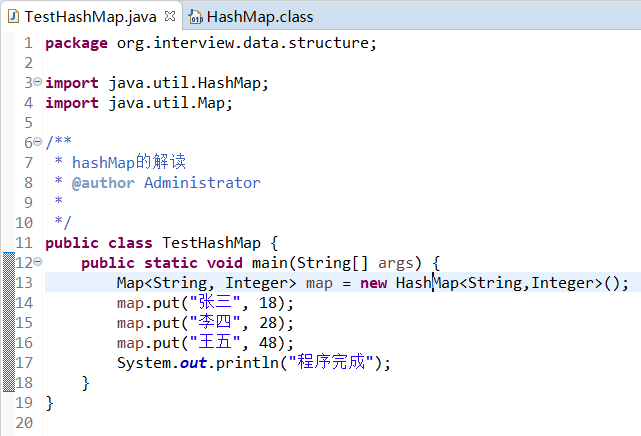
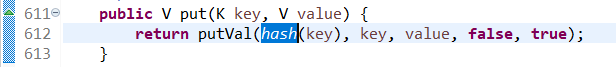
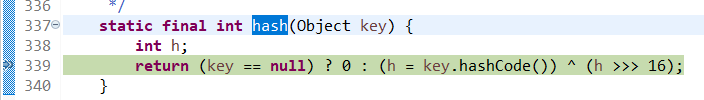
测试代码



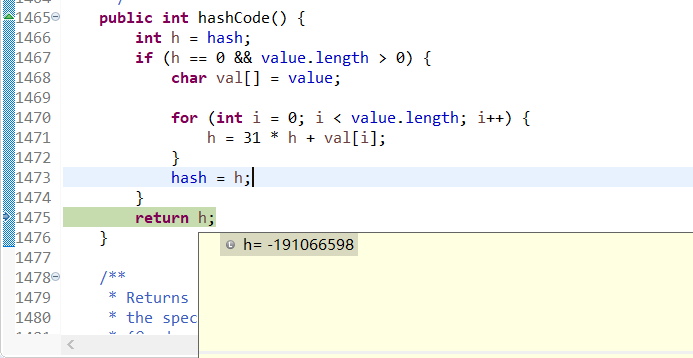
#### 扰动函数

jdk1.8中，hashmap在操作之前会有一个扰动key的hash值得操作。由于integer在Java中占用4个字节，也就是32个二进制表示，所以hashmap的设计者巧妙的将hash值的前面16位和原始值进行位异或操作，以此达到减少hash碰撞的目的。

