**Java基础**

# HashMap

## 为什么浮在因子设置为0.75

<https://juejin.im/post/5e532459e51d4526cd1de801>

HashMap采用的结构为数组+链表+红黑树(1.8),利用hash算法将数据分配到各个桶中，如果两个值计算出的hash不一样，那么这个两个值一定不相同，如果一样，这两个值有可能相同，也有可能不相同，这就发生了hash碰撞，发生了hash碰撞，hashmap将第二个值加第一个值的链表中，如果发生碰撞的概率大，那么hashmap就会退化成一个链表，导致查询的效率低下。如果减小发生碰撞的概率，有两个方法:1.扩容 2.设计一个良好的hash算法

什么时候扩容，在hashmap中是的达到了临界值的时候就会扩容，临界值=负载因子\*容量，如果负载因子过小，那么会导致频繁的扩容，耗费内存空间，过大，就会导致hash碰撞的概率增加，会导致查询的速度降低，0.75不大也不小，而且\*2的幂是整数，所以选择0.75是合适的。

# ConcurrentHashMap源码

# 三种for循环区别

1. 如果不满足条件判断，那么**for循环和while循环就不会执行**，do while循环会执行至少一次。
2. for循环的变量再小括号中定义，只有循环内部才可以使用。（初始化语句不可以输出）。
3. While循环和do-while循环初始化语句本来就在外面，所以出来循环之后还可以继续使用。（初始化语句可以输出）

# equals和hashcode的作用

1.若重写了equals(Object obj)方法，则有必要重写hashCode()方法。

2.若两个对象equals(Object obj)返回true，则hashCode（）有必要也返回相同的int数。

3.若两个对象equals(Object obj)返回false，则hashCode（）不一定返回不同的int数。

4.若两个对象hashCode（）返回相同int数，则equals（Object obj）不一定返回true。

5.若两个对象hashCode（）返回不同int数，则equals（Object obj）一定返回false。

6.同一对象在执行期间若已经存储在集合中，则不能修改影响hashCode值的相关信息，否则会导致内存泄露问题。

一般来说涉及到对象之间的比较大小就需要重写equals方法，