Java 程序设计实验报告

学号：1173710105

姓名：曾钰城

专业：软件工程

班级：1737101

哈尔滨工业大学

实验五：集合对象程序设计

一、实验目的

1）了解集合的概念和基本接口
2）掌握增强 for 循环语句
3）掌握范型的应用
4）掌握基本集合对象 ArrayList 的应用

二、 实验内容

1）将 OOBMI 改造为 CollectionBMI 类；将 CollectionBMI 中的 Student[] students 改造为 **ArrayList<Student> students**；

2）改造 genStudents 函数，将随机生成的学生对象保存到 **ArrayList<Student> students** 中。

3）增加 inputStudents 函数和相关的 isExists 函数，不要求用户输入学生人数，通过询问用户是否继续输入来决定是否继续输入学生，并判断输入的学生是否已经存在，如果不存在则将输入的学生对象保存到 **ArrayList<Student> students** 中。

4）增加五个 comparator 子类(内部类)，能够利用 Collections.sort 函数对学生分别按照学号、姓名、身高、体重、BMI 进行排序

5）改造 printStatics 函数，该函数可以打印所有学生基本信息（**利用增强 for 语句**），以及统计结果信息。打印时，每个学生的信息打印为一行，为了清晰，学号、姓名、身高、体重和计算后的 bmi 值之间用制表符(\t)隔开；打印完学生信息后，打印 BMI 统计信息。

6）分别改造统计 bmi 的均值、中值、众数、方差等统计信息的 4 个函数。

7）改造 menu 函数，提供输入学生、随机生成学生、打印学生、5 种排序、打印统计信息、退出程序等 10 个菜单功能，用户输入指定选项后，运行相应函数功能。

8）在 CollectionBMI 的 main 函数中，调用 menu 函数，测试运行各项功能。

注意，身高、体重、及 bmi 等数值均需保留两位小数的格式进行存储和显示。

三、实验结果

注意：将程序代码和运行结果截图粘贴在此处，注意源代码中注释行数不少于全部代码的 1/3，程序源代码请压缩后上传，压缩文件按照 学号.zip 进行命名，注意源程序于报告请分别上传到不同的文件夹中！