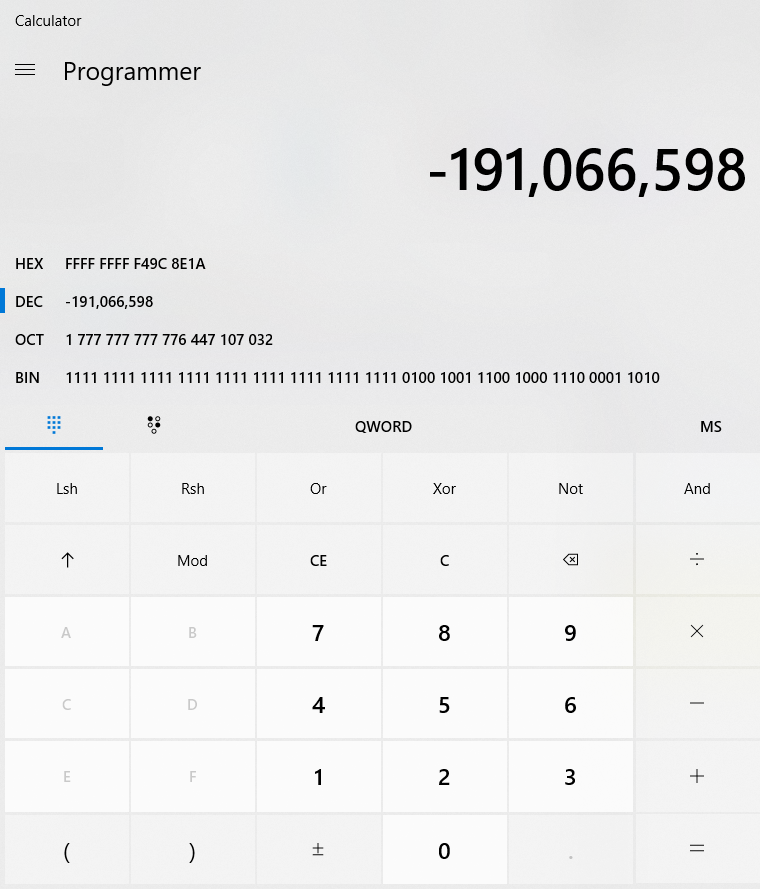




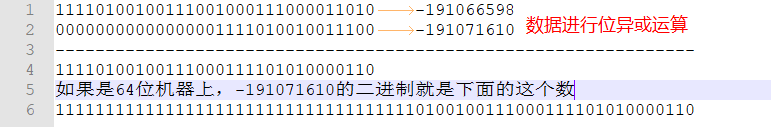
Hash值为-191066598

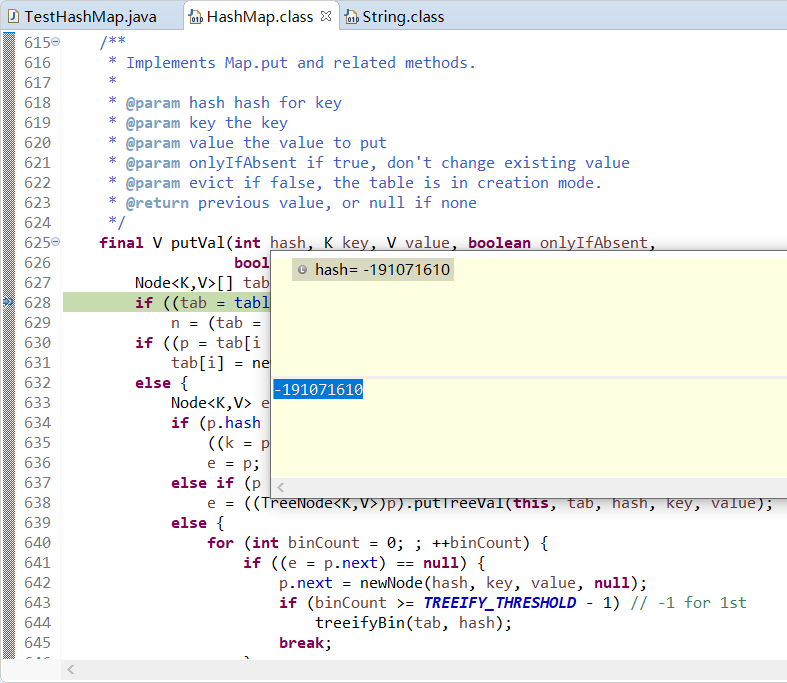


对应的二进制为



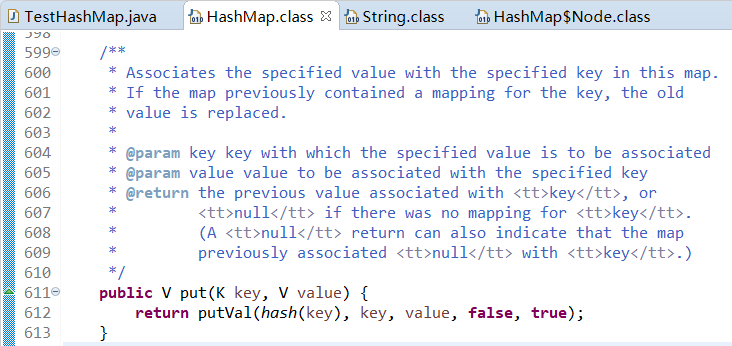
运算过程



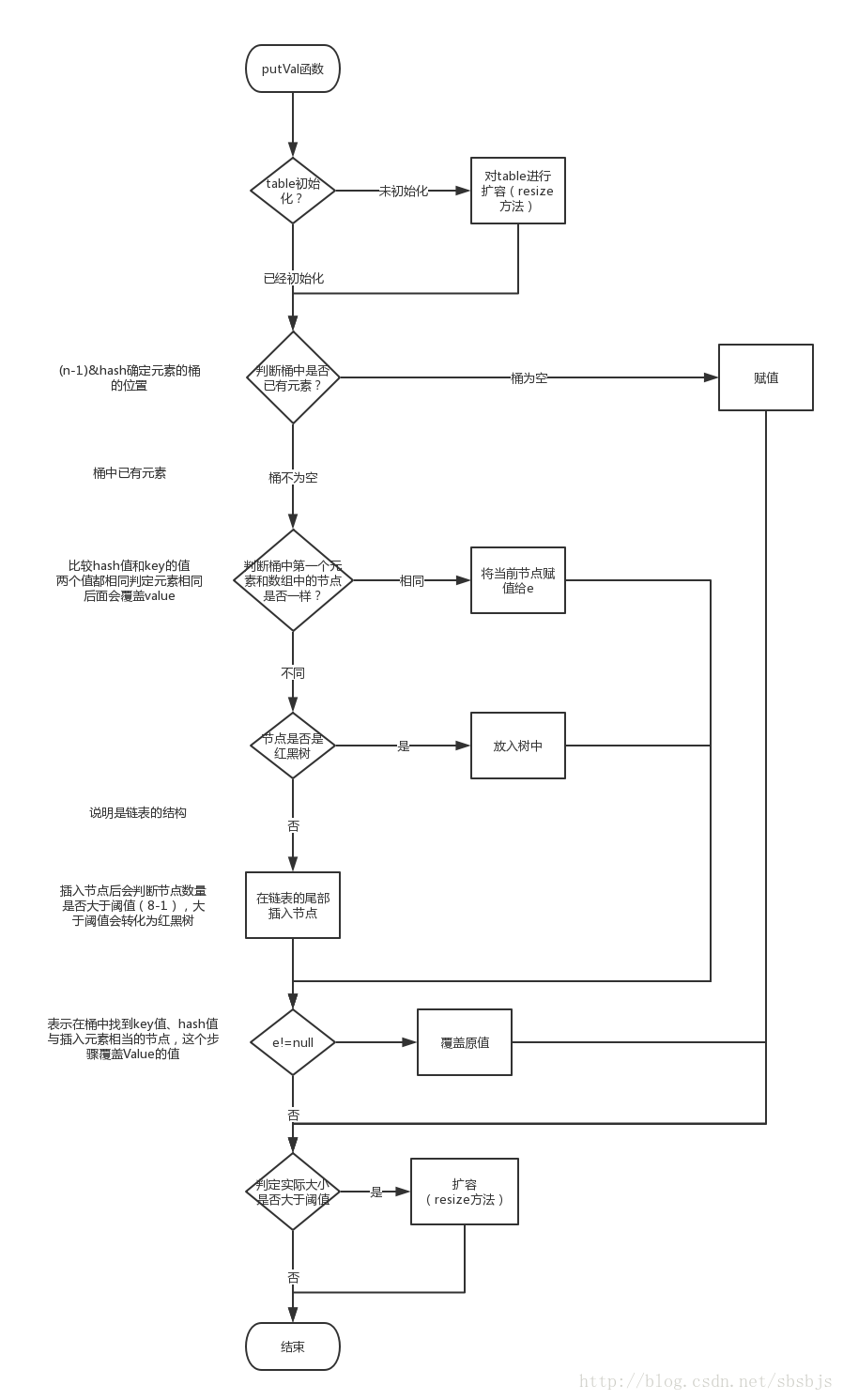


#### put(K, V)方法

put方法调用的是java.util.HashMap.putVal(int, K, V, boolean, boolean)方法



put方法对应的时序图是



解读如下：

628-629：判断table是否需要初始化，未初始化则需要进行扩容

630-631：判断桶是否为空，如果是空则赋值。注意这个地方tab[i = (n - 1) & hash]，这里用位运算而不是取余运算，其实是一个优化。这里之所以能这样优化是因为n = tab.length，由于hashmap的大小是2^n，所以应该是10000…这种，如果减一的话就是011111…，取位与运算的话，刚好能够达到取模运算的效果，并且位与运算效率会更快。

632：table不需要初始化且table[i]对应的hash桶不为空的情况

