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: Логгирование, перегрузка операторов.

Студент гр. 0303 _____ Болкунов В.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург 2021

Цель работы.

Создать набор классов для осуществление логгирования объектов игры.

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

Основные теоретические положения.

Логгирование — форма автоматической записи в хронологическом порядке операций в информационных технологиях, процесс записи информации о происходящих в рамках какого-либо процесса с некоторым объектом событиях, например, в файл регистрации или в базу данных. Например, журнал применительно к компьютерной памяти — запись в хронологическом порядке операций обработки данных, которые могут быть использованы для того, чтобы воссоздать существовавшую или альтернативную версию компьютерного файла.

Выполнение работы.

Структура LogLevel — хранит в себе перечисление уровней логов, по совместительству объект структуры определяет уровень. То есть существует возможность передать объект перечисления туда, где требуется объект уровня, так как неявно вызывается конструктор LogLevel.

- Содержит метод toString для преобразования объекта уровня в std::string.

Доступные уровни:

- *OFF*, - только для логгеров: не выводить никакие логи
- *ERROR*, - для критических ошибок
- *WARN*, - для несерьёзных ошибок
- *INFO*, - информационный лог
- *DEBUG*, - отладочный лог
- *ALL* - только для логгеров: выводить логи всех уровней

Структура Log — объект лога, хранит сообщение, временной штамп и уровень лога.

- Конструктор принимает уровень (как объект LogLevel, так и объект из перечисления уровней), и само сообщение типа std::string, временной штамп создаётся автоматически.

Для удобного создания логов каждого уровня определены макросы:

- error(msg)
- warn(msg)
- info(msg)
- debug(msg)

Класс LogTimer — синглтон для отсчёта начального времени логов (используется в TimeFormat).

- Содержит метод get для получения временного штампа начальной точки отсчёта,
- и метод reset для сброса точки отсчёта.

Класс Format — абстрактный декоратор для логов. Содержит виртуальный метод wrap для обёртки лога.

Класс EmptyFormat — формат по умолчанию (не меняет лог)

Класс TagFormat — формат с тэгом в квадратных скобках.

Класс TimeFormat — формат с временной меткой в квадратных скобках, время = разность между меткой лога и точкой отсчёт LogTimer-а по умолчанию, но возможно задать произвольный штамп точки отсчёта.

Класс Logger — абстрактный класс для логгеров,

- принимает логи через перегруженный оператор `<<`,
- имеет свой объект форматирования (указатель на объект класса Format), по умолчанию создаётся EmptyFormat.
- Способ вывода логов определяется наследниками в виртуальном методе log.
- Уровень логгера задаётся либо с помощью аргумента конструктора (по умолчанию *ALL*), либо с помощью метода setLvl.

