**hashmap 和 hashtable的区别**

**HashMap 是Hashtable 的轻量级实现（非线程安全的实现），他们都完成了Map 接口.**

**主要区别在于HashMap 允许空（null）键值（key）,由于非线程安全，效率上可能高于Hashtable。一个方便的方法就是利用Collections类的静态的synchronizedMap()方法，它创建一个[线程安全](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%AE%89%E5%85%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4uHIBuhnLrj-9PWRYuW0Y0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1RsPjDsPjRv" \t "http://zhidao.baidu.com/_blank)的Map对象，并把它作为一个封装的对象来返回。**

**HashMap 允许将null 作为一个entry 的key 或者value，而Hashtable 不允许。HashMap中只有一条记录可以是一个空的key，但任意数量的条目可以是空的value.**

**HashMap 把Hashtable 的contains 方法去掉了，改成containsvalue 和containsKey。因为contains方法容易让人引起误解。  
Hashtable 继承自Dictionary 类，而HashMap 是Java1.2 引进的Map interface 的一个实现。  
最大的不同是，Hashtable 的方法是Synchronize 的，而HashMap 不是，在多个线程访问Hashtable 时，不需要自己为它的方法实现同步，而HashMap 就必须为之提供外同步。  
Hashtable 和HashMap 采用的hash/rehash 算法都大概一样，所以性能不会有很大的差异.**

**ArrayList和Vector的区别**

**1 Vector的方法都是同步的(Synchronized),是线程安全的(thread-safe)，而ArrayList的方法不是，由于线程的同步必然要影响性能，因此,ArrayList的性能比Vector好。  
2 当Vector或ArrayList中的元素超过它的初始大小时,Vector会将它的容量翻倍,而ArrayList只增加50%的大小，这样,ArrayList就有利于节约内存空间.**

**ArrayList和Vector在从指定位置取得元素，从容器的末尾增加和删除元素都非常的有效，所有的这些操作都能在一个常数级的时间(O(1))内完成。但是从一个其他的位置增加和删除一个元素就显得颇为费时，差不多需要的时间为O(n-i)，这里的n代表元素个数，i代表要增加和删除的元素所在的位置。这些操作需花费更多的时间，因为你需要挨个移动i和更高位置的元素。**

****3. ArrayList & LinkedList****

**ArrayList的内部实现是基于内部数组Object[],所以从概念上讲,它更象数组，但LinkedList的内部实现是基于一组连接的记录，所以，它更象一个链表结构，所以，它们在性能上有很大的差别：   
       从上面的分析可知,在ArrayList的前面或中间插入数据时,你必须将其后的所有数据相应的后移,这样必然要花费较多时间，所以,当你的操作是在一列数据的后面添加数据而不是在前面或中间,并且需要随机地访问其中的元素时,使用ArrayList会提供比较好的性能；**

**而访问链表中的某个元素时,就必须从链表的一端开始沿着连接方向一个一个元素地去查找,直到找到所需的元素为止，所以,当你的操作是在一列数据的前面或中间添加或删除数据，并且按照顺序访问其中的元素时，就应该使用LinkedList了。**缺点就是查找非常麻烦 要丛第一个索引开始.