634-636，647-649，653-658：前面两个段落是为了将hashmap中和当前存放键值对相同key的那个对象找到，第三段是决定是否替换值。替换值得标准是二选一：onlyIfAbsent为false或者当前hashmap中这个对象对应的值为空。在hashmap中，只有putIfAbsent()方法会将onlyIfAbsent设置为true。所以，如果想要代码里面设置hashmap存放时不替换相同key的值得value，可以用putIfAbsent()方法

637-638：如果hash桶对象是红黑树结构，则存入到红黑树中

640-650：这里hashmap存放对象的方法，默认将存放的对象挂到最后一个节点。如果挂上节点后hash桶的长度大于等于8，则转换成红黑树结构。如果这个hash桶的节点中有和新存放的对象相同key的话，则需要退出当前逻辑，进入到653-656进行相关覆盖逻辑

661：修改次数(准确的说是减去key值相同的次数后剩余的操作次数)，因为HashMap是线程不安全，如果在迭代的过程中HashMap被其他线程修改了,modCount的数值就会发生变化, 这个时候expectedModCount和ModCount不相等, 迭代器就会抛出ConcurrentModificationException()异常。

662-663：如果hashmap中的节点总数量大于阈值，就会引起扩容操作。threshold默认大小是12，threshold大小设置是hashmap默认容量16乘上加载因子0.75。代码在hashmap第694-702行



