**Java中的equals方法和hashCode方法是Object中的，所以每个对象都是有这两个方法的，有时候我们需要实现特定需求，可能要重写这两个方法，今天就来介绍一些这两个方法的作用。**

**equals()和hashCode()方法是用来在同一类中做比较用的，尤其是在容器里如set存放同一类对象时用来判断放入的对象是否重复。**

**这里我们首先要明白一个问题：**

**equals()相等的两个对象，hashcode()一定相等，equals()不相等的两个对象，却并不能证明他们的hashcode()不相等。换句话说，equals()方法不相等的两个对象，hashCode()有可能相等。（我的理解是由于哈希码在生成的时候产生冲突造成的）**

**在这里hashCode就好比字典里每个字的索引，equals()好比比较的是字典里同一个字下的不同词语。就好像在字典里查“自”这个字下的两个词语“自己”、“自发”，如果用equals()判断查询的词语相等那么就是同一个词语，比如equals()比较的两个词语都是“自己”，那么此时hashCode()方法得到的值也肯定相等；如果用equals()方法比较的是“自己”和“自发”这两个词语，那么得到结果是不想等，但是这两个词都属于“自”这个字下的词语所以在查索引时相同，即：hashCode()相同。如果用equals()比较的是“自己”和“他们”这两个词语的话那么得到的结果也是不同的，此时hashCode() 得到也是不同的。**

**反过来：hashcode()不等，一定能推出equals()也不等；hashcode()相等，equals()可能相等，也可能不等。在object类中，hashcode()方法是本地方法，返回的是对象的地址值，而object类中的equals()方法比较的也是两个对象的地址值，如果equals()相等，说明两个对象地址值也相等，当然hashcode() 也就相等了；**

**同时hash算法对于查找元素提供了很高的效率**

**如果想查找一个集合中是否包含有某个对象，大概的程序代码怎样写呢？  
你通常是逐一取出每个元素与要查找的对象进行比较，当发现某个元素与要查找的对象进行equals方法比较的结果相等时，则停止继续查找并返回肯定的信息，否则，返回否定的信息，如果一个集合中有很多个元素，比如有一万个元素，并且没有包含要查找的对象时，则意味着你的程序需要从集合中取出一万个元素进行逐一比较才能得到结论。  
  
有人发明了一种哈希算法来提高从集合中查找元素的效率，这种方式将集合分成若干个存储区域，每个对象可以计算出一个哈希码，可以将哈希码分组(使用不同的hash函数来计算的)，每组分别对应某个存储区域，根据一个对象的哈希吗就可以确定该对象应该存储在哪个区域HashSet就是采用哈希算法存取对象的集合，它内部采用对某个数字n进行取余(这种的hash函数是最简单的)的方式对哈希码进行分组和划分对象的存储区域；Object类中定义了一个hashCode()方法来返回每个Java对象的哈希码，当从HashSet集合中查找某个对象时，**Java系统首先调用对象的hashCode()方法获得该对象的哈希码表，然后根据哈希吗找到相应的存储区域，最后取得该存储区域内的每个元素与该对象进行equals方法比较**；这样就不用遍历集合中的所有元素就可以得到结论，可见，HashSet集合具有很好的对象检索性能，但是，HashSet集合存储对象的效率相对要低些，因为向HashSet集合中添加一个对象时，要先计算出对象的哈希码和根据这个哈希码确定对象在集合中的存放位置 为了保证一个类的实例对象能在HashSet正常存储，要求这个类的两个实例对象用equals()方法比较的结果相等时，他们的哈希码也必须相等；也就是说，如果**obj1.equals(obj2)**的结果为true,那么以下表达式的结果也要为true:  
**obj1.hashCode() == obj2.hashCode()**  
换句话说：当我们重写一个对象的equals方法，就必须重写他的hashCode方法，不过不重写他的hashCode方法的话，Object对象中的hashCode方法始终返回的是一个对象的hash地址，而这个地址是永远不相等的。所以这时候即使是重写了equals方法，也不会有特定的效果的，因为hashCode方法如果都不想等的话，就不会调用equals方法进行比较了，所以没有意义了。**

**如果一个类的hashCode()方法没有遵循上述要求，那么，当这个类的两个实例对象用equals()方法比较的结果相等时，他们本来应该无法被同时存储进set集合中，但是，如果将他们存储进HashSet集合中时，由于他们的hashCode()方法的返回值不同(Object中的hashCode方法返回值是永远不同的)，第二个对象首先按照哈希码计算可能被放进与第一个对象不同的区域中，这样，它就不可能与第一个对象进行equals方法比较了，也就可能被存储进HashSet集合中了，Object类中的hashCode()方法不能满足对象被存入到HashSet中的要求，因为它的返回值是通过对象的内存地址推算出来的，同一个对象在程序运行期间的任何时候返回的哈希值都是始终不变的，所以，只要是两个不同的实例对象，即使他们的equals方法比较结果相等，他们默认的hashCode方法的返回值是不同的。**