Yo Treenawat Manomairat

Document analysis of Project 3

Professor Rafal Jabrzemski

CS 2334 Programming Structures and Abstractions

30 March 2019

Analysis of Project3

After reading Project 3 Guide, it came to mind that his project is connected to first and second one. After reading carefully we will notice that we got an mesonet.txt back so we need to read from other document to compare. Similar to the first one, we need to compare each word with the word fair in which also similar to second project, in which we need to 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 in a **logical way**, which then we find all the word with same average ascii and sorted it out. The three class we need to create is related to first creating mean for the word, second class is comparing the mean average of every word in mesonet and add all the word that have same average into hashmap, and last is sorting the hashmap.

First MesoAscii:

There nothing much around the first class, because it is similar to what we did in second, so first I use this.StID to connect it to the main method, in which is the word Fair.

```
public class MesoAscii extends MesoAsciiAbstract {
    String StID;

    public MesoAscii(MesoStation StID){
        this.StID = StID.getStID();
    }


}
```

1. create an array that hold each character from the 4 character words, in which the type of an array is an int, so it force the value of each character into Integer form.

```
@Override
int calAverage() {
    //Create array for containing int valuable of each character
    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);

}
```

This help lessen the step of taking each character out and change it into integer separately.

2. Calculate the mean by adding the value of each character and then divide by the amount(length)

```
//Calculate mean by adding all number and divide by the length
double mean = 0;
for (int i = 0; i < letter.length; i++) {
    mean += letter[i];
}
mean = mean/letter.length;
```

3. Last is to make a logical rounding, that will round math up when the first decimal is 5 or more. From this as we already got the mean from step 2, we need to find the first decimal place, in which I split the number by "." so the first half and second half will separate in to an array. After knowing that, I took the second half and determine what is the first digit number and use if-else statement to check if the first digit is 5 or more then, math.ceil, else floor if the first decimal is less than 5.

```
//Take out the first digit of number by spliting by . and then take the first digit number
String[] split = String.valueOf(mean).split("\\.");
int[] split2 = new int[2];
split2[0]=Integer.parseInt(split[0]);
split2[1]=Integer.parseInt(split[1]);
int firstDigit = Integer.parseInt(Integer.toString(split2[1]).substring(0, 1));

//If first Digit is >= 5, then use ceil, else use floor
int average = 0;
if (firstDigit >= 5) {
    average = (int) Math.ceil(mean);
}
else {
    average = (int) Math.floor(mean);
}

//return average[]
return average;
```

4. return average

Second MesoEqual:

1. Similar to before, we need to use this.

to related the parameter.

```
public class MesoEqual {
    String StID;

    //this.
    public MesoEqual(String StID) {
        this.StID = StID;
    }
```

2.

```
//create hashmap according to main (calAsciiEqual)
public HashMap<String, Integer> calAsciiEqual() throws IOException{
    //Find same average as Fair
    HashMap<String, Integer> sameAverage = new HashMap<String, Integer>();
    ArrayList<String> mesonetArray = new ArrayList<String>();

    //Read file from Mesonet and assign it as newWord, then loop it to add those newWord into mesonetArray
    File file = new File("Mesonet.txt");
    BufferedReader br = new BufferedReader(new FileReader(file));
    String newWord;
    newWord = br.readLine();
    newWord = br.readLine();
    newWord = br.readLine();
    newWord = br.readLine();
    newWord = br.readLine();
    while(newWord != null) {
        mesonetArray.add(newWord.substring(2,  6));
        newWord = br.readLine();
    }
    br.close();
```

This is wrong method, but we are reading the from the mesonet and add it into an array. For this, as we returning the type as HashMap, we first create public HashMap and HashMap SameAverage so that it can be return and an arrayList mesonetArray to add the word from meson inside.

3. Find average of the main StID(Fair), so that we can compare it to the average from Mesonet file.

```
//Use similar method from MesoAscii to find average
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);

double mean = 0;
for (int i = 0; i < letter.length; i++) {
    mean += letter[i];
}
mean = mean/letter.length;

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

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

4. Make a loop that go through all the word from the array and find average for each of them and add it to the hash map if it have the same average as "Fair". For This, it is similar to what did before, but there is loop in front and at the end there is an if statement, that if the average is the same as "fair" average, add it to the hashmap and after finish the loop return the hashmap sameAverage.

```
    //Add word with same Average from Mesonet with word "Fair" to hashmap sameAverage
    for(int i = 0; i < mesonetArray.size(); ++i)
    {
        int[] letter2 = new int[4];
        letter2[0] = StID.charAt(0);
        letter2[1] = StID.charAt(1);
        letter2[2] = StID.charAt(2);
        letter2[3] = StID.charAt(3);

        //Find average and compare them to word "Fair"
        double mean = 0;
        for (int j = 0; j < letter.length; j++) {
            mean += letter2[i];
        }
        mean = mean/letter2.length;

        String[] split3 = String.valueOf(mean).split("\\.");
        int[] split4 = new int[2];
        split4[0]=Integer.parseInt(split3[0]);
        split4[1]=Integer.parseInt(split3[1]);
        int firstDecimal = Integer.parseInt(Integer.toString(split4[1]).substring(0, 1));

        int average = 0;
        if (firstDecimal >= 5) {
            average = (int) Math.ceil(mean);
        }
        else {
            average = (int) Math.floor(mean);
        }

        if(average == StIDAscii)
        {
            sameAverage.put(StID, average);
        }
    }
return sameAverage;
}
```

Third StationLexicographical:

The last class within this project in which is use for creating an sorted hashmap. This is the most complicated as well as the less step, as it have two step.

1. Create Hashmap AsciiVal, and create return method that print out list of word and there average value

```
public class StationLexicographical extends MesoSortedAbstract {
    //create hashmap according to main (AsciiVal)
    HashMap<String, Integer> AsciiVal;

    //Sorted Map
    public StationLexicographical(HashMap<String, Integer> AsciiVal) {
        this.AsciiVal = AsciiVal;
        Map<String, Integer> sortedMap = sortedMap(AsciiVal);
        for (String StID : sortedMap.keySet()) {
            System.out.println(StID + " " + sortedMap.get(StID));
        }
    }

}
```

2. Sorted map, for this I start with setting up everything first, such as, ArrayList, set, and Iterator, and then got through first loop add the word into newWord. Then sort it using collections. Then go for second loop get the word and value and put them into sorted hashMap.

```
//Create sortedMap
@Override
Map<String, Integer> sortedMap(HashMap<String, Integer> unsortedMap) {
    //setUpfirst
    ArrayList<String> newWord = new ArrayList<String>();
    Set<Map.Entry<String, Integer>> unSorted = AsciiVal.entrySet();
    Iterator<Entry<String, Integer>> iterate = unSorted.iterator();

    //run thought unsortedmap and add them to newWord
    for(int i = 0; i < unsortedMap.size(); i++){
        newWord.add(iterate.next().getKey());
    }

    //Use collection to sorted newWord
    Collections.sort(newWord);
    Map<String, Integer> sorted = new TreeMap<String, Integer>();

    //run through for loop and add them into hashmap sorted
    for(int j = 0; j < newWord.size(); j++){
        String stid = newWord.get(j);
        int ascii = unsortedMap.get(stid);
        sorted.put(stid, ascii);
    }
    return sorted;
}
```

Overall, in this project is a lot more complicate than the first two, in which we need to change each character into an int, find the average and compare them to mesonet, in which we need to use hashMap. The first two class are relatively easy as it is combination for first two project, but the last class is hard as we need to sort the hashMap.