**你给说一下ArrayList?**

实现了List接口的可变数组，这是在ArrayList源码当中注释中的第一句话,

我上大学的时候就看过了，不知道现在还是不是记得非常清楚，另外我当时研究的是jdk1.8，现在的版本的ARrayList的实现有没有更新没有太关注，也是时间太忙了，一直管着好几个项目。

这个集合是线程不安全的，这也是和Vector 最主要的区别，好像是ArrayList的说明里边也是这么说的。

为了说明这个结构，就以添加为例说明一下。

添加元素的时候，第一步就是确认容量是否够用，确认的时候用当前的大小+1 与默认容量值10 进行比较。有人说ArrayList初始的容量是10，这是不对的，只有在添加第一个元素的时候才会容量变成10 . 只有当将要使用的数量大于现有的容量的时候才扩容，也就是他的加载因子1.

扩容后的容量是原来容量的1.5倍，源码里边的算法是用原来的大小右位移1位在加上他本身。为什么不直接乘以1.5，因为这样计算的效率高，我真的很佩服Java的代码，对这样的一点资源也斤斤计较。

扩容以后，重新申请新的内容空间，将原来的内容拷贝到新的内存空间中，如果我没有理解错误的话，这里用的是浅拷贝。这里使用了Arrays工具类的copyOf函数进行的数组拷贝，其实这个工具类的这个方法调用的还是 System里边arraycopy方法。

**LinkedList 和ArrayList 的区别**

这个好像没有啥可说的。

他们不过都是线程不安全的，一个是基于数组，一个基于链表。

LinkedList 用于增删较多，按照索引查找较少的情况下使用。

ArrayList 用于根据索引查找较多，增删较少的情况下使用。

**HashMap 实现Map的一个哈希表。**

存储结构存储键值对的一个数组，也有人给他叫做哈希桶。

以添加为例说明存储的过程

添加的时候

1、根据key 计算他的哈希值。

2 计算将要存放数据的索引地址：算法 用数组的长度减去1 与刚刚计算得到的哈希值进行位与计算，得到的结果就是下标 【(n - 1) & hash】

3，如果寻址到的地方有数据了，则遍历这里边的链表（红黑树）的数据，如果遍历到与添加的数据的key相同，则替换value，否则追加到结尾。

4、当链表长度大于8的时候，链表转换成红黑树。 红黑树是自动平衡的二叉树。

删除： 当红黑树中的节点个数少于6的时候，会在转变成链表结构。

线程安全？ 否

扩容机制:

扩容后的容量是原来的二倍，代码当中使用的是位移算法。加载因子（负载因子） 是0.75.

HashMap 是线程不安全的 ，但是效率高,而Hashtable 线程安全 但是效率低？ 怎样保证效率高而且线程安全?

使用并发包下的ConcurrentHashMap 即可，这里边使用了分段锁。