Java容器

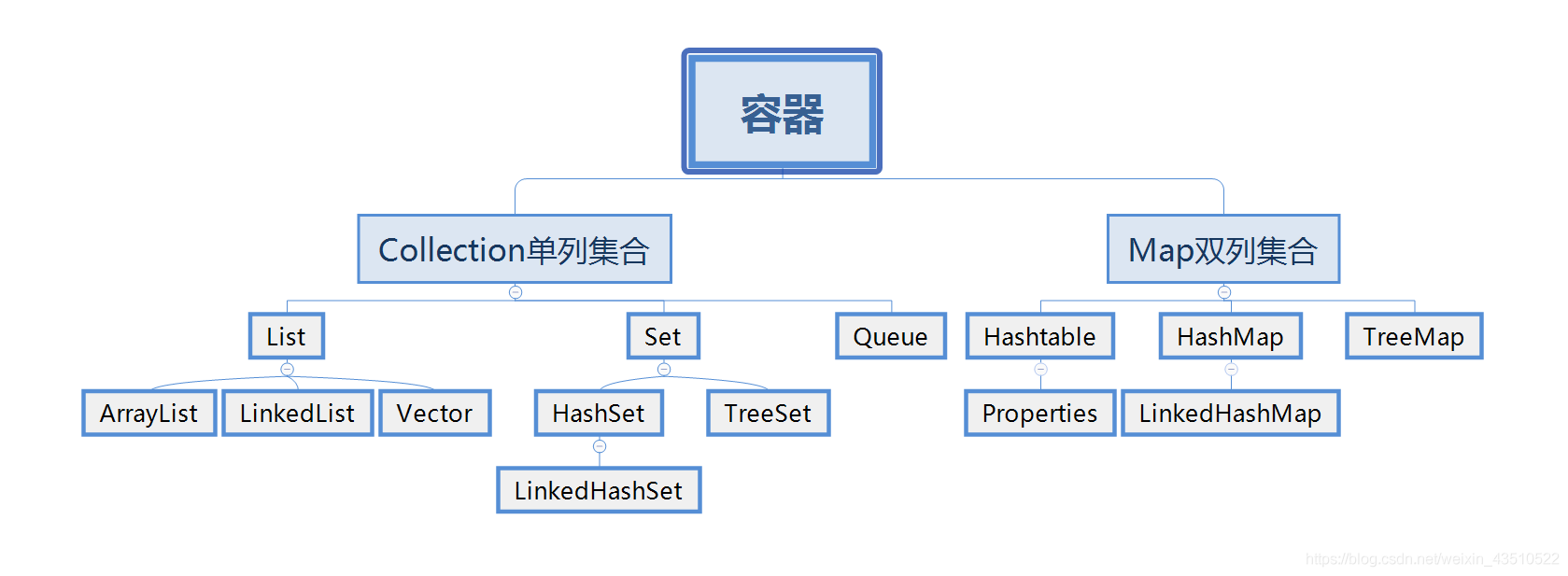
Set: HashSet，LinkedHashSet，TreeSet，EnumSet（后三个有序）

