06 集合_Set

JAVAEE高级

一：Set接口

> 1、Set 集合不允许包含相同的元素，如果试把两个相同的元素加入同一个 Set 集合中，则添加操作失败
>
> 2、Set 判断两个对象是否相同不是使用 == 运算符，而是根据 equals 方法

二、HashSet

> 1、HashSet 是 Set 接口的典型实现，大多数时候使用 Set 集合时都使用这个实现类
>
> 2、HashSet 按 Hash算法来存储集合中的元素，因此具有很好的存取和查找性能

```
1.      @Test
2.      public void getTest() {
3.              //demo1();
4.              HashSet<String> hs = new HashSet<>();
5.              hs.add("李四");
6.              hs.add("张三");
7.              hs.add("李四");
8.              hs.add("李四");
9.              System.out.println(hs);
10.     }
```

1、HashSet的特点

> 1、不能保证元素的排列顺序
> 2、HashSet 不是线程安全的
> 3、集合元素可以是 null

> 4、HashSet 集合判断两个元素相等的标准：两个对象通过 hashCode() 方法比较相等，并且两个对象的 equals() 方法返回值也相等

2、案列

①：《去除重复的对象》

> 1、键盘读取一行输入,去掉其中重复字符, 打印出不同的那些字符

```
@Test
    public void getVoid() {
        // 1,创建Scanner对象
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入一行字符串:");
        // 2,创建HashSet对象,将字符存储,去掉重复
        HashSet<Character> hs = new HashSet<>();
        // 3,将字符串转换为字符数组,获取每一个字符存储在HashSet集合中,自动去除重复
        String line = sc.nextLine();
        char[] arr = line.toCharArray();
        for (char c : arr) { // 遍历字符数组
            hs.add(c);
        }
        // 4,遍历HashSet,打印每一个字符
        for (Character ch : hs) {
            System.out.print(ch);
        }
    }
```

②：《去除重复的对象》

> 1、需要:使用HashSet去除重复的对象(值是相同)；对象自定义

```
public class Test {
     private Integer age;
 public Test(Integer age) {
        super();
        this.age = age;
    }
public static void main(String[] args) {
```

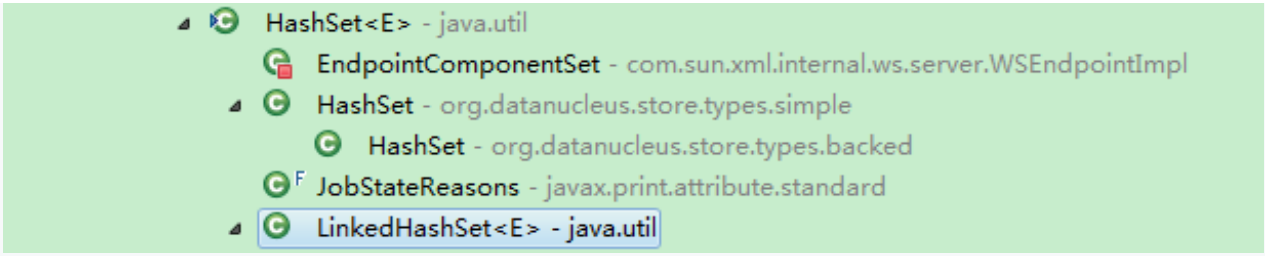
```
        HashSet<Test> hs=new HashSet();
        hs.add(new Test(12));
        hs.add(new Test(12));
        hs.add(new Test(13));
        hs.add(new Test(14));
        Iterator it=hs.iterator();
        while(it.hasNext()){
            Test p=(Test)it.next(); //强制转换
            System.out.println(p.getAge()); //有重复元素
        }
    }
    //对应的get set方法
```

