Object-Oriented Programming

Summer Semester 2023

Homework Assignment 1.

17/07/2023

Contents

Introduction

1. I strongly recommend you work on this homework assignment as soon as possible, due to reasons that are related to familiarity with the programming environment:
   - Make sure your IDE is running on your Computer.
   - Make sure you can compile C/C++ codes.
2. Try to provide clear and careful solutions.
3. You should provide comments for your code, so it will be completely clear what you are trying to achieve.

Important notice:

Some changes might be published in the first days from the HW publication.

Following the updates and HW moodle forum is required.

Definitions

In your first homework, you will be introduced to the most common concepts of working with C++ and OOP paradigm. You are going to implement part of a system that helps to manage a company. Part of this system design includes the following basic models: Manager, Shop and Company. In the next sections, you will find specifications for these classes.

Your goals are:

- Given class definitions, write C++ classes such that they conform to the requirements.
- Use const, references where it's right to do so.
- Given class definitions, distinguish private properties from public properties.
- Having method descriptions, you should provide implementation while keeping in mind basic OOP concept of *encapsulation*.
- Keep in mind that the included main files are very basic and do not cover all possible scenarios and end cases.
- Separate implementation from declaration code using the convention: implementation in "*.cpp" files, and declaration in "*.h files".
- Please note the following few points which may lead to points reduction during the submission check:
    - Avoid code duplication as much as possible.
    - You should not globally enable *std* namespace usage or any other namespaces, e.g. *using namespace std;*.
    - Avoid using magic numbers

Manager

The class “Manager” has the following attributes:

- Id – int.
    - Field name : id
- First Name – string with a maximum of 10 characters.
    - Field name : firstName
- Last Name - string with a maximum of 10 characters.
    - Field name : lastName
- Year of birth– int.
    - Field name : birthYear

And the following methods:

- Construction – should initialize all the fields in the class – by default: numerical values to 0, strings to “~”.
- Getters – for each field in the class you should provide a “get” method, which returns the demanded attribute.

  Setters – for each field in the class you should provide a “set” method, which sets a given value in the required attribute. (If input to long then output the massage "Manager first/last name length is too long" and set the default value, Refer to required output from the main file for example of exact required text.)
- **`print()`** – a method that prints the manager’s details, each attribute in new line in the following form:

  <attribute1_name>: <attribute1_value>

  <attribute2_name>: <attribute2_value>

  ..

  Refer to required output from the main file for example of exact required text.

Shop

A class called “Shop” has the following attributes:

- Manager
    - Field name : manager
- Shop name – string with a maximum of 10 characters.
    - Field name : name
- Closed day – enum of "Day".
    - Field name : closedDay
- Is online – bool
    - Field name : isOnline

And the following methods:

- Construction – should initialize all the fields in the class – by default: numerical values to 0, strings to “~”.
- Getters – for each field in the class you should provide a “get” method, which returns the demanded attribute.
- Setters – for each field in the class you should provide a “set” method, which sets a given value in the required attribute. (If input to long then output the massage "Shop name length is too long" and set the default values, Refer to required output from the main file for example of exact required text)
- **`print()`** – a method that prints the shop’s details, each attribute in new line in the following form:

  `*shop’s manager details*`

  <attribute1_name>: <attribute1_value>

  <attribute2_name>: <attribute2_value>

  ..

Company

A class called “Company” has the following attributes:

- Name – the name of the company, string with a maximum of 10 characters.
  - Field name : name
- Shops – a finite array that contains limited number of Shops.
  - use `#define MAX_SHOPS_NUMBER 10`
  - Field name : shops
- Shops number – current number of Shops in the Company.
  - Field name : shopsNumber

And the following methods:

- Construction – gets as a parameter the name of the Company, and should initialize all the fields in the class – by default: numerical values to 0, strings to “~”, booleans to false.
- Getters – for each field in the class you should provide a “get” method, which returns the demanded attribute.
- Setters –you should provide a “set” method, which sets a given value in the attribute Name.

  (If input to long then output the massage "Company name length is too long" and set the default value ,Refer to required output from the main file for example of exact required text)
- **`void addShop(…)`** - add a shop to company’s shops list. If the company is full, the method should print “There is no place for new shops! ". (Refer to required output from the main file for example of exact required text. No need to support removal of shops from the company).

- **`void printShopsByName()`** – a method that prints the details of the shops in the company in an increasing order of their lexicographic shop's name.
- **`void printShopsByDay()`** – a method that prints the details of the shops in the company in an increasing order of their closed day.

*Both prints should print all the shops in the company sorted by the selected attribute in the following form:

```
There are < shopsNumber> shops in the company. The shops:
Shop #1:
        *shop 1 details*
Shop #2:
        *shop 2 details
        ...
```

Refer to required output from the main file for example of exact required text.

Note#1: Use the exact signature for the methods detailed above.

Note#2: (For this exercise) Booleans should be printed as "True" if 1, and "False" if 0.

Evaluation

Homework exercise provided with the following example program file and corresponding output:

1. main.cpp
2. output.txt

You should be able to compile your code with “main.cpp” file and receive correct output, which placed in “output.txt” file.

*Please note that line 15 in the main.cpp file is commented out.

The function “freopen("output.txt", "w", stdout)” saves the program output to a text file named “output.txt” in the solution directory, so you can uncomment this line in order to compare your output to mine. (I recommend comparing the outputs via DiffMerge , Link: https://sourcegear.com/diffmerge/)

In order to use the “freopen” function please follow the following video tutorial:

How to handle Error C4996 in Visual Studio

Link: https://www.youtube.com/watch?app=desktop&v=qWxGZLjwKL0

Make sure you keep the C++ syntax convention as described in “Oop – Cpp – Conventions and Requirements” file in the Moodle.

GOOD LUCK! ☺