实验一：Java 基本程序设计

一、实验目的

1）掌握标准输入输出函数的使用。
2）静态函数的定义和使用（本实验要求所有函数均为静态函数）；
2）掌握简单排序算法；
3）Java 基础语法综合运行（非面向对象版本 BMI 程序）；

二、实验内容

1)编写 BMI 类，在 main 函数中增加数组 String[] ids，String[] names，float[] heights, float[] weights, float[] bmis，分别存储学生们的学号、姓名、身高、体重、计算后的 bmi 值和胖瘦健康状况。注意，上述数值均需保留两位小数存储。

2）定义 inputStudents 函数，该函数的参数为上述数组，该函数的功能是输入多个学生的相关信息，并将相关数据存储到上述数组中；

3）在 BMI 类中，增加一个函数 checkHealth，函数参数为 bmi 值，该函数按下表中 BMI 取值范围判断胖瘦健康状况，该函数的返回值为字符串，返回结果即下表中的第一列中的值，并在 inputStudents 函数中调用该函数，获得学生的胖瘦健康状况。

Category	BMI (kg/m^2)	
	from	to
Underweight		18.5
Normal Range	18.5	23
Overweight—At Risk	23	25
Overweight—Moderately Obese	25	30
Overweight—Severely Obese	30	

4）在 BMI 类中，增加 5 个排序 sortByXXX 函数，XXX 表示排序属性，可以分别按照学生学号、姓名、身高、体重、BMI 进行由小到大排序，排序算法可以利用简单排序、选择排序、冒泡排序算法或其他算法（选择其中一种算法实现即可）。排序前后必须保证同一个学生在所有数组中对应相同的下标！**为了方便实现上述功能,可定义一个排序数组 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。**

5）在 BMI 类中，增加 printStudents 函数，该函数的参数含有 int sortedIndex[],该函数可以打印排序前和排序后的结果。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。

6）定义 menu 函数，提供输入学生、打印学生，5 种排序、程序退出等 8 种选项，用户输入指定选项后，运行相应函数功能。**注意，在调用 inputStudents 函数前，需先提示用户数输入指定人数。**

7）在 BMI 类的 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.exp1.hit1170300101;

import java.util.*;
import java.math.*;

public class BMI {

    /**
     * @name main
     * @function run the function menu
     * @author Liu Qinglin
     * @version 1.2(1.0 died of using "class Student" and 1.1 was gone for not
     * using function menu)
     */
    public static void main(String[] args) {
        String[] ids = null,names = null;
        float[] heights = null,weights = null,bmis = null;
        int[] sortedIndex = null;
        menu(ids,names,heights,weights,bmis,sortedIndex);
    }

    /**
     * @name checkHealth
     * @function check where one's bmi lies
     * @author Liu Qinglin
     * @version 1.0
     * @param bmi
     * @return String type standing for the conditon
     */
    public static String checkHealth(float bmi) {
        if (bmi <= 18.5) {
            return "Underweight";
        } else if (bmi <= 23) {
            return "Normal Range";
        } else if (bmi <= 25) {
            return "Overweight--At Risk";
        } else if (bmi <= 30) {
            return "Overweight--Moderately Obese";
        } else {
```

```
            return "Severely Obese";
        }
    }

    /**
     * @name round
     * @function make the float type rounded up to 2 decimal places
     * @author Liu Qinglin
     * @version 1.0
     * @param f
     * @return String type standing for the conditon
     */
    public static float round(float f) {
        BigDecimal ff = new BigDecimal(f);
        float fff = ff.setScale(2, BigDecimal.ROUND_HALF_UP).floatValue();
        return fff;
    }

    /**
     * @param Num
     * @param ids
     * @param names
     * @param heights
     * @param weights
     * @param bmis
     * @name inputStudents
     * @function input students' information ,calculate the bmi and output the
     * bmi's condition
     * @author Liu Qinglin
     * @version 1.0
     */
    public static void inputStudents(int Num, String[] ids, String[] names, float[]
heights,
            float[] weights, float[] bmis) {
        Scanner cin = new Scanner(System.in);
        for (int i = 1; i <= Num; i++) {
            System.out.println("ID:");
            ids[i] = cin.nextLine();
            System.out.println("Name:");
            names[i] = cin.nextLine();
            System.out.println("Height:");
            float tmp = cin.nextFloat();
            heights[i] = (float) round(tmp);
            System.out.println("Weight:");
```

