集合

1. Collection集合问答

1.1 List集合特点

LinkedList　Vector　ArrayList的区别?

1.2 List、Set集合对比

简述list集合和set集合的特点。

2. Collection集合遍历

假设顺序列表ArrayList中存储的元素是整型数字1~5，遍历每个元素，将每个元素顺序输出。（你能想到的所有方式）

3. 方法应用

3.1 contains()

ArrayList去除集合中字符串的重复值(字符串的内容相同则代表重复)

3.2 retainAll()、removeAll()

有两个集合list1和list2，list1中存放了1，2，3，4，5五个元素，list2中存放了4，5，6三个元素，按要求完成下列操作

1. 打印list1和list2的交集（list1和list2中都有的元素）
2. 打印list1和list2的差集（list1中有，list2中没有的元素）

4. 集合使用

4.1 List集合的使用

定义两个集合，完成List集合的定义和遍历。

【要求】

- 定义一个集合list1存入20个随机整数
- 筛选list1中的偶数元素
- 定义一个集合list2用来接收筛选出的偶数元素
- 遍历list2

4.2 Set集合的使用

定义一个集合，完成Set集合的定义和遍历。

【要求】

- Set集合中的元素为10个整数
- 10个整数元素为1-20的随机数
- 10个整数元素互不重复

5. TreeSet排序问题

5.1 奇偶排序

将1-10按照奇数在前偶数在后，奇数正序，偶数倒序的方式保存到Set集合中（排序）

【示例】

输出结果为：

```
[1, 3, 5, 7, 9, 10, 8, 6, 4, 2]
```

5.2 成绩单应用

将学生按照成绩保存到集合中，并且名字叫tom的学生不管考多少分都位于班级的第一位。

代码结构：

```
public class Test5_2_ReportCards  {
    public static void main(String[] args) {
        // 定义Set集合并传入一个自定义比较规则的Comparator
        // 编写代码


        // 测试
        Student student1 = new Student("tom", 80);
        Student student2 = new Student("mike", 90);
```

```
10          Student student3 = new
    Student("lily", 20);
11          Student student4 = new
    Student("chris", 23);
12          set.add(student3);
13          set.add(student2);
14          set.add(student1);
15          set.add(student4);
16
17          for (Student student : set) {
18              System.out.println(student);
19          }
20      }
21  }
22
23  class Student {
24      private String name;
25      private int score;
26
27      public Student(String name, int score) {
28          super();
29          this.name = name;
30          this.score = score;
31      }
32
33      public String getName() {
34          return name;
```

```
35      }
36
37      public void setName(String name) {
38          this.name = name;
39      }
40
41      public int getScore() {
42          return score;
43      }
44
45      public void setScore(int score) {
46          this.score = score;
47      }
48
49      @Override
50      public String toString() {
51          return "Student [name=" + name + ",
    score=" + score + "]";
52      }
53  }
```

5.3 对比应用

通过两种方式实现：Comparable接口和Comparator接口

【要求】

- 设计Teacher类，包含属性id，name,age，创建Teacher对象。
- 将对象保存在TreeSet集合中并且排序。
- 排序规则为：先通过名字排序，名字相同的时候通过年龄进行排序，年龄相同时通过id排序。

6. List集合自定义实现（选做题）

6.1 自定义实现ArrayList

实现MyList代码，模拟ArrayList的部分功能。

代码结构：

```java
public interface MyList {
    public void add(int index,Object obj);
//在指定位置添加对象
    public void add(Object obj);
//在最后位置添加对象
    public Object remove(int index);
//删除指定位置上的对象
    public void set(int index,Object obj);
//修改指定位置上的数据
    public Object get(int index);
//获取指定位置上的数据
    public int size();
//获取当前数据结构当前的长度
    public void clear();
//清空所有的数据
}
```

```java
public class MyArrayList {
    // 定义数组容器
    private Object[] elementData;
    // 默认容量为10
    private static final int
DEFAULT_CAPACITY = 10;
    // 实际ArrayList的大小
    private int size;

    // 提供有参构造器 自定义初始容量
```

```
10      public MyArrayList(int initialCapacity)
    {
11          if (initialCapacity < 0) {
12              throw new
    IllegalArgumentException("Illegal Capacity:
    " + initialCapacity);
13          }
14          this.elementData = new
    Object[initialCapacity];
15      }
16
17      // 无参构造初始化 将默认10传递给有参构造器
18      public MyArrayList() {
19          this(DEFAULT_CAPACITY);
20      }
21
22      // 在尾部添加元素
23      public void add(Object object) {
24
25      }
26
27      // 在指定位置添加元素
28      public void add(int index, Object
    object) {
29          rangeCheckForAdd(index);
30
31      }
```

```
32
33      // 获取指定索引元素
34      public Object get(int index) {
35          rangeCheckForAdd(index);
36
37      }
38
39      // 检查数组是否越界
40      private void rangeCheckForAdd(int index)
    {
41          if (index < 0 || index > size) {
42              throw new
    ArrayIndexOutOfBoundsException("数组越界异
    常");
43          }
44      }
45
46      // 删除指定索引元素
47      public Object remove(int index) {
48          rangeCheckForAdd(index);
49
50      }
51
52      // 删除对象
53      public boolean remove(Object o) {
54
55      }
```

```
56
57      // 清空数据
58      public void clear() {
59
60      }
61
62      // 扩容
63      private void grow(int minCapacity) {
64
65      }
66
67      // 测试
68      public static void main(String[] args) {
69          MyArrayList my = new MyArrayList(1);
70          my.add(1);
71          my.add(2);
72          my.add(3);
73          System.out.println("获取第一个位置" +
    my.get(1));
74          my.remove(1);
75          my.clear();
76          System.out.println(my.size);
77      }
78  }
```

