对于Vector&ArrayList、Hashtable&HashMap，要记住线程安全的问题，记住Vector与Hashtable是旧的，是java一诞生就提供了的，它们是线程安全的，ArrayList与HashMap是java2时才提供的，它们是线程不安全的。

### 1、HashMap和Hashtable的区别

需要用到同步时用HashTable，不需要同步时用HashMap。HashMap可以通过调用Collections的静态方法Collections.synchronizedMap(Map map)进行同步。

*/\*\**

*\* HashMap线程不安全的，效率高于Hashtable  
 \*  
 \* 将键映射到值的对象，其中键和值都是对象  
 \*/* HashMap<Object, Object> hashMap = new HashMap<>(6);  
 hashMap.put("1", "1");  
 // 1.不能包含重复建，相同的key在Map中只会有一个与之关联的value存在  
 hashMap.put("1", "2");  
 // 2.允许null键和null值  
 hashMap.put(null, null);  
  
 System.*out*.println(hashMap);  
  
  
 */\*\*  
 \* Hashtable线程安全  
 \*  
 \* 继承自Dictionary类  
 \*  
 \* 其方法都是synchronize的  
 \*/* Hashtable<Object, Object> hashtable = new Hashtable<>();  
 hashtable.put("1", "1");  
 hashtable.put("1", "2");  
 // 不允许null键null值  
// hashtable.put(null, null);

### 2、HashMap底层结构

HashMap的主干是一个Entry数组。Entry是HashMap的基本组成单元，每一个Entry包含一个key-value键值对。整体结构图：



HashMap由数组+链表组成的。

数组是HashMap的主体，链表则是主要为了解决哈希冲突，如果定位到的数组位置不含链表（当前entry的next指向null），那么对于查找，添加等操作很快，仅需一次寻址即可；如果定位到的数组包含链表，对于添加操作，其时间复杂度为O(n)，首先遍历链表，存在即覆盖，否则新增；对于查找操作来讲，仍需遍历链表，然后通过key对象的equals方法逐一比对查找。所以，性能考虑，HashMap中的链表出现越少，性能才会越好。

### 3、ArrayList和LinkedList的区别

1. ArrayList基于动态数组，LinkedList基于链表。
2. 对于随机访问get和set，ArrayList优于LinkedList，因为LinkedList要移动指针。
3. 对于新增和删除操作add和remove，LinedList比较占优势，因为ArrayList要移动数据。

### 4、ArrayList 和 Vector 的区别

1. 同步性

Vector是线程安全的，也就是说是它的方法之间是线程同步的，而ArrayList是线程序不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用ArrayList，因为它不考虑线程安全，效率会高些；如果有多个线程会访问到集合，那最好是使用Vector。

1. 数据增长

ArrayList与Vector都有一个初始的容量大小，当存储进它们里面的元素的个数超过了容量时，就需要增加ArrayList与Vector的存储空间。Vector默认增长为原来两倍，而ArrayList的增长策略在文档中没有明确规定（从源代码看到的是增长为原来的1.5倍）。ArrayList与Vector都可以设置初始的空间大小，Vector还可以设置增长的空间大小，而ArrayList没有提供设置增长空间的方法。

### 5、Array和ArrayList的区别

* Array可以包含基本数据类型和引用类型，ArrayList只能包含引用类型。
* ArrayList是基于数组实现的，Array大小不可以调整大小，但ArrayList可以通过内部方法自动调整容量。
* ArrayList是List接口的实现类，相比Array支持更多的方法和特性。

### 6、HashSet的实现原理

1. HashSet是基于HashMap实现的，默认构造函数是构建一个初始容量为16，负载因子为0.75的HashMap。封装了一个HashMap对象来存储所有的集合元素，所有放入HashSet中的集合元素实际上由HashMap的key来保存，而HashMap的value则存储了一个PRESENT，它是一个静态的Object对象。
2. 当我们试图把某个类的对象当成HashMap的key，或试图将这个类的对象放入HashSet中保存时，重写该类的equals(Object obj)方法和hashCode() 方法很重要，而且这两个方法的返回值必须保持一致：当该类的两个的 hashCode()返回值相同时，它们通过 equals()方法比较也应该返回 true。通常来说，所有参与计算 hashCode()返回值的关键属性，都应该用于作为equals()比较的标准。
3. HashSet的其他操作都是基于HashMap的。

### 7、如何决定使用HashMap还是TreeMap

参考03

### 8、List、Set、Map之间的区别

List（列表）

List的元素以线性方式存储，可以存放重复对象，主要有以下两个实现类：

1.ArrayList: 长度可变的数组，可以对元素进行随机的访问，向ArrayList中插入与删除元素的速度慢。JDK8中ArrayList扩容的实现是通过grow()方法里使用语句newCapacity = oldCapacity + (oldCapacity >> 1)（即1.5倍扩容）计算容量，然后调用Arrays.copyof()方法进行对原数组进行复制。

2.LinkedList: 采用链表数据结构，插入和删除速度快，但访问速度慢。

Set(集合)

Set中的对象不按特定(HashCode)方式排序，没有重复对象，有两个实现类：

1.HashSet：HashSet按照哈希算法来存取集合中的对象，存取速度比较快。当HashSet中的元素个数超过数组大小\*loadFactor（默认值为0.75）时，就会进行近似两倍扩容（newCapacity = (oldCapacity << 1) + 1）。

2.TreeSet：TreeSet实现了SortedSet接口，能够对集合中的对象进行排序。

Map(映射)

Map把键对象和值对象映射的集合，每一个元素都包含一个键和值。实现类：

HashMap：HashMap基于散列表实现，其插入和查询<K,V>的开销是固定的，可以通过构造器设置容量和负载因子来调整容器的性能。

LinkedHashMap：类似于HashMap，但是迭代遍历它时，取得<K,V>的顺序是其插入次序，或者是最近最少使用(LRU)的次序。

TreeMap：TreeMap基于红黑树实现。查看<K,V>时，它们会被排序。TreeMap是唯一的带有subMap()方法的Map，subMap()可以返回一个子树。

### 9、HashMap怎样解决hash冲突

参考16