List: ArrayList，LinkedList，Stack，Vector（全部有序）

Queue：LinkedBlockingQueue，ArraysBlockingQueue（全部有序）

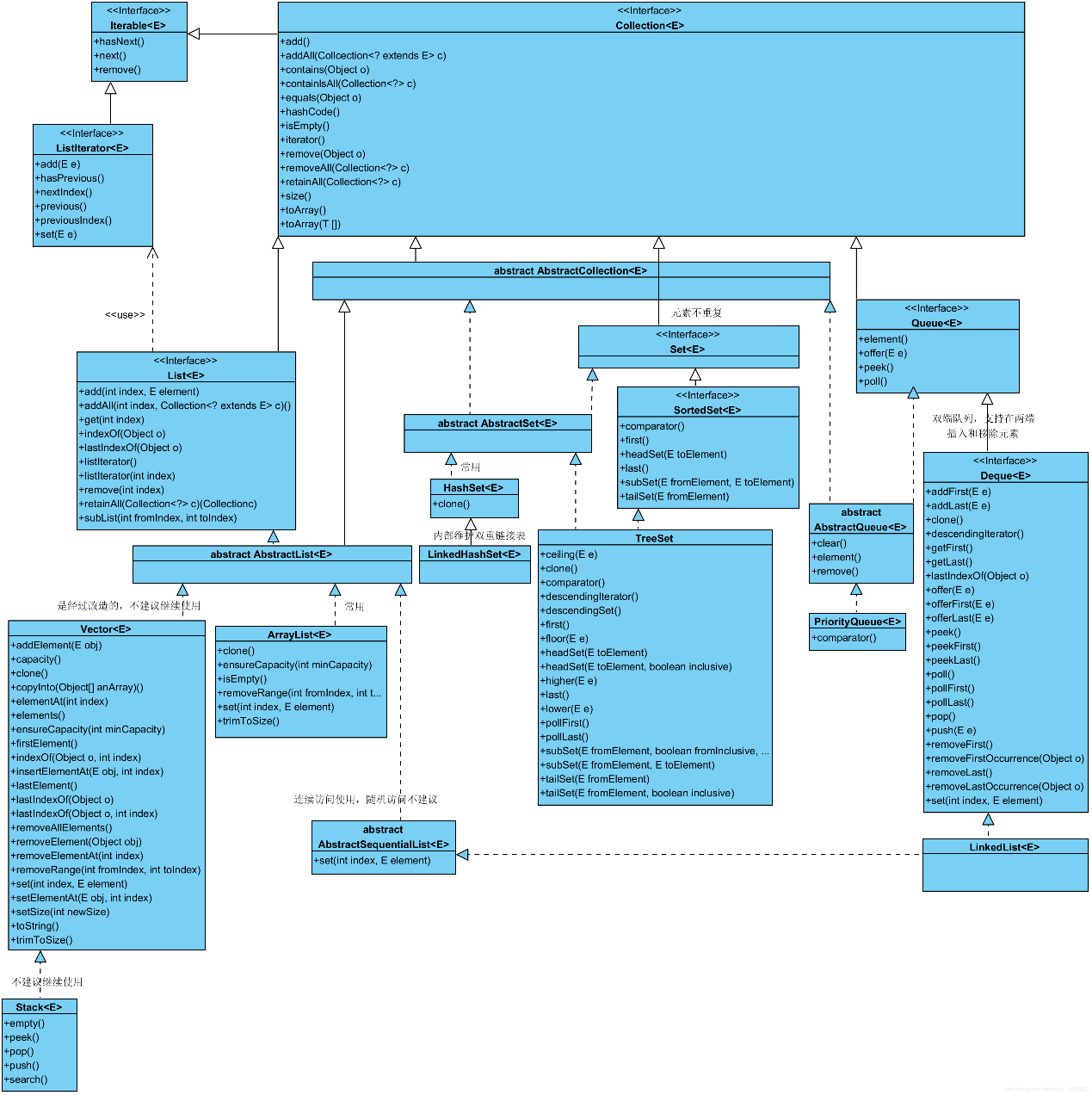
Map: HashMap，TreeMap，LinkedHashMap，EnumMap，ConcurrentHashMap，IdentityHashMap（前三有序）

