Java 程序设计实验报告

学号：1170300709

姓名：杨开来

专业：计算机类

班级：1703007

哈尔滨工业大学

实验四：面向对象程序设计

一、实验目的

1）掌握面向对象的基本概念（成员变量、成员函数等）
2）掌握类的定义、内部类的定义
3）掌握对象的声明
4）掌握对象数组的使用
5）基本算法的设计

二、实验内容

1）编写 OOBMI 类文件；并在该类文件中定义另一个类 Student，该类包含学号、姓名、身高、体重和 bmi 等属性，为 Student 类定义创建函数。

2）在 Student 类中，增加 public String toString()函数，该函数可以返回一个字符串，该字符串包含学号、姓名、身高、体重、bmi 值和胖瘦健康状况，他们之间用制表符(\t)隔开；

3）在 OOBMI 中增加成员属性 Student[] students；在 OOBMI 中增加 genStudents 函数，参数为整数，能够随机生成指定数量的名学生对象，并保存到 students 数组中。（注意，学号、姓名、身高、体重等均需随机生成，数值均需保留两位小数存储）

4）在 OOBMI 类中增加 public boolean isExists(String id) 函数，判断该学生是否已经在 students 数组中，函数返回值为 boolean 类型，如果已经存在，返回 false；否则，返回 true。并在 genStudents 函数，调用 isExists 函数避免输入或生成重复的学号的学生。

5）在 OOBMI 中增加 4 个函数，分别统计 bmi 的均值、中值、众数、方差等统计信息。

6）在 OOBMI 中增加 printStatics 函数，该函数首先打印所有学生基本信息，然后打印 bmi 的均值、中值、众数、方差等统计结果信息。

7）增加 menu 函数提供随机生成学生、打印学生、5 种排序、打印统计信息、退出执行等 9 个菜单功能，用户输入指定选项后，运行相应函数功能。

8）在 OOBMI 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、实验代码

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！

1. 试验源码

```
package oobmi;
import java.util.Random;//import Random method
import java.util.Scanner;//import Scanner method
import java.math.BigDecimal;//import Bigdecimal method
/**
 *
 * @author KG
```

```
 */
/*
    定义另一个类 Student，该类包含学号、姓名、身高、体重和 bmi 等属性，为 Student 类定义创建函数。
*/
class Student {
        String id;
        String name;
        float height;
        float weight;
        float bmi;
        String healthSituation;
        /*
        定义创建方法 Student（）
        */
        public Student(String id,String name,float weight,float height){
            this.id = id;
            this.name = name;
            this.height = height;
            this.weight = weight;
            this.bmi = weight/(height*height);

        }
        //定义方法 toString，用于返回产生的学生基本情况。在 Student 类中，该函数可以返
        //回一个字符串，该字符串包含学号、姓名、身高、体重、bmi 值和胖瘦健康状况，他
        //们之间用制表符(\t)隔开；
        public String toString(){
            String re = this.id+"\t"+this.name+"\t"+this.height+"\t"+
                    this.weight+"\t"+this.bmi+"\t"+this.healthSituation;
            return re;
        }
}
public class OOBMI {
    /*
    定义成员变量类 Studnet[] students，要产生的学生人数 num，是否继续的指示 isGoon，排序数组 sortedIndex。
    */
    static Student[] students = new Student[100];
    static int num;
    static boolean isGoon;
    static int[] sortedIndex;
    /*
```

