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. 0381 _____ Дзаппала Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Понять что такое логирование и лог-файлы. Реализовать класс логгера, который будет записывать в консоль, файл или в консоль и в файл. Должна соблюдаться идиома RAII.

Задание.

Необходимо проводить логирование того, что происходит во время игры.

Требования:

- 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

Потенциальные паттерны проектирования, которые можно использовать:

- *Адаптер (Adapter) - преобразование данных к нужному формату логирования*
- *Декоратор (Decorator) - форматирование mBBЦекста 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.

Был создан класс интерфейс Observer, у которого есть 4 чистых метода: void Update(), void Update(Character*), void SetEntity(Entity*), void DelEntity(). Это интерфейс, который будет представлять классы для логгеров. Метод Update() вызывается перегруженную операцию вывода в поток объекта Observer, в случае с файлом объект std::ofstream, если в консоль, то std::cout.

Класс Observer наследуют классы Ilogger, ConsoleLogger и FileLogger. Ilogger содержит в себе указатель на объект Entity, а также методы SetEntity, DelEntity, объявленные в Observer, и GetEntity, который возвращает указатель на отслеживаемый объект. Класс Ilogger, в свою очередь, виртуально наследуют классы ConsoleLogger и FileLogger. В этих классах перегружены опереторы вывода в поток. ConsoleLogger выводит информацию в консоль, FileLogger в файд _log.txt. Классы ConsoleLogger и FileLogger наследует класс CAFLogger (ConsoleAndFileLogger), который выводит информацию в файл и в консоль.

Также, был создан класс Observable, который является базовым для классов Character и Item. В классе определены методы Notify(), который вызывается, чтобы сказать логгерам объекта, что пора что-то записать в лог, а логгера, в свою очередь, вызывают метод Update() класса Observer. Методы AttachObserver, добавляет в вектор указателей на Observer, новый Observer для объекта отслеживания. DetachObserver(Observer*) выдергивает из вектора Observer.

В итоге, в классе Game во время игры, в методе PrintLog, вызывается метод Notify у объекта класса и врагов. Notify вызывается метод Update у объектов Observer, которые хранятся в векторе «обсерверов».

Выводы.

Реализован паттер Observer для наблюдения и логгирования объектов, которые содержаться в игре.

ПРИЛОЖЕНИЕ А

UML ДИАГРАММА КЛАССОВ