程序代码：

```
package edu.hit.java.exp3.hit1173710105;

import java.util.*;

public class CollectionBMI
{
```

```
public static void main(String[] args)
{
    ArrayList<Student> studentlist = new ArrayList<Student>();//定义 Student 类数组列表，存放学生信
息
    int i = 100;//记录用户选择想执行功能序号的索引
    int option;//记录用户对升序和降序的索引
    Student student = new Student();
    do
    {
        i = menu();// 获取用户想执行功能
        if (i>=3&&studentlist.isEmpty())// 判断是否是非法操作
        {
            System.out.println("非法操作，请输入学生信息之后再进行此项操作");
            i = 100;//重置 i
        }
        switch (i)
        {
        case 1:
            // 增加学生信息
            addStudents(studentlist);
            break;
        case 2:
            // 自动生成学生信息
            genStudents(studentlist, 10);
            break;
        case 3:
            // 打印学生信息
            printStatics(studentlist);
            break;
        case 4:
            // 按学号大小对数组列表进行排序
            option = AscOrDsc();//记录用户选择升序还是降序
            //调用 Collection.sort()，传进一个重写的比较器，实现对特定顺序的排序
            if (option == 1)
            {
                Collections.sort(studentlist, student.new SortByIDAsc());//升序
                printStatics(studentlist);// 打印学生信息
            }
            else if (option == 2)
            {
                Collections.sort(studentlist, student.new SortByIDDsc());//降序
                printStatics(studentlist);// 打印学生信息
            }
            break;
```

```
case 5:
    // 按姓名大小对数组列表进行排序
    option = AscOrDsc();//记录用户选择升序还是降序
    //调用 Collection.sort()，传进一个重写的比较器，实现对特定顺序的排序
    if (option == 1)
    {
        Collections.sort(studentlist, student.new SortByNameAsc());//升序
        printStatics(studentlist);// 打印学生信息
    }
    else if (option == 2)
    {
        Collections.sort(studentlist, student.new SortByNameDsc());//降序
        printStatics(studentlist);// 打印学生信息
    }
    break;
case 6:
    // 按身高大小对数组列表进行排序
    option = AscOrDsc();//记录用户选择升序还是降序
    //调用 Collection.sort()，传进一个重写的比较器，实现对特定顺序的排序
    if (option == 1)
    {
        Collections.sort(studentlist, student.new SortByHeightAsc());//升序
        printStatics(studentlist);// 打印学生信息
    }
    else if (option == 2)
    {
        Collections.sort(studentlist, student.new SortByHeightDsc());//降序
        printStatics(studentlist);// 打印学生信息
    }
    break;
case 7:
    // 按体重大小对数组列表进行排序
    option = AscOrDsc();//记录用户选择升序还是降序
    //调用 Collection.sort()，传进一个重写的比较器，实现对特定顺序的排序
    if (option == 1)
    {
        Collections.sort(studentlist, student.new SortByWeightAsc());//升序
        printStatics(studentlist);// 打印学生信息
    }
    else if (option == 2)
    {
        Collections.sort(studentlist, student.new SortByWeightDsc());//降序
        printStatics(studentlist);// 打印学生信息
    }
```

```
                    break;
                case 8:
                    // 按 BMI 大小对数组列表进行排序
                    option = AscOrDsc();//记录用户选择升序还是降序
                    //调用 Collection.sort()，传进一个重写的比较器，实现对特定顺序的排序
                    if (option == 1)
                    {
                        Collections.sort(studentlist, student.new SortByBMIAsc());//升序
                        printStatics(studentlist);// 打印学生信息
                    }
                    else if (option == 2)
                    {
                        Collections.sort(studentlist, student.new SortByBMIDsc());//降序
                        printStatics(studentlist);// 打印学生信息
                    }
                    break;
                case 9:
                    // 计算学生 BMI 的平均值,并打印
                    System.out.println("学生 BMI 的平均值为:" + String.format("%.2f",
calculateAverageOfBMI(studentlist)));
                    // 计算学生 BMI 的中位数,并打印
                    System.out.println("学生 BMI 的中位数为:" + String.format("%.2f",
calculateMedian(studentlist)));