在 OOBMI 中增加 genStudents 函数，参数为整数，能够随机生成指定数量的名学生对象，

并保存到 students 数组中。学号、姓名、身高、体重等均需随机生成，数值均保留两位小数

```
     */
    private static void genStudent(int num){

     for (int i = 0; i < num; i++) {
                Random random = new Random();//新建 random 对象
                String id = String.valueOf(random.nextInt(1000)+1000);

                String name = genRandomString(random);
                float x = (float)random.nextGaussian();
          /*对身高体重和 bmi 进行保留两位小数操作。*/
                float weight1 = (float)(60.0+x*5);
                BigDecimal decimal=new BigDecimal(weight1);
                float weight=(float)decimal.setScale
                        (2,BigDecimal.ROUND_HALF_UP).doubleValue();
                float height1 = (float)(1.75 + x*0.1);
                BigDecimal decimal1=new BigDecimal(height1);
                float height=(float)decimal1.setScale
                        (2,BigDecimal.ROUND_HALF_UP).doubleValue();

                if (!isExists(id,students,i)) {//判断该 id 是否已经存在。
                    i--;
                } else {
                    students[i] = new Student(id, name, weight, height);
                    students[i].healthSituation = checkHealth(students[i].bmi);//
由 checkHealth 函数返回学生的健康状况。
                    BigDecimal decimal2=new BigDecimal(students[i].bmi);//保留
为两位小数
                    students[i].bmi = (float)decimal2.setScale
                        (2,BigDecimal.ROUND_HALF_UP).doubleValue();
                }
            }

     }
    /*
    随机产生一个六位的字符串用于学生姓名。
    */
     public static String genRandomString(Random r){
          String base = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
          StringBuilder sb = new StringBuilder();
          for (int i = 0; i < 6; i++) {
```

```
            int number = r.nextInt(base.length());
            sb.append(base.charAt(number));
        }
        return sb.toString();
    }
    /*
    此函数用于判断随机产生的字符串是否已经存在，若存在则返回 false，不存在则返回 true。
    并在 genStudents 函数，调用 isExists 函数避免输入或生成重复的学号的学生。
    */
    private static boolean isExists(String id,Student[] students,int num) {
        for (int i=0;i<num;i++) {
                if (id.equals(students[i].id))
                    return false;
        }
        return true;
    }
    /*此函数用于计算学生的 BMI 值的平均数。*/
    private static float calAverage(int num,Student[] students){
        float sum = 0;
        for (int i=0;i<num;i++){
            sum = sum + students[i].bmi;
        }
        return sum/num;
    }
    /*
    此函数用于计算学生的 BMI 值的中位数。
    */
    private static float calMiddle(int num,Student[] students){
        float[] sortednum = new float[num];
        for (int i=0;i<num;i++){
            sortednum[i] = students[i].bmi;
        }
        /*
        使用冒泡法排序从而选出中位数。
        */
        for (int i=0;i<num-1;i++) {
            for (int j=0;j<num-i-1;j++) {
                if (sortednum[j]>sortednum[j+1]) {
                    float temp = sortednum[j];
                    sortednum[j] = sortednum[j+1];
                    sortednum[j+1] = temp;
                }
```

```
            }
        }
        /*
        此处用于判断返回一个已经存在的数还是计算平均数。
        */
        if (num%2==0){
           return (sortednum[num/2]+sortednum[num/2-1])/2;
        }
        else if (num%2!=0){
           return sortednum[(num-1)/2];
        }
        else{return -1;}
    }
    /*
    此函数用于计算学生 BMI 值的众数。
    */
    private static float calPublic(int num,Student[] students){
        float[][] bmis = new float[num][2];//使用一个数组存储每个数及其出现
频率
        for (int i=0;i<num;i++){
            bmis[i][0] = students[i].bmi;//初始化数组
            bmis[i][1] = 1;
        }
        /*
        判断有多少相同的数，并将其进行合并
        */
        for (int i=0;i<num-1;i++){
            if (bmis[i][1]==0){break;}
            for (int j=i+1;j<num;j++){
                if (bmis[i][0]==bmis[j][0]&&bmis[j][1]!=0){
                    bmis[i][1]+=bmis[i][1]+bmis[j][1];
                    bmis[j][1] = 0;
                }
            }
        }
        int max = 0;//max 用于存储最多数的下标
        for (int i=1;i<num;i++){
            if (bmis[i][1]>bmis[max][1]){
                max = i;
            }
        }
        for (int i=0;i<num;i++){
            if (bmis[i][1]==bmis[max][1]&&max!=i){
                return -1;//如果存在多个出现频率相同的数，则返回-1.
```

```
                }
            }
            return bmis[max][0];
        }
        /*
        此函数用于计算各数 BMI 值的方差。
        */
        private static float calVar(int num,Student[] students) {
            float sum = 0;
            for (int i=0;i<num;i++) {
                sum +=
(students[i].bmi-calAverage(num,students))*(students[i].bmi-calAverage(num,st
udents));//用于计算方差
            }
            return sum/num;
        }
        /*
        此函数用于打印各种排序后的学生基础信息。
        */
        private static void printStatics(int num,Student[] students,int[] sortedIndex)
{
            for (int i=0;i<num;i++) {
               System.out.println(students[sortedIndex[i]].toString());
               //按 sortedIndex 的顺序打印，并通过调用 toString 方法返回这些数
据。
            }
        }
        /*
        用于打印学生 BMI 值的均值，中位数，众数，方差值，并格式化输出。
        */
        private static void printSta(int num,Student[] students){
            System.out.printf("Average:%.2f\n",calAverage(num,students));
            System.out.printf("Middle:%.2f\n",calMiddle(num,students));
            if (calPublic(num,students)!=-1){
                System.out.printf("Public:%.2f\n",calPublic(num,students));
            }
            else {System.out.println("There are no public numbers.");}
            System.out.printf("Variance:%.2f\n",calVar(num,students));
        }
        /*
        增加 menu 函数提供随机生成学生、打印学生、5 种排序、打印统计信息、
退出执行等
        9 个菜单功能，用户输入指定选项后，运行相应函数功能
        */
```

```
    private static void menu(int[] sortedIndex) {
        System.out.println("Please choose the fuction you want to achieve.");
        System.out.println("1.Randomly generate the students.");
        System.out.println("2.Print the students' information.");
        System.out.println("3.Print the students by ID order.");
        System.out.println("4.Print the students by name order.");
        System.out.println("5.Print the students by height order.");
        System.out.println("6.Print the students by weight order.");
        System.out.println("7.Print the students by BMIS order.");
        System.out.println("8.Print the statistic numbers.");
        System.out.println("9.Leave.");
        Scanner in = new Scanner(System.in);
        int choice = in.nextInt();
        switch (choice) {
          case 1:
              System.out.println("How many students do you want to
generate?");
              num = in.nextInt();//接受想要随机产生的学生人数。
              for (int i=0;i<num;i++){
                  sortedIndex[i] = i;
              }
              genStudent(num); break;
          case 2: printStatics(num,students,sortedIndex); break;
          case 3: printStatics(num,students,sortIds(num,students,sortedIndex));
break;
          case 4:
printStatics(num,students,sortNames(num,students,sortedIndex)); break;
          case 5:
printStatics(num,students,sortHeights(num,students,sortedIndex)); break;
          case 6:
printStatics(num,students,sortWeights(num,students,sortedIndex)); break;
          case 7:
printStatics(num,students,sortBmis(num,students,sortedIndex)); break;
          case 8: printSta(num,students);break;
          case 9: System.out.println("Goodbye!"); isGoon = false;break;//判断
是否继续进行这个程序。
          default: System.out.println("Invalid input!"); break;
    }
    }
    /*
    此函数用于根据学生的 BMI 状况返回学生的健康状况。
    */
    public static String checkHealth(float BMI){
        //用一个多值判断来确定返回情况。
```

```
        if (BMI<18.5&&BMI>0) {
            return "Underweight.";
        }
        else if (BMI<23){
            return "Normal range.";
        }
        else if (BMI<25) {
            return "Overweight-- At risk";
        }
        else if (BMI<30){
            return "Overweight--Moderately Obease";
        }
        else if (BMI>=30){
            return "Overweight--Severely Obease";
        }
        else {return "Invalid input.";}
    }

     /*
    函数 sortIds 用于按照 ID 名进行由小到大排序，其中排序方法用冒泡法排序。
    为了方便实现上述功能,定义一个排序数组 int sortedIndex[]，该数组中保
    存了进行排序的数组排序后的下标，排序结束后，返回该数组，以便根据该数
    据进行打印显示。
    */
    private static int[] sortIds(int num,Student[] students,int[] sortedIndex) {
        for (int i=0;i<num-1;i++) {
            for (int j=0;j<num-i-1;j++) {
                if (students[sortedIndex[j]].id.compareTo(students[sortedIndex[j+1]].id)>0) {
                    int temp = sortedIndex[j];
                    sortedIndex[j] = sortedIndex[j+1];
                    sortedIndex[j+1] = temp;
                }
            }
        }
        return sortedIndex;
    }
    /*
    函数 sortNames 用于按照学生名字进行由小到大排序，其中排序方法用冒泡法排序。
    为了方便实现上述功能,定义一个排序数组 int sortedIndex[]，该数组中保
```

