CS 280
Fall 2022
Recitation Assignment 9
December 1st, 2022

Due Date: Monday, December 5, 2022, 23:59
Total Points: 9

Define an abstract class, called *UtilityCustomer*, that has one integer instance variable, an account number, *accNum*, and an abstract method, called *calculateBill*, that returns the bill amount as double. The *UtilityCustomer* class includes a default constructor that initializes the account number by -1, and a constructor that is passed a parameter for initializing the instance variable. It includes also accessor method for the instance variable, and a *Display* method that displays the account number. The *UtilityCustomer* class has two non-abstract subclasses that inherit from the *UtilityCustomer*: *GasCustomer* and *ElectricCustomer*. The subclasses definitions are as follows:

(a) *GasCustomer* class has two additional fields, the *Usage* in Cubic meters, and a constant for the price of gas per cubic meter of $3.75. Include a default constructor for initializing the *Usage* variable with 0.0 and the account number with -1, and a constructor that accepts two parameters for initializing the *Usage* and the account number fields. Include accessor and mutator methods for class's instance variable, and implement the *calculateBill* method. Write a *Display* method, that calls the *Display* method of the *UtilityCustomer* to display the gas customer's account number, the gas consumption, and the amount charged.

(b) *ElectricCustomer* class has two additional fields, *kWattHourUsed* instance variable, and a constant, for the price of electricity per kilowatt hour of $0.25. Include a default constructor for initializing the *kWattHourUsed* variable with 0.0 and the account number with -1, and a constructor that accepts two parameters for initializing the *kWattHourUsed* and the account number. Write a *Display* method, that calls the *Display* method of the *UtilityCustomer*, and implement the *calculateBill* method.

Implement each class in a separate header file that carries the class's name as: "GasCustomer.h", "ElectricCustomer.h", and "UtilityCustomer.h".

A driver program, called *RA9Prog.cpp*, is used to test your implementations of the three classes. The program reads from an input file a list of customers records and accepts a command line flag. Each customer's record includes the customers type (Gas, or Electricity), customer's account number, and the amount of usage. The program considers all Gas customers should have accounts in the range 0 to 99999, and all Electricity customers should have account numbers in the range 100000 to 199999. The customers records should not include two records with the same account number.
The accepted command flags are related to the following test cases:

- Case 1 "-new": Displays the list of accepted customers records.
- Case 2 "-rej": Displays the list of rejected records that have similar account numbers.
- Case 3 "-inv": Displays the lists of invalid customers' records based on the customer's account type.

- Case 4 "-ave": Displays the average consumption and the average bill amount for each customer type.

Vocareum Automatic Grading

- Upload your class implementation files to Vocareum.
- The driver program "RA9Prog.cpp" will be used to test your implementations using the 4 test cases that are described above.

Submission Guidelines

- Please upload your implementations to Vocareum in separate files one for each class implementation as: "Utility Customer.h", "GasCustomer.h", and "ElectricityCustomer.h".
- **Submissions after the due date are accepted with a fixed penalty of 25%. No submission is accepted after Wednesday 11:59 pm, December 7, 2022.**

Grading Table

Item	Points
Compiles Successfully	1
Case 1: Accepted customers records	2
Case 2: Rejected customers records	2
Case 3: Invalid customers records	2
Case 4: Average usages and bill amounts	2
Total	9