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. 0383 _____ Коротков А.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы:

Изучить материал по теме «логирование, перегрузка операций» и, на основе полученных знаний, реализовать класс логгера, который проводит логирование того, что происходит во время игры, и обеспечить выполнение требований к лабораторной работе.

Задание:

Необходимо проводить логирование того, что происходит во время игры.

Требуется:

• Реализован класс логгера, который будет получать объект, который необходимо отслеживать, и при изменении его состоянии записывать данную информацию.

• Должна быть возможность записывания логов в файл, в консоль или одновременно в файл и консоль.

• Должна быть возможность выбрать типа вывода логов

• Все объекты должны логироваться через перегруженный оператор вывода в поток.

• Должна соблюдаться идиома RAII

Основные теоретические положения:

Потенциальные паттерны проектирования, которые можно использовать:

• Адаптер (Adapter) - преобразование данных к нужному формату логирования

• Декоратор (Decorator) - форматирование текста для логирования

• Мост (Bridge) - переключение между логированием в файл/консоль

• Наблюдатель (Observer) - отслеживание объектов, которые необходимо логировать

• Синглтон (Singleton) - гарантия логирования в одно место через одну сущность

• Заместитель (Proxy) - подстановка и выбор необходимого логирования

Выполнение работы.

В ходе лабораторной работы были реализованы следующие классы:

class Observer – шаблонный класс наблюдателя, отлавливающий изменения в объектах классов, которые необходимо логировать, и отправляющий сообщение об изменении в адаптер

class Adapter – класс, совершающий преобразование данных об изменении к нужному формату логирования и перенаправляющий их в логгер.

class LoggerPool – класс, производящий логирование полученных от адаптера данных, в соответствии с выставленным режимом логирования: в файл, в консоль или одновременно в файл и консоль.

class ILogger – интерфейс логгера, содержащий общую для всех классов, реализующих этот интерфейс, функцию вывода лога.

class ConsoleLogger – класс, реализующий интерфейс логгера и производящий логирование в консоль.

class FileLogger – класс, реализующий интерфейс логгера и производящий логирование в файл.

Каждый из классов, требующих логирования был унаследован от абстрактного класса *Observed*, который содержит лишь одно поле *Observer obs*, позволяющее фиксировать изменения в объектах классов и через шаблонный паттерн-класс *Adapter* отправлять данные в *LoggerPool*, реализованный с помощью паттерна Singleton.

При изменениях в каждом из наблюдаемых классов, вызывается шаблонный метод класса Observer *sumbit(T* sender, logEvent event)*. Посредством этого метода адаптер узнает в каком объекте произошло изменение, и какое оно. За получение информации о типе изменения отвечает параметр *logSignal* event, представляющий собой элемент из перечисления возможных типов изменений.

В классе *LoggerPool*, предназначенном для хранения файлового *FileLogger* и консольного *ConsoleLogger* логгеров.

Для реализации необходимого вывода в поток в классах *ConsoleLogger* и *FileLogger* используется класс *LogMessage* с перегруженными операторами перенаправления в файловый и стандартный потоки.

Логируемые изменения:

- Создание героя
- Передвижение героя
- Блокирование передвижения героя
- Получение героем урона
- Гибель героя
- Выход героя с поля
- Создание каждого из врагов
- Получение врагами урона
- Передвижение каждого из врагов

- Гибель каждого из врагов
- Создание предмета
- Использование героем предмета
- Начало битвы между героем и врагом
- Выполнение героем условия выхода с поля (победы над всеми врагами)
- Окончание игры
- Закрытие окна игры

UML диаграмму см. в приложении А.

Выводы.

В ходе выполнения лабораторной работы была написана программа, реализующая систему логирования в файл и/или в консоль и создана UML-диаграмма.

Приложение А

UML-диаграмма