Java源码与底层

1. Java源码技术

# [HashMap为什么线程不安全(hash碰撞与扩容导致)](https://www.cnblogs.com/baizhanshi/p/7724656.html)

一直以来都知道HashMap是线程不安全的，但是到底为什么线程不安全，在多线程操作情况下什么时候线程不安全？

让我们先来了解一下HashMap的底层存储结构，HashMap底层是一个Entry数组，一旦发生Hash冲突的的时候，HashMap采用拉链法解决碰撞冲突，Entry内部的变量：

1. Java底层技术

# HashMap底层实现原理及面试问题:

①HashMap的工作原理

HashMap基于hashing原理，我们通过**put()和get()**方法储存和获取对象。当我们将键值对传递给put()方法时，它调用键对象的**hashCode()**方法来计算hashcode，让后找到**bucket位置**来储存值对象。当获取对象时，通过键对象的equals()方法找到正确的键值对，然后返回值对象。***HashMap使用链表来解决碰撞问题，当发生碰撞了，对象将会储存在链表的下一个节点中。 HashMap在每个链表节点中储存键值对对象。***

当两个不同的键对象的hashcode相同时会发生什么？ 它们会储存在同一个bucket位置的链表中。键对象的equals()方法用来找到键值对。

因为HashMap的好处非常多，我曾经在电子商务的应用中使用HashMap作为缓存。因为金融领域非常多的运用Java，也出于性能的考虑，我们会经常用到HashMap和**ConcurrentHashMap。**

### ②HashMap和Hashtable的区别

1.HashMap和Hashtable都实现了Map接口，但决定用哪一个之前先要弄清楚它们之间的分别。主要的区别有：**线程安全性，同步(synchronization)**，以及**速度**。

2.HashMap是非synchronized，而Hashtable是synchronized，这意味着Hashtable是线程安全的，多个线程可以共享一个Hashtable；而如果没有正确的同步的话，多个线程是不能共享HashMap的。Java 5提供了ConcurrentHashMap，它是HashTable的替代，比**HashTable的扩展性更好。**

**3.** 另一个区别是HashMap的迭代器(Iterator)是**fail-fast迭代器**，而Hashtable的enumerator迭代器不是fail-fast的。所以当有其它线程改变了HashMap的结构（增加或者移除元素），将会抛出ConcurrentModificationException，但迭代器本身的remove()方法移除元素则不会抛出ConcurrentModificationException异常。但这并不是一个一定发生的行为，要看JVM。这条同样也是**Enumeration和Iterator**的区别。

4. 由于Hashtable是线程安全的也是**synchronized**，所以在单线程环境下它比HashMap要慢。如果你不需要同步，只需要单一线程，那么使用HashMap性能要好过Hashtable。

5.HashMap不能保证随着时间的推移Map中的元素次序是不变的。

### 要注意的一些重要术语：

1) **sychronized意味着在一次仅有一个线程能够更改Hashtable。就是说任何线程要更新Hashtable时要首先获得同步锁**，其它线程要等到同步锁被释放之后才能再次获得同步锁更新Hashtable。

2) Fail-safe和iterator迭代器相关。如果某个集合对象创建了Iterator或者ListIterator，然后其它的线程试图“结构上”更改集合对象，将会抛出ConcurrentModificationException异常。但其它线程可以通过set()方法更改集合对象是允许的，因为这并没有从“结构上”更改集合。但是假如已经从结构上进行了更改，再调用set()方法，将会抛出IllegalArgumentException异常。

3) 结构上的更改指的**是删除或者插入一个元素，这样会影响到map的结构。**

### 我们能否让HashMap同步？

HashMap可以通过下面的语句进行同步：  
**Map m = Collections.synchronizeMap(hashMap);**