МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «ООП»

Тема: Добавление логирования

Студент гр. 9304 _____ Тиняков С.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Создать набор классов для отслеживания игрока, элемнтов поля и прочего.

Задание.

Создан набор классов, которые отслеживают игрока и элементы на поле, и выводят/сохраняют информацию об их изменениях.

Обязательные требования:

- Реализована возможность записи логов в терминал и/или файл
- Взаимодействие с файлом реализовано по идиоме RAII
- Перегружен оператор вывода в поток для всех классов, которые должны быть логированы

Дополнительные требования:

- Классы, которые отслеживают элементы, реализованы через паттерн Наблюдатель
- Разделение интерфейса и реализации класса логирования через паттерн Мост

Выполнение работы.

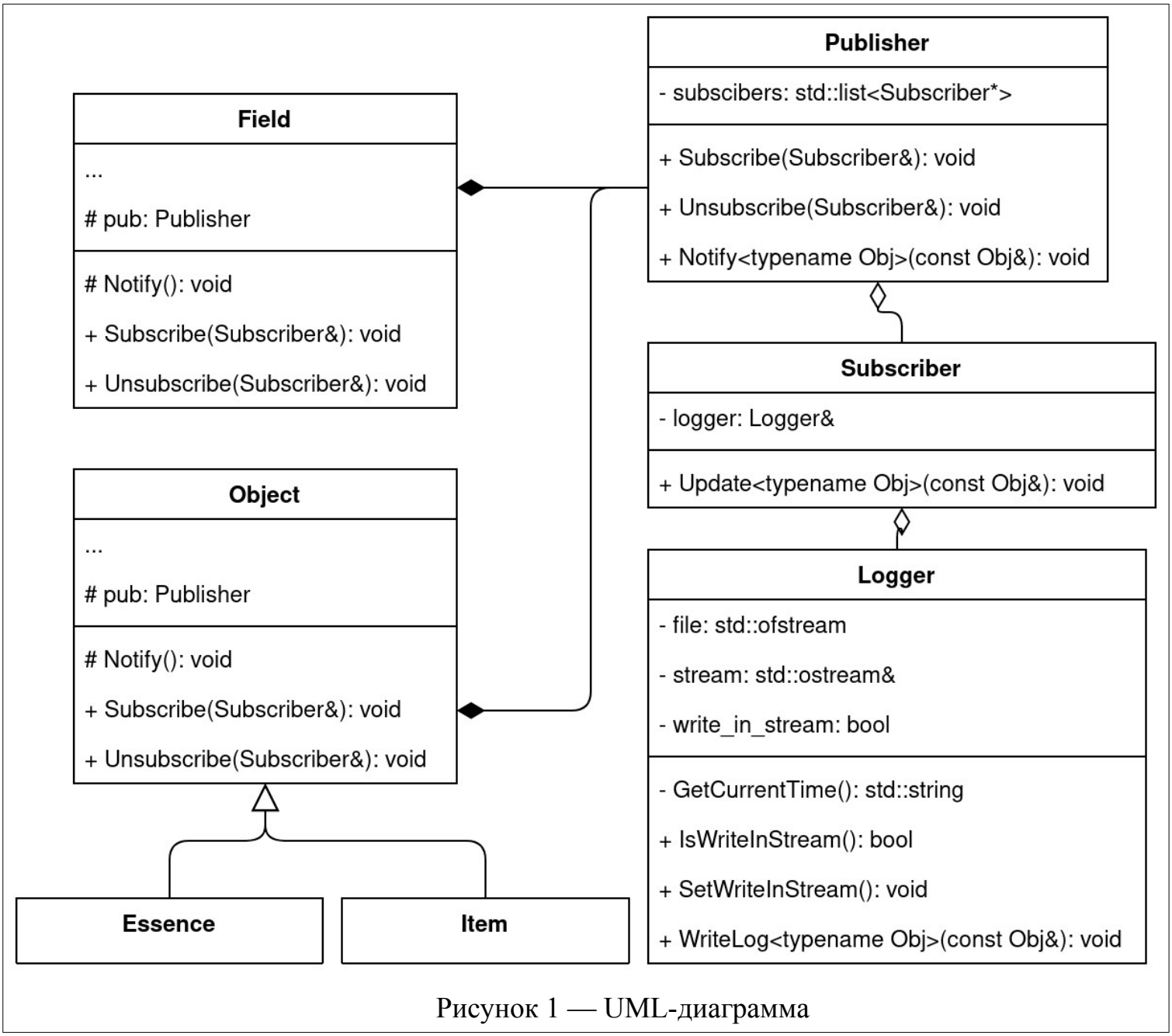

Рисунок 1 — UML-диаграмма

Класс *Logger* является главным классом в наборе классов для отслеживания. *Logger* сохраняет логи в файл. Также присутствует возможность вывода в поток. За это отвечает поле *write_in_stream*. Для изменения и получения значения данного поля были реализованы соответственно методы *SetWriteInStream* и *IsWriteInStream*. В поле *stream*

содержится ссылка на поток вывода, а в поле *file* файл для логов. Метод *GetCuurentTime* возвращает строку с текущем временем в формате *«[Час:Минуты:Секунды]»*. Метод *WriteLog* записывает аргумент в файл и при необходимости в поток вывода. Для универсальности, метод *WriteLog* реализован через шаблон.

Для отслеживания необходимых объектов была реализованна пара классов: *Publisher* и *Subscriber*. Поле *logger* в классе *Subscriber* содержит ссыслку на *Logger*, который будет записывать изменения объектов, на которые подписан данный подписчик. Для записи при помощи *Logger* используется метод *Update*, который для универсальности реализован при помощи шаблона и который вызывает метод *WriteLog* для *logger*. Класс *Publisher* создан для оповещения всех подписчиков. Указатели на них хранятся в поле *subcribers*. Методы *Subscribe* и *Unsubscribe* соответственно подписывают и отписывают подписчика от данного издателя. Метод *Notify* вызывает метод *Udate* у всех подписчиков.

Все классы, которые необходимо отслеживать, имеют в себе поле *pub* — издателя. Также имеется виртуальный метод *Notify* с модификатором доступа *protected*. Это сделано по следующей причине: во время работы программы довольно сложно установить какой сейчас класс(особенно при наследовании и использовании интерфейсов), а методу *Notify* в классе *Publisher* нужен тип объекта. Поэтому каждый класс должен переопределить метод *Notify*, в котором вызовет метод *Notify* для *pub* с необходимым типом. Если этого не сделать, то логгирование будет некорректно.

Для проверки правильности работы классов были созданы *unit*-тесты.

Выводы.

Был создан набор классов для отслеживания игрока, элемнтов поля и прочего. Класс *Logger* отвечает за запись логов в файл и поток вывода. Для удобства отслеживания были созданы классы *Publisher* и *Subscriber*. Запись

лога для универсальности реализована через шаблон. Во всех классах, которые необходимо отслеживать, перегружен оператор вывода в поток. Для проверки правильности работы классов были созданы *unit*-тесты.