1、简单概括容器间继承关系



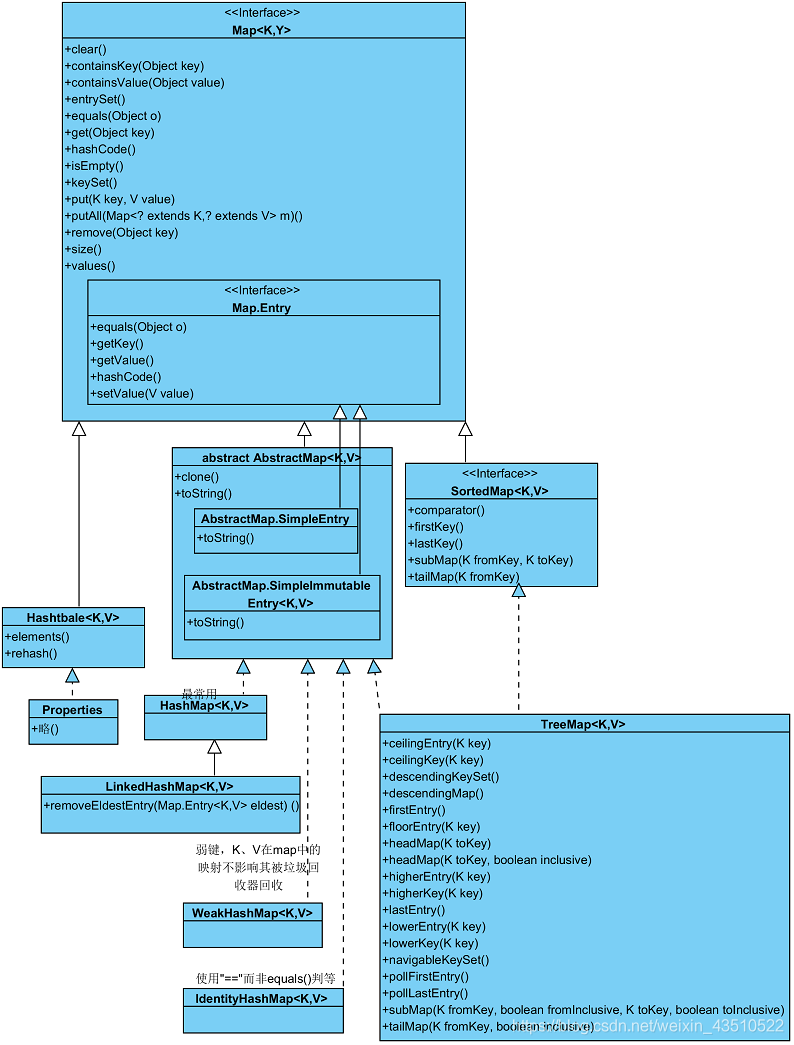
**2、Collection及子类/接口容器具体继承关系**

​ Collection接口并不是一个根接口，它的超级接口是Iterator，需要提供其遍历、移除元素(可选操作)的能力。



##### 3、Map及子类/接口容器具体继承关系

​ Map是将键(key)映射到值(value)的对象。不同的映射不能包含相同的键；每个键最多只能映射到一个值。



4、几个面试常见问题：

1.Q:ArrayList和Vector有什么区别？HashMap和HashTable有什么区别？

A:Vector和HashTable是线程同步的（synchronized）。性能上，ArrayList和HashMap分别比Vector和Hashtable要好。

2.Q:大致讲解java集合的体系结构

A:List、Set、Map是这个集合体系中最主要的三个接口。

其中List和Set继承自Collection接口。

Set不允许元素重复。HashSet和TreeSet是两个主要的实现类。

List有序且允许元素重复。ArrayList、LinkedList和Vector是三个主要的实现类。

Map也属于集合系统，但和Collection接口不同。Map是key对value的映射集合，其中key列就是一个集合。key不能重复，但是value可以重复。HashMap、TreeMap和Hashtable是三个主要的实现类。

SortedSet和SortedMap接口对元素按指定规则排序，SortedMap是对key列进行排序。

3.Q：Comparable和Comparator区别

A:调用java.util.Collections.sort(List list)方法来进行排序的时候，List内的Object都必须实现了Comparable接口。

java.util.Collections.sort(List list，Comparator c)，可以临时声明一个Comparator 来实现排序。

Collections.sort(imageList, new Comparator() {

public int compare(Object a, Object b) {

int orderA = Integer.parseInt( ( (Image) a).getSequence());

int orderB = Integer.parseInt( ( (Image) b).getSequence());

return orderA - orderB;

}

});