                    // 计算学生 BMI 的众数
                    findMode(studentlist);
                    // 计算学生 BMI 的方差,并打印
                    System.out.println("学生的 BMI 的方差为:" + String.format("%.2f",
calculateVariance(studentlist)));
                    break;
                case 10:
                    // 程序结束，退出
                    System.out.println("It is the end!");
                    System.exit(0);
                    break;
                default:
                    break;
                }
            } while (true);
        }

        /**
         * 通过键盘输入，手动增加学生信息*/
        public static void addStudents(ArrayList<Student> studentlist)
        {
```

```
        Scanner scanner = new Scanner(System.in);
        String id = "";
        String name;
        float height;
        float weight;
        do
        {
            do
            {
                System.out.println("退出请在学号处输入 exit");//给出退出学生信息输入的条件
                System.out.print("请输入学生学号:");
                id = scanner.next();
                if(id.toUpperCase().equals("EXIT"))//判断是否是退出输出
                    return;
                if(id!="" && isExists(id, studentlist))//判断输入的学号是否是已经存在
                {
                    System.out.println("该学号已经存在，请重新输入");
                    continue;
                }
            } while (id==""||id==null);//判断用户输入的学号是否是合法

            System.out.print("请输入学生姓名:");
            name = UpperFirstLetter(scanner.next());
            System.out.print("请输入学生身高(单位 m):");
            height = scanner.nextFloat();
            System.out.print("请输入学生体重(单位 kg):");
            weight = scanner.nextFloat();
            //根据输入的信息生成一个新的 Student 对象实例，并增加到数组列表的末尾
            studentlist.add(new Student(id,name,height,weight));
        }while(true);

}

/**打印生徐还是降序的选择菜单，并返回用户的选择*/
public static int AscOrDsc()
{
        Scanner scanner = new Scanner(System.in);
        System.out.println("1.升序");
        System.out.println("2.降序");
        System.out.print("请选择:");
        return scanner.nextInt();
}

/**
```

```
 * 随机生成指定数量的学生信息
 */
public static void genStudents(ArrayList<Student> studentlist, int num)
{
	String id;// 学生 ID
	String name;// 学生姓名
	float height;// 学生身高
	float weight;// 学生体重
	for (int i = 0; i < num; i++)
	{
		do
		{
			id = RandomGenerateID();// 生成学生 ID
		} while (isExists(id, studentlist));// 判断学生的 ID 时候已经存在

		name = RandomGenerateName();// 随机生成学生名字
		height = nextFloat(1.5f, 2.0f);// 生成指定范围 1.5~2.0 的浮点数
		weight = nextFloat(45f, 100f);// 生成指定范围 455~100 的浮点数
		//根据生成的的信息生成一个新的 Student 对象实例，并增加到数组列表的末尾
		studentlist.add(new Student(id, name, height, weight));
	}
}

/**
 * 打印菜单，并且获取用户输入 返回这个输入所对应的功能的索引
 */
public static int menu()
{
	String num = null;// 获取用户想执行功能
	int i = 8;// 获取用户想执行功能
	boolean flag = true;// 错误输入标志;
	Scanner scanner = new Scanner(System.in);
	do
	{
		if (flag)// 只打印一次菜单，避免重复
		{
			printMenuMessage();
		}
		num = scanner.nextLine();
		// 检查非法输入，如果用户输入非数字，这再次输入
		try
		{
			i = Integer.parseInt(num);
			flag = true;
```

```
            } catch (Exception e)
            {
                flag = false;
            }
            if (!flag)
            {
                System.out.println("输入有误，请重新输入:");
            }
        } while (!flag || i > 10 || i < 0);
        return i;
}

/**
 * 打印菜单信息
 */
public static void printMenuMessage()
{
        System.out.println("本程序功能：");
        System.out.println("1.手动输入学生信息");
        System.out.println("2.随机生成学生信息");
        System.out.println("3.打印学生信息");
        System.out.println("4.按学号进行排序");
        System.out.println("5.按姓名进行排序");
        System.out.println("6.按身高进行排序");
        System.out.println("7.按体重进行排序");
        System.out.println("8.按 BMI 进行排序");
        System.out.println("9.查看学生 BMI 统计信息");
        System.out.println("10.退出程序");
