CS 2334: Programming Structures and Abstraction

Project 3

HashMap

Due Date: Oct 26, 2019

Spring 2019

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 and date in Java.

2. Project Specification:

2.1. Overall Program Behavior:

For this project you will perform some operations on Date/Time and Ascii code using HashMap. In different section you may see different format of date/time, don't think this as an error, rather you have to follow the format given for each section. You will also work with the stations from Mesonet. 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 files differently. Some 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 class, you have to determine what classes/methods are necessary to complete the project; however, don't create unnecessary class, since your project will be evaluated by Zylab, therefore, you will not be able to upload any unnecessary class, but you can write any number of methods if you think they are necessary.

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

I have changed the coding style a lot to make it readable with zylab; however, still, some items Zylab can't check, they will be checked manually during review.

Section 1 Extension:

```
	/**
		 * Now add two more time zone like this: (These two may not be real time
zone)
		 * "ZST", "11/05/2018 19:59" and "AST", "10/01/2020 19:59".
		 * After adding these two, your Hashmap may look like:
		 *
```

Time zone in Hashmap:
BST:10/09/2019 02:48
AST:10/01/2020 19:59
CST:10/08/2019 15:48
ZST:11/05/2018 19:59
GMT:10/08/2019 20:48
 * You can see the First Hashmap is not sorted.
 *
 * Now, declare a second Hashmap, again as <String, String>
 * Store the values of first Hashmap as the key of the second Hashmap.
 *
 * Declare an array with type LocalDateTime.
 * Store the values from the first Hashmap (or key of the second hashmap) in the array.
 * You will need to convert from String to LocalDateTime to store in the array. You can use any
 * function/method to convert but make sure the array content are formatted as "2019-10-09T02:48"
 * means year-month-dayThour:minute
 *
 * So far, you have two hashmap and an array.
 *
 * Now sort the first hashmap using keys and print. It should be look like:
 *

Print Style 1:
AST 10/01/2020 19:59
BST 10/09/2019 02:48
CST 10/08/2019 15:48
GMT 10/08/2019 20:48
ZST 11/05/2018 19:59
 *
 * Now print the second hashmap and print it like:
 *

Print Style 2:
10/08/2019 20:48
10/01/2020 19:59
11/05/2018 19:59
10/08/2019 15:48
10/09/2019 02:48
 *

```
 * Now sort the second hashmap using it keys and print like:
 *
Print Style 3:
10/01/2020 19:59
10/08/2019 15:48
10/08/2019 20:48
10/09/2019 02:48
11/05/2018 19:59
 *
 * Now print the unsorted array you have:
 *
Print Style 4:
2019-10-09T02:48
2020-10-01T19:59
2019-10-08T15:48
2018-11-05T19:59
2019-10-08T20:48
 *
 * Sort the array and print it like:
 *
Print Style 5: Final sorted Array:
2020-10-01T19:59
2019-10-09T02:48
2019-10-08T20:48
2019-10-08T15:48
2018-11-05T19:59
 *
 * Look at the three style (1, 3, 5). They are giving you different
types of sorted list that are confusing. For grading you have to print Style 1, 3,
and 5 only (see Main class).

 * This section is complete, now we will move to the next section.
 */
```

Section 3 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. Please, look at the Project 2 description, if needed.

ASCII average is the average of two averages.

To get the ASCII value for 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 part is greater than or equal to 0.25, the Average would be ceiling.

This is the first average. First string will be tested for different stations.

For the second average, the station is fixed, “NRMN”. Calculation of the second average is same as the first average. Here, value is 79.

Final average will be the average of first average and second average; however, the average will be ceiling, in case, it is a fraction.

ASCII average is this final average, here 79.

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

Description: Here, you should compare 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.

```
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.

```
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, though emphasize has been given on accuracy of the result.

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** (You should write your documentation in **README .md** on Github).

The **documentation** should consist of discussion of your understanding of the problem, your problem-solving approach, an UML and details analysis of your implemented codes (classes, methods, input/output etc.).

4. Points Distribution:

Bits and pieces	Points
Code on Zylab	80
Github (at least 20 submissions)	10
Documentation (README.md)	10

5. Submission Instructions: **(Due on October 26, 2019, 11:59 pm)**

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

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

Plagiarism will not be tolerated under any circumstances. Participating students will be penalized depending on the degree of plagiarism.

6. Late Penalty:

Submit your project before the due date/time to avoid any 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.

** For any update/modification/announcement on the project (regarding understanding, error, submission error etc.), pls post/follow a single place, the discussion section on Canvas under Project3.

Good Luck!!