```
            tmp = cin.nextFloat();
            weights[i] = (float) round(tmp);
            cin.nextLine();
            if (heights[i] != 0) {
                tmp = weights[i] / heights[i] / heights[i];
                bmis[i] = (float) round(tmp);
            } else {
                bmis[i] = 0.00f;
            }
            System.out.println(checkHealth(bmis[i]));
        }
    }

    /**
     * @name printHint
     * @function print hint on the screen
     * @author Liu Qinglin
     * @version 1.0
     */
    public static void printHint() {
        System.out.println("1.Input Students' Information");
        System.out.println("2.Print Students' Information");
        System.out.println("3.Sort By Ids");
        System.out.println("4.Sort By Names");
        System.out.println("5.Sort By Heights");
        System.out.println("6.Sort By Weights");
        System.out.println("7.Sort By BMIs");
        System.out.println("8.Exit");
    }

    /**
     * @name menu
     * @function operate 8 functions,including input,output,5 kinds of sorts and
     * exit
     * @author Liu Qinglin
     * @version 1.0
     */
    public static void menu(String[] ids,String[] names,float[] heights,float[] weights,
            float[] bmis,int[] sortedIndex) {
        Scanner cin = new Scanner(System.in);
        boolean state = false;
        // this boolean type is used to control when to operate several exit functions
        while (!state) {
            /*
```

```
                The biggest loop is to make the operate continuous as long as the value of
state is true,
                and the value only changes when function exit is operated.
                First, only function 1 and 8 is allowed to operate, for before 1 is operated,
2 - 7 are not able to operate.
                So, before 1 or 8 is operated, the input only allows value 1 or 8, before
which the input is nonstop.
                 */
                int tmp = 0;
                while (tmp != 1 && tmp != 8 && !state) {
                    printHint();
                    // use printHint to print the hint for the first time
                    tmp = cin.nextInt();
                    // if the value of tmp is 8, get out of every loop
                    if (tmp == 8) {
                        state = true;
                        break;
                    }
                }
                // if state is equal to true, get out of the loop
                if (state) {
                    break;
                }
                // input the number of students before input students' information
                System.out.println("Input The Number Of Students：");
                int Num = cin.nextInt();
                // create the arrays
                ids = new String[Num + 1];
                names = new String[Num + 1];
                heights = new float[Num + 1];
                weights = new float[Num + 1];
                bmis = new float[Num + 1];
                sortedIndex = new int[Num + 1];
                /*
                set the sortedIndex's initial value as 1 - Num, so if execute function 2
before any sort,
                output's order is the same as the input's
                 */
                for (int i = 1; i <= Num; i++) {
                    sortedIndex[i] = i;
                }
                // execute function inputStudents
                inputStudents(Num, ids, names, heights, weights, bmis);
```

```
                // again, execute a nonstop loop, also, the boolean variable state is used to
control when to exit
                while (true) {
                    printHint();
                    tmp = cin.nextInt();
                    // tmp is used to control which function to execute, not use case
because it takes more code length
                    if (tmp == 2) {
                        printStudents(Num, ids, names, heights, weights, bmis,
sortedIndex);
                    } else if (tmp == 3) {
                        sortedIndex = sortByIds(Num, ids);
                    } else if (tmp == 4) {
                        sortedIndex = sortByNames(Num, names);
                    } else if (tmp == 5) {
                        sortedIndex = sortByHeights(Num, heights);
                    } else if (tmp == 6) {
                        sortedIndex = sortByWeights(Num, weights);
                    } else if (tmp == 7) {
                        sortedIndex = sortByBmis(Num, bmis);
                    } else if (tmp == 8) {
                        state = true;
                        break;
                    }
                }
                if (state) {
                    break;
                }
            }
        }

        /**
         * @name quicksortByString
         * @function quicksort the tmp array
         * @author Liu Qinglin
         * @version 1.0
         * @param beg,ed,tmp,index
         */
        public static void quicksortByString(int beg, int ed, String[] tmp, int[] index) {
            if (beg >= ed) {
                return;
            }
            // if beg >= ed, the sort is to an end, and the recursion is over
            int l = beg, r = ed;
```