存了进行排序的数组排序后的下标，排序结束后，返回该数组，以便根据该数
据进行打印显示。
*/
```
private static int[] sortNames(int num,Student[] students,int[] sortedIndex) {
    for (int i=0;i<num-1;i++) {
        for (int j=0;j<num-i-1;j++) {
            if (students[sortedIndex[j]].name.compareTo(students[sortedIndex[j+1]].name)>0) {
                int temp = sortedIndex[j];
                sortedIndex[j] = sortedIndex[j+1];
                sortedIndex[j+1] = temp;
            }
        }
    }
    return sortedIndex;
}
```
/*
函数 sortHeights 用于按照身高进行由小到大排序，其中排序方法用冒泡法排序。
为了方便实现上述功能,定义一个排序数组 int sortedIndex[]，该数组中保
存了进行排序的数组排序后的下标，排序结束后，返回该数组，以便根据该数
据进行打印显示。
*/
```
private static int[] sortHeights(int num,Student[] students,int[] sortedIndex) {
    for (int i=0;i<num-1;i++) {
        for (int j=0;j<-i-1;j++) {
            if (students[sortedIndex[j]].height>students[sortedIndex[j+1]].height) {
                int temp = sortedIndex[j];
                sortedIndex[j] = sortedIndex[j+1];
                sortedIndex[j+1] = temp;
            }
        }
    }
    return sortedIndex;
}
```
/*
函数 sortWeights 用于按照学生体重进行由小到大排序，其中排序方法用冒泡法排序。
为了方便实现上述功能,定义一个排序数组 int sortedIndex[]，该数组中保

```
    存了进行排序的数组排序后的下标，排序结束后，返回该数组，以便根据该数
    据进行打印显示。
    */
    private static int[] sortWeights(int num,Student[] students,int[] sortedIndex)
{
        for (int i=0;i<num;i++) {
            for (int j=0;j<num-i-1;j++) {
                if
(students[sortedIndex[j]].weight>students[sortedIndex[j+1]].weight) {
                    int temp = sortedIndex[j];
                    sortedIndex[j] = sortedIndex[j+1];
                    sortedIndex[j+1] = temp;
                }
            }
        }
        return sortedIndex;
}
    /*
    函数 sortBmis 用于按照学生 BMI 值进行由小到大排序，其中排序方法用冒泡法排序。
    为了方便实现上述功能,定义一个排序数组 int sortedIndex[]，该数组中保
    存了进行排序的数组排序后的下标，排序结束后，返回该数组，以便根据该数
    据进行打印显示。
    */
    private static int[] sortBmis(int num,Student[] students,int[] sortedIndex) {
        for (int i=0;i<num-1;i++) {
            for (int j=0;j<num-i-1;j++) {
                if
(students[sortedIndex[j]].bmi>students[sortedIndex[j+1]].bmi) {
                    int temp = sortedIndex[j];
                    sortedIndex[j] = sortedIndex[j+1];
                    sortedIndex[j+1] = temp;
                }
            }
        }
        return sortedIndex;
    }
    public static void main(String[] args) {
        isGoon = true;//初始化 isGoon
        int[] sortedIndex = new int[100];//初始化排序数组。
        while (isGoon){
            menu(sortedIndex);//调用菜单函数。
```

```
        }
    }
}
```

