【济南中心】JAVA 编程阶梯：基础篇之第十七章

集合框架：

Set 集合概述和特点：

Set 集合与 Collection 基本上完全一样，它没有提供任何额外的方法。实际上 Set 就是 Collection，只是行为略有不同。（Set 不允许包含重复元素）。例：

```
import java.util.HashSet;
import java.util.Set;
public class SetTest {
        public static void main(String[] args) {
                Set books = new HashSet();
                books.add(new String("java"));
                boolean result = books.add(new String("java"));
                System.out.println(result + "-->" + books);
        }
}
```

HashSet：

(1)HashSet 是 Set 接口的实现。HashSet 按 Hash 算法来存储集合中的元素，具有很好的存取和查找性能。

(2)HashSet 不能保证元素的排列顺序，顺序可能与添加顺序不同，顺序也有可能发生变化。

(3)当向 HashSet 集合中存入一个元素时，HashSet 会调用该对象的 hashCode() 方法来得到该对象的 hashCode 值，然后根据该 HashCode 值决定该对象在 HashSet 中的存储位置。如果有两个元素。例子：

```
HashSet<Person> hs = new HashSet<>();
hs.add(new Person("张三", 23));
hs.add(new Person("张三", 23));
hs.add(new Person("李四", 23));
hs.add(new Person("李四", 23));
hs.add(new Person("王五", 23));
hs.add(new Person("赵六", 23));
```

HashSet 重写 hashCode()和 equals()方法，目的和实现：如果有两个元素通过 equals()方法比较返回 true ,但它们的 hashCode()方法返回值不相等 ,HashSet 将会把它们存储在不同的位置，依然可以添加成功。即，HashSet 集合判断两个元素相等的标准是两个。

对象通过 equals()方法比较相等，并且两个对象的 hashCode()方法返回值也相等。

HashSet 保证元素唯一性的原理：

1.HashSet 原理

* 我们使用 Set 集合都是需要去掉重复元素的，如果在存储的时候逐个 equals()比较，效率较低,哈希算法提高了去重复的效率，降低了使用 equals()方法的次数

* 当 HashSet 调用 add()方法存储对象的时候，先调用对象的 hashCode()方法得到一个哈希值，然后在集合中查找是否有哈希值相同的对象

* 如果没有哈希值相同的对象就直接存入集合

* 如果有哈希值相同的对象，就和哈希值相同的对象逐个进行 equals()比较,比较结果为 false 就存入，true 则不存

2.将自定义类的对象存入 HashSet 去重复

* 类中必须重写 hashCode()和 equals()方法

* hashCode()：属性相同的对象返回值必须相同，属性不同的返回值尽量不同(提高效率)

* equals()：属性相同返回 true，属性不同返回 false,返回 false 的时候存储

LinkedHashSet 的概述和使用：

LinkedHashSet 集合也是根据元素的 hashCode 值来决定元素的存储位置，但它同时使用链表维护元素的次序，这样使得元素看起来是以插入的顺序保存的。特点：当遍历 LinkedHashSet 集合里的元素时，LinkedHashSet 将会按元素的添加顺序来访问集合里的元素。所以说 LinkedHashSet 可以保证怎么存就怎么取 。

案例演示：

1.需求：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。并把最终的随机数输出到控制台。代码如下：

```
HashSet<Integer> hs = new HashSet<>();                    //创建集合对象
Random r = new Random();                  //创建随机数对象
while(hs.size() < 10) {
int num = r.nextInt(20) + 1;                  //生成 1 到 20 的随机数
hs.add(num);
}
for (Integer integer : hs) {                  //遍历集合
System.out.println(integer);                  //打印每一个元素
}
```

2. 使用 Scanner 从键盘读取一行输入,去掉其中重复字符, 打印出不同的那些字符，aaaabbbcccddd。代码如下：

```
Scanner sc = new Scanner(System.in);                    //创建键盘录入对象
System.out.println("请输入一行字符串:");
String line = sc.nextLine();                  //将键盘录入的字符串存储在 line 中
char[] arr = line.toCharArray();                  //将字符串转换成字符数组
HashSet<Character> hs = new HashSet<>();                    //创建 HashSet 集合对象
for(char c : arr) {                  //遍历字符数组
hs.add(c);                  //将字符数组中的字符添加到集合中
}
for (Character ch : hs) {                  //遍历集合
System.out.println(ch);
}
```

