**Java学习笔记**

1. **HashMap**
   1. **HashMap1.7**

**（**<https://www.bilibili.com/video/bv1x741117jq/?p=2>**）**

数据结构：数组+链表

设计原理：

1. 为什么长度是2的幂次方?
2. 因为2的幂次方-1对应2进制正好全是1，能够使得hash更加的均与分布，解决hash碰撞问题
3. 在数据扩容的时候，原来位置的数组要么落在原位置，要么落在长度+扩容的长度位置
4. 如何解决hash碰撞
5. 数组长度幂次方，取长度的时候会进行与运算
6. 高16位和低16位进行或运算
7. 在jdk参数设置jdk.map.althashing.threshold会引起hash不为默认值0（如果为0则禁用替代哈希。），改为与此实例相关联的随机值键的哈希码，使哈希冲突更难找到。

存在问题：

1. 链表插入采用头插法，会造成死循环

（线程在这挂起的时候可能会引起）

Entry*<*K,V*>* next = e.next;

1. 会存在concurrentModificationException异常

重现场景：hashmap循环迭代的时候，用hashmap对象去删除元素，导致modCount（操作次数）不一样发生异常

Map*<*String, String*>* map = new HashMap*<>(*100*)*;  
map.put*(*"1","1"*)*;  
map.put*(*"2","2"*)*;  
for*(*String key : map.keySet*()){* if*(*key.equals*(*"2"*)){* map.remove*(*"2"*)*;  
 *}  
}*

* 1. **HashMap1.8**

数据结构：数组+链表+红黑树

升级亮点：

1. 链表插入改用尾插法，避免头插法多线程可能存在的死循环情况。
2. 采用链表+红黑树，解决链表过长效率低问题
3. **CurrentHashMap**
   1. **CurrentHashMap1.7**

**（**<https://www.bilibili.com/video/bv1x741117jq/?p=2>**）**

数据结构：数组+链表+segment+reentrantlock

原理：在hashmap的基础上一层加了分段锁，根据hash找到分段锁数组位置，在put进去的时候用reentranlock锁住，其他线程用trylock不断判断锁状态决定是否进入