1、HashMap常见面试题：

**①HashMap的底层数据结构？**

数组和链表组合构成的数据结构，java8之前是头插法，作者认为后来的值被查找的可能性更大一点，提升查找的效率。在java8之后，都是所用尾部插入了。之所以采用尾部插入和扩容机制有关系，在扩容再hash的时候，会导致原来在链头的节点到达链尾，且如果再多线程，如果此时多线程插入节点，会出现环形链表。使用头插会改变链表的上的顺序，但是如果使用尾插，在扩容时会保持链表元素原本的顺序，就不会出现链表成环的问题了。

java8之后链表有红黑树的部分，红黑树的引入巧妙的将原本O(n)的时间复杂度降低到了O(logn)。

**②HashMap的存取原理？**

**③Java7和Java8的区别**？

**④为啥会线程不安全？**

put/get方法都没有加同步锁，多线程情况最容易出现的就是：无法保证上一秒put的值，下一秒get的时候还是原值，所以线程安全还是无法保证。

**⑤有什么线程安全的类代替么?**

一般都会使用HashTable或者CurrentHashMap，但是因为前者的并发度的原因基本上没啥使用场景了，所以存在线程不安全的场景我们都使用的是CorruentHashMap。

⑥默认初始化大小是多少？为啥是这么多？为啥大小都是2的幂？

为了位运算的方便，位与运算比算数计算的效率高了很多。因为位运算直接对内存数据进行操作，不需要转成十进制，所以位运算要比取模运算的效率更高，所以HashMap在计算元素要存放在数组中的index的时候，使用位运算代替了取模运算。之所以可以做等价代替，前提是要求HashMap的容量一定要是2^n 。

在使用不是2的幂的数字的时候，Length-1的值是所有二进制位全为1，这种情况下，index的结果等同于HashCode后几位的值，**实现均匀分布。**

使用16，太小了就有可能频繁发生扩容，影响效率。太大了又浪费空间，不划算。所以，16就作为一个经验值被采用了。

**⑦HashMap的扩容方式？负载因子是多少？为什是这么多？**

扩容即resize，有两个因素：

Capacity：HashMap当前长度。

LoadFactor：负载因子，默认值0.75f

分为两步：

扩容：创建一个新的Entry空数组，长度是原数组的2倍。

ReHash：遍历原Entry数组，把所有的Entry重新Hash到新数组。

**⑧HashMap的主要参数都有哪些？**

**⑨HashMap是怎么处理hash碰撞的？**

**⑩hash的计算规则？**

为啥重写equals方法的时候需要重写hashCode方法呢？

Ojbect类中有两个方法equals、hashCode，这两个方法都是用来比较两个对象是否相等的。

通过key的hashCode去寻找index，然后使用equals方法，对确认对象值；默认的equals是比较两个对象的内存地址。

2、ConcurrentHashMap & HashTable

①Hashmap中的链表大小超过八个时会自动转化为红黑树，当删除小于六时重新变为链表，为啥呢？

根据泊松分布，在负载因子默认为0.75的时候，单个hash槽内元素个数为8的概率小于百万分之一，所以将7作为一个分水岭，等于7的时候不转换，大于等于8的时候才进行转换，小于等于6的时候就化为链表。

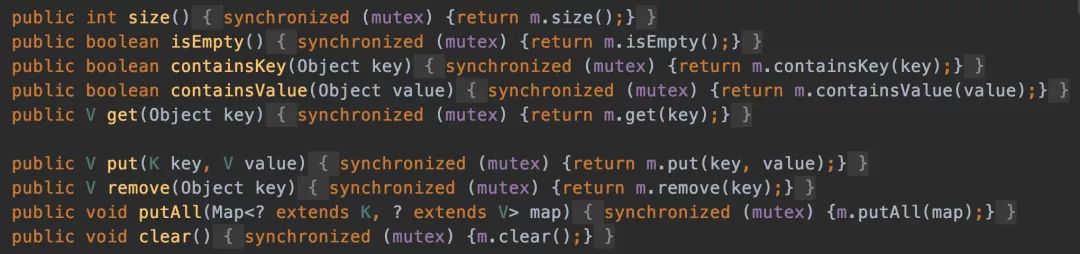
②多线程的场景：

Collections.synchronizedMap(Map)创建线程安全的map集合；

Hashtable

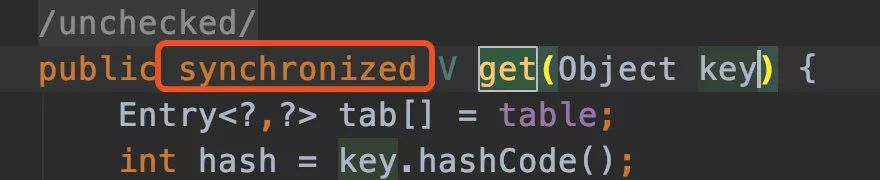
ConcurrentHashMap

Collections.synchronizedMap(Map)：构造器，传入mutex参数，则将对象排斥锁赋值为传入的对象。



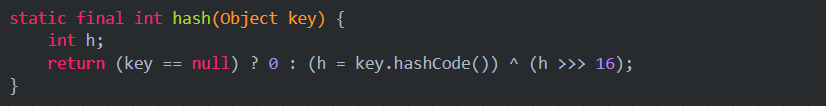
Hashtable

对数据操作的时候都会上锁，所以效率比较低下。



③Hashtable 是不允许键或值为 null 的，HashMap 的键值则都可以为 null

因为Hashtable在我们put 空值的时候会直接抛空指针异常，但是HashMap却做了特殊处理。



Hashtable使用的是安全失败机制（fail-safe），这种机制会使你此次读到的数据不一定是最新的数据。

如果你使用null值，就会使得其无法判断对应的key是不存在还是为空，因为你无法再调用一次contain(key）来对key是否存在进行判断，ConcurrentHashMap同理。

④ConcurrentHashMap

1.7中的数据结构，仍然是数组加链表。

谈谈你理解的 Hashtable，讲讲其中的 get put 过程。ConcurrentHashMap同问。

1.8 做了什么优化？

线程安全怎么做的？

不安全会导致哪些问题？

如何解决？有没有线程安全的并发容器？

ConcurrentHashMap 是如何实现的？

ConcurrentHashMap并发度为啥好这么多？

1.7、1.8 实现有何不同？为什么这么做？

快速失败（fail—fast）是啥，应用场景有哪些？安全失败（fail—safe）同问。