}

/**
 * 生成一个指定范围 min~max 的浮点数
 */
public static float nextFloat(final float min, final float max)
{
        return min + ((max - min) * new Random().nextFloat());
}

/**
 * 随机生成一个 201001~211001 的学生 ID*/
public static String RandomGenerateID()
{
        Random random = new Random();
        int i = random.nextInt(10000);
```

```
        return (201001+i)+"";
    }
    /**
     * 随机生成学生的名字
     */
    public static String RandomGenerateName()
    {
        Random random = new Random();
        String name = "";
        for (int i = 0; i < 5; i++)// 学生名字长度为 5
        {
            name += (char) (random.nextInt(26) + 'a');// 随机生成一个 0~26 的数，计算其余字母 a 的相对偏移量，得出一个 ASCII 码字符，连接成一个字符串
        }
        return UpperFirstLetter(name);
    }

    /**
     * 将字符串的首字母改成大写，其余的字母改成小写，并返回*/
    public static String UpperFirstLetter(String name)
    {
        name = name.toLowerCase();
        return name.substring(0, 1).toUpperCase() + name.substring(1);// 将字符串的第一个字母变成大写，并且返回
    }
    /**
     * 判断 ID 是否与已经存在，避免重复
     */
    public static boolean isExists(String id, ArrayList<Student> studentlist)
    {
        for (int i = 0; i < studentlist.size(); i++)
        {
            if (studentlist.get(i) != null)// 数据合法性判断
            {
                if (id.equals(studentlist.get(i).getID()))
                {
                    return true;// ID 重复返回 true
                }
            }
        }
        return false;// ID 不重复返回 false
    }

    /**
```

```
     * 按格式打印所有学生的信息*/
    public static void printStatics(ArrayList<Student> studentlist)
    {
        System.out.println("学生信息总览:");
        System.out.println("学号" + "\t" + "姓名" + "\t" + "  身高" + "\t" + "  体重" + "\t" + "BMI 指数" + "\t"
+ "      健康状况");
        for (Student result : studentlist)//用 for—loop 循环，从数组列表 studentlist 里面获取每一个实例化
对象
        {
            System.out.print(result.getID() + "\t");//  打印学号
            System.out.print(result.getName() + "\t");//  打印姓名
            System.out.print(String.format("%.2f", result.getHeight()) + "\t");//  打印身高
            System.out.print(String.format("%.2f", result.getWeight()) + "\t");//  打印体重
            System.out.print(String.format("%.2f", result.getBMI()) + "\t");//  打印 BMI 的值
            System.out.println(result.getPhysicalCondition());//  打印健康状况
        }
    }

    /**  计算所有学生的 BMI 的平均值  */
    public static float calculateAverageOfBMI(ArrayList<Student> studentlist)
    {
        float average = 0;
        for (Student result : studentlist)//用 for—loop 循环，从数组列表 studentlist 里面获取每一个实例化
对象
            average += result.getBMI();//获取每一个实例化对象对 BMI 值
        return average / studentlist.size();
    }

    /**
     * 计算所有学生的 BMI 的中位数
     */
    public static float calculateMedian(ArrayList<Student> studentlist)
    {
        Student student = new Student();//创建一个新对象，用与创建比较器对象
        //调用 Collection.sort()，传进一个重写的比较器，实现对 BMI 升序排序
        Collections.sort(studentlist, student.new SortByBMIAsc());
        int lenght = studentlist.size();//记录学生人数
        float Median;
        if (lenght % 2 != 0)//  如果 bmi 数组长度是奇数，即有奇数个学生，则中位数就是数组索引值为
length/2 的值
        {
            Median = studentlist.get(lenght / 2).getBMI();
        } else //  如果 bmi 数组长度是偶数，即有偶数个学生，则中位数就是数组索引值为 length/2 与
length/2+1 的值的平均值
```

```
        {
            Median = (studentlist.get(lenght / 2 + 1).getBMI() + studentlist.get(lenght / 2).getBMI()) / 2;
        }
        return Median;
    }

    /**
     * 用于寻找 student 对象数组里面所有学生各自 BMI 值的众数
     */
    public static void findMode(ArrayList<Student> studentlist)
    {

        Student student = new Student();//创建一个新对象，用与创建比较器对象
        //调用 Collection.sort()，传进一个重写的比较器，实现对 BMI 升序排序
        Collections.sort(studentlist, student.new SortByBMIAsc());
