МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Объектно-ориентированное программирование»
Тема: Логирование

Студент гр. 9382 Демин В.В.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Создать логирования для элементов поля, использую паттерны наблюдатель и мост.

Задание.

Создан набор классов, которые отслеживают игрока и элементы на поле, и выводят/сохраняют информацию об их изменениях.

Обязательные требования:

- Реализована возможность записи логов в терминал и/или файл
- Взаимодействие с файлом реализовано по идиоме RAII
- Перегружен оператор вывода в поток для всех классов, которые должны быть логированы

Дополнительные требования:

- Классы, которые отслеживают элементы, реализованы через паттерн Наблюдатель
- Разделение интерфейса и реализации класса логирования через паттерн Мост

Выполнение работы.

В ходе выполнения работы был использован паттерн Наблюдатель для реализации логирования элементов поля. В данном случае субъектом являются элементы, которые с помощью функции notify() сообщают о своем состоянии Наблюдателю,классу логирования. Которая выводит данные либо в консоль, либо в файл.

Вся реализация логирования вынесена в интерфейс LoggerImpl, которая в свою очередь разделяется на логирование в файл и консоль. Это реализовано с помощью паттерна Мост.

Взаимодействие с файлом реализовано по идеоме RAII.

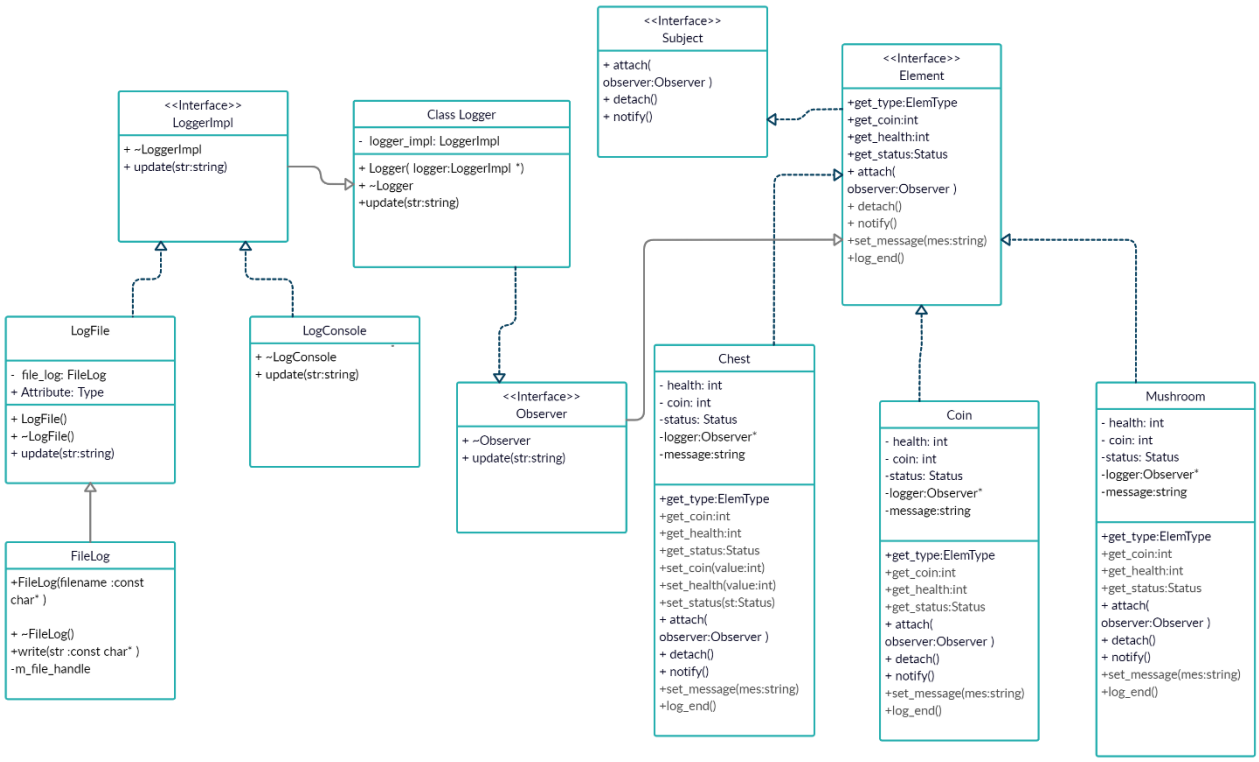

Рисунок 1. Uml диаграмма

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 2	Player(Cord:(0, 5), Health:(1), Coins(0)):time(22:42:5) |_____| | | | O | | | | | + $ | | | | | | |_____| Health:1 Coins:0 You need coins to win:1	Выбирается режим отрисовки клеток(1- в консоле). Выбирается рижим логирования

Выводы.

