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, перегрузка операций

Студент гр. 1383 _____ Куликов М.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

Цель работы.

Реализовать класс/набор классов отслеживающих изменения состояний в программе. Отслеживание должно быть 3-х уровней:

Изменения состояния игрока и поля, а также срабатывание событий

Состояние игры (игра начата, завершена, сохранена, и.т.д.)

Отслеживание критических состояний и ошибок (поле инициализировано с отрицательными размерами, игрок попытался перейти на непроходимую клетку, и.т.д.)

Реализованы классы для вывода информации разных уровней для в консоль и в файл с перегруженным оператором вывода в поток.

Задание.

Разработан класс/набор классов, отслеживающий изменения разных уровней

Разработаны классы для вывода в консоль и файл с соблюдением идиомы RAII и перегруженным оператором вывода в поток.

Разработанные классы спроектированы таким образом, чтобы можно было добавить новый формат вывода без изменения старого кода (например, добавить возможность отправки логов по сети)

Выбор отслеживаемых уровней логирования должен происходить в runtime

В runtime должен выбираться способ вывода логов (нет логирования, в консоль, в файл, в консоль и файл)

Выполнение работы.

Был разработан набор классов, ответственный за логирование. Используя паттерн «наблюдатель», был создан абстрактный класс для наблюдаемых объектов, от которого был унаследован интерфейс событий, класс игрового поля и класс игрока. По такому принципу логгеры могут обращаться к любому наблюдаемому объекту, вне зависимости от его типа.

Observed_Obj – Класс наблюдаемых объектов, содержащий в себе одно поле – флаг для обозначения того, что объект был изменен в течение текущего шага и это необходимо отобразить в логах.

Методы класса:

1)void notify() – устанавливает флаг изменения в 1 для последующего использования.

2)void set_notify_flag(int num) – позволяет установить флаг изменения в нужное значение.

3)int get_notify_flag() – позволяет получить флаг изменения текущего объекта.

4)Перегруженный оператор вывода в поток, который выводит в поток лог объекта из метода get_log(), описанного далее.

5) virtual std::string get_log() – виртуальный метод для получения сообщения для лога. Для каждого дочернего класса наблюдаемых объектов определяется по разному, в зависимости от того, какую информацию нужно получить. (Для игрока – его характеристики, для поля – исключительные ситуации или изменение положения игрока, для события – его срабатывание и его характеристики)

Observer – Класс – наблюдатель, который хранит в себе 2 поля, ответственные за хранение логгеров (поля типа интерфейса) и 1 поле, хранящее игровое поле. Класс имеет единственный метод void update(), в котором все события на игровом поле, игрок и само поле проверяются на изменение, после чего в выбранные логгеры отправляются измененные объекты, чтобы информация о них была выведена в поток.

ILogger – интерфейс группы логгеров, содержащий в себе единственный метод virtual void output_logs(Observed_Obj* obj), который выполняет запись информации в поток логгера.

Console_Log – класс, ответственный за вывод логов в консоль. Имеет в себе только 1 перегруженный метод output_logs, который выводит информацию о полученном объекте в std:cout.

File_log – класс, ответственный за вывод логов в файл. Имеет в себе поле типа std::ofstream, для хранения потока(файла). Из методов содержит в себе конструктор, принимающий путь к файлу, деструктор, в котором файл закрывается и перегруженный метод output_logs, который выводит информацию в файл.

Выбор метода логгирования происходит с помощью считывания команды от пользователя в классе Controller и нового метода set_logs(int num), в котором в поле объекта Controller записывается созданный с нужными параметрами объект-наблюдатель. После каждого шага игрока вызывается метод update(), который проверяет изменения и записывает информацию о них.

Тестирование.

На рисунках 1 и 2 показан вывод логов об исключительной ситуации, сработавшем событии и его воздействии на игрока, новых координатах игрока в консоль.

```
-5 -5
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
P F F F F F F F F F

Choose the way of logging. 0 - No logs, 1 - Console logs, 2 -logs in logs.txt logs , 3 - both logs
3
d
```

Рисунок 1 – Выбор размера поля и метода логгирования.

```
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F P F F F F F F F F

Player's frost armor now: 2
The player tried to make a field with wrong size.
Current hero position: [1][0]
Health: 15
Frost armor: 2
Fire armor: 0
Agility: 0
```

Рисунок 2 – Результаты логгирования в консоль после первого шага

На рисунке 3 показаны результаты логгирования в файл после тех же действий, что и на прошлом рисунке.

Рис

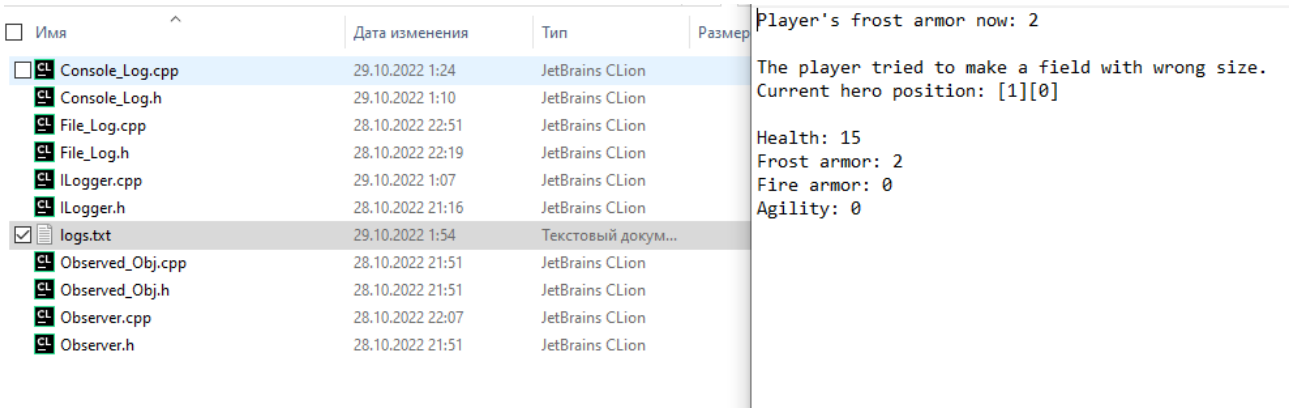

Рисунок 3 – Результаты логгирования в файл.

Выводы.

Была реализован набор классов, который позволил осуществить логгирование разных уровней с возможностью вывода логов в разные потоки.

ПРИЛОЖЕНИЕ А

UML ДИАГРАММА