CS 2334: Programming Structures and Abstraction

Project 3

HashMap

Due Date: Oct 31, 2020

Fall 2020

100 pts

1. Objective:

The objective of this programming project is to implement a java program where mainly HashMap will be used to store/retrieve/sort values along with some other java functions. After completing the project, students will have an intermediate understanding of HashMap.

2. Project Specification:

2.1. Overall Program Behavior:

For this project you will perform some operations on Date/Time and Ascii code using HashMap. Be aware that the given files may look like the same as the previous projects but obviously these are little different from the previous files. i.e., you may have to parse the files differently. Concepts of Abstraction and ASCII code also have been used in this project. For Project 3, the same 4-letter station code will be used as the input, however, you must use HashMap to solve all the problems. Not all but some of the classes have been provided. From Driver/Main class, you have to determine what classes/methods are necessary to complete the project; however, don't create any unnecessary class, since your project will be evaluated by Zylab, therefore, you will not be able to upload any unnecessary class files, but you can write any number of methods.

In this project, based on the Main class and abstract classes, you are required to think and write all the necessary Classes, Methods etc. to generate intended output. Your code will be tested for varieties of input combination.

2.2 Input Format:

This PDF should read along with the description provided in the Driver/Main class to get all the information needed for this project. In the Main class, you will find the description of the intended output from a specific line of code.

2.3 Output Format:

Start from the Main/Driver class, it is mentioned in the Main class when to come to pdf for details. I have attached a text file containing the sample output so that you don't get confuse with the final output, since a text file is easier to read than PDF for formatting.

Section 2 Extension

```
System.out.print("ASCII average: ");
System.out.println(AsciiVal.get(StID));
```

Output: ASCII average: 79

Description: ASCII average is calculated as you did in project 2. It's not floor or ceiling, rather the average.

ASCII average in this project is the average of two Averages. How to get two averages is described below.

To get the ASCII value for a city, for example, N R M and N as 78, 82, 77, and 78. Sum of these ASCII value is 315. Dividing 315 by 4, we get 78.75

Taking the ceiling of 78.75, you will get the first part: Ascii Ceiling is 79

Taking the floor, you will get the second part: Ascii Floor is 78

For the third part, if the fraction part is less than 0.25, the Average would be floor. Otherwise, if the fraction is greater than or equal to 0.25 (for this project we are using 0.25), the Average would be ceiling.

This is the First Average.

For the Second Average will come from a fixed station, "NRMN". Calculation process of the second average is same as the first average. Here, value is 79.

ASCII average will be the average of First Average and Second Average; however, the ASCII average will be ceiling, in case, the average of first average and second average is a fraction.

ASCII average, in this case, is 79.

Next Output: Stations are: {NRMN=79, OKMU=79, STIL=79, JAYX=79, NEWP=79, STUA=79, WATO=79, MRSH=79}

Description: Here, you will compute ASCII average of all the stations. You will get some stations have the same ASCII average (79, in this case, NRMN). You will add these stations (who have same ASCII average) and Print.

Next code:

```
asciiVal=mesoequal.calAsciiEqual();
for (String stid : asciiVal.keySet())
{
    System.out.println(stid + " " + asciiVal.get(stid));
}
```

```
Output: Unsroted:
NRMN 79
OKMU 79
```

```
STIL 79
JAYX 79
NEWP 79
STUA 79
WATO 79
MRSH 79
```

Description: Same output is printed in a different way, here you can see the list of stations is unsorted.

Next code:

```
new StationLexicographical(asciiVal);
```

```
Output: Sorted:
JAYX 79
MRSH 79
NEWP 79
NRMN 79
OKMU 79
STIL 79
STUA 79
WATO 79
```

Description: You need to sort the list of the stations in lexicographical order. This alphabetical order is also called lexicographic or lexical order. You will find a description here: https://en.wikipedia.org/wiki/Lexicographical_order

Pls, print in the specific format as provided above, since Zylab will grade you automatically.

Note: In the development of coding process, you will need to inherit some classes from other classes.

3. File names:

File names for this project are as follows:

We will provide Main.java, some classes (complete and incomplete), and text files.

You will write *necessary classes, **and*** **Documentation** (in the form of Javadoc).

The **documentation** should consist of details analysis of your implemented codes (classes, methods, input/output etc.). Javadoc will be discussed in the lab section before the due date.

4. Points Distribution:

Bits and pieces	Points
Code on Zylab	60
Code Review	20
Documentation (Javadoc, during code review)	20

5. Submission Instructions: (Due on October 31, 2020, 11:59 pm)

This project requires the submission of a *soft copy on*

- *ZyLab for grading* and
- *Github for review.*
- ***Documentation*** *(Javadoc on Github only).*

Due date for code submission and Javadoc is the same date (unlike project 1 or 2).

Plagiarism will not be tolerated under any circumstances. Participating students will be penalized depending on the degree of plagiarism. It includes "No-code" sharing among the students. It can lead to academic misconduct reporting to the authority if identical code is found among the students.

6. Late Penalty:

Submit your project before the due date/time to avoid late penalty. **A late penalty of 20*% per hour* will be imposed after the *due date/time***. After five hours from the due date/time, you will not be allowed to submit the project.

Good Luck!!