В ходе выполнения лабораторной работы был создан класс логирования о состоянии игрока и элементов поля на поле.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Observer.h

```
//
// Created by vikto on 01.11.2020.
//

#ifndef MYGAME_OBSERVER_H
#define MYGAME_OBSERVER_H
#include <iostream>

class Observer{
public:
    virtual ~Observer() {};
    virtual void update(std::string str) const=0;
};


#endif //MYGAME_OBSERVER_H
```

Название файла: Cell.cpp

```
//
// Created by vikto on 13.09.2020.
//

#include "Subject.h"

//
// Created by vikto on 01.11.2020.
//

#ifndef MYGAME_SUBJECT_H
#define MYGAME_SUBJECT_H

#include "Observer.h"

class Subject {
public:

    virtual void attach(Observer *observer)  = 0;

    virtual void detach()  = 0;

    virtual void notify()  = 0;
};
```

```
#endif //MYGAME_SUBJECT_H
```
Название файла: Logger.h
```
//
// Created by vikto on 02.11.2020.
//

#ifndef MYGAME_LOGGER_H
#define MYGAME_LOGGER_H


#include "LoggerImpl.h"
#include "../Observer/Observer.h"


class Logger : public Observer {

    LoggerImpl *logger_impl = nullptr;
public:

    Logger(LoggerImpl *logger);

    ~Logger() override;

    void update(std::string str) const override;
};


#endif //MYGAME_LOGGER_H
```
Название файла: Feild.cpp
```
//
// Created by vikto on 13.09.2020.
//

#include "LoggerImpl.h"

//
// Created by vikto on 02.11.2020.
//

#ifndef MYGAME_LOGGERIMPL_H
#define MYGAME_LOGGERIMPL_H
#include <iostream>
#include <ctime>
```

```
class LoggerImpl {
public:
    virtual  ~LoggerImpl() {  };
    virtual void update(std::string str)=0;
};


#endif //MYGAME_LOGGERIMPL_H
```
Название файла: Iterator.h
```
//
// Created by vikto on 13.09.2020.
//

#ifndef MYGAME_ITERATOR_H
#define MYGAME_ITERATOR_H
#include "../../Game/Field/Cell/Cell.h"


template <class U>
class Iterator {
public:
    virtual U *Next()=0;
    virtual U *begin()=0;
    virtual U *getElem()=0;
    virtual U *Prev()=0;
    virtual bool hasMore()=0;
};


class CellIterator: public Iterator<Cell> {
//методы для iteratinga
    Cell* temp;
    Cell* head;
    public:
    CellIterator(Cell &temp);
    Cell * getElem() override;
    Cell* begin() override;
    Cell* Next() override;
    Cell* Prev() override;
    bool hasMore() override;
};


#endif //MYGAME_ITERATOR_H
```

Название файла: LogConsole.h

```
//
// Created by vikto on 01.11.2020.
//

#ifndef MYGAME_LOGCONSOLE_H
#define MYGAME_LOGCONSOLE_H

#include "LoggerImpl.h"

class LogConsole : public LoggerImpl {
public:
    ~LogConsole() override;
    void update(std::string str) override;
};


#endif //MYGAME_LOGCONSOLE_H
```

Название файла: LogFile.h

```
//
// Created by vikto on 01.11.2020.
//

#ifndef MYGAME_LOGFILE_H
#define MYGAME_LOGFILE_H

#include "LoggerImpl.h"
#include <direct.h>
class FileLog;
class LogFile : public LoggerImpl {
   FileLog *file_log;
public:
   LogFile();
   ~LogFile() override;
   void update(std::string str) override;
};

class FileLog {
public:
   FileLog(const char* filename ) : m_file_handle(std::fopen(filename, "w+"))
   {
      if( !m_file_handle )
         throw std::runtime_error("file open failure") ;
   }
   ~FileLog()
   {
      if( std::fclose(m_file_handle) != 0 )
      {
```

```
        }
    }

    void write( const char* str )
    {
        if(std::fputs(str,m_file_handle) == EOF )
            throw std::runtime_error("file write failure") ;
    }

private:
    std::FILE* m_file_handle ;
    FileLog(const FileLog & ) ;
    FileLog & operator=(const FileLog & ) ;
};
#endif //MYGAME_LOGFILE_
```