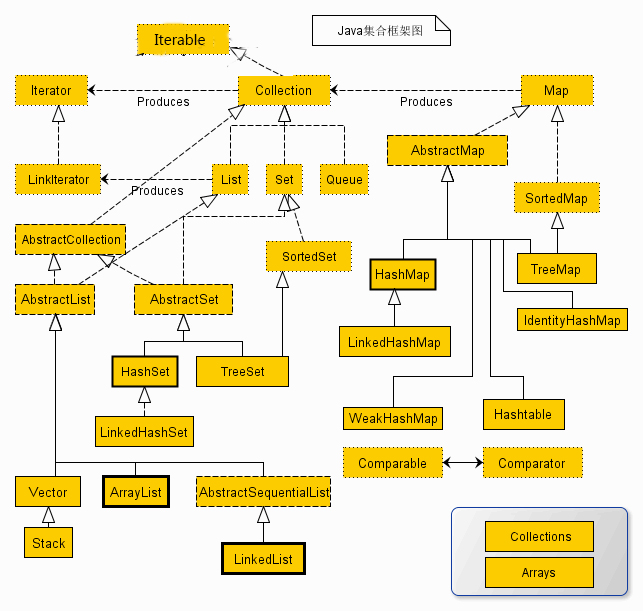
### 介绍Collection框架的结构



上述类图中，**实线边框**的是**实现类**，比如ArrayList，LinkedList，HashMap等，**折线边框**的是**抽象类**，比如AbstractCollection，AbstractList，AbstractMap等，而**点线边框**的是**接口**，比如Collection，Iterator，List等。

　　发现一个特点，上述所有的集合类，都实现了Iterator接口，这是一个用于遍历集合中元素的接口，主要包含hashNext(),next(),remove()三种方法。它的一个子接口LinkedIterator在它的基础上又添加了三种方法，分别是add(),previous(),hasPrevious()。也就是说如果实现Iterator接口，那么在遍历集合中元素的时候，只能往后遍历，被遍历后的元素不会在遍历到，通常无序集合实现的都是这个接口，比如HashSet，HashMap；而那些元素有序的集合，实现的一般都是LinkedIterator接口，实现这个接口的集合可以双向遍历，既可以通过next()访问下一个元素，又可以通过previous()访问前一个元素，比如ArrayList。

　　还有一个特点就是抽象类的使用。如果要自己实现一个集合类，去实现那些抽象的接口会非常麻烦，工作量很大。这个时候就可以使用抽象类，这些抽象类中给我们提供了许多现成的实现，我们只需要根据自己的需求重写一些方法或者添加一些方法就可以实现自己需要的集合类，工作量大大降低。

### 常见java容器的比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口** | **特性** | **实现类** | **实现类特性** | **成员要求** |
| List | 线性、有序的存储容器，可通过索引访问元素 | ArrayList | 数组实现。非同步。 |  |
| Vector | 类似ArrayList，同步。 |  |
| LinkedList | 双向链表。非同步。 |  |
| Map | 保存键值对成员 | HashMap | 基于哈希表的 Map 接口的实现，满足通用需求 | 任意Object对象，如果修改了equals方法，需同时修改hashCode方法 |
| TreeMap | 默认根据自然顺序进行排序，  或者根据创建映射时提供的 [Comparator](mk:@MSITStore:E:\TechnologyMaterial\JDK6API%E4%B8%AD%E6%96%87%E5%8F%82%E8%80%83%5b%E6%B2%88%E4%B8%9C%E8%89%AF%5d(070114).chm::/java/util/Comparator.html)进行排序 | 键成员要求实现Comparable接口，或者使用Comparator构造TreeMap。键成员一般为同一类型。 |
| LinkedHashMap | 类似于HashMap，但迭代遍历时取得“键值对”的顺序是其插入顺序或者最近最少使用的次序 | 与HashMap相同 |
| IdentityHashMap | 使用==取代equals()对“键值”进行比较的散列映射 | 成员通过==判断是否相等 |
| WeakHashMap | 弱键映射，允许释放映射所指向的对象 |  |
| ConcurrentHashMap | 线性安全的Map |  |
| Set | 成员不能重复 | HashSet | 为快速查找设计的Set | 元素必须定义hashCode() |
| TreeSet | 保持次序的Set，底层为树结构 | 元素必须实现Comparable接口 |
| LinkedHashSet | 内部使用链表维护元素的顺序（插入的次序） | 元素必须定义hashCode() |

    注：在满足要求的情况下，Map应尽量使用HashMap，Set应尽量使用HashSet

### Collection 和 Collections的区别。

Collection是集合类的上级接口，继承与他的接口主要有Set 和List.

