**关于equals和hashCode，看这一篇真的就够了**

**1、equals在Java中含义**

首先要解释清楚这个，**equals方法在Java中代表逻辑上的相等**，什么叫逻辑上的相等？这个就涉及到Java本身的语法特性。

我们知道，**Java中存在着==来判断基本数据类型的相等，但是对于对象，==只能判断内存地址是否相等，也就是说是否是同一个对象**：

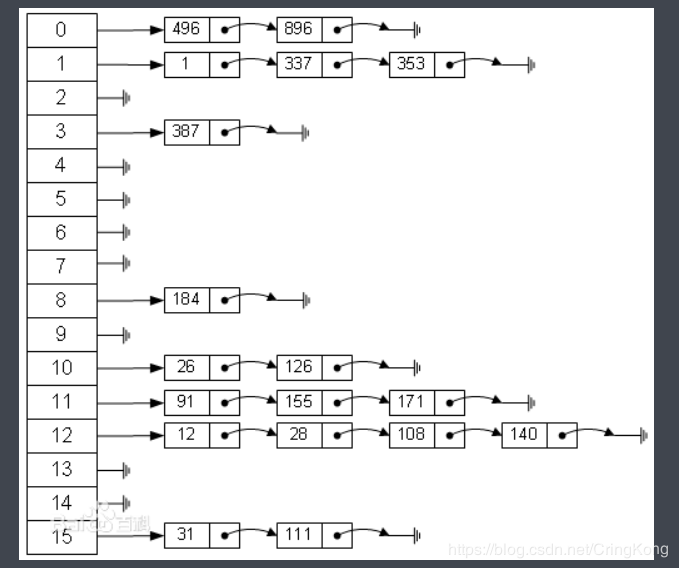
如果我们自己创建一个类的话， **要实现判断两个实例逻辑上是否相等，就需要重写他的equals()方法。**

## 2、hashCode在Java中的作用

网上有很多博客都把hashCode()和equals()混为一谈，**但实际上hashCode()就是他的字面意思，代表这个对象的哈希码。**

但是为什么JavaDoc明确的告诉我们，**hashCode()和equals()要一起重写呢**？原因是因为，在Java自带的容器HashMap和HashSet中，**都需同时要用到对象的hashCode()和equals()方法来进行判断，然后再插入删除元素，这点我们一会再谈。**

那么我们还是单独来看hashCode()，为什么HashMap需要用到hashCode？这个就涉及到HashMap底层的数据结构 – 散列表的原理：



**HashMap底层用于存储数据的结构其实是散列表（也叫哈希表）**，散列表是通过哈希函数将元素映射到数组指定下标位置，**在Java中，这个哈希函数其实就是hashCode()方法。**

hashCode相等的两个对象不一定逻辑相等，逻辑相等的两个对象hashCode必须相等！

## 3、为什么hashCode和equals要一起重写

刚刚我们知道，**equals()是用来判断对象是否逻辑相等，hashCode()就是获得一个对象的hash值，同时再HashMap中用来得到数组下标位置。**

那么为什么很多地方都说到，hashCode()和equals()要一起重写呢？**明明通过对象hashCode就可以定位数组下标了啊，那我们直接用把对象存进去取出来不就行了吗？**

答案是这样的：**设计再良好的哈希函数，也会出现哈希冲突的情况，什么是哈希冲突呢**？举个例子来说，我设计了这样一种哈希函数：



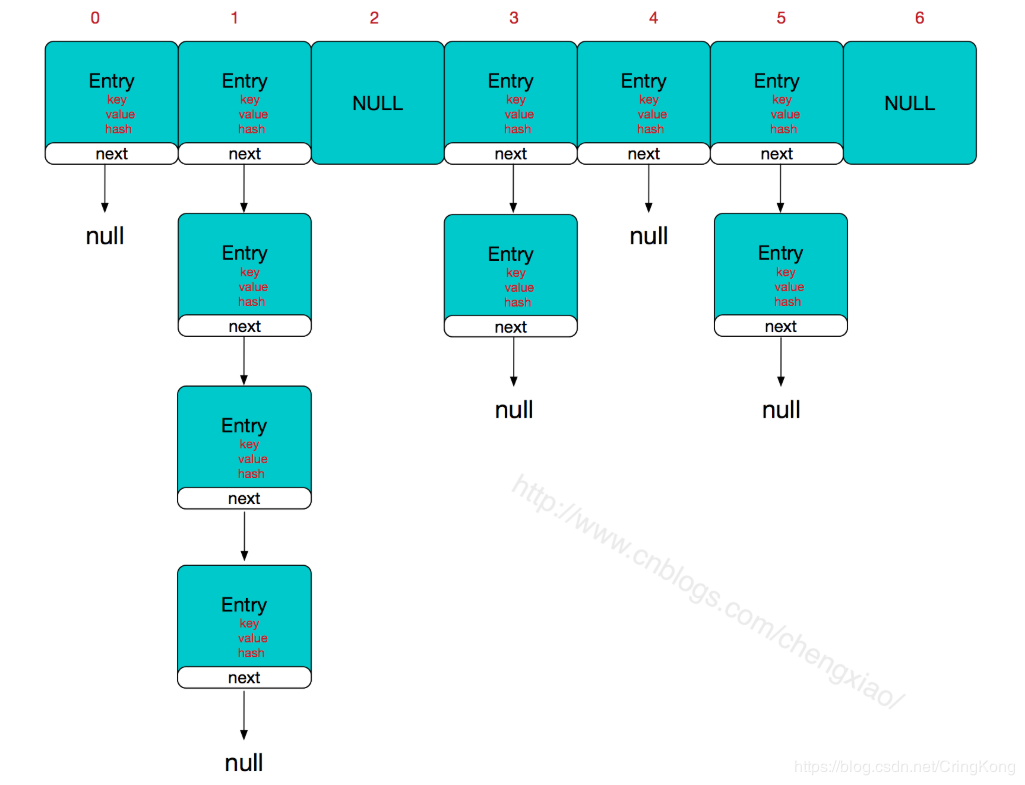
可以看到， "fish" 和 "fuck"出现了哈希冲突，这是我们不想看到的，一旦出现了哈希冲突，我们的哈希表就需要解决哈希冲突，一般解方式有：

* 开发定址法（线性探测再散列，二次探测再散列，伪随机探测再散列）
* 再哈希法
* 链地址法
* 建立一个公共溢出区

这都是数据结构课本上的东西，我就不再细讲了，不懂的同学自行搜索！

就像我之前说的，**设计再精良的哈希函数，也会有哈希冲突的情况出现，Java中的hashCode()本身就是一种哈希函数，必然会出现哈希冲突，更怕一些程序员写出某些硬核哈希函数。**

既然存在哈希冲突，我们就得解决，**HashMap采用的是链地址法来解决**



这里就存在一种极端情况，如何判断是究竟是两个 **逻辑相等**的对象**重复写入**，还是**两个逻辑不等**的对象出现了**哈希冲突**呢？

很简单，用equals()方法判断不就完事了吗，我们之前说了，equals()方法就是用来设计判断两个对象是否逻辑相等的啊！