### 1、回答

HashMap是一种存取高效但不保证有序的常用容器。它的数据结构为“数组+链表”，是解决哈希冲突的产物，也就是我们常说的链地址法。它实现了Map接口采用K-V键值对存储数据，并实现了浅拷贝和序列化。

HashMap的默认初始大小为16，初始化大小必须为2的幂，最大大小为2的30次方。数组中存储的链表节点Entry 类实现于Map.Entry 接口，它实现了对节点的通用操作。

HashMap的阈值默认为“容量\*0.75f”，当存储节点数量超过该值，则对map 进行扩容处理。

HashMap提供了4种构造方法，分别是默认构造方法；可以指定初始容量的构造方法；可以指定初始容量和阈值的构造方法以及基于一个Map的构造方法。虽然是构造函数，但是真正的初始化都是在第一次添加操作里面实现的。

在第一次添加操作中，HashMap 会先判断存储数组有没有初始化，如果没有先进行初始化操作，初始化过程中会取比用户指定的容量大的最近的2 的幂次方数作为数组的初始容量，并更新扩容的阈值。

接着添加操作讲解。添加操作的执行流程为：

* 先判断有没有初始化
* 再判断传入的key 是否为空，为空保存在table[o] 位置
* key不为空就对key 进hash，hash 的结果再& 数组的长度就得到存储的位置
* 如果存储位置为空则创建节点，不为空就说明存在冲突
* 解决冲突HashMap 会先遍历链表，如果有相同的value 就更新旧值，否则构建节点添加到链表头
* 添加还要先判断存储的节点数量是否达到阈值，到达阈值要进行扩容
* 扩容扩2倍，是新建数组所以要先转移节点，转移时都重新计算存储位置，可能保持不变可能为旧容量+位置。
* 扩容结束后新插入的元素也得再hash 一遍才能插入。

获取节点的操作和添加差不多，也是

* 先判断是否为空，为空就在table[0] 去找值
* 不为空也是先hash,&数组长度计算下标位置
* 再遍历找相同的key 返回值

HashMap的其他操作大同小异，再讲讲HashMap1.7 的问题还有1.7 和1.8 的差别。

HashMap 是一个并发不安全的容器，在迭代操作是采用的是fast-fail 机制；在并发添加操作中会出现丢失更新的问题；因为采用头插法在并发扩容时会产生环形链表的问题，导致CPU 到达100%，甚至宕机。

解决并发问题可以采用

* Java 类库提供的Collections 工具包下的Collections.synchronizedMap()方法，返回一个线程安全的Map
* 或者使用并发包下的ConcurrentHashMap，ConcurrentHashMap采用分段锁机制实现线程安全
* 使用HashTable （不推荐）

Hash1.7 和1.8 最大的不同在于1.8 采用了“数组+链表+红黑树”的数据结构，在链表长度超过8 时，把链表转化成红黑树来解决HashMap 因链表变长而查询变慢的问题；其次

* 在hash 取下标时将1.7 的9次扰动（5次按位与和4次位运算）改为2次（一次按位与和一次位运算）
* 1.7 的底层节点为Entry，1.8 为node ，但是本质一样，都是Map.Entry 的实现
* 还有就是在存取数据时添加了关于树结构的遍历更新与添加操作，并采用了尾插法来避免环形链表的产生
* 但是并发丢失更新的问题依然存在。

回答顺序：数据结构+继承结构+基本字段+构造方法+添加操作+扩容操作+获取操作+并发问题+与1.8的区别。

### 2、为什么初始容量必须为2的幂？为什么负载因子为0.75？为什么要做那么多扰动处理？

减少哈希冲突

1. 容量必须为2的幂是为了增加取值的可能性

2的n次幂转化为二进制为1后面n个0，在计算下标的时候是hash&(length-1)，也就是&(n-1)个1：初始容量为4 -> 100，length-1 -> 11。所有的二进制都为1有什么好处？

* 0/1 & 1都为它本身
* 0/1 & 0都为0

可以看出&1保证了取值的平均。如果某一位为0，比如最后一位，那么它&出来下标就一定是个偶数，减少了HashMap数组一般的取值，大大增加了冲突的可能。

1. 负载因子为0.75f是空间与时间的均衡

如果负载因子小，意味着阈值变小。比如容量为10 的HashMap，负载因子为0.5f，那么存储5个就会扩容到20，出现哈希冲突的可能性变小，但是空间利用率不高。适用于有足够内存并要求查询效率的场景。

相反如果阈值为1 ，那么容量为10，就必须存储10个元素才进行扩容，出现冲突的概率变大，极端情况下可能会从O(1)退化到O(n)。适用于内存敏感但不要求要求查询效率的场景

1. hash()的意义在于使hash结果不同

hash 算法的好坏直接影响hash 结构的效率，坏的hash 算法极端情况下可能会使hash 结构的存取效率从O(1)退化到O(n)。1.8 之所以把9 次扰动降到2 次，是出于计算效率的考虑。

### 2、& 字符虽然和%效果一样，但是操作效率更高

### 3、为什么int，String最适合为key

int和String的好处在于hash出来的值不会改变。如果是一个对象，那么他们可能会因为内部引用的改变而hashCode值的改变，会导致存储重复的数据或找不到数据的情况。

### 4、并发操作导致的添加丢失和环形链表的产生过程

### 5、知识点拓展

拓展一：解决Hash 冲突的不同方案

* 链地址法
* 开发地址：线性探测法、平方探测法
* 完全散列：布谷鸟散列

拓展二：HashMap 是浅拷贝，说一说浅拷贝和深拷贝的区别

拓展三：说一说Collections.synchronizedMap()和HashTable 的区别

拓展四：说一说HashMap 如何实现有序(LinkHashMap 和TreeMap)以及他们的差别

拓展五：说一说ConcurrentHashMap 如何实现线程安全