Collections是针对集合类的一个帮助类，此类不能实例化，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

### Collection框架中实现比较要实现什么接口

comparable/comparator

### ArrayList和Vector的区别

答：

这两个类都实现了List接口（List接口继承了Collection接口），他们都是有序集合，即存储在这两个集合中的元素的位置都是有顺序的，相当于一种动态的数组，我们以后可以按位置索引号取出某个元素，，并且其中的数据是允许重复的，这是与HashSet之类的集合的最大不同处，HashSet之类的集合不可以按索引号去检索其中的元素，也不允许有重复的元素（本来题目问的与hashset没有任何关系，但为了说清楚ArrayList与Vector的功能，我们使用对比方式，更有利于说明问题）。

接着才说ArrayList与Vector的区别，这主要包括两个方面：.   
（1）同步性：

Vector是线程安全的，也就是说是它的方法之间是线程同步的，而ArrayList是线程不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用ArrayList，因为它不考虑线程安全，效率会高些；如果有多个线程会访问到集合，那最好是使用Vector，因为不需要我们自己再去考虑和编写线程安全的代码。

备注：对于Vector&ArrayList、Hashtable&HashMap，要记住线程安全的问题，记住Vector与Hashtable是旧的，是java一诞生就提供了的，它们是线程安全的，ArrayList与HashMap是java2时才提供的，它们是线程不安全的。所以，我们讲课时先讲老的。  
（2）数据增长：

ArrayList与Vector都有一个初始的容量大小，当存储进它们里面的元素的个数超过了容量时，就需要增加ArrayList与Vector的存储空间，每次要增加存储空间时，不是只增加一个存储单元，而是增加多个存储单元，每次增加的存储单元的个数在内存空间利用与程序效率之间要取得一定的平衡。Vector默认增长为原来两倍，而ArrayList的增长策略在文档中没有明确规定（从源代码看到的是增长为原来的1.5倍）。ArrayList与Vector都可以设置初始的空间大小，Vector还可以设置增长的空间大小，而ArrayList没有提供设置增长空间的方法。

总结：即Vector增长原来的一倍，ArrayList增加原来的0.5倍。

### HashMap和Hashtable的区别

（条理上还需要整理，也是先说相同点，再说不同点）

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，在只有一个线程访问的情况下，效率要高于Hashtable。

HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。

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。

就HashMap与HashTable主要从三方面来说。   
一.历史原因:Hashtable是基于陈旧的Dictionary类的，HashMap是Java 1.2引进的Map接口的一个实现   
二.同步性:Hashtable是线程安全的，也就是说是同步的，而HashMap是线程序不安全的，不是同步的   
三.值：只有HashMap可以让你将空值作为一个表的条目的key或value

### List 和 Map 区别?

一个是存储单列数据的集合，另一个是存储键和值这样的双列数据的集合，List中存储的数据是有顺序，并且允许重复；Map中存储的数据是没有顺序的，其键是不能重复的，它的值是可以有重复的。

### List, Set, Map是否继承自Collection接口?

List，Set是，Map不是 . HashSet内部用的是HashMap实现的,只是没有用到HashMap的value部分。

### List、Map、Set三个接口，存取元素时，各有什么特点？

这样的题属于随意发挥题：这样的题比较考水平，两个方面的水平：一是要真正明白这些内容，二是要有较强的总结和表述能力。如果你明白，但表述不清楚，在别人那里则等同于不明白。

首先，List与Set具有相似性，它们都是单列元素的集合，所以，它们有一个功共同的父接口，叫Collection。Set里面不允许有重复的元素，所谓重复，即不能有两个相等（注意，不是仅仅是相同）的对象 ，即假设Set集合中有了一个A对象，现在我要向Set集合再存入一个B对象，但B对象与A对象equals相等，则B对象存储不进去，所以，Set集合的add方法有一个boolean的返回值，当集合中没有某个元素，此时add方法可成功加入该元素时，则返回true，当集合含有与某个元素equals相等的元素时，此时add方法无法加入该元素，返回结果为false。Set取元素时，没法说取第几个，只能以Iterator接口取得所有的元素，再逐一遍历各个元素。

List表示有先后顺序的集合， 注意，不是那种按年龄、按大小、按价格之类的排序。当我们多次调用add(Obj e)方法时，每次加入的对象就像火车站买票有排队顺序一样，按先来后到的顺序排序。有时候，也可以插队，即调用add(int index,Obj e)方法，就可以指定当前对象在集合中的存放位置。一个对象可以被反复存储进List中，每调用一次add方法，这个对象就被插入进集合中一次，其实，并不是把这个对象本身存储进了集合中，而是在集合中用一个索引变量指向这个对象，当这个对象被add多次时，即相当于集合中有多个索引指向了这个对象，如图x所示。List除了可以以Iterator接口取得所有的元素，再逐一遍历各个元素之外，还可以调用get(index i)来明确说明取第几个。

Map与List和Set不同，它是双列的集合，其中有put方法，定义如下：put(obj key,obj value)，每次存储时，要存储一对key/value，不能存储重复的key，这个重复的规则也是按equals比较相等。取则可以根据key获得相应的value，即get(Object key)返回值为key 所对应的value。另外，也可以获得所有的key的结合，还可以获得所有的value的结合，还可以获得key和value组合成的Map.Entry对象的集合。

