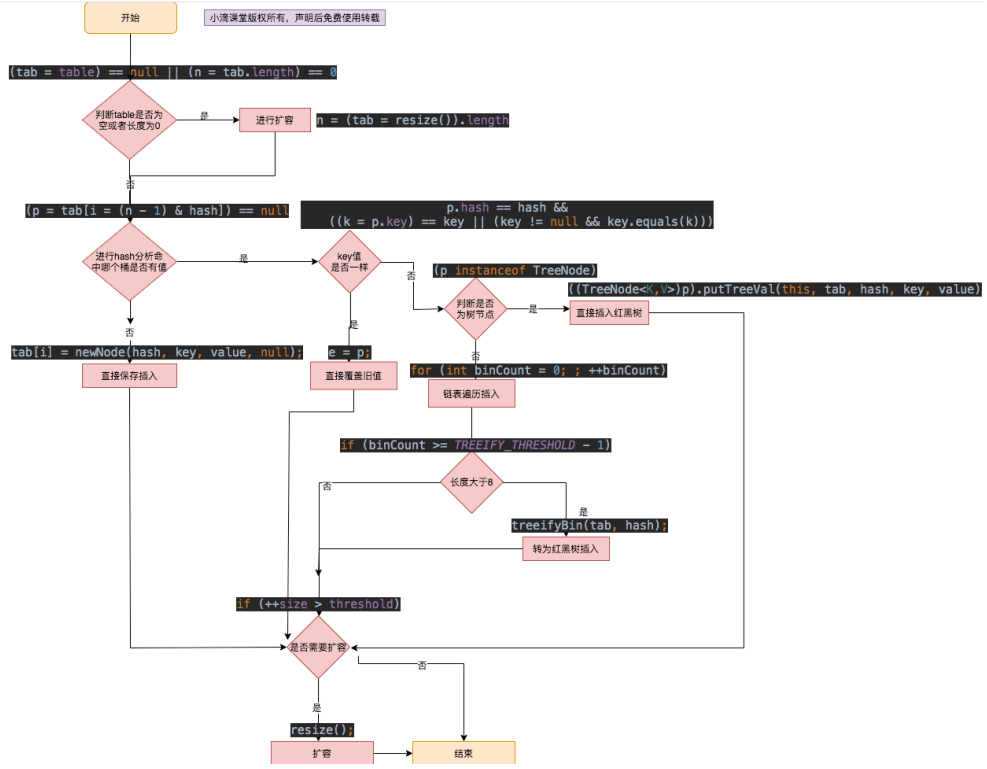
开放地址法<https://cloud.tencent.com/developer/article/1361248>







instanceof用于判断前面的对象是否是后面的类,或者其子类、实现类的实例



HashMap get流程

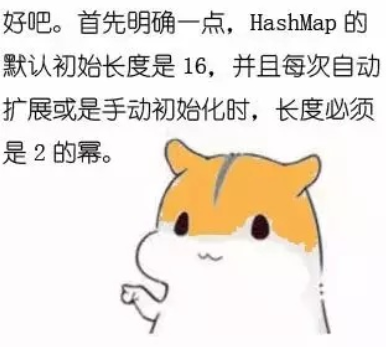
遍历每个桶

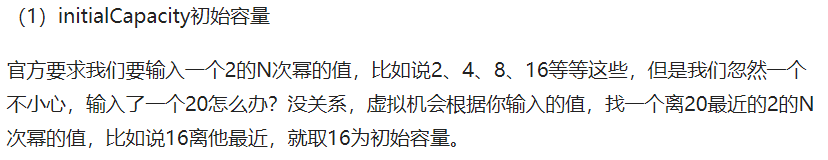
//获取桶⾸节点，hash碰撞概览⼩，通常链表第⼀个节点就是值，没必要去循环遍历，处于效率

