day13

集合的排序

java.util.Collections类

Collections是集合的工具类,里面定义了很多静态方法用于操作集合.

Collections.sort(List list)方法

可以对List集合进行自然排序(从小到大)

```
package collection;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

/**
 * 集合的排序
 * 集合的工具类:java.util.Collections提供了一个静态方法sort,可以对List集合
 * 进行自然排序
 */
public class SortListDemo1 {
```

```
    public static void main(String[] args)
{
        List<Integer> list = new
ArrayList<>();
        Random random = new Random();
        for(int i=0;i<10;i++){
            list.add(random.nextInt(100));
        }
        System.out.println(list);
        Collections.sort(list);
        System.out.println(list);
    }
}
```

排序自定义类型元素

```
package collection;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * 排序自定义类型元素
 */
public class SortListDemo2 {
    public static void main(String[] args)
{
        List<Point> list = new ArrayList<>
();
```

```
            list.add(new Point(1,2));
            list.add(new Point(97,88));
            list.add(new Point(7,6));
            list.add(new Point(9,9));
            list.add(new Point(5,4));
            list.add(new Point(2,3));
            System.out.println(list);
            /*
                编译不通过的原因:
                Collections.sort(List list)该方法要求集合中的元素类型必须实现接口:
                Comparable,该接口中有一个抽象方法compareTo,这个方法用来定义元素之间比较
                大小的规则.所以只有实现了该接口的元素才能利用这个方法比较出大小进而实现排序
                操作.
             */
            Collections.sort(list);//编译不通过 compare比较  comparable可以比较的
            System.out.println(list);
        }
}
```

实际开发中,我们并不会让我们自己定义的类(如果该类作为集合元素使用)去实现Comparable接口,因为这对我们的程序有**侵入性**.

侵入性:当我们调用某个API功能时,其要求我们为其修改其他额外的代码,这个现象就是侵入性.侵入性越强的API越不利于程序的后期可维护性.应当尽量避免.

重载的Collections.sort(List list,Comparator c)方法

```
package collection;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

/**
 * 排序自定义类型元素
 */
public class SortListDemo2 {
    public static void main(String[] args)
{
        List<Point> list = new ArrayList<>
();
        list.add(new Point(1,2));
        list.add(new Point(97,88));
        list.add(new Point(7,6));
        list.add(new Point(9,9));
        list.add(new Point(5,4));
        list.add(new Point(2,3));

        System.out.println(list);
        /*
```

```
                Collections.sort(List list)在排
序List集合时要求集合元素必须实现了
                Comparable接口。实现了该接口的类必须
重写一个方法compareTo用与定义比较
                大小的规则，从而进行元素间的比较后排
序。否则编译不通过。

                侵入性:
                当我们调用某个API时，其反过来要求我们
为其修改其他额外的代码，这种现象就
                成为侵入性。侵入性不利于程序后期的维
护，尽可能避免。
                compare:比较
            */
//          Collections.sort(list);

            //匿名内部类的形式创建一个比较器
          Comparator<Point> com = new
Comparator<Point>() {
                @Override
                /**
                 * 实现比较器接口后必须重写方法
compare.
                 * 该方法用来定义参数o1与参数o2的比
较大小规则
                 * 返回值用来表示o1与o2的大小关系
                 */
                public int compare(Point o1,
Point o2) {
```

```
                int len1 = o1.getX() *
o1.getX() + o1.getY() * o1.getY();
                int len2 = o2.getX() *
o2.getX() + o2.getY() * o2.getY();
                return len1-len2;
            }
        };
        Collections.sort(list,com);//回调模式



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

最终没有侵入性的写法

```
package collection;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
```

```
/**
 * 排序自定义类型元素
 */
public class SortListDemo2 {
    public static void main(String[] args)
{
        List<Point> list = new ArrayList<>
();
        list.add(new Point(1,2));
        list.add(new Point(97,88));
        list.add(new Point(7,6));
        list.add(new Point(9,9));
        list.add(new Point(5,4));
        list.add(new Point(2,3));

        System.out.println(list);
        /*
            Collections.sort(List list)在排
序List集合时要求集合元素必须实现了
            Comparable接口。实现了该接口的类必须
重写一个方法compareTo用与定义比较
            大小的规则，从而进行元素间的比较后排
序。否则编译不通过。

            侵入性:
            当我们调用某个API时，其反过来要求我们
为其修改其他额外的代码，这种现象就
            称为侵入性。侵入性不利于程序后期的维
护，尽可能避免。
            compare:比较
```

```
//          */
//          Collections.sort(list);

//        //匿名内部类的形式创建一个比较器
//          Comparator<Point> com = new
Comparator<Point>() {
//              @Override
//              /**
//               * 实现比较器接口后必须重写方法
compare.
//               * 该方法用来定义参数o1与参数o2的
比较大小规则
//               * 返回值用来表示o1与o2的大小关系
//               */
//              public int compare(Point o1,
Point o2) {
//                  int len1 = o1.getX() *
o1.getX() + o1.getY() * o1.getY();
//                  int len2 = o2.getX() *
o2.getX() + o2.getY() * o2.getY();
//                  return len1-len2;
//              }
//          };
//          Collections.sort(list,com);//回调
模式

//          Collections.sort(list,new
Comparator<Point>() {
//              public int compare(Point o1,
Point o2) {
```

```
//                int len1 = o1.getX() *
o1.getX() + o1.getY() * o1.getY();
//                int len2 = o2.getX() *
o2.getX() + o2.getY() * o2.getY();
//                return len1-len2;
//            }
//        });

        Collections.sort(list,(o1,o2)->
                o1.getX() * o1.getX() +
o1.getY() * o1.getY() -
                o2.getX() * o2.getX() -
o2.getY() * o2.getY()
        );

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

排序字符串

java中提供的类,如:String,包装类都实现了Comparable接口,但有时候这些比较规则不能满足我们的排序需求时,同样可以临时提供一种比较规则来进行排序.

```java
package collection;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class SortListDemo3 {
    public static void main(String[] args)
{
        List<String> list = new ArrayList<>
();
//          list.add("Tom");
//          list.add("jackson");
//          list.add("rose");
//          list.add("jill");
//          list.add("ada");
//          list.add("hanmeimei");
//          list.add("lilei");
//          list.add("hongtaoliu");
//          list.add("Jerry");

        list.add("传奇");
        list.add("小泽老师");
        list.add("苍老师");
        System.out.println(list);

        //按照字符多少排序
//          Collections.sort(list);
```

```
//        Collections.sort(list, new
Comparator<String>() {
//            public int compare(String o1,
String o2) {
////                return o1.length()-
o2.length();
//                return o2.length()-
o1.length();//反过来减就是降序
//            }
//        });

        Collections.sort(list,(o1,o2)-
>o2.length()-o1.length());
        System.out.println(list);
    }
}
```