Program Design（25%）

Programs use log libraries to write logs. Suppose we are writing a very simple log library that sends log message both to a log file and the console. To write logs to file, we already have implemented a class FileWriter:

class FileWriter

{ public:

static void writeStringToFile(const char \*strMessage);

};

To display logs in console, we already have implemented a class ConsoleDisplayer:

class ConsoleDisplayer

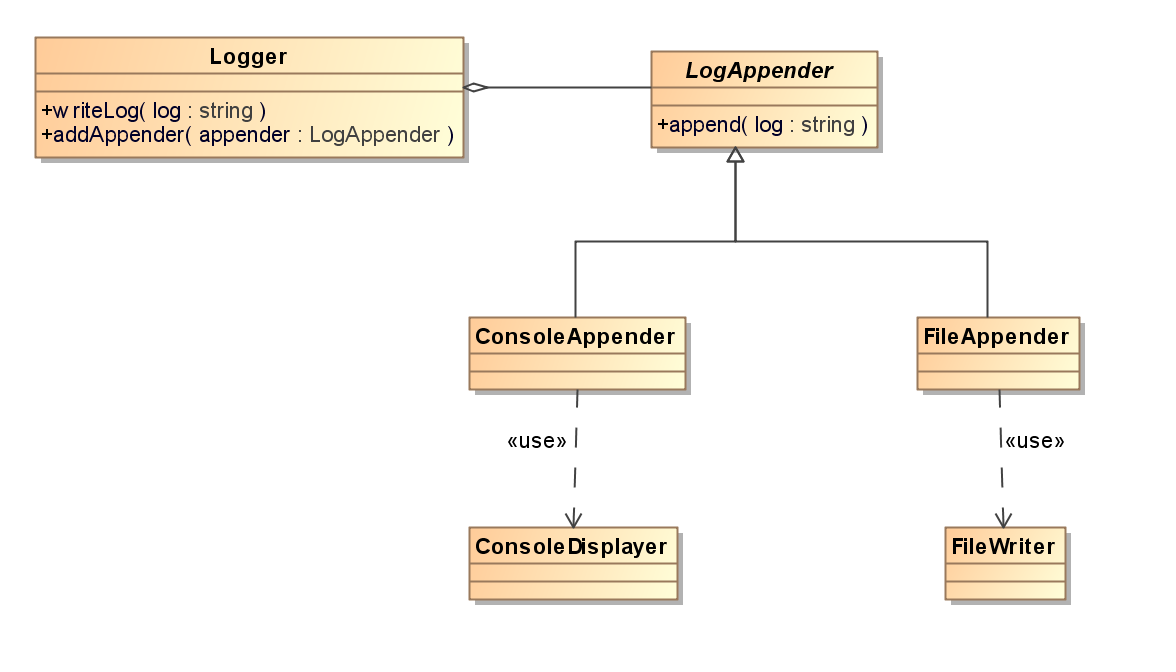
{

public:

static void displayString(const char \*strMessage);

};

Our log library contains four classes: Logger, LogAppender, FileAppender, and ConsoleAppender.



The Logger is used to write logs. The LogAppender is an *abstract* class which shall write log messages to files or consoles. The ConsoleAppender inherits from LogAppender and displays log messages in console using ConsoleDisplayer. The FileAppender also inherits from LogAppender but writes log messages to files using FileWriter. The Logger can attach several LogAppenders to record log messages to different targets at the same time.

Logger is declared as:

class Logger

{

public:

Logger() : m\_count(0){}

void addAppender(LogAppender \*appender);

void writeLog(const char \*log);

private:

LogAppender \*m\_appenders[256];

int m\_count;

};

LogAppender is declared as

class LogAppender

{

public:

virtual void writeLog(const char \*log) = 0;

};

The log library is supposed to be used as the following example:

Logger logger;

logger.addAppender(new ConsoleAppender());

logger.addAppender(new FileAppender());

logger.write("Hello world.");

Please finish the log library:

1. write the unimplemented methods of class Logger.

2. write the declaration and implementation of class FileAppender.

3. write the declaration and implementation of class ConsoleAppender.