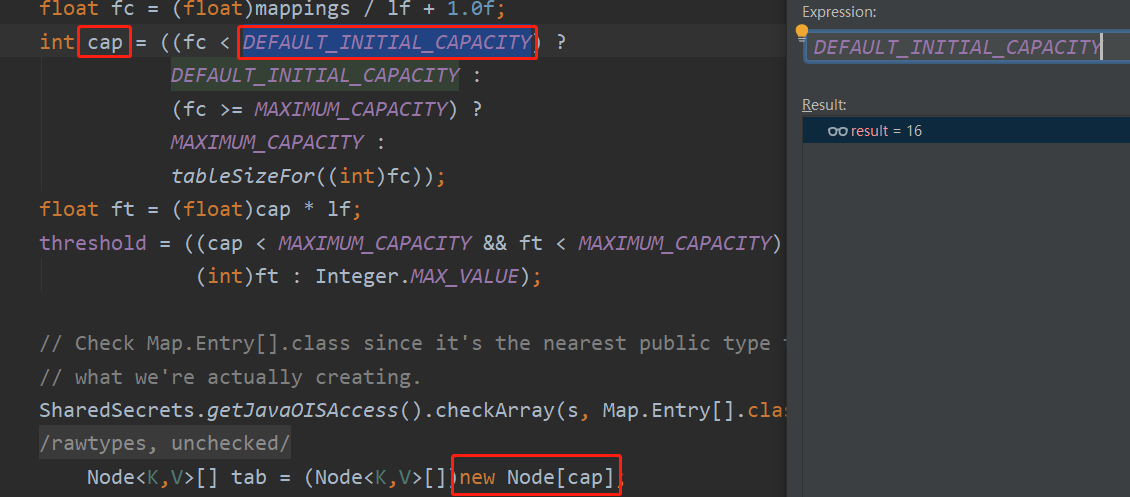
//如果不⽌⼀个节点，就需要循环遍历，存在多个hash碰撞

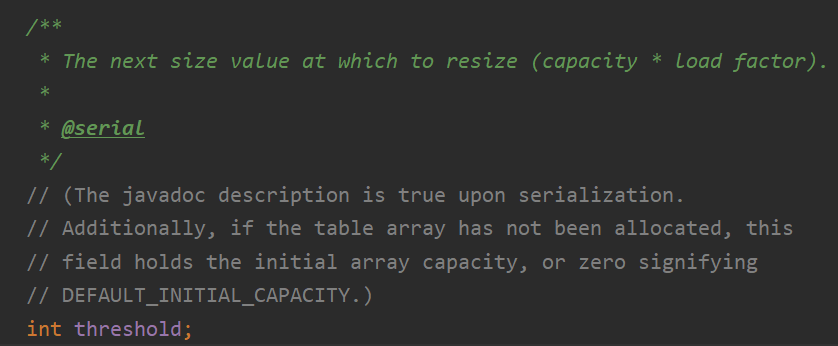
//判断是否是红⿊树，如果是则调⽤树的查找

//链表结构，则循环遍历获取节点

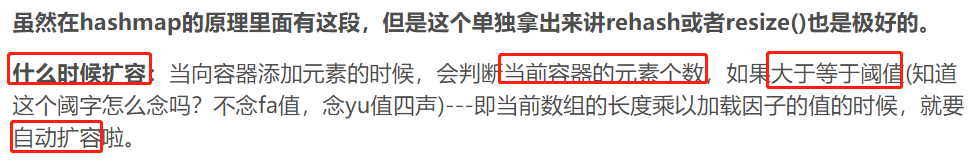




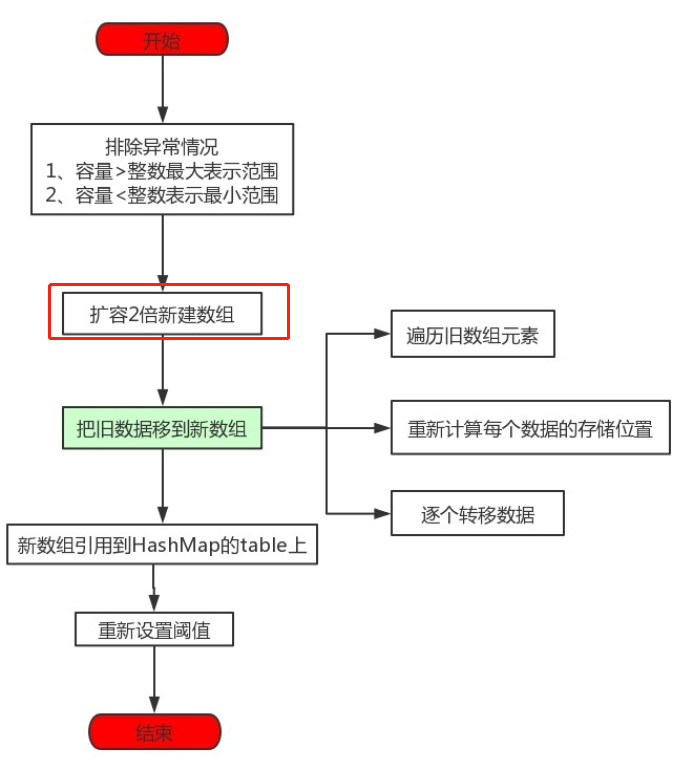


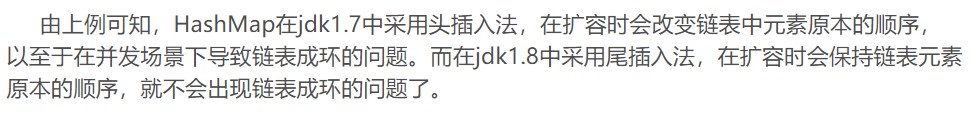


threshold为capacity容量\*load factor负载因子



默认的负载因子大小为0.75，也就是说，当一个map填满了75%的bucket时候，和其它集合类(如ArrayList等)一样，将会创建原来HashMap大小的两倍的bucket数组，来重新调整map的大小，并将原来的对象放入新的bucket数组中。这个过程叫作rehashing，因为它调用hash方法找到新的bucket位置。



