**各种list之间的却别**

-ArrayList:底层数据结构是 数组结构.是线程不同步的(不安全的).查询速度很快，但是增删较慢。

构造一个初始容量为 10 的空列表，当放满了10个元素后，以50%的长度加长集合容器的长度。

List 接口的大小可变数组的实现,即：可以直接指定容量大小

|--LinkedList:底层数据结构是 双向链表结构。对元素的增删速度很快。但是查询速度很慢。线程是不同步的。

|--Vector:底层数据结构也是数组结构，是线程同步的(安全的),效率低,被ArrayList替代了。1.2版本以后的集合都是不同步的

**HashMap和Hashtable的区别：**

1. Hashtable的大部分方法做了同步，HashMap没有，因此，HashMap不是线程安全的。
2. Hashtable不允许key或者value使用null值，而HashMap可以。
3. 在内部算法上，它们对key的hash算法和hash值到内存索引的映射算法不同。

**String、StringBuffer与StringBuilder的区别。**

Java 平台提供了两种类型的字符串：String和StringBuffer / StringBuilder，它们可以储存和操作字符串。

String是只读字符串，也就意味着String引用的字符串内容是不能被改变的。

StringBuffer和StringBulder类表示的字符串对象可以直接进行修改。

StringBuilder是JDK1.5引入的，它和StringBuffer的方法完全相同，区别在于它是单线程环境下使用的，因为它的所有方面都没有被synchronized修饰，因此它的效率也比StringBuffer略高。

# Android为什么要设计出Bundle而不是直接使用HashMap来进行数据传递？

* Bundle内部是由ArrayMap实现的，ArrayMap的内部实现是两个数组，一个int数组是存储对象数据对应下标，一个对象数组保存key和value，内部使用二分法对key进行排序，所以在添加、删除、查找数据的时候，都会使用二分法查找，只适合于小数据量操作，如果在数据量比较大的情况下，那么它的性能将退化。而HashMap内部则是数组+链表结构，所以在数据量较少的时候，HashMap的Entry Array比ArrayMap占用更多的内存。因为使用Bundle的场景大多数为小数据量，我没见过在两个Activity之间传递10个以上数据的场景，所以相比之下，在这种情况下使用ArrayMap保存数据，在操作速度和内存占用上都具有优势，因此使用Bundle来传递数据，可以保证更快的速度和更少的内存占用。
* 另外一个原因，则是在Android中如果使用Intent来携带数据的话，需要数据是基本类型或者是可序列化类型，HashMap使用Serializable进行序列化，而Bundle则是使用Parcelable进行序列化。而在Android平台中，更推荐使用Parcelable实现序列化，虽然写法复杂，但是开销更小，所以为了更加快速的进行数据的序列化和反序列化，系统封装了Bundle类，方便我们进行数据的传输。