# Java 程序设计实验报告

学号：L170300302

姓名：金英雄

专业：计算机科学与技术

班级：L170300302

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！**

```
package edu.hit.java.exp2.HitL170300302;


import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
```

```java
import java.util.Random;
import java.util.Scanner;

public class OOBMI {
    Student[] students = new Student[100];
    int n;

    public void genStudents(int n) {
        for (int i = 0; i < n; i++) {
            String id = createData(10);// -生成 ID
            for (int j = 0; j < i; j++) {
                if (id.equals(students[j].ids))
                    id = createData(10);
            }
            String name = getStringRandom(10);
            Random rand = new Random();
            float height = (float) ((rand.nextInt(50) + 150) * 0.01);
            float weight = (float) ((rand.nextInt(50) + 50));
            Student stu = new Student(id, name, height, weight);
            students[i] = stu;
        }
        this.n = n;
    }

    public static String createData(int length) {
        StringBuilder sb = new StringBuilder();
        Random rand = new Random();
        for (int i = 0; i < length; i++) {
            sb.append(rand.nextInt(10));
        }
        String data = sb.toString();
        return data;
    }

    public String getStringRandom(int length) {

        String val = "";
        Random random = new Random();

        // 随机
        for (int i = 0; i < length; i++) {

            int temp = random.nextInt(2) % 2 == 0 ? 65 : 97;
            val += (char) (random.nextInt(26) + temp);
```

```
        }
        return val;
    }

    public boolean isExists(String id) {
        for (int i = 0; i < n; i++) {
            if (students[i].ids.equals(id))
                return true;
        }
        return false;
    }

    public float getMedian() {
        List<Float> bims = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            bims.add(students[i].bmi);
        }
        float mid = 0;
        Collections.sort(bims);
        if (n % 2 == 0)
            mid = (bims.get(n / 2) + bims.get(n / 2 + 1)) / 2;
        else
            mid = bims.get(n / 2);
        return (float) (Math.round(mid * 100) * 0.01);
    }

    public float getAverage() {
        float sum = 0;
        for (int i = 0; i < n; i++) {
            sum = sum + students[i].bmi;
        }
        return (float) (Math.round(sum / n * 100) * 0.01);
    }

    public float getMode() {
        List<Float> bims = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            bims.add(students[i].bmi);
        }
        HashSet<Float> uniqueData = new HashSet<Float>(bims);
        HashMap<Integer, Float> mass = new HashMap<Integer, Float>();
```

```
        int[] count = new int[uniqueData.size()];
        int j = 0;
        for (Float float1 : uniqueData) {
            for (Float float2 : bims) {
                if (float1.equals(float2))
                    count[j]++;
            }
            mass.put(count[j], float1);
            j++;
        }
        int k = 0;
        for (int i : count) {
            k = Math.max(k, i);
        }
        return mass.get(k);
    }

    public float getVariance() {
        float variance = 0;// 方差
        float average = 0;// 均值
        List<Float> bims = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            bims.add(students[i].bmi);
        }
        float sum = 0, sum2 = 0;
        for (int i = 0; i < n; i++) {
            sum += bims.get(i);
        }
        average = sum / n;
        for (int i = 0; i < n; i++) {
            sum2 += (bims.get(i) - average) * (bims.get(i) - average);
        }
        variance = sum2 / n;
        return variance;
    }

    public void printStatics() {
        printStudents();
        System.out.println("Average:" + getAverage() + "Wn" + "Median" + getMedian() + "Wn" + "mode" + getMode() + "Wn"
                + "Variance" + getVariance());

    }
```

```
public void printStudents() {
    for (int i = 0; i < n; i++)
        System.out.println(students[i].toString());

}

public void printStudents(int[] sortedIndex) {
    for (int i = 0; i < n; i++)
        System.out.println(students[sortedIndex[i]].toString());

}

public int[] sortByIds() {
    int sortedIndex[] = new int[100];
    for (int i = 0; i < 100; i++) {
        sortedIndex[i] = i;
    }
    for (int i = 0; i < n; i++) {
        int k = i;
        for (int j = k + 1; j < n; j++) {
            if (students[j].ids.compareTo(students[k].ids) < 0) {
                k = j;
            }
        }
        if (i != k) {
            int temp = sortedIndex[i];
            sortedIndex[i] = sortedIndex[k];
            sortedIndex[k] = temp;
        }
    }
    return sortedIndex;

}

public int[] sortByNames() {
    int sortedIndex[] = new int[100];
    for (int i = 0; i < 100; i++) {
        sortedIndex[i] = i;
    }
    for (int i = 0; i < n; i++) {
        int k = i;
        for (int j = k + 1; j < n; j++) {
            if (students[j].name.compareTo(students[k].name) < 0) {
                k = j;
```

```
            }
        }
        if (i != k) {
            int temp = sortedIndex[i];
            sortedIndex[i] = sortedIndex[k];
            sortedIndex[k] = temp;
        }
    }
    return sortedIndex;

}

public int[] sortByHeights() {
    int sortedIndex[] = new int[100];
    for (int i = 0; i < 100; i++) {
        sortedIndex[i] = i;
    }
    for (int i = 0; i < n; i++) {
        int k = i;
        for (int j = k + 1; j < n; j++) {
            if (students[j].height < students[k].height) {
                k = j;
            }
        }
        if (i != k) {
            int temp = sortedIndex[i];
            sortedIndex[i] = sortedIndex[k];
            sortedIndex[k] = temp;
        }
    }
    return sortedIndex;

}

public int[] sortByWeights() {
    int sortedIndex[] = new int[100];
    for (int i = 0; i < 100; i++) {
        sortedIndex[i] = i;
    }
    for (int i = 0; i < n; i++) {
        int k = i;
        for (int j = k + 1; j < n; j++) {