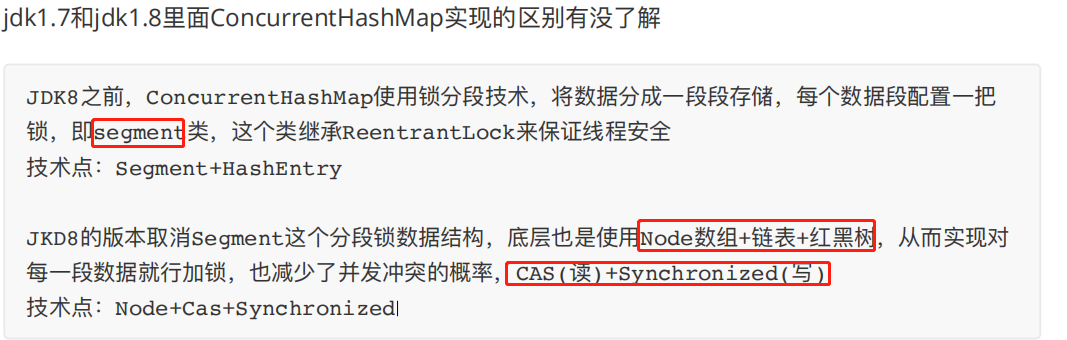


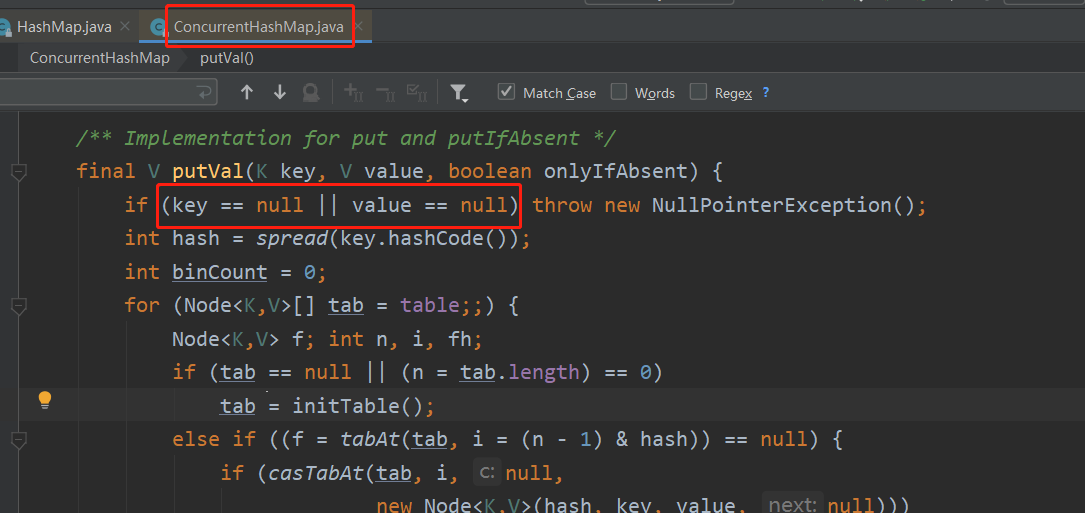
简单来说，两个线程同时触发扩容，现在某个桶有链表1->3,使用头插法会出现1->3->1->3此时就死循环了，而尾插法插入时每经过一次链表节点会进行重复判断，若重复直接break了，所以不会死循环



