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 Прохоров Б.В.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Изучить основы объектно-ориентированного программирования на языке C++.

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

Выполнение работы.

Чтобы выполнить задание, нужно было реализовать 4 класса: *Logger*, *LoggingListener*, *ConsoleLoggingListener*, *FileLoggingListener* и немного модифицировать класс *Model*.

У класса *Model*, представляющего собой модель, проецируемую пользователю, появились конструкторы, операторы копирования и перемещения. Также объекты *Model* теперь можно сравнивать.

Logger необходимо было создать, чтобы у нас появился наблюдатель, который следит за моделью и оповещает своих подписчиков (см. классы дальше) об изменениях модели. Класс *Logger* является другом для класса *Model*. Так нужно было сделать для того, чтобы можно было обращаться к приватным полям объекта *Model*, не захламляя *Model* геттерами. У логгера есть метод *respond_to_changes (Model *model)* – логгер реагирует на изменение модели, записывая логи в подписчиков. Есть функционал подписки / отписки.

LoggingListener – подписчик, список таких подписчиков хранится у логгера. При изменении модели логгер проходит по списку подписчиков (соблюдается полиморфизм) и оповещает каждого. У подписчиков есть общий метод *get_stream ()*, который возвращает ссылку на *ostream*. Нужен, чтобы логгер понимал, куда записывать логи.

ConsoleLoggingListener – класс конкретного подписчика, в котором хранится поток вывода. Метод *get_stream ()* возвращает ссылку на *cout.*

FileLoggingListener – класс конкретного подписчика, в котором хранится файловый поток. Работает идиома, происходит проверка на открытие при открытии и закрытии файла. Метод *get_stream ()* возвращает ссылку на файловый поток.

Выводы.

Были изучены основы объектно-ориентированного программирования на языке C++.

ПРИЛОЖЕНИЕ А
UML ДИАГРАММА