**Java 集合List、Set、Map特性总结**

一、前言

　　从接触集合到现在以来，我发现我确实知道集合的大概的概念以及特性，但是却没有一个很清晰的逻辑梳理和使用概念。在使用的时候总是只随性的使用ArrayList、HashMap，几乎不会使用其它类型的集合。所以今天来和它们做个了结。

二、集合结构

Collection

1

├List

│├LinkedList

│├ArrayList

│└Vector

└Set

|-HashSet

|-LinkedHashSet

|-TreeSet

Map

├Hashtable

├HashMap

│├LinkedHashMap

└TreeMap

三、集合详解

1. Collection Map特定的规则

List 必须保持元素特定的顺序

Set 不能有重复元素

Map 一组成对的“键值对”对象

2. Collection

　　Collection 是最基本的集合接口，一个Collcetion代表一组Object的集合，这些Object被称作Collection的元素。Collection是一个接口，用以提供规范定义。

**2-1. List**

　　List集合代表一个元素有序、可以重复的集合，集合中每个元素都有对应的顺序索引。List集合允许加入重复的元素是因为它是通过索引访问指定的集合元素。List元素默认按元素的添加顺序设置元素的索引。

　　除了具有Collection接口必备的iterator()方法外，List还提供一个listIterator()方法，返回一listIterator接口和标准的Iterator接口相比，listIterator多了一些add()之类的方法，允许添加，删除，设定元素，还可以向前或者向后遍历。

　　实现List接口的常用类有LinkedList，ArrayList，Vector和Stack。

**2-1-1. ArrayList**

　　ArrayList 基于的大小可变数组的实现。实现了所有可选列表操作，并允许包括 null 在内的所有元素。除了实现 List 接口外，此类还提供一些方法来操作内部用来存储列表的数组的大小。（此类大致上等同于 Vector 类，除了此类是不同步的。）

　　值得一说的是每一个ArrayList实例都有一个初始容量（Capacity），该容量是指用来存储列表元素数组的大小。随着向ArrayList中不断添加元素，容量也自动增长。

　　\*List arrayList = new ArrayList()；如果像这样使用默认的构造方法，初始容量被设置为10。

　　在添加大量元素前，应用程序可以使用 ensureCapacity 操作来增加 ArrayList 实例的容量。这可以减少递增式再分配的数量。\*

**2-1-2. LinkedList**

　　LinkedList 基于链表结构。实现了所有可选列表操作，并允许包括 null 在内的所有元素。和ArrayList一样LinkedList也是不同步的。

　　该类除了实现了List接口外，还为在列表的开头及结尾的 get、remove 和 insert 元素提供了统一的命名方法。这些操作允许将链接列表用作堆栈、队列或双端队列。实现 Deque 接口，为 add、poll 提供先进先出队列操作，以及其他堆栈和双端队列操作。

2-1-3. Vector

　　 Vector和ArrayList在用法上几乎完全相同，Vector由于使用了synchronized方法（线程安全）所以性能上比ArrayList要差

**2-2. Set**

　　 一个不包含重复元素的 collection。确切地说，set 不包含满足 e1.equals(e2) 的元素对 e1 和 e2，并且最多包含一个 null 元素。

**2-2-1. HashSet**

　　HashSet是Set接口的典型实现，HashSet使用HASH算法来存储集合中的元素，因此具有良好的存取和查找性能。当向HashSet集合中存入一个元素时，HashSet会调用该对象的hashCode()方法来得到该对象的hashCode值，然后根据该HashCode值决定该对象在HashSet中的存储位置。

值得注意的是，HashSet集合判断两个元素相等的标准是两个对象通过equals()方法比较相等，并且两个对象的hashCode()方法的返回值相等，因为任何对象默认集成Object类，而Object类的equals实现比较方法是通过“==”也就是比较内存地址是否相等。所以在没有重写equals方法的情况下要求hashCode()方法的返回值也相等才可以

**2-2-2. LinkedHashSet**

LinkedHashSet集合也是根据元素的hashCode值来决定元素的存储位置，但和HashSet不同的是，它同时使用链表维护元素的次序，这样使得元素看起来是以插入的顺序保存的。

　　当遍历LinkedHashSet集合里的元素时，LinkedHashSet将会按元素的添加顺序来访问集合里的元素。

LinkedHashSet需要维护元素的插入顺序，因此性能略低于HashSet的性能，但在迭代访问Set里的全部元素时(遍历)将有很好的性能(链表很适合进行遍历)

**2-2-3. TreeSet**

　　TreeSet是一个有序的二叉树实现的，所以它是有序的，它的作用是提供有序的Set集合，确保集合元素处于排序状态。

**2-2-4. EnumSet**

　　EnumSet是一个专门为枚举类设计的集合类，EnumSet中所有元素都必须是指定枚举类型的枚举值，该枚举类型在创建EnumSet时显式、或隐式地指定。EnumSet的集合元素也是有序的，

　　EnumSet以枚举值在Enum类内的定义顺序来决定集合元素的顺序

**3. Map**

　　Map 将键映射到值的对象。一个映射不能包含重复的键；每个键最多只能映射到一个值。

**3-1. Hashtable**

　　Hashtable是一个陈旧的Map实现类，此类实现一个哈希表，该哈希表将键映射到相应的值。任何非 null 对象都可以用作。改类是线程同步的而且

　　Hashtable 的实例有两个参数影响其性能：初始容量 和加载因子。容量 是哈希表中桶 的数量，初始容量 就是哈希表创建时的容量。

　　基于哈希表的 Map 接口的实现。此实现提供所有可选的映射操作，并允许使用 null 值和 null 键。（除了非同步和允许使用 null 之外，HashMap 类与 Hashtable 大致相同。）此类不保证映射的顺序，特别是它不保证该顺序恒久

**3-2. HashMap**

HashMap基于哈希表的 Map 接口的实现。此实现提供所有可选的映射操作，并允许使用 null 值和 null 键。（除了非同步和允许使用 null 之外，HashMap 类与 Hashtable 大致相同。）此类不保证映射的顺序，特别是它不保证该顺序恒久不变。

**3-2-1. LinkedHashMap**

LinkedHashMap是HashMap的一个子类，保存了记录的插入顺序，在用Iterator遍历LinkedHashMap时，先得到的记录肯定是先插入的.也可以在构造时用带参数，按照应用次数排序。LinkedHashMap的遍历速度只和实际数据有关，和容量无关，而HashMap的遍历速度和他的容量有关。

**3-3. TreeMap**

　　TreeMap是一个红黑树数据结构，能够把它保存的记录根据键排序,默认是按键值的升序排序，也可以指定排序的比较器，当用Iterator 遍历TreeMap时，得到的记录是排过序的。

**四、总结**

ArrayList和LinkedList：都是线程不安全的，LinkedList基于链表实现所以对于集合的插入和删除要优于动态数组实现的ArrayList，查找却是慢了很多。

HashMap和Hashtable : Hashtable不可以存null值为key或者value。两者效率大致相同，因为它们的实现机制几乎完全一样。但HashMap通常比Hashtable要快一点，因为Hashtable需要额外的线程同步控制。

HashMap和Treemap：HashMap适用于在Map中插入、删除和定位元素。Treemap适用于按自然顺序或自定义顺序遍历键(key)。