CS 2334: Programming Structures and Abstraction

Project 2

Abstraction

Due Date: Oct 14, 2020

Fall 2020

100 pts

1. Objective:

The objective of this programming project is to implement a java program where Abstract class/method can be inherited. After completing the project, students will have an intermediate understanding of Abstraction. Pls, start the project early so that you can get answer of any confusion/clarification. For clarification of any part, pls, post on Discord discussion channel of this project.

2. Project Specification:

2.1. Overall Program Behavior:

For the first part of the project, pls read the Driver/Main file uploaded which is self-explanatory. For help related to Date/Year, pls see: https://docs.oracle.com/javase/8/docs/api/java/time/YearMonth.html.

For the second part of the project, you are already familiar with the Mesonet text file where a station ID is a 4-letter code. For project 2, a 4-letter Station ID will be used as the input. Mesonet.txt is a different version of the file used in the previous project. Using abstraction, you will write classes (Empty class will be provided) to compute some averages of the 4-letter code. Calculation method of the averages have been discussed in Section 2.4.

In this project, you are required:

1. to write DateTimeInherit.java, MesoInherit.java, PosAvg.java, and LetterAvg.java;
2. to write a README.md (will be reviewed and graded by an instructor during code review).

2.2 Input Format: (from Driver.java)

This PDF should be read along with the Driver/Main class uploaded for this project. In Section 2.3 is the output and Section 2.4 describes the output. One station ID will be entered. For example, NRMN:

```
String stID = "NRMN";
```

Your program will be tested with a variety of inputs. In this example, we are using NRMN, which is the code for Norman.

2.3 Output Format:

```
The Index of the city is in the Mesonet: 77
This index is average of NOWA and OILT, NEWP and OKCE, and so on.
```

```
Ascii Ceiling is 79
Ascii Floor is 78
Ascii Average is 79

Letter Avg: O

Total number of stations starting with the letter 'O' is 4.
These stations are: OILT OKCE OKEM OKMU
```

** This is a sample output, for ‘NRMN’ as input. Your code will be tested with many different inputs.

2.4 Description of the output:

You have 4 letters: ‘N’, ‘R’, ‘M’, and ‘N’

```
The Index of the city is in the Mesonet: 77
```

/*If you calculate the Index of NRMN in the Mesonet.txt file, starting from 1 is 77. Remember, array index 0 is the first element, i.e., for the first element, index is 1.*/

```
This index is average of NOWA and OILT, NEWP and OKCE, and so on.
```

/*If the index is 77, then, 77 is the average of 76(NOWA) and 78(OILT). It’s also the average of 75(NEWP) and 79(OKCE). Showing up to two stations is enough.
Note: since you are calculating up to N+2 or N-2, if you give the first two stations or last two stations as input, it will generate error. Therefore, you can skip giving the first two stations or last two stations as the input.*/

```
Ascii Ceiling is 79
Ascii Floor is 78
Ascii Average is 79

Letter Avg: O
```

/*You have to retrieve the ASCII value for N R M and N which are 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, for this project we are considering, if the fraction is less than 0.75, the Average would be floor of the value. Otherwise, if the fraction part is greater than or equal to 0.75, the Average would be ceiling of the value. In this case, we are getting the average as 79 since the average is 78.75 (here 0.75 is greater or equal to 0.75).

For the fourth part, the letter value of the average which is 79 (ceiling of 78.75), which is equivalent to letter ‘O’. [It is not ‘zero’, rather the first letter of Oklahoma]*/

```
Total number of stations starting with the letter 'O' is 4.
```

/*In the station list, number of stations starting with ‘O’ is 4.*/

```
These stations are: OILT OKCE OKEM OKMU
```

/*Finally, here is the list of those four stations.*/

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

3. File names:

File names for this project are as follows:

We will provide: Main.java, DateTimeAbstract.java, MesoAbstract.java, MesoStation.java, and Mesonet.txt

You will write: *DateTimeInherit.java, MesoInherit.java, PosAvg.java, LetterAvg.java, and* README.md

Based on the given DateTimeAbstract.java and MesoAbstract.java, you have to inherit necessary methods.

DateTimeInherit.java: You need to write this class to generate the first part. Here, you must inherit the abstract class/methods, as necessary.

PosAvg.java: You will write necessary code to generate the output related with the position/index.

MesoInherit.java: You need to write this class to generate intended output. Here, you must inherit the abstract class/methods, as necessary.

LetterAvg.java: This class will contain the code to generate the output of the last part of the Driver.

The documentation (README.md) should consist of discussion of your problem-solving approach and details analysis of your implemented code including description of each class/method/variable.

4. Points Distribution:

Bits and pieces	Points
Java files (on Zylab for automatic grading)	60
README.md (will be graded during Project review)	20
Code review (will be graded during Project review)	20

5. Submission Instructions:

This project requires the submission of *soft copy only.*

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.

README.md: This documentation should contain understanding of the problem and your problem-solving approach, description of methods and variables used. This file will be reviewed by an instructor along with your code during the code review.

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 are not allowed to submit the project.

Good Luck!!