[https://www.cnblogs.com/williamjie/p/9099141.html#top](https://www.cnblogs.com/williamjie/p/9099141.html" \l "top)

hashMap vs hashTable的异同

Hashtable同样是基于哈希表实现的，同样每个元素是一个key-value对，其内部也是通过单链表解决冲突问题，容量不足（超过了阀值）时，同样会自动增长。

      Hashtable也是JDK1.0引入的类，是线程安全的，能用于多线程环境中。

      Hashtable同样实现了Serializable接口，它支持序列化，实现了Cloneable接口，能被克隆。

HashMap是基于哈希表实现的，每一个元素是一个key-value对，其内部通过单链表解决冲突问题，容量不足（超过了阀值）时，同样会自动增长。

      HashMap是非线程安全的，只是用于单线程环境下，多线程环境下可以采用concurrent并发包下的concurrentHashMap。

      HashMap 实现了Serializable接口，因此它支持序列化，实现了Cloneable接口，能被克隆。

HashMap中key和value都允许为null。key为null的键值对永远都放在以table[0]为头结点的链表中。

加载因子，如果加载因子越大，对空间的利用更充分，但是查找效率会降低（链表长度会越来越长）；如果加载因子太小，那么表中的数据将过于稀疏（很多空间还没用，就开始扩容了），对空间造成严重浪费。如果我们在构造方法中不指定，则系统默认加载因子为0.75，这是一个比较理想的值，一般情况下我们是无需修改的。

       另外，无论我们指定的容量为多少，构造方法都会将实际容量设为不小于指定容量的2的次方的一个数，且最大值不能超过2的30次方

# HashTable和HashMap区别

## 1、继承的父类不同

      Hashtable继承自Dictionary类，而HashMap继承自AbstractMap类。但二者都实现了Map接口。

## 2、线程安全性不同

      javadoc中关于hashmap的一段描述如下：此实现不是同步的。如果多个线程同时访问一个哈希映射，而其中至少一个线程从结构上修改了该映射，则它必须保持外部同步。

      Hashtable 中的方法是Synchronize的，而HashMap中的方法在缺省情况下是非Synchronize的。在多线程并发的环境下，可以直接使用Hashtable，不需要自己为它的方法实现同步，但使用HashMap时就必须要自己增加同步处理。（结构上的修改是指添加或删除一个或多个映射关系的任何操作；仅改变与实例已经包含的键关联的值不是结构上的修改。）这一般通过对自然封装该映射的对象进行同步操作来完成。如果不存在这样的对象，则应该使用 [Collections.synchronizedMap](http://write.blog.csdn.net/postedit/51556314" \l "synchronizedMap(java.util.Map)) 方法来“包装”该映射。最好在创建时完成这一操作，以防止对映射进行意外的非同步访问，如下所示：

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

      Hashtable 线程安全很好理解，因为它每个方法中都加入了Synchronize。

3.是否提供contains方法

 HashMap把Hashtable的contains方法去掉了，改成containsValue和containsKey，因为contains方法容易让人引起误解。

      Hashtable则保留了contains，containsValue和containsKey三个方法，其中contains和containsValue功能相同。

4.Hashtable中，key和value都不允许出现null值。但是如果在Hashtable中有类似put(null,null)的操作，编译同样可以通过，因为key和value都是Object类型，但运行时会抛出NullPointerException异常，这是JDK的规范规定的。  
HashMap中，null可以作为键，这样的键只有一个；可以有一个或多个键所对应的值为null。当get()方法返回null值时，可能是 HashMap中没有该键，也可能使该键所对应的值为null。因此，在HashMap中不能由get()方法来判断HashMap中是否存在某个键， 而应该用containsKey()方法来判断。

## 5、两个遍历方式的内部实现上不同

      Hashtable、HashMap都使用了 Iterator。而由于历史原因，Hashtable还使用了Enumeration的方式 。

## 6、hash值不同

      哈希值的使用不同，HashTable直接使用对象的hashCode。而HashMap重新计算hash值。

      hashCode是jdk根据对象的地址或者字符串或者数字算出来的int类型的数值。

      Hashtable计算hash值，直接用key的hashCode()，而HashMap重新计算了key的hash值，Hashtable在求hash值对应的位置索引时，用取模运算，而HashMap在求位置索引时，则用与运算，且这里一般先用hash&0x7FFFFFFF后，再对length取模，&0x7FFFFFFF的目的是为了将负的hash值转化为正值，因为hash值有可能为负数，而&0x7FFFFFFF后，只有符号外改变，而后面的位都不变。

## 7、内部实现使用的数组初始化和扩容方式不同

      HashTable在不指定容量的情况下的默认容量为11，而HashMap为16，Hashtable不要求底层数组的容量一定要为2的整数次幂，而HashMap则要求一定为2的整数次幂。  
      Hashtable扩容时，将容量变为原来的2倍加1，而HashMap扩容时，将容量变为原来的2倍。

      Hashtable和HashMap它们两个内部实现方式的数组的初始大小和扩容的方式。HashTable中hash数组默认大小是11，增加的方式是 old\*2+1。