> 重写hashCode()；equals();方法就无重复的元素

```
    /**
     * 其他代码如上：
     * 重写equals方法
     * */
    @Override
    public boolean equals(Object arg0) {
       if(this==arg0) return true;
       if(!(arg0 instanceof Test)) //这里是判断obj对象是否是Test类的一个实例。
           throw new ClassCastException("类型错误");  //输入类型错误
       Test p = (Test)arg0;//强制转换
       return  this.age==p.age;  //说明姓名和年龄相同则为同一元素
    }
    /**
     * 重写hashCode方法
     * */
    @Override
    public int hashCode() {
        return age;
    }
```

三、LinkedHashSet

> 1、LinkedHashSet 是 HashSet 的子类



2、LinkedHashSet插入性能略低于 HashSet，但在迭代访问 Set 里的全部元素时有很好的性能

3、LinkedHashSet 不允许集合元素重复；保证元素唯一的,与HashSet的原理一样

4、底层是链表实现的,是set集合中唯一一个能保证怎么存就怎么取的集合对象

```
@Test
public void getTest() {
    LinkedHashSet<String> lhs = new LinkedHashSet<>();
    lhs.add("a");
    lhs.add("a");
    lhs.add("a");
    lhs.add("a");
    lhs.add("b");
    lhs.add("c");
    lhs.add("d");
}
```

四、TreeSet

1、TreeSet 可以确保集合元素处于排序状态,同样他也可以保证元素的唯一

2、TreeSet 两种排序方法：自然排序和定制排序

3、默认情况下，TreeSet 采用自然排序

1、自然排序

1、TreeSet 会调用集合元素的 compareTo(Object obj) 方法来比较元素之间的大小关系，然后将集合元素按升序排列

2、如果试图把一个对象添加到 TreeSet 时，则该对象的类必须实现 Comparable 接口

3、实现 Comparable 的类必须实现 compareTo(Object obj) 方法，两个对象即通过 compareTo(Object obj) 方法的返回值来比较大小

4、向 TreeSet 中添加元素时，只有第一个元素无须比较compareTo()方法，后面添加的所有元素都会调用compareTo()方法进行比较

5、因为只有相同类的两个实例才会比较大小，所以向 TreeSet 中添加的应该是同一个类的对象

6、对于 TreeSet 集合而言，它判断两个对象是否相等的唯一标准是：两个对象通过 compareTo(Object obj) 方法比较返回值

```
1.      @Test
2.      public void getTest() {
3.          TreeSet<String> ts = new TreeSet<>();
4.          ts.add("D");
5.          ts.add("B");
6.          ts.add("C");
7.          ts.add("A");
8.          ts.add("F");
9.          System.out.println(ts);
10.     }
```

2、定制排序

1、如降序排列，可通过Comparator接口的帮助。需要重写compare(T o1,T o2)方法

2、利用int compare(T o1,T o2)方法，比较o1和o2的大小

3、如果方法返回正整数，则表示o1大于o2；如果返回0，表示相等；返回负整数，表示o1小于o2

4、要实现定制排序，需要将实现Comparator接口的实例作为形参传递给TreeSet的构造器

5、此时，仍然只能向TreeSet中添加类型相同的对象。否则发生ClassCastException异常

> 6、使用定制排序判断两个元素相等的标准是：通过Comparator比较两个元素返回了0

①：存入字符串

```
class CompareByLen implements Comparator<String> {
    @Override
    public int compare(String s1, String s2) { // 按照字符串的长度比较
        int num = s1.length() - s2.length(); // 长度为主要条件
        return num == 0 ? s1.compareTo(s2) : num; // 内容为次要条件
    }
}

/***
 *将字符串按照长度排序
 */

    @Test
    public void getTest() {
        TreeSet<String> ts = new TreeSet<>(new CompareByLen());
        ts.add("AAA");
        ts.add("BB");
        ts.add("CCCC");
        ts.add("DDD");
        ts.add("FFF");
        System.out.println(ts);
    }
```

