Assignment 5: Computing Income

Due: 23:59, Sat 10 Dec 2016 Full marks: 100

Introduction

In a company, there are three different types of staffs and they earn their income differently. *Normal staffs* are permanent staffs who earn monthly salaries. *Contract staffs* are employed on one-year contracts from January to December. They earn monthly salaries, as well as a contract-end gratuity which is a percentage (the gratuity rate) of their total monthly salaries. *Part-time staffs* are like contract staffs, but they enjoy different benefits. One such benefit is the transport subsidy scheme. A normal staff is eligible to apply for transport subsidy if his/her annual income is less than $125,000; a contract staff is eligible if his/her annual income (including the gratuity) is less than $130,000; and a part-time staff is eligible if his/her lowest monthly salary in the year is less than $8,300. In this assignment, you will apply inheritance and polymorphism to write three classes `Staff`, `ContractStaff`, and `PartTimeStaff` to model normal staffs, contract staffs, and part-time staffs respectively, and a client program to read the salaries and gratuity rates (if any) of the staffs from a data file and determine whether each staff is eligible to apply for transport subsidy.

Program Specification

You have to write your program in four source files (1) `Staff.cpp`, (2) `ContractStaff.cpp`, (3) `PartTimeStaff.cpp`, and (4) `calcIncome.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 `Staff.h`, `ContractStaff.h`, and `PartTimeStaff.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 Staff (Staff.cpp)

The `Staff` class models normal staffs. Its interface is given as follows.

```
class Staff {
public:
    Staff(string sid = "00000000");
    string getID() const;
    double getSalary(int m) const;
    void setSalary(int m, double wage);
    virtual double annualIncome() const;
    virtual bool transportSubsidy() const;
private:
    string id;
    double salary[12];
};
```

Private Data Members

`string id;`
The staff ID of the staff, which is a string containing eight digits.

`double salary[12];`
An array of `double` for storing the monthly salaries from January to December of the staff. Salary of January is stored in `salary[0]`; salary of February is stored in `salary[1]`; etc.

Public Constructor and Member Functions

`Staff(string sid = "00000000");`
The parameter `sid` is supposedly the staff ID for initializing the data member `id`. However, `sid` 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 (< 8), add enough '0's to the front to make it eight-digit. If it has more than eight digits, trim it to retain only the *rightmost* eight digits. (E.g., suppose `sid` is "dq689a358sdc964569", then `id` should be initialized as "58964569".) Besides, the salaries of all 12 months should be initialized as 0.0.

`string getID() const;`
Returns the staff ID of the staff.

`double getSalary(int m) const;`
Returns the salary for month `m` of the staff. You can assume that parameter `m` is always 1–12, meaning January–December.

`void setSalary(int m, double wage);`
Sets the salary for month `m` to the parameter `wage`. However, if `wage` is less than 0, then set the salary for that month to 0. If parameter `m` is not 1–12 (January–December), then this member function does not set anything.

`virtual double annualIncome() const;`
Returns the annual income of the staff, which is the sum of his/her salaries from January to December. For example, suppose the staff earns $12000 from January to July, and $13500 from August to December, then the member function should return 151500.0 ($= 12000 \times 7 + 13500 \times 5$).

`virtual bool transportSubsidy() const;`
Returns `true` if the staff is eligible to apply transport subsidy, i.e., the annual income of the staff is less than 125000.0; returns `false` otherwise.

Class `ContractStaff` (`ContractStaff.cpp`)
The `ContractStaff` class models contract staffs and is a subclass of `Staff`. Its interface is given as follows.

```
class ContractStaff : public Staff {
public:
    ContractStaff(string sid = "00000000", double gr = 0.0);
    double getGratuityRate() const;
    void setGratuityRate(double gr);
    virtual double annualIncome() const;
    virtual bool transportSubsidy() const;
private:
    double gratuityRate;
};
```

Private Data Members

`double gratuityRate;`
The gratuity rate of the contract staff.

Public Constructor and Member Functions

`ContractStaff(string sid = "00000000", double gr = 0.0);`
The parameter `sid` is supposedly the staff ID of the contract staff. This constructor (1) calls the constructor of its superclass `Staff` and (2) initializes the gratuity rate as the parameter `gr` or 0.0, whichever is higher.

`double getGratuityRate() const;`
Returns the gratuity rate of the contract staff.

`void setGratuityRate(double gr);`
Sets the gratuity rate of the contract staff to the parameter `gr`. However, if `gr` is less than 0.0, then set the gratuity rate to 0.0.

`virtual double annualIncome() const;`
This member function *overrides* the one in its superclass. It returns the annual income of the contract staff, which is the sum of his/her salary from January to December plus a percentage (the gratuity rate) of this sum. For example, suppose the contract staff earns $11000.0 from January to May, and $12400.0 from June to December, and has an 8% gratuity rate (0.08), then the member function should return 153144.0 ($= (11000 \times 5 + 12400 \times 7) \times 1.08$).

