Program #5
Due: Tuesday, November 15th, 2016 at 11:30PM

Instructor	Dr. Stephen Perkins	
Office Location	ECSS 4.702	
Office Phone	(972) 883-3891	
Email Address	stephen.perkins@utdallas.edu	
Office Hours	Tuesday and Thursday 10:30am – 11:30am Tuesday and Thursday 1:00pm – 2:15pm and by appointment	
Grader	Section 502:	Sai Vamsi Muvva sxm154231@utdallas.edu Open Lab 2.103B1
	Section 504:	Gopichand Vanka gxv151030@utdallas.edu Open Lab 2.104A1 Tuesday/Thursday 3:00pm – 5:00pm

Purpose

Demonstrate the ability to extend and use an abstract class. Demonstrate the advanced feature of implementing a class that uses the *Singleton* design pattern. Demonstrate the advanced feature of implementing a class that uses the Factory Method design pattern. Exercise your new class with a driver program.

Assignment

We will be creating a useful logging class. This class can be used as a means of advanced logging in other C++ programs. It will use a *factory method design pattern* such that debugging can be implemented with different classes but the main body of code will not know which class it is using. It will use *a singleton design pattern* that will allow us to get a logging object from within whatever function requires logging output.

You are to create an abstract class named *CS1337Logger*. This class must contain a private boolean variable named *loggingEnabled*. You need to define a setter method to set its value. The constructor for the class should initialize *loggingEnabled* to false;

You are to create a **pure virtual function** called *displayMessage* that takes a *const char** as an argument. This pure virtual function makes the class *CS1337Logger* an abstract class.

You are to create a *logMessage* function that will accept a *const char** message as an argument. This method should then check the value of *loggingEnabled*. If it is true, then you should call the pure virtual function *displayMessage* and pass it the *const char **.

You must then create two derived classes.

The first is called *ScreenLogger*. It must override *displayMessage* with the result that the message is displayed to the screen.

The second is called *FileLogger*. It must override *displayMessage* with the result that the message is displayed BOTH to the screen and to a log file with a fixed name of "Log.txt". The constructor must open the output file. The singleton pattern will ensure that the open is only called once.

You must then create a Class called *LogSingleton*. This class should implement the Singleton design pattern and should create and return a single instance of either the *ScreenLogger* or the *FileLogger*. The Class should have a private constructor and a static method named *GetLogger*(). You will hard code which logging class it will use (*ScreenLogger*, *FileLogger*). A single change in your code will completely change the behavior.

Once your code is implemented, you must create a main routine that demonstrates the use of the log object returned by the Singleton, enables and disables logging, and displays messages to both files and the screen. Switching which class the factory method returns will require a recompile and new run of your program. You should create a few functions that do not do anything other than log messages. You might, for instance, log the entry into functions and the exit from those functions. Call these functions from main or within themselves. Some of the functions should get the logger instance from the Singleton and display log messages. You DO NOT NEED to pass the logger into each function. The goal is to exercise the logger and show that you can turn logging off and on in various sections of the code.

Requirements

- Your code must extend and use the described abstract debugging class.
- Your code must exhibit the use of the Singleton design pattern
- Your code must exhibit the use of the factory method design pattern.

Deliverables

You must submit your homework through ELearning. You must include your .h and .cpp files.

Notes

No late homework is accepted.