List 以特定次序来持有元素，可有重复元素。Set 无法拥有重复元素,内部排序。Map 保存key-value值，value可多值。

HashSet按照hashcode值的某种运算方式进行存储，而不是直接按hashCode值的大小进行存储。例如，"abc" ---> 78，"def" ---> 62，"xyz" ---> 65在hashSet中的存储顺序不是62,65,78，这些问题感谢以前一个叫崔健的学员提出，最后通过查看源代码给他解释清楚，看本次培训学员当中有多少能看懂源码。LinkedHashSet按插入的顺序存储，那被存储对象的hashcode方法还有什么作用呢？学员想想!hashset集合比较两个对象是否相等，首先看hashcode方法是否相等，然后看equals方法是否相等。new 两个Student插入到HashSet中，看HashSet的size，实现hashcode和equals方法后再看size。

同一个对象可以在Vector中加入多次。往集合里面加元素，相当于集合里用一根绳子连接到了目标对象。往HashSet中却加不了多次的。

### 说出ArrayList,Vector, LinkedList的存储性能和特性

ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector由于使用了synchronized方法（线程安全），通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

LinkedList也是线程不安全的，LinkedList提供了一些方法，使得LinkedList可以被当作堆栈和队列来使用。

### 去掉一个Vector集合中重复的元素

Vector newVector = new Vector();

For (int i=0;i<vector.size();i++)

{

Object obj = vector.get(i);

if(!newVector.contains(obj);

newVector.add(obj);

}

还有一种简单的方式，HashSet set = new HashSet(vector);

### Set里的元素是不能重复的，那么用什么方法来区分重复与否呢? 是用==还是equals()? 它们有何区别?

Set里的元素是不能重复的，元素重复与否是使用equals()方法进行判断的。

equals()和==方法决定引用值是否指向同一对象equals()在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

### 你所知道的集合类都有哪些？主要方法？

最常用的集合类是 List 和 Map。 List 的具体实现包括 ArrayList 和 Vector，它们是可变大小的列表，比较适合构建、存储和操作任何类型对象的元素列表。 List 适用于按数值索引访问元素的情形。

Map 提供了一个更通用的元素存储方法。 Map 集合类用于存储元素对（称作"键"和"值"），其中每个键映射到一个值。

ArrayList/Vector🡪List

🡪Collection

HashSet/TreeSet🡪Set

Propeties🡪HashTable

🡪Map

Treemap/HashMap

我记的不是方法名，而是思想，我知道它们都有增删改查的方法，但这些方法的具体名称，我记得不是很清楚，对于set，大概的方法是add,remove, contains；对于map，大概的方法就是put,remove，contains等，因为，我只要在eclispe下按点操作符，很自然的这些方法就出来了。我记住的一些思想就是List类会有get(int index)这样的方法，因为它可以按顺序取元素，而set类中没有get(int index)这样的方法。List和set都可以迭代出所有元素，迭代时先要得到一个iterator对象，所以，set和list类都有一个iterator方法，用于返回那个iterator对象。map可以返回三个集合，一个是返回所有的key的集合，另外一个返回的是所有value的集合，再一个返回的key和value组合成的EntrySet对象的集合，map也有get方法，参数是key，返回值是key对应的value。

### 两个对象值相同(x.equals(y) == true)，但却可有不同的hash code，这句话对不对?

对。

如果对象要保存在HashSet或HashMap中，它们的equals相等，那么，它们的hashcode值就必须相等。

如果不是要保存在HashSet或HashMap，则与hashcode没有什么关系了，这时候hashcode不等是可以的，例如arrayList存储的对象就不用实现hashcode，当然，我们没有理由不实现，通常都会去实现的。

### TreeSet里面放对象，如果同时放入了父类和子类的实例对象，那比较时使用的是父类的compareTo方法，还是使用的子类的compareTo方法，还是抛异常！

（应该是没有针对问题的确切的答案，当前的add方法放入的是哪个对象，就调用哪个对象的compareTo方法，至于这个compareTo方法怎么做，就看当前这个对象的类中是如何编写这个方法的）

实验代码：

**public** **class** Parent **implements** Comparable {

**private** **int** age = 0;

**public** Parent(**int** age){

**this**.age = age;

}

**public** **int** compareTo(Object o) {

// **TODO** Auto-generated method stub

System.*out*.println("method of parent");

Parent o1 = (Parent)o;

**return** age>o1.age?1:age<o1.age?-1:0;

}

}

**public** **class** Child **extends** Parent {

**public** Child(){

**super**(3);

}

**public** **int** compareTo(Object o) {

// **TODO** Auto-generated method stub

System.*out*.println("method of child");

// Child o1 = (Child)o;

**return** 1;

}

}

**public** **class** TreeSetTest {

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

TreeSet set = **new** TreeSet();

set.add(**new** Parent(3));

set.add(**new** Child());

set.add(**new** Parent(4));

System.*out*.println(set.size());

}

}