        int length = studentlist.size();// 记录学生的个数，即数组 BMI 的长度
        float[] bmis = new float[length];// 用于获得 student 对象数组列表里面每个学生的 BMI 值
        for (int i = 0; i < length; i++)
        {
            bmis[i] = studentlist.get(i).getBMI();//获取象数组列表里面每个学生的 BMI 值
        }
        float[] num = new float[length];// 用于存储记录不同得 BMIS 值
        int[] count = new int[length];// 用于记录不同的 BMI 在数组中出现的次数
        int time = 0;// 用于计数，没出现一个相同的 BMI 值，自增一次
        int k = 0;// 用于 num 与 count 数组下标的自增

        for (int i = 0; i < length; i++)
        {
            if (bmis[i] > 0)// 判断 bims[i]里面存放的是否是有用数据
            {
                num[k] = bmis[i];// 记录这个数值
                for (int j = 0; j < length; j++)
                {
                    if (bmis[j] > 0 && bmis[j] == num[k])// 判断 bims[j]里面存放的是否是有用数据
且判断是否与 num[k]相等
                    {
                        time++;// 这个数出现次数加一
                        bmis[j] = 0f;// 将这个数置为无用数据，以后不再参与统计
                    }
                }
                count[k] = time;// 记录出现次数
                k++;
                time = 0;// 重置计数器
            }
```

```
        }

        float max = count[0];
        // 找到出现次数最多的 BMI 值,若果 max = 1，
        // BMI 中所有数均只出现一次，每个数都不一样，那么众数不存在
        for (int i = 0; i < count.length; i++)
        {
            if (count[i] > max)
            {
                max = count[i];
            }
        }

        // 找到出现次数最多的数据的下标，并记录下来
        int[] ModeSub = new int[length];
        k = 0;
        for (int i = 0; i < length; i++)
        {
            if (count[i] == max)
            {
                ModeSub[k] = i;
                k++;
            }
        }

        if (max == 1)// BMI 中所有数均只出现一次，每个数都不一样，那么众数不存在
        {
            System.out.println("所有学生的 BMI 值均只出现一次，众数不存在！");
            return;
        }

        System.out.print("\n" + "众数为:");
        for (int i = 0; i < length; i++)
        {
            if (ModeSub[i] != 0)// 过滤非法数据
            {
                System.out.print(String.format("%.2f", num[ModeSub[i]]) + "\t");
            }
        }
    }

    /**
     * 计算所有学生的 BMI 的方差
     */
```

```
    public static float calculateVariance(ArrayList<Student> studentlist)
    {
        float average = 0;// 计算所有学生的 BMI 平均值
        float sum = 0;// 计算总值
        int length = studentlist.size();// 记录学生的个数，即数组 BMI 的长度
        float[] bmis = new float[length];//用于获得 student 对象数组列表里面每个学生的 BMI 值
        for (int i = 0; i < length; i++)
        {
            bmis[i] = studentlist.get(i).getBMI();//获取象数组列表里面每个学生的 BMI 值
            average = bmis[i];//计算每个学生的 BMI 值总和
        }
        average = average / length;//计算出平均值
        for (int i = 0; i < length; i++)
        {
            sum += (bmis[i] - average) * (bmis[i] - average);// 迭代计算出方差
        }
        return sum / length;
    }
}

class Student
{
    private String id;// 学生学号
    private String name;// 学生姓名
    private float height;// 学生身高
    private float weight;// 学生体重
    private float bmi;// 学生 bmi 值
    private String physicalCondition;// 学生健康状况

    public Student()//空构造方法，用于创建对象，调用比较器
    {
    }

    public Student(String id, String name, float height, float weight)// 构造方法，初始化数据
    {
        this.id = id;
        this.name = name;
        this.height = height;
        this.weight = weight;
        bmi = calculateBMI(height, weight);// 计算出根据输入的数据计算出学生 BMI
        physicalCondition = checkHealth(bmi);// 根据 BMI 推算出学生的健康状况
    }

    @Override
```

```
/** 返回学生的个人信息 */
public String toString()
{
	return id + "\t" + name + "\t" + height + "\t" + weight + "\t" + bmi + "\t" + physicalCondition;
}

/** 计算出学生的 bmi */
private float calculateBMI(float height, float weight)
{
	return weight / (height * height);
}

/** 返回私有域 id，即学生的学号 */
public String getID()
{
	return id;
}

/** 返回私有域 BMI */