```
        // choose a random pivot so that the algorithm cannot be slowed down by some
special cases
        int p = (int) (Math.random() * (ed - beg + 1) + beg);
        String key = tmp[p];
        // get the random pivot
        int key1 = index[p];
        // pitch on the corresponding index
        String t = tmp[p];
        int t1 = index[p];
        tmp[p] = tmp[l];
        index[p] = index[l];
        tmp[l] = t;
        index[l] = t1;
        // switch the left element and the random pivot, and the same is true of the
index
        while (l < r) {
            while (l < r && key.compareTo(tmp[r]) <= 0) {
                r--;
            }
            // find the first element smaller than the random pivot from the right side
            if (l < r) {
                tmp[l] = tmp[r];
                index[l] = index[r];
                l++;
            }
            // move it and its index and go on
            while (l < r && key.compareTo(tmp[l]) >= 0) {
                l++;
            }
            // find the first element bigger than the random pivot from the left side
            if (l < r) {
                tmp[r] = tmp[l];
                index[r] = index[l];
                r--;
            }
            // move it and its index and go on
        }
        tmp[l] = key;
        index[l] = key1;
        // set the remaining element as the key, and the index
        quicksortByString(beg, l - 1, tmp, index); //recursion
        quicksortByString(l + 1, ed, tmp, index); //recursion
    }
```

```
    /**
     * @name quicksortByFloat
     * @function quicksort the tmp array
     * @author Liu Qinglin
     * @version 1.0
     * @param beg,ed,tmp,index
     */
    public static void quicksortByFloat(int beg, int ed, float[] tmp, int[] index) {
        if (beg >= ed) {
            return;
        }
        // if beg >= ed, the sort is to an end, and the recursion is over
        int l = beg, r = ed;
        // choose a random pivot so that the algorithm cannot be slowed down by some
special cases
        int p = (int) (Math.random() * (ed - beg + 1) + beg);
        float key = tmp[p];
        // get the random pivot
        int key1 = index[p];
        // pitch on the corresponding index
        float t = tmp[p];
        int t1 = index[p];
        tmp[p] = tmp[l];
        index[p] = index[l];
        tmp[l] = t;
        index[l] = t1;
        // switch the left element and the random pivot, and the same is true of the
index
        while (l < r) {
            while (l < r && key <= tmp[r]) {
                r--;
            }
            // find the first element smaller than the random pivot from the right side
            if (l < r) {
                tmp[l] = tmp[r];
                index[l] = index[r];
                l++;
            }
            // move it and its index and go on
            while (l < r && key >= tmp[l]) {
                l++;
            }
            // find the first element bigger than the random pivot from the left side
            if (l < r) {
```

```
                tmp[r] = tmp[l];
                index[r] = index[l];
                r--;
            }
            // move it and its index and go on
        }
        tmp[l] = key;
        index[l] = key1;
        // set the remaining element as the key, and the index
        quicksortByFloat(beg, l - 1, tmp, index); //recursion
        quicksortByFloat(l + 1, ed, tmp, index); //recursion
}

/**
 * @name SortByIds
 * @function quicksort and return the index array
 * @author Liu Qinglin
 * @version 1.0
 * @param Num,ids
 * @return an index array
 */
public static int[] sortByIds(int Num, String[] ids) {
        String[] tmp = new String[Num + 1];
        for (int i = 1; i <= Num; i++) {
            tmp[i] = ids[i];
        }
        int[] index = new int[Num + 1];
        for (int i = 1; i <= Num; i++) {
            index[i] = i;
        }
        quicksortByString(1, Num, tmp, index);
        return index;
}

/**
 * @name SortByNames
 * @function quicksort and return the index array
 * @author Liu Qinglin
 * @version 1.0
 * @param Num,ids
 * @return an index array
 */
public static int[] sortByNames(int Num, String[] names) {
        String[] tmp = new String[Num + 1];
```

```
        for (int i = 1; i <= Num; i++) {
            tmp[i] = names[i];
        }
        int[] index = new int[Num + 1];
        for (int i = 1; i <= Num; i++) {
            index[i] = i;
        }
        quicksortByString(1, Num, tmp, index);
        return index;
    }

