Assignment 6: Shopping Customers

Due: 23:59, Sat 8 Dec 2018 Full marks: 100

Introduction

An online shopping platform has a loyalty membership program which distinguishes different kinds of customers: *standard*, *gold*, and *diamond* customers. Different customer types have different membership benefits such as rebate amount and shipping fee discount. *Standard customers* have a rebate rate of 1% on the first \$500 purchase, 1.5% on the next \$1000 purchase, and 2% on the remainder purchase. E.g., if a standard customer purchases \$2300, s/he can earn a rebate of $\$500 \times 1\% + \$1000 \times 1.5\% + (\$2300 - \$500 - \$1000) \times 2\% = \$36$. *Gold customers* have a rebate rate of 1% on the first \$500 purchase, 1.7% on the next \$1000 purchase, and 2.4% on the remainder purchase. E.g., a gold customer purchasing \$2300 can earn a rebate of $\$500 \times 1\% + \$1000 \times 1.7\% + (\$2300 - \$500 - \$1000) \times 2.4\% = \$41.2$. *Diamond customers* earn rebates like gold customers but can be multiplied by a premium (%). (E.g., $\$41.2 \times 1.1 = \$45.32$)

Besides, shipping fee in the shopping platform is also different for different customers. Standard customers' shipping fee is determined as follows:

	Weight < 8kg	**Weight ≥ 8kg**
Purchase < \$400	Shipping fee: \$40	Shipping fee: \$60
Purchase ≥ \$400	Shipping fee: \$0	Shipping fee: \$40

Shipping fee of gold customers is similar to standard customers, but they enjoy a shipping discount rate (%). (E.g., $\$40 \times (1 - 10\%) = \$36$) Diamond customers enjoy a fixed \$30 markdown on the shipping fee compared to gold customers. (E.g., $\$36 - \$30 = \$6$)

In this assignment, you will apply inheritance and polymorphism to write three classes `Customer`, `GoldCustomer`, and `DiamondCustomer` to model standard, gold, and diamond customers respectively, and a client program to read some customers data from a file and compute the rebates and shipping fees accordingly.

Program Specification

You have to write your program in four source files (1) `Customer.cpp`, (2) `GoldCustomer.cpp`, (3) `DiamondCustomer.cpp`, and (4) `rebate.cpp`. The first three files are for implementing the corresponding classes organized in an inheritance hierarchy. The fourth is a client program of these classes. You are given three header files `Customer.h`, `GoldCustomer.h`, and `DiamondCustomer.h`. Do *not* modify their contents. You are recommended to write the files in order (1, 2, 3, 4). *Implement the member functions of each class and test them individually one by one. Your four files will be graded separately*, so you should not mix the functionalities of them. Also, you *cannot declare any global variables* in *all* your source files (except `const` ones).

Class `Customer` (`Customer.cpp`)

The `Customer` class models standard customers. Its interface is given as follows.

```
class Customer {
public:
    Customer(string id = "0000000000");
    string getCustomerID() const;
    string getCustomerName() const;
    void setCustomerName(string n);
    double getPurchaseAmount() const;
    void setPurchaseAmount(double p);
    double getWeight() const;
    void setWeight(double w);
    virtual double rebate() const;
    virtual double shippingFee() const;
private:
    string cid, name;
    double purchase, weight;
};
```

Private Data Members

`string cid, name;`

The data member `cid` is the customer ID, which is a string consisting of ten digits, while `name` is the customer name.

`double purchase, weight;`

The purchase amount ($) and the shipping weight (kg) of the customer respectively.

Public Constructor and Member Functions

`Customer(string id = "0000000000");`

The parameter `id` is supposedly the customer ID for initializing the data member `cid`. However, `id` can be a string with any possible contents. You have to remove all non-digit characters from it. Then if it has not enough digits (< 10), add enough '0's to the front to make it 10-digit. If it has more than ten digits, trim it to retain only the *rightmost* ten digits. E.g., suppose `id` is "dq689a358sdc964569", then `cid` should be initialized as "9358964569". Besides, the customer name should be initialized as the empty string "", and purchase amount as 0.0, and shipping weight as 0.5.

`string getCustomerID() const;`

Returns the customer ID of the customer.

`string getCustomerName() const;`

Returns the name of the customer.

`void setCustomerName(string n);`

Sets the customer name to the parameter n.

`double getPurchaseAmount() const;`

Returns the purchase amount of the customer.

`void setPurchaseAmount(double p);`
Sets the purchase amount of the customer to the parameter p or 0.0, whichever is higher. (That is, if p is less than 0.0, then set the purchase amount to 0.0 instead.)