3. 将集合中的重复元素去掉

```
public static void main(String[] args) {
                        ArrayList<String> list = new ArrayList<>();
                        list.add("a");
                        list.add("a");
                        list.add("a");
                        list.add("b");
                        list.add("b");
                        list.add("b");
                        list.add("b");
                        list.add("c");
                        list.add("c");
                        list.add("c");
                        list.add("c");

                        System.out.println(list);
                        System.out.println("去除重复后:");
                        getSingle(list);
                        System.out.println(list);
                }

                /*
                 * 将集合中的重复元素去掉
                 * 1,void
                 * 2,List<String> list
                 */

                public static void getSingle(List<String> list) {
                        LinkedHashSet<String> lhs = new LinkedHashSet<>();
                        lhs.addAll(list);
                       //将 list 集合中的所有元素添加到 lhs
                        list.clear();
                         //清空原集合
                        list.addAll(lhs);
                      //将去除重复的元素添回到 list 中
                }
```

TreeSet：

1.特点

* TreeSet 是用来排序的，可以指定一个顺序，对象存入之后会按照指定的顺序排列

* 2.使用方式

* a.自然顺序(Comparable)

* TreeSet 类的 add()方法中会把存入的对象提升为 Comparable 类型

* 调用对象的 compareTo()方法和集合中的对象比较

* 根据 compareTo()方法返回的结果进行存储

* b.比较器顺序(Comparator)

* 创建 TreeSet 的时候可以制定 一个 Comparator

* 如果传入了 Comparator 的子类对象, 那么 TreeSet 就会按照比较器中的顺序排序

* add()方法内部会自动调用 Comparator 接口中 compare()方法排序

* 调用的对象是 compare 方法的第一个参数,集合中的对象是 compare 方法的第二个参数

* c.两种方式的区别

* TreeSet 构造函数什么都不传, 默认按照类中 Comparable 的顺序(没有就报错 ClassCastException)

* TreeSet 如果传入 Comparator, 就优先按照 Comparator

练习：

在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。

```
public static void main(String[] args) {
                                ArrayList<String> list = new ArrayList<>();
```

```
                list.add("ccc");
                list.add("ccc");
                list.add("aaa");
                list.add("aaa");
                list.add("bbb");
                list.add("ddd");
                list.add("ddd");

                sort(list);
                System.out.println(list);
        }

        /*
          * 对集合中的元素排序,并保留重复
          * 1,void
          * 2,List<String> list
          */
        public static void sort(List<String> list) {
                TreeSet<String> ts = new TreeSet<>(new Comparator<String>()
{               //定义比较器(new Comparator(){}是 Comparator 的子类对象)

                        @Override
                        public int compare(String s1, String s2)
{                               //重写 compare 方法
                                int num =
s1.compareTo(s2);                                                       //比
较内容
                                return num == 0 ? 1 :
num;                                                            //如果内容一样返
回一个不为 0 的数字即可
                        }
                });

                ts.addAll(list);
                                                //将 list 集合中的所有元素添加到 ts
中
                list.clear();
                                                        //清空 list
                list.addAll(ts);
                                                //将 ts 中排序并保留重复的结果在添
加到 list 中
        }
```

2. 从键盘接收一个字符串, 程序对其中所有字符进行排序,例如键盘输

入: helloitcast 程序打印:acehillostt

```
Scanner sc = new Scanner(System.in);                    //创建键盘录入对象
        System.out.println("请输入一行字符串:");
        String line = sc.nextLine();                                    //将键盘录入的字符串存储在 line 中
        char[] arr = line.toCharArray();                                //将字符串转换成字符数组
        TreeSet<Character> ts = new TreeSet<>(new Comparator<Character>() {

                @Override
                public int compare(Character c1, Character c2) {
                        //int num = c1.compareTo(c2);
                        int num = c1 - c2;                              //自动拆箱

                        return num == 0 ? 1 : num;
                }
        });

        for(char c : arr) {
                ts.add(c);
        }

        for(Character ch : ts) {
                System.out.print(ch);
        }
```

3. 程序启动后, 可以从键盘输入接收多个整数, 直到输入 quit 时结束输入. 把所有输入的整数倒序排列打印.

```
Scanner sc = new Scanner(System.in);                    //创建键盘录入对象
                System.out.println("请输入:");
                TreeSet<Integer> ts = new TreeSet<>(new Comparator<Integer>() {//将比较器传给 TreeSet 的构造方法

                        @Override
                        public int compare(Integer i1, Integer i2) {

                                //int num = i2 - i1;                    //自动拆箱

                                int num = i2.compareTo(i1);
```

```
                return num ==
0 ? 1 : num;
            }
        });

        while(true) {
            String line =
sc.nextLine();                          //将键盘录入的字符串存储在 line
中
            if("quit".equals(line))
                //如果字符串常量
和变量比较,常量放前面,这样不会出现空指针异常,变量里面可能存储 null
                break;
            try {
                int num =
Integer.parseInt(line);                 //将数字字符串转换成数字
                ts.add(num);
            } catch (Exception e) {
                System.out.p
rintln("您录入的数据有误,请输入一个整数");
            }

        }

        for (Integer i : ts)
{
    //遍历 TreeSet 集合

            System.out.println(i);
        }
```

