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. 1384 _____ Алиев Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

Цель работы.

Разработать классы, отслеживающие изменение состояний игры. Перегрузить операторы вывода в поток.

Задание.

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

1. Изменения состояния игрока и поля, а также срабатывание событий
2. Состояние игры (игра начата, завершена, сохранена, и.т.д.)
3. 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.

Требования:

- 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)

Примечания:

- Отслеживаемые сущности не должны ничего знать о сущностях, которые их логируют

- Уровни логирования должны быть заданными отдельными классами или перечислением
- Разные уровни в логах должны помечаться своим префиксом
- Рекомендуется сделать класс сообщения
- Для отслеживания изменений можно использовать наблюдателя
- Для вывода сообщений можно использовать адаптер, прокси и декоратор

Выполнение работы.

Для выполнения данной лабораторной работы был использован паттерн проектирования «Наблюдатель». Таким образом, класс логгера может реагировать на изменения, происходящие в классах.

Разработан класс наблюдаемого и интерфейс наблюдателя. Класс, отвечающий за логирование, реализует интерфейс наблюдателя. Соответственно, сущности, за которыми необходимо вести наблюдение, наследуются от класса наблюдаемого.

В качестве аргумента при логировании принимается класс сообщения, хранящий текст сообщения и уровень логирования.

От абстрактного класса Logger были унаследованы два конкретных класса логирования, ConsoleLogger и FileLogger. Первый класс выводит информацию в консоль, второй – в файл, название которого указывается в конструкторе.

Класс FileLogger был реализован с соблюдением идиомы RAII: при создании объекта класса файл логирования загружается в память, а при удалении объекта – файл закрывается, взаимодействие с содержимым файла производится только вызывая методы класса.

Были перегружены операторы вывода в поток класса сообщения.

Разработаны классы-хранители всевозможных сообщений, возникающих при исполнении программы. Классы PlayerMessages, EventMessages, GameStateMessages хранят сообщения относительно действий игрока, исполнения действий и изменения состояния игры соответственно. Все они наследуются от класса MessageStorage.

Для контроля памяти был создан класс LoggerPool – вспомогательный класс, сохраняющий и удаляющий все логгеры, создаваемые в процессе исполнения программы. Класс был разработан с применением паттерна «Одиночка», таким образом из любого места в программе можно получить объект класса LoggerPool и добавить в него класс логирования. При вызове деструктора у LoggerPool, удаляются все хранящиеся в нём логгеры, что позволяет очистить память и закрыть файлы, которые были открыты при создании объектов класса FileLogger.

Диаграмма разработанных в рамках данной лабораторной работы классов представлена ниже:

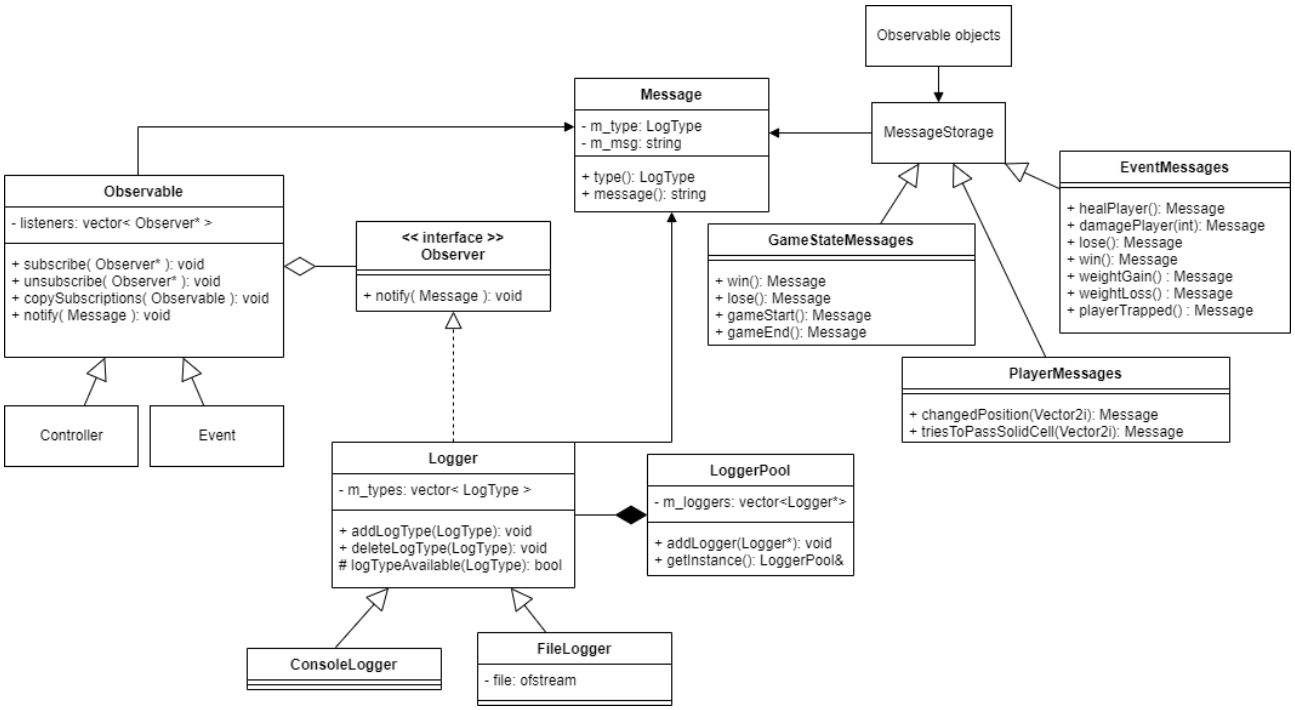

Рисунок 1 – Диаграмма классов

Тестирование.

