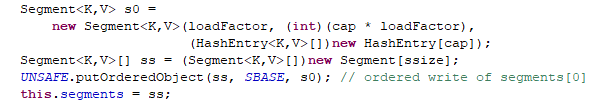
# ConcurrentHashMap（jdk1.7.0\_67）

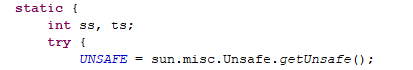
jdk1.7与jdk1.6实现上有很大的区别。jdk1.6的实现是弱一致性的，而jdk1.7通过Unsafe类中的一些方法实现了强一致性。下面我们来具体看一下。

# 延迟了Segment数组中元素的赋值。

jdk1.7中的实现：



在jdk1.7的实现中，在构造器ConcurrentHashMap(int, float, int)中，只初始化了segments数组中的第一个元素，第一个元素在初始化其他数组元素时提供了原型的作用，在put方法中会讲到。



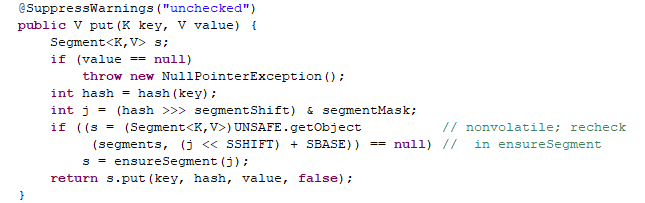
在静态初始化块中初始化了UNSAFE常量。putOrderedObject是putObjectVolatile的延迟实现，putObjectVolatile这个方法的作用使设置的值可以立即被其他线程看到，而putOrderedObject不会立即被其他线程看到，只对final修饰的字段并且被修改时起作用。

jdk1.6中的实现：

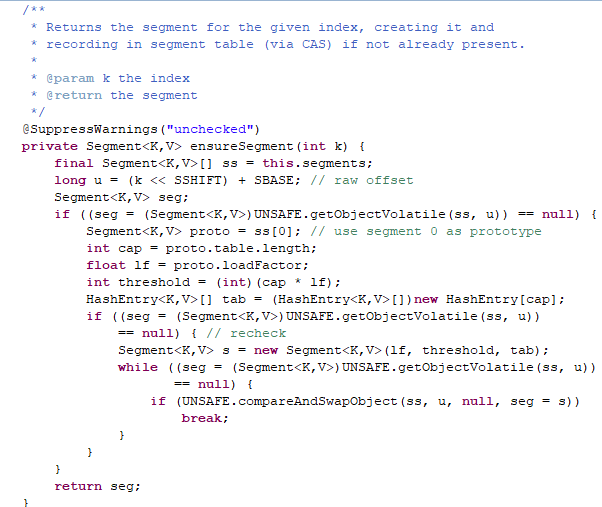


在jdk1.6的实现中，在构造器中就对数组中每一个元素进行了初始化。

# put方法

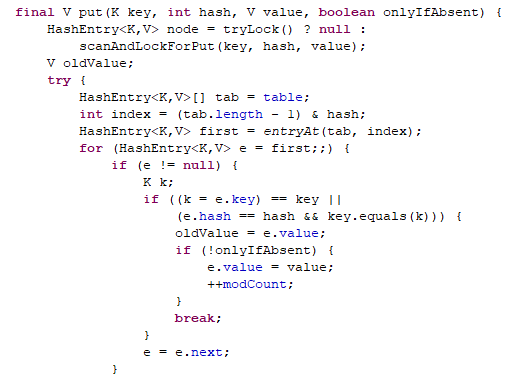
 这里使用了Unsafe的getObject方法获取Segment对象，替代了jdk1.6通过数组下标获取对象的方法。但是getObject不是线程安全，有可能在获取对象的同时有其他线程修改了Segment，导致获取到Segment为null，而实际上Segement已经被设置了值。所以这里重新确认了Sement的值。

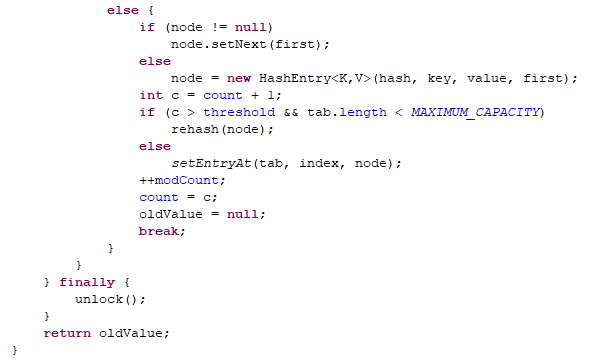
下面看一下ensureSegment方法：

 这个方法的目的是给定一个索引，返回Segment，如果这个Segment不存在那么创建并且设置table属性的值。这里使用了自旋锁的机制和CAS比较交换的方式来设置table的值，确保了线程安全。

我们详细看一下这个方法，前两步获取Segment在segments中的偏序位置，第四步通过getObjectVolatile方法来获取Segment，前面提到过getObjectVolatile，主要是在其他线程修改了Segment之后能立即看到结果。如果Segment为空，拷贝segments[0]的loadFactory， table.length，用来创建HashEntry对象和之后创建的Segment对象。然后在重新检查Segment对象是否为空，为空则设置创建Segment对象，之后采用自旋锁机制和CAS方式将创建的Segment赋值到数组元素中。

