HashMap原理与字典

在展开HashMap原理的讲解之前，首先回忆一下大家初中和高中使用的汉英字典。  
  
比如要找一个单词对应的中文意思，假设单词是Lengendary,首先在目录找到Lengendary在第 555页。   
  
然后，翻到第555页，这页不只一个单词，但是量已经很少了，逐一比较，很快就定位目标单词Lengendary。  
  
555相当于就是Lengendary对应的****hashcode****

分析HashMap性能卓越的原因

**-----hashcode概念-----**  
所有的对象，都有一个对应的**hashcode（散列值）**  
比如字符串“gareen”对应的是1001 (实际上不是，这里是方便理解，假设的值)  
比如字符串“temoo”对应的是1004  
比如字符串“db”对应的是1008  
比如字符串“annie”对应的**也是1008**  
  
**-----保存数据-----**  
准备一个数组，其长度是2000，并且设定特殊的hashcode算法，使得所有字符串对应的hashcode，都会落在0-1999之间  
要存放名字是"gareen"的英雄，就把该英雄和名称组成一个**键值对**，存放在数组的1001这个位置上  
要存放名字是"temoo"的英雄，就把该英雄存放在数组的1004这个位置上  
要存放名字是"db"的英雄，就把该英雄存放在数组的1008这个位置上  
要存放名字是"annie"的英雄，然而 "annie"的hashcode 1008对应的位置**已经有db英雄了**，那么就在这里创建一个链表，**接在db英雄后面存放annie**  
  
**-----查找数据-----**  
比如要查找gareen，首先计算"gareen"的hashcode是1001，根据1001这个下标，到数组中进行定位，（**根据数组下标进行定位，是非常快速的**） 发现1001这个位置就只有一个英雄，那么该英雄就是gareen.  
比如要查找annie，首先计算"annie"的hashcode是1008，根据1008这个下标，到数组中进行定位，发现1008这个位置**有两个英雄**，那么就对两个英雄的名字进行逐一比较(**equals**)，因为此时需要比较的量就已经少很多了，很快也就可以找出目标英雄  
这就是使用hashmap进行查询，非常快原理。  
  
  
这是一种用**空间换时间**的思维方式

