HashMap和Hashtable的区别

HashMap和Hashtable都实现了Map接口，但决定用哪一个之前先要弄清楚它们之间的分别。主要的区别有：线程安全性，同步(synchronization)，以及速度。

1. HashMap几乎可以等价于Hashtable，除了HashMap是非synchronized的，并可以接受null(HashMap可以接受为null的键值(key)和值(value)，而Hashtable则不行)。
2. HashMap是非synchronized，而Hashtable是synchronized，这意味着Hashtable是线程安全的，多个线程可以共享一个Hashtable；而如果没有正确的同步的话，多个线程是不能共享HashMap的。Java 5提供了ConcurrentHashMap，它是HashTable的替代，比HashTable的扩展性更好。
3. 另一个区别是HashMap的迭代器(Iterator)是fail-fast迭代器，而Hashtable的enumerator迭代器不是fail-fast的。所以当有其它线程改变了HashMap的结构（增加或者移除元素），将会抛出ConcurrentModificationException，但迭代器本身的remove()方法移除元素则不会抛出ConcurrentModificationException异常。但这并不是一个一定发生的行为，要看JVM。这条同样也是Enumeration和Iterator的区别。
4. 由于Hashtable是线程安全的也是synchronized，所以在单线程环境下它比HashMap要慢。如果你不需要同步，只需要单一线程，那么使用HashMap性能要好过Hashtable。
5. HashMap不能保证随着时间的推移Map中的元素次序是不变的。

建议少用hashtable,在单线程中，无需做线程控制，运行效率更高；在多线程中，synchronized会造成线程饥饿，死锁，可以用concurrentHashMap替代.

Hashtable默认大小是11是因为除（近似）质数求余的分散效果好

int oldCapacity = table.length;

int newCapacity = oldCapacity \* 2 + 1;

虽然不保证capacity是一个质数，但至少保证它是一个奇数。

Hashtable的寻址是这样做的：

Entry tab[] = table;

int hash = key.hashCode();

int index = (hash & 0x7FFFFFFF) % tab.length;

直接用key的hashCode()，不像HashMap里为了增强hash的分散效果而要做二次hash

要注意的一些重要术语：

1) sychronized意味着在一次仅有一个线程能够更改Hashtable。就是说任何线程要更新Hashtable时要首先获得同步锁，其它线程要等到同步锁被释放之后才能再次获得同步锁更新Hashtable。

2) Fail-safe和iterator迭代器相关。如果某个集合对象创建了Iterator或者ListIterator，然后其它的线程试图“结构上”更改集合对象，将会抛出ConcurrentModificationException异常。但其它线程可以通过set()方法更改集合对象是允许的，因为这并没有从“结构上”更改集合。但是假如已经从结构上进行了更改，再调用set()方法，将会抛出IllegalArgumentException异常。

3) 结构上的更改指的是删除或者插入一个元素，这样会影响到map的结构。