**List,Set,Map用法以及区别**

List,Set,Map是否继承自Collection接口？   
答：List，Set是，Map不是。 Collection是最基本的集合接口，一个Collection代表一组Object，即Collection的元素。一些Collection允许相同的元素而另一些不行。一些能排序而另一些不行。Java JDK不能提供直接继承自Collection的类，Java JDK提供的类都是继承自Collection的"子接口"，如:List和Set。   
注意：Map没有继承Collection接口，Map提供key到value的映射。一个Map中不能包含相同key，每个key只能映射一个value。Map接口提供3种集合的视图，Map的内容可以被当做一组key集合，一组value集合，或者一组key-value映射。 

List按对象进入的顺序保存对象，不做排序或编辑操作。Set对每个对象只接受一次，并使用自己内部的排序方法(通常，你只关心某个元素是否属于Set,而不关心它的顺序--否则应该使用List)。Map同样对每个元素保存一份，但这是基于"键"的，Map也有内置的排序，因而不关心元素添加的顺序。如果添加元素的顺序对你很重要，应该使用 LinkedHashSet或者LinkedHashMap.

详细介绍：   
List特点：元素有放入顺序，元素可重复   
Map特点：元素按键值对存储，无放入顺序   
Set特点：元素无放入顺序，元素不可重复（注意：元素虽然无放入顺序，但是元素在set中的位置是有该元素的HashCode决定的，其位置其实是固定的）   
List接口有三个实现类：LinkedList，ArrayList，Vector   
LinkedList：底层基于链表实现，链表内存是散乱的，每一个元素存储本身内存地址的同时还存储下一个元素的地址。链表增删快，查找慢   
ArrayList和Vector的区别：ArrayList是非线程安全的，效率高；Vector是基于线程安全的，效率低   
Set接口有两个实现类：HashSet(底层由HashMap实现)，LinkedHashSet   
SortedSet接口有一个实现类：TreeSet（底层由平衡二叉树实现）   
Query接口有一个实现类：LinkList   
Map接口有三个实现类：HashMap，HashTable，LinkeHashMap   
  HashMap非线程安全，高效，支持null；HashTable线程安全，低效，不支持null   
SortedMap有一个实现类：TreeMap   
其实最主要的是，list是用来处理序列的，而set是用来处理集的。Map是知道的，存储的是键值对   
set 一般无序不重复.map kv 结构 list 有序 。

**List的功能方法**

　　实际上有两种List: 一种是基本的ArrayList,其优点在于随机访问元素，另一种是更强大的LinkedList,它并不是为快速随机访问设计的，而是具有一套更通用的方法。

　　List : 次序是List最重要的特点：它保证维护元素特定的顺序。List为Collection添加了许多方法，使得能够向List中间插入与移除元素(这只推荐LinkedList使用。)一个List可以生成ListIterator,使用它可以从两个方向遍历List,也可以从List中间插入和移除元素。

　　ArrayList : 由数组实现的List。允许对元素进行快速随机访问，但是向List中间插入与移除元素的速度很慢。ListIterator只应该用来由后向前遍历ArrayList,而不是用来插入和移除元素。因为那比LinkedList开销要大很多。

　　LinkedList : 对顺序访问进行了优化，向List中间插入与删除的开销并不大。随机访问则相对较慢。(使用ArrayList代替。)还具有下列方法：addFirst(), addLast(), getFirst(), getLast(), removeFirst() 和 removeLast(), 这些方法 (没有在任何接口或基类中定义过)使得LinkedList可以当作堆栈、队列和双向队列使用。

**Set的功能方法**

　　Set具有与Collection完全一样的接口，因此没有任何额外的功能，不像前面有两个不同的List。实际上Set就是Collection,只是行为不同。(这是继承与多态思想的典型应用：表现不同的行为。)Set不保存重复的元素(至于如何判断元素相同则较为负责)

　　Set : 存入Set的每个元素都必须是唯一的，因为Set不保存重复元素。加入Set的元素必须定义equals()方法以确保对象的唯一性。Set与Collection有完全一样的接口。Set接口不保证维护元素的次序。

　　HashSet : 为快速查找设计的Set。存入HashSet的对象必须定义hashCode()。

　　TreeSet : 保存次序的Set, 底层为树结构。使用它可以从Set中提取有序的序列。

　　LinkedHashSet : 具有HashSet的查询速度，且内部使用链表维护元素的顺序(插入的次序)。于是在使用迭代器遍历Set时，结果会按元素插入的次序显示。

**Map的功能方法**

　　方法put(Object key, Object value)添加一个“值”(想要得东西)和与“值”相关联的“键”(key)(使用它来查找)。方法get(Object key)返回与给定“键”相关联的“值”。可以用containsKey()和containsValue()测试Map中是否包含某个“键”或“值”。标准的Java类库中包含了几种不同的Map：HashMap, TreeMap, LinkedHashMap, WeakHashMap, IdentityHashMap。它们都有同样的基本接口Map，但是行为、效率、排序策略、保存对象的生命周期和判定“键”等价的策略等各不相同。

　　执行效率是Map的一个大问题。看看get()要做哪些事，就会明白为什么在ArrayList中搜索“键”是相当慢的。而这正是HashMap提高速度的地方。HashMap使用了特殊的值，称为“散列码”(hash code)，来取代对键的缓慢搜索。“散列码”是“相对唯一”用以代表对象的int值，它是通过将该对象的某些信息进行转换而生成的。所有Java对象都能产生散列码，因为hashCode()是定义在基类Object中的方法。

　　HashMap就是使用对象的hashCode()进行快速查询的。此方法能够显著提高性能。

　　Map : 维护“键值对”的关联性，使你可以通过“键”查找“值”

　　HashMap : Map基于散列表的实现。插入和查询“键值对”的开销是固定的。可以通过构造器设置容量capacity和负载因子load factor，以调整容器的性能。

　　LinkedHashMap : 类似于HashMap，但是迭代遍历它时，取得“键值对”的顺序是其插入次序，或者是最近最少使用(LRU)的次序。只比HashMap慢一点。而在迭代访问时发而更快，因为它使用链表维护内部次序。

　　TreeMap : 基于红黑树数据结构的实现。查看“键”或“键值对”时，它们会被排序(次序由Comparabel或Comparator决定)。TreeMap的特点在于，你得到的结果是经过排序的。TreeMap是唯一的带有subMap()方法的Map，它可以返回一个子树。

　　WeakHashMao : 弱键(weak key)Map，Map中使用的对象也被允许释放: 这是为解决特殊问题设计的。如果没有map之外的引用指向某个“键”，则此“键”可以被垃圾收集器回收。

　　IdentifyHashMap : 使用==代替equals()对“键”作比较的hash map。专为解决特殊问题而设计。