Yo Treenawat Manomairat

Document analysis of Project 2

Professor Rafal Jabrzemski

CS 2334 Programming Structures and Abstractions

7 March 2019

Analysis of Project1

After reading Project 2 Guide, it came to mind that his project is connected to first one. After reading carefully we will notice that we don't get an mesonet.txt so we don't need to read from other document to compare, but what we need to do in this project is related to **ascii** number. From this, my understanding about the project is to change each character in 4 characters word into **ascii** number and calculate the mean of number, in three ways, **ceil**, **floor**, and **logical rounding**. After finish finding the number we then need to change the average number back into a **char.** For this, we already got the main and MesoStation, so we need to create an abstract class and Inherit class on our own.

First MesoAbstract:

There is nothing much in this state, as we only create an abstract class, in which I create three things, which is protected String, an abstract to link inherited class to, and an array that will be used in inherited class. Knowing the out put the size of an array is 3

```
import java.util.Arrays;

public abstract class MesoAbstract
{
        //Protected String
        protected String StID;

        //Abstract
        public MesoAbstract(MesoStation mesoStation) {
                StID = mesoStation.getStID();
        }

        //Array created for MesoInherit
        int[] average = new int[3];
}
```

Second MesoInherited:

1. Create two method

After reading guideline and looking into the code, we can see that there are only two method in inherited class, which one use to calculate average and pass it back into an array, while another change the average into char

```
public int[] calAverage() {

}
public char letterAverage() {

}
```

2. Create two array, first to separate each character and other to return average

For this, I first created an array and pass each character as an **ASCII** number into the array as can be seen.

I then create another array in which suppose to put three type of average into them(**ceil, floor, and logical rounding**). I take out StID and int[] average, after knowing what I do wrong later on

```
public int[] calAverage() {
        String StID = "OKCE";
        int[] letter = new int[4];
        letter[0] = StID.charAt(0);
        letter[1] = StID.charAt(1);
        letter[2] = StID.charAt(2);
        letter[3] = StID.charAt(3);
        int[] average = new int[4];
        double mean = 0;
        for (int i = 0; i < letter.length; i++) {
                mean += letter[i];
        }
        mean = mean/letter.length;
```

3. Input different mean into an Array

For this, the first mean is **ceil**, in which I cast from double into int, which is the same for second average **floor**, which done in similar way.

```
average[0] = (int) Math.ceil(mean);
average[1] = (int) Math.floor(mean);

String[] split = String.valueOf(mean).split("\\.");
int[] split2 = new int[2];
split2[0]=Integer.parseInt(split[0]);
split2[1]=Integer.parseInt(split[1]);
average[2] = split[1].;
int firstDigit = Integer.parseInt(Integer.toString(split2[1]).substring(0, 1));

average[0] = (int) Math.ceil(mean);
average[1] = (int) Math.floor(mean);
if (firstDigit >= 5) {
        average[2] = (int) Math.ceil(mean);
}
else {
        average[2] = (int) Math.floor(mean);
}
return average;
```

The **complicated** part is **logical rounding**, in which need to check what the number of first digit of decimal. In which if the first decimal place is 5 or more, we need to **round up**, while lower than 5, we need to **round down**. For this I create two array each having **length of 2.** After create an array, I split the number with ".", the first half of number before "." go to [0], while decimal number half go to [1]. After finish splitting them, I then took out the first digit of the **decimal String,** to see what is the value of the first decimal place. For this, I use if-else statement and see if the firstDigit is more than or equal to 5, then we round up, else we round the number down.

3.5 I accidentally said that this class is public abstract, so I need to change the class back to normal or else the main class won't work because it can't initiate an abstract class.

4. After finish the average, I then finish letterAverage() by changing the mean back to char

```
char answer = (char) average[0];
return answer;
```

For this, I just change the average number from the array and cast it back to char. (According to picture, it is wrong and average should be 2, in which I change later on.)

5. Comment on every step so that people could understand

Overall, in this project is easier than the first one, in which we need to change each character into an int, find the average and pass it back as a **char**. We don't really need to worry about output because main class already have everything, so we only need to calculate averages array that will be use for output and how we implement the abstract and inherited using super for example. The hard part would be the logical rounding in which I don't really know if there an easier way to check the first decimal place.