public float getBMI()
{
	return bmi;
}

/** 返回私有域 height，即学生的身高 */
public float getHeight()
{
	return height;
}

/** 返回私有域 weight，即学生的体重 */
public float getWeight()
{
	return weight;
}

/** 返回私有域 name，即学生的姓名 */
public String getName()
{
	return name;
}

/** 返回私有域 PhysicalCondition，即学生的健康状况 */
public String getPhysicalCondition()
```

```
{
        return physicalCondition;
}

/** 根据学生 bmi 退出其健康状况 */
public String checkHealth(float bmis)
{
        String HealthType = null;// 记录学生的健康类型
        if (bmis < 18.5)
        {
                HealthType = "Underweight";// 体重过轻
        } else if (18.5 <= bmis && bmis < 23)
        {
                HealthType = "Normal Range";// 正常范围
        } else if (23 <= bmis && bmis < 25)
        {
                HealthType = "Overweight-At Risk";// 有肥胖的趋势
        } else if (25 <= bmis && bmis < 30)
        {
                HealthType = "Overweight-Moderately Obese";// 超重
        } else if (30 <= bmis)
        {
                HealthType = "Overweight-Severely Obese";// 严重超重
        } else
        {
        }
        return HealthType;
}

/**
 *Comparator 接口，重写 compare 方法，实现按 ID 值对学生进行升序排序*/
class SortByIDAsc implements Comparator<Student>
{
        public int compare(Student o1, Student o2)
        {
                return o1.getID().compareTo(o2.getID());
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按 ID 值对学生进行降序排序*/
class SortByIDDsc implements Comparator<Student>
{
        public int compare(Student o1, Student o2)
```

```
        {
            return o2.getID().compareTo(o1.getID());
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按姓名值对学生进行升序排序*/
class SortByNameAsc implements Comparator<Student>
{
        public int compare(Student s1, Student s2)
        {
            return s1.getName().compareTo(s2.getName());
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按姓名值对学生进行降序排序*/
class SortByNameDsc implements Comparator<Student>
{
        public int compare(Student s1, Student s2)
        {
            return s2.getName().compareTo(s1.getName());
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按身高值对学生进行升序排序*/
class SortByHeightAsc implements Comparator<Student>
{
        public int compare(Student s1, Student s2)
        {
            if (s1.getHeight() < s2.getHeight())
                return -1;
            return 1;
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按身高值对学生进行降序排序*/
class SortByHeightDsc implements Comparator<Student>
{
        public int compare(Student s1, Student s2)
        {
            if (s2.getHeight() < s1.getHeight())
```

```
                return -1;
            return 1;
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按体重值对学生进行升序排序*/
class SortByWeightAsc implements Comparator<Student>
{
        public int compare(Student s1, Student s2)
        {
            if (s1.getWeight() < s2.getWeight())
                return -1;
            return 1;
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按体重值对学生进行降序排序*/
class SortByWeightDsc implements Comparator<Student>
{
        public int compare(Student s1, Student s2)
        {
            if (s2.getWeight() < s1.getWeight())
                return -1;
            return 1;
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按 BMI 值对学生进行升序排序*/
class SortByBMIAsc implements Comparator<Student>
{
        public int compare(Student s1, Student s2)
        {
            if (s1.getBMI() < s2.getBMI())
                return -1;
            return 1;
        }
}

/**
 *Comparator 接口，重写 compare 方法，实现按 BMI 值对学生进行降序排序*/
class SortByBMIDsc implements Comparator<Student>
```

```
    {
        public int compare(Student s1, Student s2)
        {
            if (s2.getBMI() < s1.getBMI())
                return -1;
            return 1;
        }
    }
}
```