`double getWeight() const;`
Returns the shipping weight of the purchase.

`void setWeight(double w);`
Sets the shipping weight to the parameter w or 0.5, whichever is higher.

`virtual double rebate() const;`
This `virtual` member function computes and returns the rebate of the customer in this purchase. See the progressive rebate calculation of a standard customer in the Introduction section.

`virtual double shippingFee() const;`
This `virtual` member function returns the shipping fee of the purchase of the customer. Again, see how the shipping fee of a standard customer purchase is determined in the Introduction section.

Class GoldMember (GoldMember.cpp)

The `GoldCustomer` class models gold customers and is a subclass of `Customer`. Its interface is given as follows.

```
class GoldCustomer : public Customer {
public:
    GoldCustomer(string id = "0000000000", double d = 0.1);
    double getShippingDiscount() const;
    void setShippingDiscount(double d);
    virtual double rebate() const;
    virtual double shippingFee() const;
private:
    double shippingDiscount;
};
```

Private Data Member

`double shippingDiscount;`
The discount off (%) on the shipping fee. E.g., if the shipping discount is stored as 0.1 (10%), that means the shipping fee would be multiplied by 0.9.

Public Constructor and Member Functions

`GoldCustomer(string id = "0000000000", double d = 0.1);`
The parameter `id` is supposedly the customer ID of the gold customer. This constructor (1) calls the constructor of its superclass `Customer` and (2) initializes the shipping discount as the parameter d or 0.1, whichever is higher. But if d > 1.0, then set the shipping discount to 1.0 only.

`double getShippingDiscount() const;`
Returns the shipping discount of the gold customer.

`void setShippingDiscount(double d);`

Sets the shipping discount of the gold customer to the parameter d. But if d < 0.1, then set the shipping discount to 0.1. Besides, if d > 1.0, then set the shipping discount to 1.0. (That is, shipping discount is at least 10% off and at most 100% off.)

`virtual double rebate() const;`

This member function *overrides* the one in its superclass. It returns the rebate of the purchase of the gold customer. See the progressive rebate calculation of a gold customer in the Introduction section.

`virtual double shippingFee() const;`

This member function *overrides* the one in its superclass. It returns the shipping fee of the purchase of the gold customer, which is that of a standard customer but with a shipping discount.

Class DiamondCustomer (DiamondCustomer.cpp)

The `DiamondCustomer` class models diamond customers and is a subclass of `DiamondCustomer`. Its interface is given as follows.

```
class DiamondCustomer : public GoldCustomer {
public:
    DiamondCustomer(string id = "0000000000", double d = 0.1,
                                              double p = 1.1);
    double getPremium() const;
    void setPremium(double p);
    virtual double rebate() const;
    virtual double shippingFee() const;
private:
    double premium;
};
```

Private Data Member

`double premium;`

The rebate premium (%) of the diamond customer.

Public Constructor and Member Functions

`DiamondCustomer(string id = "0000000000", double d = 0.1, double p = 1.1);`

The parameter `id` is supposedly the customer ID of the diamond customer. This constructor (1) calls the constructor of its superclass `GoldCustomer` and (2) initializes the premium as the parameter p or 1.1, whichever is higher. (That is, rebate is at least 10% more.)

`double getPremium() const;`

Returns the rebate premium of the diamond customer.

`void setPremium(double p);`

Sets the premium of the diamond customer to the parameter p or 1.1, whichever is higher.

`virtual double rebate() const;`

This member function *overrides* the one in its superclass. It returns the rebate of the purchase of the diamond customer, which is that of a gold customer multiplied by the premium.

`virtual double shippingFee() const;`

This member function *overrides* the one in its superclass. It returns the shipping fee of the purchase of the diamond customer, which is that of a gold customer but with a $30 markdown. However, if the fee becomes negative after markdown, then the member function simply returns 0.0.

Client Program (`rebate.cpp`)

Your main program is a client of the three classes `Customer`, `GoldCustomer`, and `DiamondCustomer` with the following flow.