6.2 自定义实现LinkedList

完成MyLinkedList代码，模拟LinkedList的部分功能

代码结构：

```
public class MyLinkedList{
    private class Node{
        //存放元素的值
        private Object data;
        //存放上一个节点
        private Node pre;
        //存放下一个节点
        private Node next;
    }

    //补全以下四种方法
    public void add(Object o){}
    public void add(int index,Object o){}
    public void remove(int index){}
    public Object get(int index){}
}
```

范型类版本：

```

public class MyLinkedList<T> {
    private Node<T> first;// 定义头节点
```

```
4       private Node<T> last;// 定义尾节点
5       private int size;// 链表的长度
6
7       private class Node<T> {
8           private T data;// 存放元素的值
9           private Node<T> pre; // 存放上一个节点
10          private Node<T> next;// 存放下一个节点
11      }
12
13      // 补全以下四种方法
14      // 添加元素方法
15      public void add(T t) {
16
17      }
18
19      // 将元素添加到指定索引方法
20      public void add(int index, T t) throws
    Exception {
21          // 校验参数
22          checkIndex(index);
23
24      }
25
26      // 根据索引删除元素方法
27      public void remove(int index) throws
    Exception {
28          // 校验参数
```

```
29          checkIndex(index);
30
31      }
32
33      // 根据索引获取元素方法
34      private Node<T> get(int index) throws
    Exception {
35
36      }
37
38      // 获取MyLinkedList长度方法
39      public int getSize() {
40          return this.size;
41      }
42
43      // 展示数据方法
44      public void show() throws Exception {
45          for (int i = 0; i < getSize(); i++)
    {
46              if (i == getSize() - 1) {
47
     System.out.println(get(i).data);
48              }else {
49                  System.out.print(get(i).data
    + " -> ");
50              }
51          }
```

```
52      }
53
54      // 校验参数方法
55      private void checkIndex(int index)
    throws Exception {
56          if (index < 0 || index >= getSize())
    {
57              throw new Exception("参数index不
    合法");
58          }
59      }
60
61      // 测试
62      public static void main(String[] args)
    throws Exception {
63          MyLinkedList<Integer> linkedList =
    new MyLinkedList<>();
64          linkedList.add(1);
65          linkedList.add(2);
66          linkedList.add(3);
67          linkedList.add(4);
68          linkedList.add(5);
69          linkedList.show();
70          linkedList.add(2, 1);
71          linkedList.show();
72          linkedList.remove(2);
73          linkedList.show();
```

```
74          }
75      }
```

7. Map集合问答

1）map可以使用迭代器遍历吗？为什么？

2）map的键和值能为空吗？为什么？

8. Map集合使用和遍历

1）创建一个Map集合，里面有如下元素{香蕉 = 5.6 ， 樱桃 = 25 ， 桃子 = 5.6 ， 苹果 = 2.3}，按要求完成：

- 遍历该map集合，并且统计有多少种水果（key）用两种方式进行遍历
- 将“香蕉”的价格修改为10.9，并删除桃子这组数据，将修改后的map中所有key和value都输出到控制台

2）统计字符串中每个字符出现的次数，String s = "aabbddccaefdd";

【提示】

- 可使用Map<Character, Integer>集合，键存放字符，值存放出现的次数

9. Map集合常用方法

按要求完成以下步骤：

1. 键盘录入整行字符串(sc.nextLine())，遇到quit结束录入，录入字符串格式为：学号.姓名.年龄.分数

   【示例】

   001.zs.20.68
   003.tom.19.78
   quit

2. 拆解上述整行的字符串得到属性值，然后实例化学生对象，按照"."分割字符串

   1） split()方法的应用
   2）String --> Integer 的转换

3. 将学生对象添加到Map<学生,学号>集合中
   要求：添加时 按照成绩逆序排列

4. 遍历集合

5. 判断 003号学生是否存在
   如果存在，删除该学生
6. 再次遍历集合(采用 另一种方式 遍历)

Student实体类代码结构：

```
public class Student {
    private String id;
    private String name;
    private int age;
    private double score;

    public Student() {
    }

    public Student(String id, String name, int age, double score) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.score = score;
    }

    public String getId() {
        return id;
```

```
19      }
20
21      public void setId(String id) {
22          this.id = id;
23      }
24
25      public String getName() {
26          return name;
27      }
28
29      public void setName(String name) {
30          this.name = name;
31      }
32
33      public int getAge() {
34          return age;
35      }
36
37      public void setAge(int age) {
38          this.age = age;
39      }
40
41      public double getScore() {
42          return score;
43      }
44
45      public void setScore(double score) {
```

```
46          this.score = score;
47      }
48
49      @Override
50      public String toString() {
51          return "Student{" +
52                  "id='" + id + '\'' +
53                  ", name='" + name + '\'' +
54                  ", age=" + age +
55                  ", score=" + score +
56                  '}';
57      }
58  }
```

泛型

10. 泛型概念

使用泛型的好处是，什么是泛型擦除

11. 泛型使用

定义一个通用的(带泛型)的方法，该方法可以返回Collection集合中的最大值

注解

12. 注解概念

简述以下四个元注解的作用

- @Target
- @Retention
- @Documented
- @Inherited