`virtual bool transportSubsidy() const;`
This member function *overrides* the one in its superclass. It returns `true` if the contract staff is eligible to apply transport subsidy, i.e., the annual income of the contract staff is less than 130000.0; and returns `false` otherwise.

Class `PartTimeStaff` (`PartTimeStaff.cpp`)

The `PartTimeStaff` class models part-time staffs and is a subclass of `ContractStaff`. Its interface is given as follows.

```
class PartTimeStaff : public ContractStaff {
public:
    PartTimeStaff(string sid = "00000000", double gr = 0.0);
    virtual bool transportSubsidy() const;
};
```

Public Constructor and Member Functions

`PartTimeStaff(string sid = "00000000", double gr = 0.0);`

The parameter `sid` is supposedly the ID of the part-time staff. This constructor simply calls the constructor of its superclass `ContractStaff`.

`virtual bool transportSubsidy() const;`

This member function *overrides* the one in its superclass. It returns `true` if the part-time staff is eligible to apply transport subsidy, i.e., the lowest monthly salary of the part-time staff in the year is less than 8300.0; and returns `false` otherwise.

Client Program (`calcIncome.cpp`)

Your main program is a client of the three classes `Staff`, `ContractStaff`, and `PartTimetaff` 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 >> …;`.)
2. Then, use the file name to open the data file and read the staffs' income data for the company. 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 staffs in the company.
   - Subsequently, there are n lines of data. Each line of data denotes the data of one staff, and contains 14 or 15 space-delimited fields, depending on the type of the staff.
   - The first field denotes the staff type, which must be either "NS", "CS", or "PT", denoting normal staffs, contract staffs, or part-time staffs respectively. The second field is the staff ID.
   - The third to 14th fields are the monthly salaries from January to December. The 15th field exists for contract staffs only (including part-time), denoting the gratuity rate. These fields must always be numbers (can be non-integers).

   You can see some sample data files in Blackboard.
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 `Staff` as follows:
   ```
       vector<Staff *> staffs(n);  // Vector of superclass pointers
   ```
5. Read the data from the data file. Create objects of `Staff`, `ContractStaff`, or `PartTimeStaff` accordingly, and set up the pointers `staffs[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:
   - `staffs[i] = new Staff(…);`
   - `staffs[i] = new ContractStaff(…);`
   - `staffs[i] = new PartTimeStaff(…);`

   Note that (1) the constructors are responsible for handling the case where a staff ID is not 8-digit long; and (2) the `new` operator returns a pointer which can be assigned to `staffs[i]`.

6. Set the January–December monthly salaries and gratuity rate (if applicable) for the objects accordingly. The `setSalary(…)` and `setGratuityRate(…)` member functions are responsible for handling the range validity checking.
7. Then, the program should call the `virtual` member functions `annualIncome()` and `transportSubsidy()` of the objects to compute and print out the annual income and transport subsidy eligibility of all the staffs in the following format:
   - There are $n + 1$ lines of output, where n is the number of staffs. The first line is a header line and the subsequent n lines are the annual income and eligibility of the n staffs.
   - In each of the subsequent n lines, there are three fields separated by spaces. The first field is the 8-digit staff ID. The second field is the annual income of the staff in *two decimal places*. The third field is either "Eligible" or "Ineligible", denoting whether the staff is eligible to apply transport subsidy. You can use `cout << fixed << setprecision(2);` to control the printing of the annual income in two decimal places.

Program Output

The following shows some sample program output given some sample data files which can be found in Blackboard. The **bold blue** text is user input and the other text is the program output. You can try the provided sample program with some other 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* (`calcIncome.cpp`), not the classes.

```
Enter file name: allstaffs1.txt↵
Annual Income of 5 staff(s):
46709394 220206.80 Ineligible
05201314 91585.07 Eligible
25277177 123331.88 Eligible
25266366 125019.90 Ineligible
34567890 98692.90 Eligible
```

```
Enter file name: allstaffs2.txt↵
Annual Income of 10 staff(s):
92932838 135353.13 Eligible
02976121 132547.81 Ineligible
35790796 133746.12 Eligible
00088636 128233.70 Ineligible
00252126 121108.30 Eligible
00013165 119927.64 Ineligible
00020970 146431.09 Ineligible
00066296 127865.10 Ineligible
00036831 128472.05 Eligible
00005090 133284.80 Ineligible
```

Submission and Marking

- Your program file names should be `Staff.cpp`, `ContractStaff.cpp`, `PartTimeStaff.cpp`, and `calcIncome.cpp`. Submit the four files in Blackboard (https://elearn.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 beginning of all your source files.
- Besides the above information, 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.