Приведён пример результата логирования процесса исполнения программы.

Для логирования в консоль задан один уровень - изменения состояний объектов. Результаты логирования выводятся после каждого шага вместе с полем, состояние которого было изменено после действия игрока.

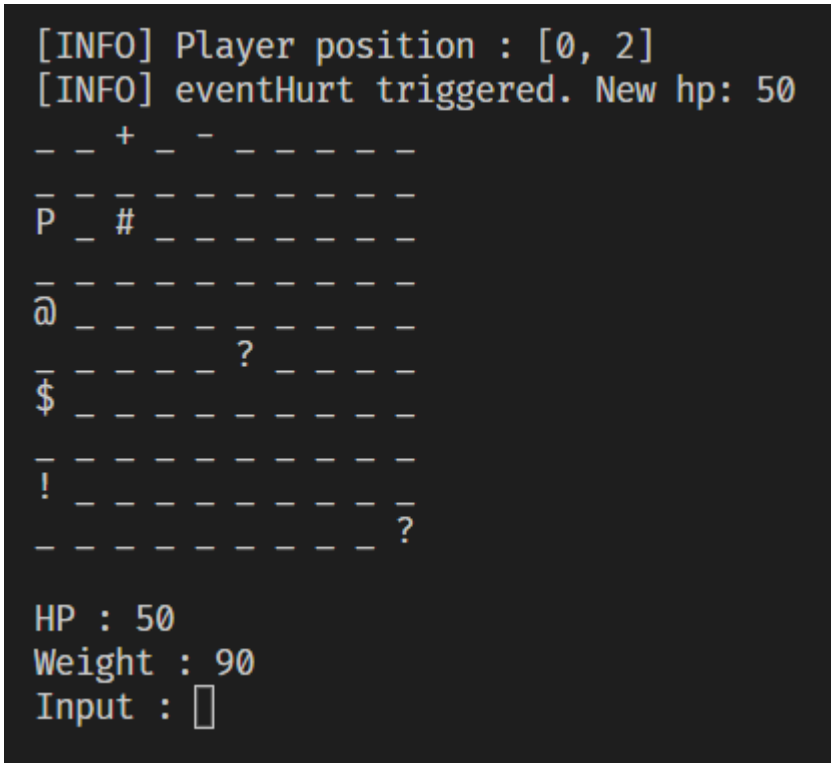

Рисунок 2 – Логирование в консоль

Для логирования в файл задано два уровня логирования – уровень критических состояний программы и состояния самой игры:

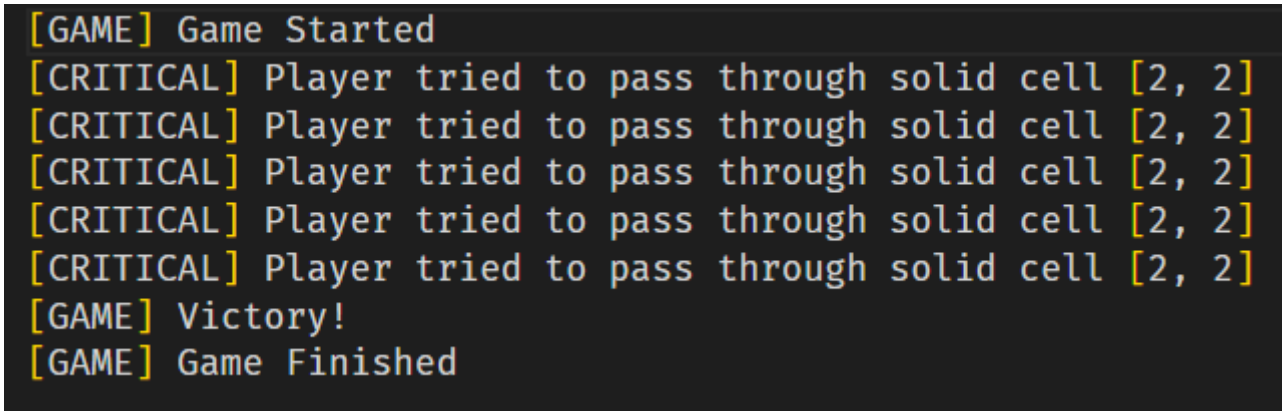

Рисунок 3 – Логирование в файл

Выводы.

Были разработаны классы, выполняющие логирование. Класс, отвечающий за логирование в файл, соблюдает идиому RAII.

Отслеживаемые сущности ничего не знают об отслеживающих их сущностях.