散列表 又叫哈希表(hash table)。哈希表是种数据结构，它可以提供快速的插入操作和查找操作。通过访问key而直接访问存储的value值。它的key - value之间存在一个映射函数，我们可以通过key值和“看不到”的映射函数（散列函数）访问对应的value值。这加快了查找的速度！存放记录的数组称做散列表。散列方法不同于顺序查找、二分查找、二叉排序树及B-树上的查找。它不以关键字的比较为基本操作，采用直接寻址技术 （就是说，它是直接通过key映射[映射函数，实现的方式有多种]到内存地址上去的）。在理想情况下，无须任何比较就可以找到待查关键字，查找的期望时间为O(1）。 哈希表也有一些缺点它是基于数组的，数组创建后难于扩展某些哈希表被基本填满时，性能下降得非常严重，所以程序虽必须要清楚表中将要存储多少数据（或者准备好定期地把数据转移到更大的哈希表中，这是个费时的过程）。而且，也没有一种简便的方法可以以任何一种顺序〔例如从小到大〕遍历表中数据项。如果需要这种能力，就只能选择其他数据结构。然而如果不需要有序遍历数据，井且可以提前预测数据量的大小。那么哈希表在速度和易用性方面是无与伦比的。

**散列表的冲突现象**

（1）冲突

两个不同的关键字，由于散列函数值相同，因而被映射到同一表位置上。该现象称为冲突(Collision)或碰撞。

（2）安全避免冲突的条件（选择合适的散列函数。）

（3）冲突不可能完全避免

**哈希表算法-哈希表的构造方法**

１.直接定址法

２.数字分析法

３.平方取中法：  取关键字平方后的中间几位为哈希地址

 4.折叠法：将关键字分割成位数相同的几部分（最后一部分的位数可以不同），然后取这几部分的叠加和（舍去进位）作为哈希地址，这方法称为折叠法。

５.除留余数法： 取关键字被某个不大于哈希表表长m的数p除后所得余数为哈希地址。                H(key)=key MOD p (p<=m)

６.随机数法：选择一个随机函数，取关键字的随机函数值为它的哈希地址，即H(key)=random(key) ,其中random为随机函数。通常用于关键字长度不等时采用此法。

      在Java中也一样，hashCode方法的主要作用是为了配合基于散列的集合一起正常运行，这样的散列集合包括HashSet(HashSet底层是HashMap)、HashMap以及HashTable。  
      当集合要添加新的对象时，先调用这个对象的hashCode方法，得到对应的hashcode值，实际上在HashMap的具体实现中会用一个table保存已经存进去的对象的hashcode值，如果table中没有该hashcode值，它就可以直接存进去，不用再进行任何比较了；如果存在该hashcode值，就调用它的equals方法与新元素进行比较，相同的话就不存了，不相同就散列其它的地址，所以这里存在一个冲突解决的问题，这样一来实际调用equals方法的次数就大大降低了，说通俗一点：Java中的hashCode方法就是根据一定的规则将与对象相关的信息（比如对象的存储地址，对象的字段等）映射成一个数值，这个数值称作为散列值。put方法是用来向HashMap中添加新的元素，从put方法的具体实现可知，会先调用hashCode方法得到该元素的hashCode值，然后查看table中是否存在该hashCode值，如果存在则调用equals方法重新确定是否存在该元素，如果存在，则更新value值，否则将新的元素添加到HashMap中。从这里可以看出，hashCode方法的存在是为了减少equals方法的调用次数，从而提高程序效率。