4. 键盘录入 5 个学生信息(姓名,语文成绩,数学成绩,英语成绩),按照总分从高到低输出到控制台。

```
Scanner sc = new Scanner(System.in);

        System.out.println("请输入 5 个学生成绩格式
是:(姓名,语文成绩,数学成绩,英语成绩)");

        TreeSet<Student> ts = new TreeSet<>(new
Comparator<Student>() {

            @Override
            public int compare(Student s1,
Student s2) {

                int num =
s2.getSum() - s1.getSum();              //根据学生的总成绩降
序排列
```

```
                return num ==
0 ? 1 : num;
            }
        });

        while(ts.size() < 5) {
            String line = sc.nextLine();
            try {
                String[] arr
= line.split(",");
                int chinese =
Integer.parseInt(arr[1]);
                //转
换语文成绩
                int math =
Integer.parseInt(arr[2]);
                //转
换数学成绩
                int english =
Integer.parseInt(arr[3]);
                //转
换英语成绩
                ts.add(new
Student(arr[0], chinese, math, english));
            } catch (Exception e) {
                System.out.p
rintln("录入格式有误,输入 5 个学生成绩格式是:(姓名,语文成绩,数学成绩,英语成绩");
            }

        }

        System.out.println("排序后的学生成绩是:");
        for (Student s : ts) {
            System.out.println(s);
        }
```

EnumSet 类

EnumSet 是一个专为枚举类设计的集合类 ,EnumSet 中的所有元素都必须是指定枚举类型的枚举值。

EnumSet 类没有暴露任何构造器来创建该类的实例，程序应该通过它提供的 static 方法来创建 EnumSet 对象。

static EnumSet allOf(Class elementType)：创建一个包含指定枚举类里所有

枚举值的 EnumSet 集合。

static EnumSet complementOf(EnumSet s)：创建一个其元素类型与指定 EnumSet 里元素类型相同的 EnumSet 集合 新 EnumSet 集合包含原 EnumSet 集合所不包含的、此枚举类剩下的枚举值（

即新 EnumSet 集合和原 EnumSet 集合的集合元素加起来就是该枚举类的所有枚举值）。

static EnumSet copyOf(Collection c)：使用一个普通集合来创建 EnumSet 集合。

static EnumSet copyOf(EnumSet s)：创建一个与指定 EnumSet 具有相同元素类型、相同集合元素的 EnumSet 集合。

static EnumSet noneOf(Class elementType)：创建一个元素类型为指定枚举类型的空 EnumSet。

static EnumSet of(E first, E...rest)：创建一个包含一个或多个枚举值的 EnumSet 集合，传入的多个枚举值必须属于同一个枚举类。

static EnumSet range(E from, E to)：创建一个包含从 from 枚举值到 to 枚举值范围内所有枚举值的 EnumSet 集合。

例子：

```
import java.util.EnumSet;

enum Season{
        SPRING,SUMMER,FAIL,WINTER
}
public class EnumSetTest {
        public static void main(String[] args) {
                EnumSet es1 = EnumSet.allOf(Season.class);
                System.out.println(es1);
                EnumSet es2 = EnumSet.noneOf(Season.class);
```

```
                System.out.println(es2);
                es2.add(Season.WINTER);
                es2.add(Season.SPRING);
                System.out.println(es2);
                EnumSet es3 = EnumSet.of(Season.SUMMER , Season.WINTER);
                System.out.println(es3);
                EnumSet es4 = EnumSet.range(Season.SUMMER, Season.WINTER);
                System.out.println(es4);
                EnumSet es5 = EnumSet.complementOf(es4);
                System.out.println(es5);
        }
}
```

总结：

1.List

* a.普通 for 循环，使用 get()逐个获取

* b.调用 iterator()方法得到 Iterator，使用 hasNext()和 next()方法

* c.增强 for 循环，只要可以使用 Iterator 的类都可以用

* d.Vector 集合可以使用 Enumeration 的 hasMoreElements()和 nextElement()方法

2.Set

* a.调用 iterator()方法得到 Iterator，使用 hasNext()和 next()方法

* b.增强 for 循环，只要可以使用 Iterator 的类都可以用

3.普通 for 循环,迭代器,增强 for 循环是否可以在遍历的过程中删除

识别二维码 关注黑马程序员视频库
免费获得更多 IT 资源