ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №9  
по предмету: Разработка программных модулей

09.02.07 Информационные системы и программирование

Выполнил:

студент группы Y2437

\_\_\_\_\_\_\_\_\_\_\_Князев А. А.

Проверил:

\_\_\_\_\_\_\_\_ Ржевский В. А.

Дата: «\_\_» \_\_\_\_\_ 20\_\_ г.

Оценка: \_\_\_\_\_\_\_\_\_\_

УНИВЕРСИТЕТ ИТМО

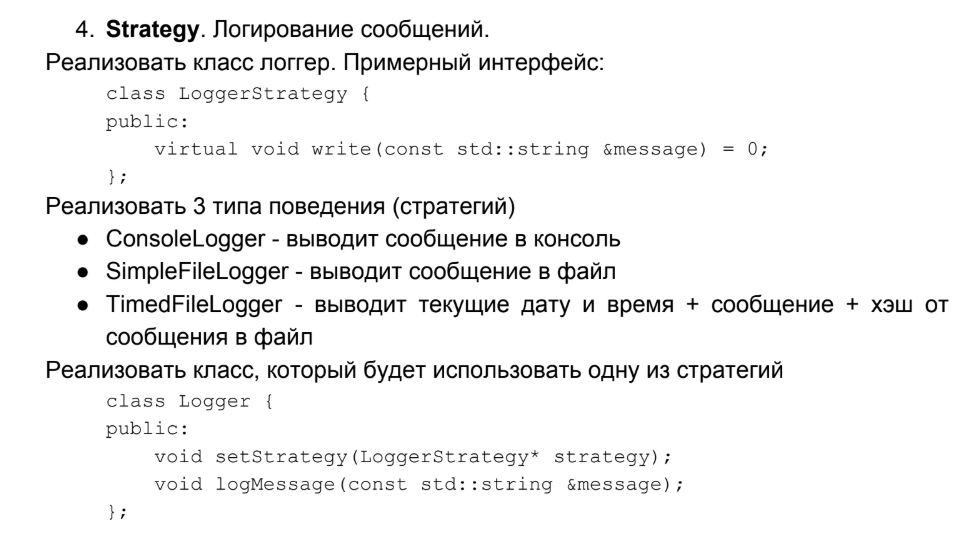
Факультет среднего профессионального образования

Санкт-Петербург 2019

1. Общее задание

Требуется решить задачу на тему реализации поведенческих паттернов (стратегии в данном случае).

1. Индивидуальное задание



Код задания №1:

Main.cpp:

#include "Loggers.hpp"

int main()

{

std::ofstream myfile;

myfile.open ("example.txt", std::ios::out);

std::ofstream myfile2;

myfile2.open ("example2.txt", std::ios::out);

Logger \*loggers[3] =

{

new ConsoleLogger(),

new SimpleFileLogger(myfile), // file cerr

new TimedFileLogger(myfile2) // file cerr

};

for (int i = 0; i < 10; i++)

{

(loggers[0])->log("hello, std::cout");

(loggers[1])->log("hello, file");

(loggers[2])->log("hello, file with time");

}

myfile.close();

myfile2.close();

}

Loggers.hpp:

#pragma once

#include <iostream>

#include <fstream>

#include <string>

class Logger

{

protected:

Logger() = default;

public:

virtual ~Logger() = default;

virtual void log(const std::string &msg) = 0;

};

class ConsoleLogger : public Logger

{

public:

ConsoleLogger() = default;

void log(const std::string &msg) override;

};

class SimpleFileLogger : public Logger

{

private:

std::ofstream &os;

public:

SimpleFileLogger(std::ofstream &f)

: os(f) { }

void log(const std::string &msg) override;

};

class TimedFileLogger : public Logger

{

private:

std::ofstream &os;

public:

TimedFileLogger(std::ofstream &f)

:os(f) { }

void log(const std::string &msg) override;

};

Loggers.cpp:

#include "Loggers.hpp"

#include <chrono>

#include <ctime>

void ConsoleLogger::log(const std::string &msg)

{

std::cout << msg << std::endl;

return;

}

void SimpleFileLogger::log(const std::string &msg)

{

os << msg << std::endl;

return;

}

void TimedFileLogger::log(const std::string &msg)

{

auto start = std::chrono::system\_clock::now();

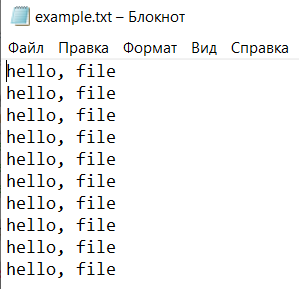
std::time\_t end\_time = std::chrono::system\_clock::to\_time\_t(start);

os << "time: " << std::ctime(&end\_time) << "message: "

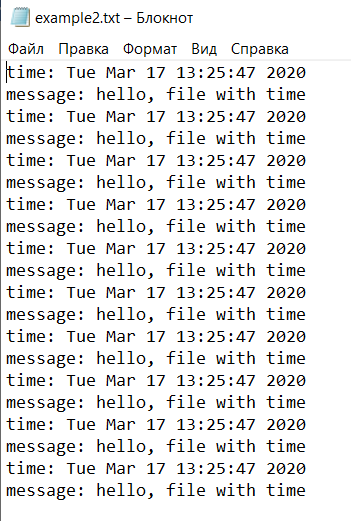
<< msg << std::endl;

return;

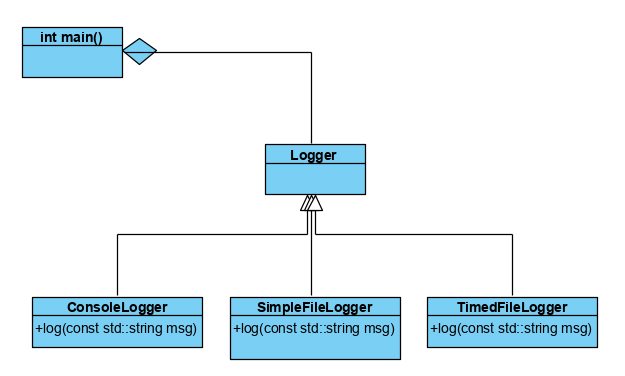
}



*Рисунок 1. Исходный файл с текстом.*



*Рисунок 2. Выходной файл с текстом.*



*Рисунок 3. Uml диаграмма классов реализации поведенческого паттерна*