②:存入对象

```
public class Person implements Comparable<Person> {
    private String name;
    private int age;
    //对应的get set 方法和构造方法

    public int compareTo(Person o) {
        int length = this.name.length() - o.name.length(); // 比较长度为主要条件
        int num = length == 0 ? this.name.compareTo(o.name) : length; // 比较内容为次要条件
        return num == 0 ? this.age - o.age : num; // 比较年龄为次要条件
    }
    }


```

```
14.
15.     @Test
16.     public void getTest() {
17.         TreeSet<Person> ts = new TreeSet<>();
18.         ts.add(new Person("AAAA", 23));
19.         ts.add(new Person("AAA", 13));
20.         ts.add(new Person("AA", 33));
21.         ts.add(new Person("A", 43));
22.         ts.add(new Person("BBBBB", 53));
23.     }
```

3、案列《获取10个1至20的随机数，要求随机数不能重复》

> 1、编写一个程序，获取10个1至20的随机数，要求随机数不能重复，并进行升序的方式进行

```
1.     @Test
2.     public void getTest() {
3.         // 1,有Random类创建随机数对象
4.         Random r = new Random();
5.         // 2,需要存储10个随机数,而且不能重复,所以我们用TreeSet集合
6.         TreeSet<Integer> hs = new TreeSet<>();
7.         // 3,如果TreeSet的size是小于10就可以不断的存储,如果大于等于10就停止存储
8.         while (hs.size() < 10) {
9.             // 4,通过Random类中的nextInt(n)方法获取1到20之间的随机数,并将这些随机数存储在HashSet集合中
10.             hs.add(r.nextInt(20) + 1);
11.         }
12.         // 5,遍历HashSet
13.         for (Integer integer : hs) {
14.             System.out.println(integer);
15.         }
16.     }
```

五、综合案列

1、字符串有序(字典顺序)

> 1、需求：一个集合中存储了无序并且重复的字符串,定义一个方法,让其有序(字典顺序),而且还不能去除重复

```
@Test
    public void getVoid() {
            // 1,定义一个List集合,并存储重复的无序的字符串
            ArrayList<String> list = new ArrayList<>();
            list.add("aaaa");
            list.add("aaa");
            list.add("ccc");
            list.add("ddd");
            list.add("fffffffffff");
            list.add("bbbb");
            list.add("a");
            list.add("aa");
            // 2,定义方法对其排序保留重复
            sort(list);
            // 3,打印list
            System.out.println(list);
        }

    /*
     * 定义方法,排序并保留重复
     *    分析:
     * 1,创建TreeSet集合对象,因为String本身就具备比较功能,但是重复不会保留,所以我们用比较器
     * 2,将list集合中所有的元素添加到TrreSet集合中,对其排序,保留重复 ,清空list集合
     * 4,将TreeSet集合中排好序的元素添加到list中
     */
    public static void sort(List<String> list) {
        // 1,创建TreeSet集合对象,因为String本身就具备比较功能,但是重复不会保留,所以我们用比较器
        TreeSet<String> ts = new TreeSet<>(new Comparator<String>() {
            @Override
            public int compare(String s1, String s2) {
                int num = s1.compareTo(s2); // 比较内容为主要条件
                return num == 0 ? 1 : num; // 保留重复
            }
        });
        // 2,将list集合中所有的元素添加到TrreSet集合中,对其排序,保留重复
        ts.addAll(list);
        // 3,清空list集合
```

```
            list.clear();
            // 4,将TreeSet集合中排好序的元素添加到list中
            list.addAll(ts);
        }
```

六、HashSet 和 TreeSet 的区别?

1、HashSet

> 存储结构：采用 Hashtable 哈希表存储结构
> 优点：添加速度快，查询速度快，删除速度快
> 缺点：无序

2、TreeSet

> 存储结构： 采用二叉树的存储结构
> 优点：有序（排序后的升序）查询速度比 List 快（按照内容查询）
> 缺点：查询速度没有 HashSet 快