回到put方法，获取完Segment对象之后调用了Segment的put方法：

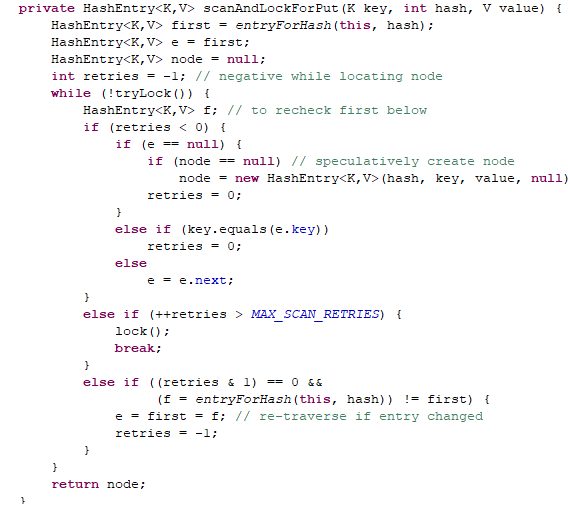




jdk1.7中使用了tryLock方法替代了jdk1.6中的lock方法。tryLock方法当获取到锁时返回true，获取不到时返回false，并不会阻塞线程。利用tryLock的这些特性可以在等待获取锁的时候做一些额外工作，这里就是调用了scanAndLockForPut方法。

tryLock立即获取到锁的情况很简单，无非就是遍历链表，执行插入或者修改操作。

我们重点看一下不能立即获取锁的情况，到底做了哪些额外工作：



这个方法一直在尝试获取锁，当尝试获取锁失败时，遍历HashEntry，如果没有找到，则创建并返回node，再返回之前，首先保证当前线程已经获取了锁。

while代码块一共有四条执行路径：

1. retries < 0：即遍历HashEntry，如果没有找到则创建，并把retries设置为0，如果找到了，把retries设置为0。
2. ++retries > MAX\_SCAN\_RETRIES：



availableProcessors方法返回java虚拟机可用的处理器的数量，MAX\_SCAN\_RETRIES即1或者64，在单核的处理器为1，多核为64。

当找到或创建节点之后，再尝试获取MAX\_SCAN\_RETRIES次锁，如果还没有获取到，则调用lock，阻塞当前线程，直到获取到锁为止。这样做的原因是当其他线程一直持有锁不是放，如果当前线程一直再尝试获取锁，会一直消耗cpu的资源。

1. (retries & 1) == 0：当retries二进制最后一位为0时结果为true，为1时结果为false。

没两次检查一次当前HashEntry是否被修改，如果被修改将e，first重新赋值，retries重新赋值为-1。

1. ：不做任何操作。

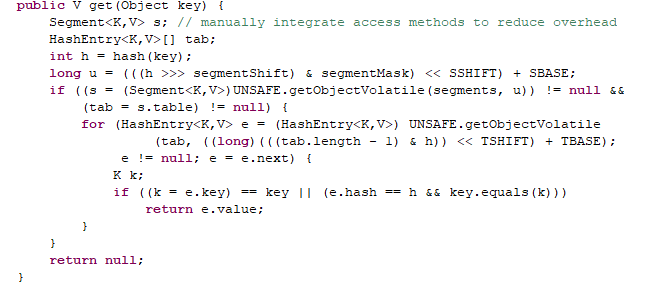
这个方法当当前链表存储了key时，返回null，否则返回新创建的节点。这样做的目的应该是提前缓存了node节点，以便在获取锁之后能够直接使用。

回到put方法：



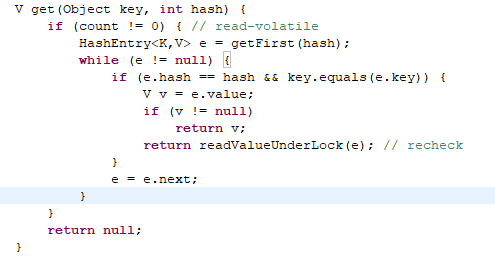
这里当node不为null时，设置了next属性的值。

# get方法



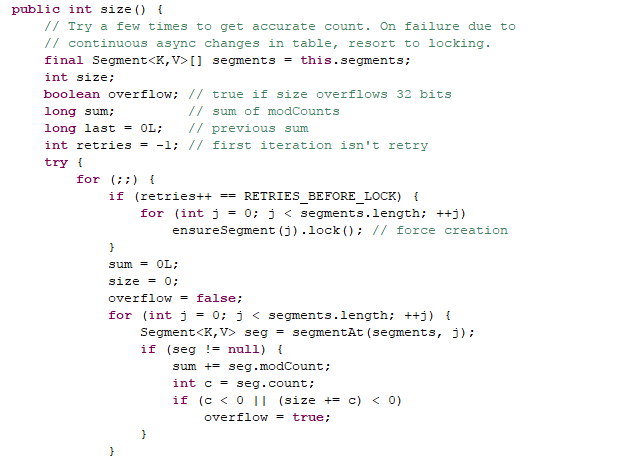
get方法相对比较简单，这里使用的getObjectVolatile方法保证了HashEntry的一致性。

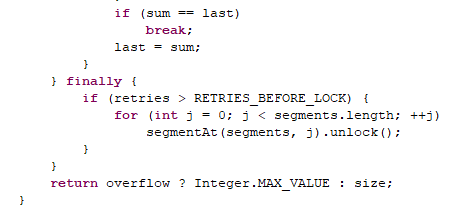
在1.6的实现中：



虽然count是volatile修饰的，但不能保证get获取对象happens-before put设置对象，不能保证HashEntry的强一致性。

# size方法





overflow：超过32位为true，即超出了int的范围为true。

sum：modCount的总和。

last：上一次统计的sum值。

retries：统计重试的次数。当大于RETRIES\_BEFORE\_LOCK时，加锁统计。

当sum == last即两次统计的modCount总和相等时，即在统计过程中没有线程修改ConcurrentHashMap的结构，则跳出循环，返回size；否则，将last设置为sum，再重新统计。当统计次数大于RETRIES\_BEFORE\_LOCK时，加锁统计。