# 1. Set集合

Set也是接口，同样继承自接口Collection。Set集合常用的实现类有HashSet、TreeSet和LinkedHashSet。Set接口中没有什么特有的方法。

Set集合的特点在于Set集合中元素时唯一的，即不允许集合中存储重复的元素。并且HashSet和TreeSet集合元素是无序的，即元素的存储顺序和取出顺序无关，无固定取出顺序，这与他们的实现原理是相关的，学习之后就会了解。

先用HashSet体会一下Set集合，和普通的集合用法都有普遍性。

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.HashSet; **import** java.util.Iterator;  **public class** Demo {  **public static void** main(String[] args) {  HashSet<String> set = **new** HashSet<>();  set.add(**"hello"**);  set.add(**"world"**);  set.add(**"java"**);  *// 用迭代器遍历  // set集合不像List集合有get(index)方法，因此现在只能用迭代器遍历* **for**(Iterator<String> it = set.iterator(); it.hasNext(); ) {  System.***out***.println(it.next());  }  */\* 从运行的结果：  world  java  hello  看，就知道存储与取出顺序不一致了。  \*/* } } |

## 1.1 HashSet

HashSet的底层数据结构是哈希表（就是元素为链表的数组），线程不安全，效率较高。

HashSet的特点是不存储相同的元素，依赖的是对象的hashCode()和equals()方法。最好先研究一下哈希表的结构和存储的过程（数据结构的知识），才好理解下面讲的HashSet的存储原理。

HashSet存储数据的步骤如下：

（1）当向集合添加一个新对象时，先调用该对象的hashCode()方法得到返回值，由此返回值算出此元素在哈希表中的数组索引，也就是决定了该元素存储在哪个数组中（如果能被添加的话）。

（2）如果哈希表中该数组位置还没有元素，那么直接将该元素添加到此索引处，反之，如果此位置已经有元素了（即该元素哈希值和之前添加过来的某元素相同了），那么就调用该对象的equals()方法和该数组索引处所有的元素进行比较，如果此元素和该索引处某元素相等，则此元素不会添加到集合中，否则将此对象添加在该数组索引处，与原来的元素构成链表。

通过上述步骤，能够实现HashSet存储的元素唯一。我们可以重写自定义对象的equals()方法，来实现HashSet集合不会存储我们认为的重复的元素，否则HashSet只会根据对象的内存地址判断两个对象是否相等。

与此同时，我们也应该重写hashCode()方法，使hashCode()返回值能尽可能区分两个不同的对象，这样能避免链表太长而影响查找元素的效率。一般hashCode()返回该对象所有属性值相加总值的hashCode码。（可参见我之前写的Student类示例）

String类本身重写了equals()和hashCode()方法，因此在HashSet<String>对象中，不能存储相同的字符串，可自行实验。

调用原理:先判断hashCode() 方法的值,如果相同才会去判断equals 如果不相同,是不会调用equals方法的。

当调用add()方法添加元素没有添加成功时，add()返回值是false（已存在相同元素了）。

## 1.2 LinkedHashSet

LinkedHashSet底层同样是哈希表和链表，存储的元素是唯一的，这点原理和HashSet都相同，可重写自定义类的hashCode()和equals()方法。LinkedHashSet和HashSet相比，特点就是能保持元素存储顺序和取出顺序的一致性（保证元素有序），可根据需要选用。

## 1.3 TreeSet

TreeSet的底层是红黑树结构，红黑树是一种自平衡的二叉树。TreeSet也不能存储相同的元素，TreeSet的特点是能够对元素进行排序。要先了解存储的原理才能明白是怎么实现元素的唯一性和排序的：

以下面的代码为例：

|  |
| --- |
| **package** com.zhang.test;  **import** java.util.TreeSet;  **public class** Demo {  **public static void** main(String[] args) {  TreeSet<Integer> ts = **new** TreeSet<>();  ts.add(20);  ts.add(18);  ts.add(23);  ts.add(17);  ts.add(18);  ts.add(24);  System.***out***.println(ts);  } } |