HashMap的小总结

HashMap实现了Map的标准接口，内部成员的存储是无序的。创建hashMap时要避免创建过大的出示容量，引起内部会否则会导致迭代性能下降。因为一个HashMap实例有两个参数会影响其性能：初始容量和负载因子（默认0.75）。容量是指哈希表中的桶数，初始容量只是创建哈希表时的容量。负载因子是衡量哈希表在其容量自动增加之前，允许其变得多满。当哈希表中的条目数超过负载因子和当前容量的乘积时，哈希表就会被重新洗牌（也就是重建内部数据结构），从而使哈希表中的桶数大约增加一倍。如果负载因子太高，会影响 get和 put的查找成本，如果初始容量>最大条目数/负载因子，则永远不会发生内存的再分配和拷贝动作。

其次，在填充的数据中，要避免重复的hashcode，否则会影响性能。

并发

默认实现不能保证线程同步，如果要保证同步，需要使用Collections.synchronizedMap方法来保证集合修改时的非同步访问。

Map m = Collections.synchronizedMap(new HashMap(...));

同时，针对此HashMap的迭代器创建之后，如果发生结构性修改，迭代器会立刻抛出ConcurrentModificationException异常。但是，程序不能依赖这种异常来保证程序的正确性，因为在异步并发修改的情况下，无法保证此异常可以立即引发。所以，此异常一般用来调试和检测bug。

源代码分析

初始化构建时，当给定一个容量n，其内部会按照2的n次方进行扩容，最大到2的30次方。

Put操作时，先对key进行hash运算，如果出现碰撞则采用Tree方式存储。