    /**
     * @name SortByHeights
     * @function quicksort and return the index array
     * @author Liu Qinglin
     * @version 1.0
     * @param Num,ids
     * @return an index array
     */
    public static int[] sortByHeights(int Num, float[] heights) {
        float[] tmp = new float[Num + 1];
        for (int i = 1; i <= Num; i++) {
            tmp[i] = heights[i];
        }
        int[] index = new int[Num + 1];
        for (int i = 1; i <= Num; i++) {
            index[i] = i;
        }
        quicksortByFloat(1, Num, tmp, index);
        return index;
    }

    /**
     * @name SortByWeights
     * @function quicksort and return the index array
     * @author Liu Qinglin
     * @version 1.0
     * @param Num,ids
     * @return an index array
     */
    public static int[] sortByWeights(int Num, float[] weights) {
        float[] tmp = new float[Num + 1];
        for (int i = 1; i <= Num; i++) {
            tmp[i] = weights[i];
```

```
            }
            int[] index = new int[Num + 1];
            for (int i = 1; i <= Num; i++) {
                index[i] = i;
            }
            quicksortByFloat(1, Num, tmp, index);
            return index;
        }

        /**
         * @name SortByBmis
         * @function quicksort and return the index array
         * @author Liu Qinglin
         * @version 1.0
         * @param Num,ids
         * @return an index array
         */
        public static int[] sortByBmis(int Num, float[] bmis) {
            float[] tmp = new float[Num + 1];
            for (int i = 1; i <= Num; i++) {
                tmp[i] = bmis[i];
            }
            int[] index = new int[Num + 1];
            for (int i = 1; i <= Num; i++) {
                index[i] = i;
            }
            quicksortByFloat(1, Num, tmp, index);
            return index;
        }

        /**
         * @name printStudents
         * @function print the Students' information after sort (or not)
         * @author Liu Qinglin
         * @version 1.0
         * @param Num,ids,names,heights,weights,bmis,sortedIndex
         */
        public static void printStudents(int Num, String[] ids, String[] names, float[]
heights,
                float[] weights, float[] bmis, int[] sortedIndex) {
            for (int i = 1; i <= Num; i++) {
                System.out.println(ids[sortedIndex[i]] + "\t" + names[sortedIndex[i]] +
"\t" + round(heights[sortedIndex[i]])
```

```
                    + "\t" + round(weights[sortedIndex[i]]) + "\t" +
round(bmis[sortedIndex[i]]));
                }
            }
        }
```

运行结果截图

```
1. Input Students' Information
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
1
Input The Number Of Students:
3
ID:
1170300101
Name:
LQL
Height:
1.82
Weight:
70.0
Normal Range

ID:
1170300102
Name:
LJR
Height:
1.70
Weight:
55.0
Normal Range
ID:
1111111111
Name:
GIRL
Height:
1.85
Weight:
95.5
Overweight--Moderately Obese
```

2. Print Students' Information

```
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
2
1170300101    LQL          1.82          70.0          21.13
1170300102    LJR          1.7           55.0          19.03
1111111111    GIRL         1.85          95.5          27.9
```

3. Sort By Ids(And output with function 2)

```
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
3
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
2
1111111111    GIRL         1.85          95.5          27.9
1170300101    LQL          1.82          70.0          21.13
1170300102    LJR          1.7           55.0          19.03
```

4. Sort By Names(And output with function 2)

```
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
4
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
2
1111111111   GIRL        1.85        95.5        27.9
1170300102   LJR         1.7         55.0        19.03
1170300101   LQL         1.82        70.0        21.13
```

5. Sort By Heights(And output with function 2)

```
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
5
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
2
1170300102   LJR         1.7         55.0        19.03
1170300101   LQL         1.82        70.0        21.13
1111111111   GIRL        1.85        95.5        27.9
```

6. Sort By Weights(And output with function 2)

```
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
6
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
2
1170300102    LJR          1.7          55.0          19.03
1170300101    LQL          1.82         70.0          21.13
1111111111    GIRL         1.85         95.5          27.9
```

7. Sort By BMIs(And output with function 2)

```
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
7
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
2
1170300102    LJR          1.7          55.0          19.03
1170300101    LQL          1.82         70.0          21.13
1111111111    GIRL         1.85         95.5          27.9
```

8. Exit

```
1.Input Students' Information
2.Print Students' Information
3.Sort By Ids
4.Sort By Names
5.Sort By Heights
6.Sort By Weights
7.Sort By BMIs
8.Exit
8
成功构建 (总时间: 11 分钟 48 秒)
```