为何hashmap头插法并发情况下会发生死循环

<https://blog.csdn.net/littlehaes/article/details/105241194>

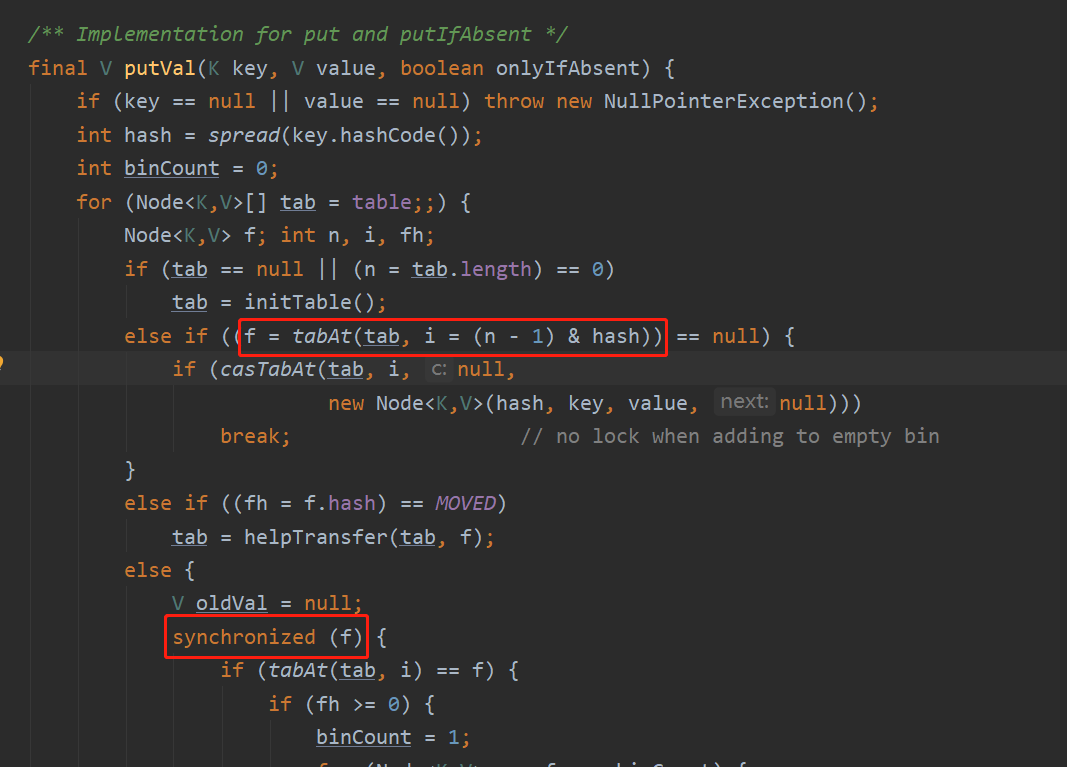




concurrentHashMap key，value不允许为null，允许为‘‘

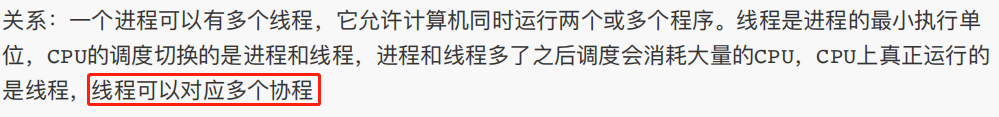


第三点初始化是因为table采用懒加载模式，需要用时才初始化

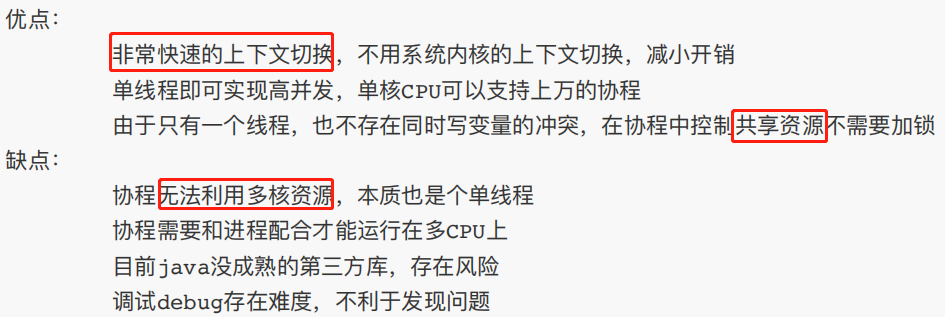


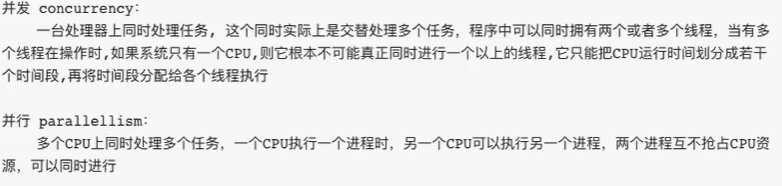
f是第一个node下去的相当于桶，锁住某个桶(而不是整个table,所以叫分段锁)

线程需要CPU的切换



协程是轻量级线程





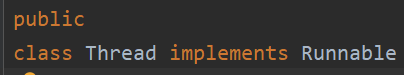
并发指在⼀段时间内宏观上去处理多个任务。 并⾏指同⼀个时刻，多个任务确实真的同时运⾏。

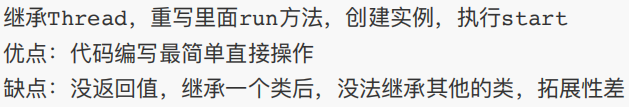
⼀个项⽬经理A和3个程序B C D的故事

单线程 A给B讲完需求再A给c讲完需求再A给D讲完需求

并发：A给B讲完需求，B⾃⼰去实现，期间A继续给C和D讲，不⽤等待某个程序员去完成，期间项⽬经理没空闲下来

并⾏：直接找3个项⽬经理分别分配给3个程序员







Runnable：没返回值，不能直接启动，需要通过构造⼀个Thread实例传递进去启动

是接口，thread是类



线程池：安全⾼性能，复⽤线程