2. 运行结果截图

菜单

```
Please choose the fuction you want to achieve.
1.Randomly generate the students.
2.Print the students' information.
3.Print the students by ID order.
4.Print the students by name order.
5.Print the students by height order.
6.Print the students by weight order.
7.Print the students by BMIS order.
8.Print the statistic numbers.
9.Leave.
```

按学号排序

```
1009    PTYCSR  1.72    58.66   19.83   Normal range.
1551    PLPKRZ  1.62    53.47   20.37   Normal range.
1630    CEOQEO  1.79    62.09   19.38   Normal range.
```

按随机产生顺序排序

```
1630    CEOQEO  1.79    62.09   19.38   Normal range.
1009    PTYCSR  1.72    58.66   19.83   Normal range.
1551    PLPKRZ  1.62    53.47   20.37   Normal range.
```

按姓名排序

```
4
1630    CEOQEO  1.79    62.09   19.38   Normal range.
1551    PLPKRZ  1.62    53.47   20.37   Normal range.
1009    PTYCSR  1.72    58.66   19.83   Normal range.
```

按姓名排序

```
4
1630    CEOQEO  1.79    62.09   19.38   Normal range.
1551    PLPKRZ  1.62    53.47   20.37   Normal range.
1009    PTYCSR  1.72    58.66   19.83   Normal range.
```

按身高排序

```
5
1680    RDUFCZ  1.69    56.91   19.93   Normal range.
1135    VDDERI  1.71    58.12   19.88   Normal range.
1227    APMSDL  1.82    63.6    19.2    Normal range.
```

按体重排序

```
6
1551    PLPKRZ  1.62    53.47   20.37   Normal range.
1009    PTYCSR  1.72    58.66   19.83   Normal range.
1630    CEOQEO  1.79    62.09   19.38   Normal range.
```

按 BMI 排序

```
1630    CEOQEO  1.79    62.09   19.38   Normal range.
1009    PTYCSR  1.72    58.66   19.83   Normal range.
1551    PLPKRZ  1.62    53.47   20.37   Normal range.
```

输出平均数，中位数，众数，方差

```
8
Average:19.67
Middle:19.88
There are no public numbers.
Variance:0.11
```

再见

```
9
Goodbye!
```

无效输入

```
10
Invalid input!
```