程序运行结果：

菜单：

```
本程序功能：
1.手动输入学生信息
2.随机生成学生信息
3.打印学生信息
4.按学号进行排序
5.按姓名进行排序
6.按身高进行排序
7.按体重进行排序
8.按BMI进行排序
9.查看学生BMI统计信息
10.退出程序
```

功能一：手动增加学生信息

```
1
退出请在学号处输入exit
请输入学生学号:201001
请输入学生姓名:Jack
请输入学生身高(单位m):1.78
请输入学生体重(单位kg):72
退出请在学号处输入exit
请输入学生学号:exit
```

功能二：随机生成 10 名学生信息

```
学生信息总览：
学号      姓名      身高      体重      BMI指数     健康状况
201001   Jack     1.78     72.00    22.72    Normal Range
207786   Fzwof    1.96     80.79    21.08    Normal Range
208249   Tqqzz    1.69     77.63    27.33    Overweight-Moderately Obese
204328   Iaxfa    1.86     86.61    24.97    Overweight-At Risk
210562   Wwlsa    1.99     95.78    24.16    Overweight-At Risk
204772   Ovhcp    1.56     98.66    40.62    Overweight-Severely Obese
204828   Nduzj    1.61     51.86    20.01    Normal Range
205899   Nmluy    1.51     88.05    38.47    Overweight-Severely Obese
208248   Ajvpn    1.50     57.72    25.52    Overweight-Moderately Obese
210726   Fypnk    1.64     70.15    26.01    Overweight-Moderately Obese
209127   Hiauo    1.60     62.56    24.49    Overweight-At Risk
```

功能三：打印学生信息

```
学生信息总览:
学号      姓名      身高      体重      BMI指数     健康状况
201001   Jack     1.78     72.00    22.72    Normal Range
207786   Fzwof    1.96     80.79    21.08    Normal Range
208249   Tqqzz    1.69     77.63    27.33    Overweight-Moderately Obese
204328   Iaxfa    1.86     86.61    24.97    Overweight-At Risk
210562   Wwlsa    1.99     95.78    24.16    Overweight-At Risk
204772   Ovhcp    1.56     98.66    40.62    Overweight-Severely Obese
204828   Nduzj    1.61     51.86    20.01    Normal Range
205899   Nmluy    1.51     88.05    38.47    Overweight-Severely Obese
208248   Ajvpn    1.50     57.72    25.52    Overweight-Moderately Obese
210726   Fypnk    1.64     70.15    26.01    Overweight-Moderately Obese
209127   Hiauo    1.60     62.56    24.49    Overweight-At Risk
```

功能四：按学号进行排序

升序

```
4
1.升序
2.降序
请选择:1
学生信息总览:
学号      姓名      身高      体重      BMI指数     健康状况
201001   Jack     1.78     72.00    22.72    Normal Range
204328   Iaxfa    1.86     86.61    24.97    Overweight-At Risk
204772   Ovhcp    1.56     98.66    40.62    Overweight-Severely Obese
204828   Nduzj    1.61     51.86    20.01    Normal Range
205899   Nmluy    1.51     88.05    38.47    Overweight-Severely Obese
207786   Fzwof    1.96     80.79    21.08    Normal Range
208248   Ajvpn    1.50     57.72    25.52    Overweight-Moderately Obese
208249   Tqqzz    1.69     77.63    27.33    Overweight-Moderately Obese
209127   Hiauo    1.60     62.56    24.49    Overweight-At Risk
210562   Wwlsa    1.99     95.78    24.16    Overweight-At Risk
210726   Fypnk    1.64     70.15    26.01    Overweight-Moderately Obese
```

降序

```
4
1.升序
2.降序
请选择:2
学生信息总览:
学号      姓名      身高      体重      BMI指数     健康状况
210726   Fypnk    1.64     70.15    26.01    Overweight-Moderately Obese
210562   Wwlsa    1.99     95.78    24.16    Overweight-At Risk
209127   Hiauo    1.60     62.56    24.49    Overweight-At Risk
208249   Tqqzz    1.69     77.63    27.33    Overweight-Moderately Obese
208248   Ajvpn    1.50     57.72    25.52    Overweight-Moderately Obese
207786   Fzwof    1.96     80.79    21.08    Normal Range
205899   Nmluy    1.51     88.05    38.47    Overweight-Severely Obese
204828   Nduzj    1.61     51.86    20.01    Normal Range
204772   Ovhcp    1.56     98.66    40.62    Overweight-Severely Obese
204328   Iaxfa    1.86     86.61    24.97    Overweight-At Risk
201001   Jack     1.78     72.00    22.72    Normal Range
```

