

# Collection

public interface Collection<E> extends Iterable<E> {

...

}

泛型<E>即该Collection中元素对象的类型，继承的Iterable是定义的一个遍历操作接口，采用hasNext next的方式进行遍历。具体实现还是放在具体类中去实现。

List是存储的元素容器是有个有序的可以索引到元素的容器，并且里面的元素可以重复。

Set里面和List最大的区别是Set里面的元素对；

## List

List提供了一种特殊的iterator遍历器，叫做ListIterator。这种遍历器允许遍历时插入，替换，删除，双向访问。 并且还有一个重载方法允许从一个指定位置开始遍历。

### ArrayList

1. ArrayList是一个实现了List接口的可变数组
2. 可以插入null
3. 它的size, isEmpty, get, set, iterator,add这些方法的时间复杂度是O(1),如果add n个数据则时间复杂度是O(n).
4. ArrayList不是synchronized的。

所有元素都是保存在一个Object数组中，然后通过size控制长度。

transient Object[] elementData;

private int size;

### LinkedList

LinkedList是一个链表维护的序列容器。和ArrayList都是序列容器，一个使用数组存储，一个使用链表存储。

 查找方面。数组的效率更高，可以直接索引出查找，而链表必须从头查找。

 插入删除方面。特别是在中间进行插入删除，这时候链表体现出了极大的便利性，只需要在插入或者删除的地方断掉链然后插入或者移除元素，然后再将前后链重新组装，但是数组必须重新复制一份将所有数据后移或者前移。

 在内存申请方面，当数组达到初始的申请长度后，需要重新申请一个更大的数组然后把数据迁移过去才行。而链表只需要动态创建即可。

## Set

### HashSet

HashSet实现了Set接口，基于HashMap进行存储。遍历时不保证顺序，并且不保证下次遍历的顺序和之前一样。HashSet中允许null元素。

private transient HashMap<E,Object> map;

private static final Object PRESENT = new Object();

HashSet的集合其实就是HashMap的key的集合，然后HashMap的value默认都是PRESENT。HashMap的定义即是key不重复的集合。使用HashMap实现，这样HashSet就不需要再实现一遍。所以所有的add，remove等操作其实都是HashMap的add、remove操作。遍历操作其实就是HashMap的keySet的遍历；

public Iterator<E> iterator() {

return map.keySet().iterator();

}

public boolean contains(Object o) {

return map.containsKey(o);

}

public boolean add(E e) {

return map.put(e, PRESENT)==null;

}

public void clear() {

map.clear();

}

### LinkedHashSet

LinkedHashSet的核心概念相对于HashSet来说就是一个可以保持顺序的Set集合。HashSet是无序的，LinkedHashSet会根据add，remove这些操作的顺序在遍历时返回固定的集合顺序。这个顺序不是元素的大小顺序，而是可以保证2次遍历的顺序是一样的。

类似HashSet基于HashMap的源码实现，LinkedHashSet的数据结构是基于LinkedHashMap。

### TreeSet

TreeSet即是一组有次序的集合，如果没有指定排序规则Comparator，则会按照自然排序。（自然排序即e1.compareTo(e2) == 0作为比较）

注意：TreeSet内的元素必须实现Comparable接口。

TreeSet源码的算法即基于TreeMap

| **Set实现** | **使用场景** | **数据结构** |
| --- | --- | --- |
| HashSet | 无序的、无重复的数据集合 | 基于HashMap |
| LinkedSet | 维护次序的HashSet | 基于LinkedHashMap |
| TreeSet | 保持元素大小次序的集合，元素需要实现Comparable接口 | 基于TreeMap |

# Map

Map是一个保存键值映射的对象。 映射Map中不能包含重复的key，每一个key最多对应一个value。

public interface Map<K,V> {

...

interface Entry<K,V> {

K getKey();

V getValue();

...

}

}

## 遍历Map的方法

### 1、Set<K> keySet()

1）通过keySet()获取到所有key的集合，因为key是不可重复的是，所以返回set集合格式；

2）对返回的set集合进行遍历，然后get方法获取value；

### 2、Collection<V> values()

直接获取到所有的value集合，但无法获取到key。适用于不需要key。然后通过迭代器进行遍历；

Collection<String> collection = map.values();//返回值是个值的Collection集合

### 3、Set< Map.Entry< K, V>> entrySet()

将整个Entry对象作为元素返回所有的数据。然后遍历Entry，分别再通过getKey和getValue获取key和value；

//通过entrySet()方法将map集合中的映射关系取出（这个关系就是Map.Entry类型）

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

//将关系集合entrySet进行迭代，存放到迭代器中

Iterator<Map.Entry<String, String>> it = entrySet.iterator();

while(it.hasNext()) {

Map.Entry<String, String> me = it.next();//获取Map.Entry关系对象me

String key = me.getKey();//通过关系对象获取key

String value = me.getValue();//通过关系对象获取value

}

| **Map实现** | **使用场景** | **数据结构** |
| --- | --- | --- |
| HashMap | 哈希表存储键值对，key不重复，无序 | 哈希散列表 |
| LinkedHashMap | 是一个可以记录插入顺序和访问顺序的HashMap | 存储方式是哈希散列表，但维护了头尾指针用来记录顺序 |
| TreeMap | 具有元素排序功能 | 红黑树 |
| WeakHashMap | 弱键映射，映射之外无引用的键，可以被垃圾回收 | 哈希散列表 |