如果需要改变排列顺序

改成return orderb - orderA 即可。

List接口对Collection进行了简单的扩充，它的具体实现类常用的有ArrayList和LinkedList。你可以将任何东西放到一个List容器中，并在需要时从中取出。ArrayList从其命名中可以看出它是一种类似数组的形式进行存储，因此它的随机访问速度极快，而LinkedList的内部实现是链表，它适合于在链表中间需要频繁进行插入和删除操作。在具体应用时可以根据需要自由选择。前面说的Iterator只能对容器进行向前遍历，而ListIterator则继承了Iterator的思想，并提供了对List进行双向遍历的方法。

Set接口也是Collection的一种扩展，而与List不同的时，在Set中的对象元素不能重复，也就是说你不能把同样的东西两次放入同一个Set容器中。它的常用具体实现有HashSet和TreeSet类。HashSet能快速定位一个元素，但是你放到HashSet中的对象需要实现hashCode()方法，它使用了前面说过的哈希码的算法。而TreeSet则将放入其中的元素按序存放，这就要求你放入其中的对象是可排序的，这就用到了集合框架提供的另外两个实用类Comparable和Comparator。一个类是可排序的，它就应该实现Comparable接口。有时多个类具有相同的排序算法，那就不需要在每分别重复定义相同的排序算法，只要实现Comparator接口即可。集合框架中还有两个很实用的公用类：Collections和Arrays。Collections提供了对一个Collection容器进行诸如排序、复制、查找和填充等一些非常有用的方法，Arrays则是对一个数组进行类似的操作。

Map是一种把键对象和值对象进行关联的容器，而一个值对象又可以是一个Map，依次类推，这样就可形成一个多级映射。对于键对象来说，像Set一样，一个Map容器中的键对象不允许重复，这是为了保持查找结果的一致性;如果有两个键对象一样，那你想得到那个键对象所对应的值对象时就有问题了，可能你得到的并不是你想的那个值对象，结果会造成混乱，所以键的唯一性很重要，也是符合集合的性质的。当然在使用过程中，某个键所对应的值对象可能会发生变化，这时会按照最后一次修改的值对象与键对应。对于值对象则没有唯一性的要求。你可以将任意多个键都映射到一个值对象上，这不会发生任何问题（不过对你的使用却可能会造成不便，你不知道你得到的到底是那一个键所对应的值对象）。Map有两种比较常用的实现：HashMap和TreeMap。HashMap也用到了哈希码的算法，以便快速查找一个键，TreeMap则是对键按序存放，因此它便有一些扩展的方法，比如firstKey(),lastKey()等，你还可以从TreeMap中指定一个范围以取得其子Map。键和值的关联很简单，用pub(Object key,Object value)方法即可将一个键与一个值对象相关联。用get(Object key)可得到与此key对象所对应的值对象。

遍历Map的方式：

//最常规的一种遍历方法，最常规就是最常用的，虽然不复杂，但很重要，这是我们最熟悉的，就不多说了！！

public static void work(Map<String, Student> map) {

Collection<Student> c = map.values();

Iterator it = c.iterator();

for (; it.hasNext();) {

System.out.println(it.next());

}

}

// 利用keyset进行遍历，它的优点在于可以根据你所想要的key值得到你想要的 values，更具灵活性！！

public static void workByKeySet(Map<String, Student> map) {

Set<String> key = map.keySet();

for (Iterator it = key.iterator(); it.hasNext();) {

String s = (String) it.next();

System.out.println(map.get(s));

}

}

// 比较复杂的一种遍历在这里，呵呵~~他很暴力哦，它的灵活性太强了，想得到什么就能得到什么~~

public static void workByEntry(Map<String, Student> map) {

Set<Map.Entry<String, Student>> set = map.entrySet();

for (Iterator<Map.Entry<String, Student>> it = set.iterator(); it

.hasNext();) {

Map.Entry<String, Student> entry = (Map.Entry<String, Student>) it

.next();

System.out.println(entry.getKey() + "—>" + entry.getValue());

}

}

//Map.Entry的另外一种简练写法(foreach遍历方式)

public static void workByEntry(Map<String, Student> map) {

Set<Map.Entry<String, Student>> set = map.entrySet();

for (Map.Entry<String, Student> me : set) {

System.out.println(me.getKey() + "—>" + me.getValue());

}

}