1. Prompt the user to enter a string, which is the name of the data file to be read from. (Use `getline(…)`, not `cin >> …`, because a file name can contain spaces.)
2. Then, use the file name to open the data file and read the customers purchase data for the online shopping platform. You can assume that the data file has the following file format:
   - The first line contains a positive integer n denoting the total number of customer purchases of the platform in a day's business.
   - Subsequently, there are n lines of data. Each line of data denotes the data of one customer purchase, and contains five to seven space-delimited fields, depending on the customer type.
   - The first field denotes the customer type, which must be either "SC", "GC", or "DC", denoting standard customers, gold customers, or diamond customers respectively. The second field is the customer ID. The third field is the customer name. You can assume that a customer name does not contain spaces.
   - All subsequent fields on a line are then always numbers. The fourth and fifth fields are the purchase amount and shipping weight of the purchase respectively.
   - The sixth field exists for gold and diamond customers only, denoting the shipping discount.
   - The seventh field exists for diamond customers only, denoting the rebate premium.

   Figure 1 shows a sample data file. You can see more sample data files in Blackboard.

```
5
SC 467093946    ChanTaiMan  312.8  0.6
GC 5205201314   LamYauChing 532.7  1.2  0.15
DC 25277177     LeeManHo    354.0  0.8  0.2  1.15
DC 25266366     WongWaiWang 99.3   9.2  0.1  1.2
GC 123456789012 HoTaiWan    2597.3 17.9 0.3
```

Figure 1: Sample Data File `customers1.txt`

3. In case the data file cannot be opened successfully, print a warning message *"Cannot open data file. Program exit!"* and then terminate program execution with the function call `exit(1)`.
4. Create an vector of n pointers to (super)class `Customer` as follows:
   ```
   vector<Customer *> customers(n);     // Vector of superclass pointers
   ```
5. Read the data from the data file. Create objects of `Customer`, `GoldCustomer`, or `DiamondCustomer` accordingly, and set up the pointers `customers[`i`]` to aim at these objects one by one. To create the objects and set up the pointers, you can write either one of the followings:
   - `customers[`i`] = new Customer(…);`

- `customers[i] = new GoldCustomer(…);`
- `customers[i] = new DiamondCustomer(…);`

  Note that (1) the constructors are responsible for handling the case where a customer ID is not 10-digit long; and (2) the `new` operator returns a pointer which can be assigned to `customers[i]`.

6. Set the customer name, purchase amount, shipping weight, shipping discount (if applicable), and rebate premium (if applicable) for the objects accordingly. The `set`*`XXX`*`(…)` member functions are responsible for handling the range validity checking.
7. Then, the program should call the `virtual` member functions `rebate()` and `shippingFee()` of the objects to compute and print out the rebate and shipping fee of all the customers in the following format:
   - There are $n + 1$ lines of output, where n is the number of customer purchases. The first line is a header line and the subsequent n lines are the rebate and shipping fee of the n customers.
   - In each of the subsequent n lines, there are three fields separated by spaces. The first field is the 10-digit customer ID. The second and third fields are the rebate amount and shipping fee of the customer respectively, printed in *two decimal places*. (You can use `cout << fixed << setprecision(2);` to format numbers to two decimal places.)

Program Output

The following shows some sample program output given some sample data files which can be found in Blackboard. The blue text is user input and the other text is the program output. You can try the provided sample program with some self-generated data files as well. *Your program output should be exactly the same as the sample program* (i.e., same text, same symbols, same letter case, same number of spaces, etc.). Otherwise, it will be considered as *wrong*, even if you have computed the correct result. Note that *all printout should be printed from the client program* (`rebate.cpp`), not the classes.

```
Enter file name: customers1.txt↵
Process 5 customer(s):
0467093946 3.13 40.00
5205201314 5.56 0.00
0025277177 4.07 2.00
0025266366 1.19 24.00
3456789012 48.34 28.00
```

Updated on 26 Nov!

```
Enter file name: customers2.txt↵
Process 10 customer(s):
9855171152 17.56 0.00
1676993123 11.23 18.80
2779950282 2.65 13.60
0068446649 11.81 0.00
9076154209 4.51 0.00
1200146177 30.61 0.00
8263825098 9.42 0.00
```

Updated on 26 Nov!

```
0691439088 8.09 0.00
9109889533 6.00 0.00
0869722898 3.16 60.00
```

Submission and Marking

- Your program file names should be `Customer.cpp`, `GoldCustomer.cpp`, `DiamondCustomer.cpp`, and `rebate.cpp`. Submit the four files in Blackboard (https://blackboard.cuhk.edu.hk/). You do not have to submit the `.h` files.
- Insert your name, student ID, and e-mail address as comments at the top of all your source files.
- Besides the above, your program should further *include suitable comments as documentation*.
- You can submit your assignment multiple times. Only the latest submission counts.
- Your program should be *free of compilation errors and warnings*.
- *Plagiarism is strictly monitored and heavily punished if proven.* Lending your work to others is subjected to the same penalty as the copier.