功能五：按姓名大小进行排序

升序：

```
5
1.升序
2.降序
请选择:1
学生信息总览:
学号      姓名      身高      体重      BMI指数     健康状况
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
207786  Fzwof   1.96    80.79   21.08   Normal Range
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
201001  Jack    1.78    72.00   22.72   Normal Range
204828  Nduzj   1.61    51.86   20.01   Normal Range
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
```

降序：

```
5
1.升序
2.降序
请选择:2
学生信息总览:
学号      姓名      身高      体重      BMI指数     健康状况
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
204828  Nduzj   1.61    51.86   20.01   Normal Range
201001  Jack    1.78    72.00   22.72   Normal Range
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
207786  Fzwof   1.96    80.79   21.08   Normal Range
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
```

功能六：按身高大小进行排序

升序：

```
6
1.升序
2.降序
请选择:1
学生信息总览:
学号      姓名      身高      体重      BMI指数     健康状况
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
204828  Nduzj   1.61    51.86   20.01   Normal Range
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
201001  Jack    1.78    72.00   22.72   Normal Range
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
207786  Fzwof   1.96    80.79   21.08   Normal Range
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
```

降序：

```
6
1.升序
2.降序
请选择:2
学生信息总览:
学号      姓名      身高      体重      BMI指数     健康状况
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
207786  Fzwof   1.96    80.79   21.08   Normal Range
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
201001  Jack    1.78    72.00   22.72   Normal Range
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
204828  Nduzj   1.61    51.86   20.01   Normal Range
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
```

功能七：按体重大小进行排序

升序：

```
7
1.升序
2.降序
请选择:1
学生信息总览:
学号      姓名      身高      体重      BMI指数    健康状况
204828  Nduzj   1.61    51.86   20.01   Normal Range
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
201001  Jack    1.78    72.00   22.72   Normal Range
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
207786  Fzwof   1.96    80.79   21.08   Normal Range
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
```

降序:

```
7
1.升序
2.降序
请选择:2
学生信息总览:
学号      姓名      身高      体重      BMI指数    健康状况
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
207786  Fzwof   1.96    80.79   21.08   Normal Range
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
201001  Jack    1.78    72.00   22.72   Normal Range
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
204828  Nduzj   1.61    51.86   20.01   Normal Range
```

功能八：按 BMI 大小进行排序

升序:

```
8
1.升序
2.降序
请选择:1
学生信息总览:
学号      姓名      身高      体重      BMI指数    健康状况
204828  Nduzj   1.61    51.86   20.01   Normal Range
207786  Fzwof   1.96    80.79   21.08   Normal Range
201001  Jack    1.78    72.00   22.72   Normal Range
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
```

降序：

```
8
1.升序
2.降序
请选择:2
学生信息总览:
学号      姓名      身高      体重      BMI指数    健康状况
204772  Ovhcp   1.56    98.66   40.62   Overweight-Severely Obese
205899  Nmluy   1.51    88.05   38.47   Overweight-Severely Obese
208249  Tqqzz   1.69    77.63   27.33   Overweight-Moderately Obese
210726  Fypnk   1.64    70.15   26.01   Overweight-Moderately Obese
208248  Ajvpn   1.50    57.72   25.52   Overweight-Moderately Obese
204328  Iaxfa   1.86    86.61   24.97   Overweight-At Risk
209127  Hiauo   1.60    62.56   24.49   Overweight-At Risk
210562  Wwlsa   1.99    95.78   24.16   Overweight-At Risk
201001  Jack    1.78    72.00   22.72   Normal Range
207786  Fzwof   1.96    80.79   21.08   Normal Range
204828  Nduzj   1.61    51.86   20.01   Normal Range
```

功能九：查看学生 BMI 统计信息

```
9
学生BMI的平均值为:26.85
学生BMI的中位数为:24.97
所有学生的BMI值均只出现一次，众数不存在!
学生的BMI的方差为:576.47
```

功能十：退出程序

```
10
It is the end!
```