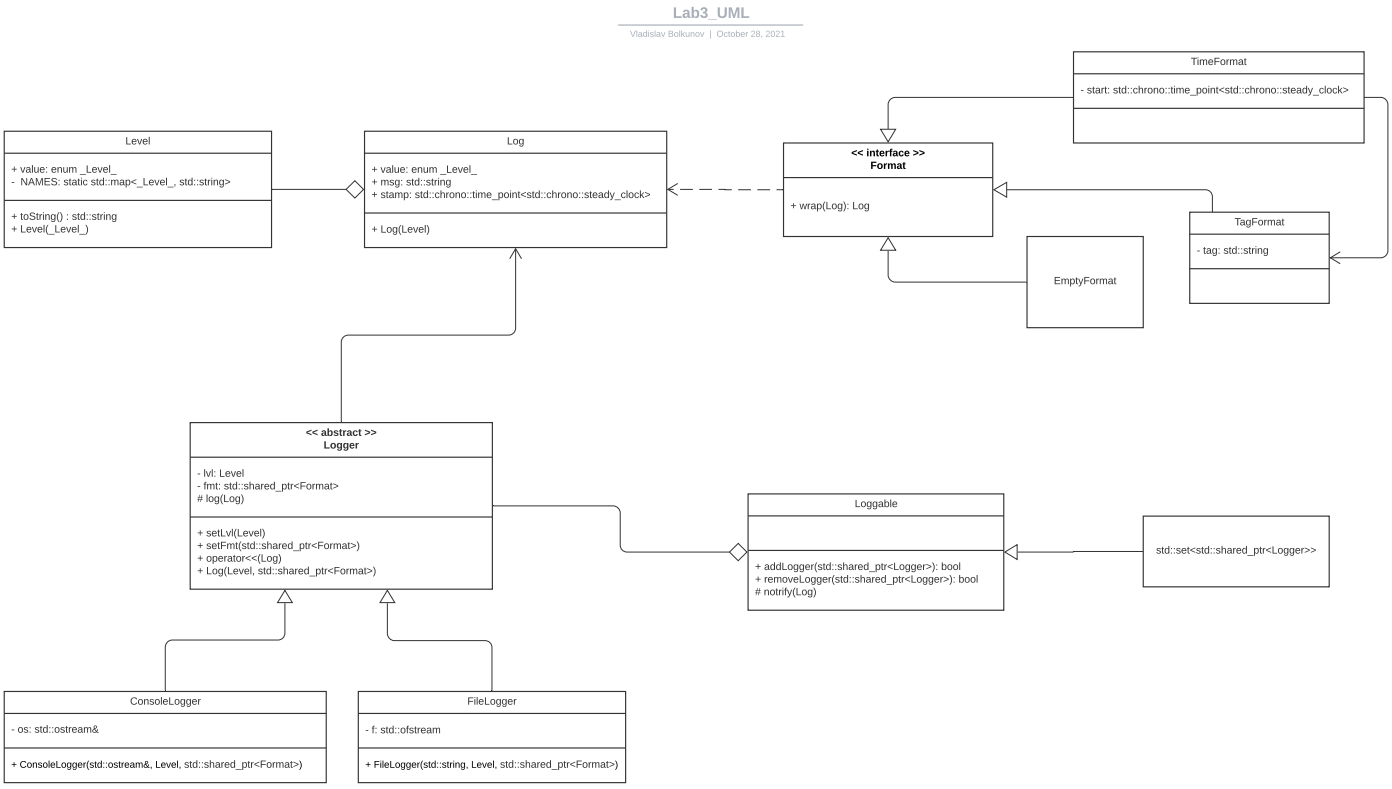
Класс FileLogger — логгер вывода в файл, принимает имя файла.

- Время жизни файла контролируется объектом логгера, то есть файл открывается конструкторе (в который передаётся имя файла) и закрывается в деструкторе.

Класс ConsoleLogger — логгер вывода в консоль, принимает ссылку на поток вывода (std::cout по умолчанию).

Класс Loggable — класс логируемого объекта. Классы, объекты которых необходимо логгировать, наследуясь от данного класса получают возможность записывать логи в заданные логгеры.

- Приватно наследуется от std::set (в качестве шаблона — указатель на логгер), чтобы предотвратить возможность многократной записи в один и тот же логгер.
- Содержит методы добавления и удаления логгера. Методы для добавления и удаления логгера виртуальные, но определены по умолчанию, пример ситуации где требуется переопределение — класс Field, который «подписывает» логгеры на каждую свою клетку.
- Методом notify выводит лог во все имеющиеся у него логгеры.
- При использовании с объектами класса Logger реализует паттерн Observer, объекты-логгеры «подписываются» на обновления объекта класса наследуемого от Loggable



Разработанный программный код см. в директории /*game_lib*.

Разработанную диаграмму классов UML см. в Lab3_UML.pdf .

Тестирование.

Разработанные тесты см. в директории /tests.

Выводы.

Был реализован набор классов для осуществления удобного логгирования и в игровые объекты добавлены выводы логов.