            if (students[j].weight < students[k].weight) {
                k = j;
```

```
            }
        }
        if (i != k) {
            int temp = sortedIndex[i];
            sortedIndex[i] = sortedIndex[k];
            sortedIndex[k] = temp;
        }
    }
    return sortedIndex;

}

public int[] sortByBmis() {
    int sortedIndex[] = new int[100];

    for (int i = 0; i < 100; i++) {
        sortedIndex[i] = i;
    }
    for (int i = 0; i < n; i++) {
        int k = i;
        for (int j = k + 1; j < n; j++) {
            if (students[j].bmi < students[k].bmi) {
                k = j;
            }
        }
        if (i != k) {
            int temp = sortedIndex[i];
            sortedIndex[i] = sortedIndex[k];
            sortedIndex[k] = temp;
        }
    }
    return sortedIndex;

}

public void menu() {

    Scanner num = new Scanner(System.in);
    int[] index = new int[50];
    int order = 1;
    int n = 0;
    for (int i = 0; i < 50; i++)
        index[i] = i;
    while (order != -1) {
```

```
        System.out.printf("1.Generate studentsWn" + "2.Print Students'InformationWn" + "3.Sort By IdsWn"
                + "4.Sord By NamesWn" + "5.Sord By HeightsWn" + "6.Sord By WeightsWn" + "7.Sord By BmisWn"
                + "8.GetstaticsWn" + "9.ExitWn");
        order = num.nextInt();
        switch (order) {
            case 1:
                System.out.println("Please input n");
                n = num.nextInt();
                genStudents(n);
                break;
            case 2:
                printStudents();
                break;
            case 3:
                System.out.println("Before:");
                printStudents();
                index = sortByIds();
                System.out.println("After:");
                printStudents(index);
                break;
            case 4:
                System.out.println("Before:");
                printStudents();
                index = sortByNames();
                System.out.println("After:");
                printStudents(index);

                break;
            case 5:
                System.out.println("Before:");
                printStudents();
                index = sortByHeights();
                System.out.println("After:");
                printStudents(index);

                break;
            case 6:
                System.out.println("Before:");
                printStudents();
                index = sortByWeights();
                System.out.println("After:");
                printStudents(index);

                break;
```

```
                case 7:
                    System.out.println("Before:");
                    printStudents();
                    index = sortByBmis();
                    System.out.println("After:");
                    printStudents(index);

                    break;
                case 8:
                    printStatics();
                    break;
                case 9:
                    order = -1;

            }
        }

    }

    public static void main(String[] args) {
        OOBMI oo = new OOBMI();

        oo.menu();

    }

}

class Student {
    String ids;
    String name;
    float height;
    float weight;
    float bmi;

    Student(String ids, String name, float height, float weight) {
        this.ids = ids;
        this.name = name;
        this.height = height;
        this.weight = weight;
        this.bmi = (float) (Math.round(weight / (height * height) * 100) * 0.01);
    }

    public String toString() {
```

```
        return ids + "Wt" + name + "Wt" + height + "Wt" + weight + "Wt" + bmi + "Wt" + checkHealth(bmi);
    }


    public static String checkHealth(float bmis) {
        if (bmis < 18.5) {
            return "Underweight";
        } else if (bmis >= 18.5 && bmis < 23) {
            return "Normal Range";
        } else if (bmis >= 23 && bmis < 25) {
            return "Overweight-At Rist";
        } else if (bmis >= 25 && bmis < 30) {
            return "Oberweight-Moderately Obese";
        } else {
            return "Overweight-Severely Obese";
        }


    }
}
```

```
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
1
Please input n
5
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
2
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLI  1.75    51.0    16.65   Underweight
```

```
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
3
Before:
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
After:
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
```

```
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
4
Before:
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
After:
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
```

```
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
5
Before:
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLI  1.75    51.0    16.65   Underweight
After:
5178826958  MqvsZZpFLI  1.75    51.0    16.65   Underweight
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
```

```
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
6
Before:
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
After:
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
```

```
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
7
Before:
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
After:
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
```

```
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
8
1751809626  gyaZcnexQs  1.79    92.0    28.71   Oberweight-Moderately Obese
2134936268  GuoevKGvBC  1.92    90.0    24.41   Overweight-At Rist
3987156360  FxbbiQxFHL  1.89    94.0    26.32   Oberweight-Moderately Obese
2254866192  KoezhJJHSq  1.79    67.0    20.91   Normal Range
5178826958  MqvsZZpFLl  1.75    51.0    16.65   Underweight
Average:23.4
Median24.41
mode26.32
Variance17.901037
1.Generate students
2.Print Students'Information
3.Sort By Ids
4.Sord By Names
5.Sord By Heights
6.Sord By Weights
7.Sord By Bmis
8.Getstatics
9.Exit
9

Process finished with exit code 0
```