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. 1384 _____ Тапеха В. А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2022

Цель работы.

Научиться выполнять перегрузку операций и логирование путём реализации набора классов, которые отслеживают изменения состояний в программе и классов, которые выводят информацию в разные уровни (консоль/файл) с перегруженным оператором вывода в поток.

Задание.

Реализовать класс/набор классов отслеживающих изменения состояний в программе. Отслеживание должно быть 3-х уровней:

1. Изменения состояния игрока и поля, а также срабатывание событий
2. Состояние игры (игра начата, завершена, сохранена, и т.д.)
3. Отслеживание критических состояний и ошибок (поле инициализирован с отрицательными размерами, игрок попытался перейти на непроходимую клетку, и т.д.)

Реализованы классы для вывода информации разных уровней для в консоль и в файл с перегруженным оператором вывода в поток.

Требования:

- Разработан класс/набор классов отслеживающий изменения разных уровней
- Разработаны классы для вывода в консоль и файл с соблюдением идиомы RAII и перегруженным оператором вывода в поток.
- Разработанные классы спроектированы таким образом, чтобы можно было добавить новый формат вывода без изменения старого кода (например, добавить возможность отправки логов по сети)
- Выбор отслеживаемых уровней логирования должен происходить в runtime
- В runtime должен выбираться способ вывода логов (нет логирования, в консоль, в файл, в консоль и файл)

Примечания:

- Отслеживаемые сущности не должны ничего знать о сущностях, которые их логируют
- Уровни логирования должны быть заданными отдельными классами или перечислением
- Разные уровни в логах должны помечаться своим префиксом
- Рекомендуется сделать класс сообщения
- Для отслеживания изменений можно использовать наблюдателя
- Для вывода сообщений можно использовать адаптер, прокси и декоратор

Выполнение работы.

Для выполнения лабораторной работы были созданы классы, отвечающие за сообщение, которое выводится во время логирования, логирование конкретных объектов, и отвечающие за вывод в разные уровни(консоль/файл).

Новые классы:

1) Реализован интерфейс Logger, от которого реализуются FileLog и ConsoleLog. В интерфейсе есть общий метод print. В ConsoleLog печатается сообщение в консоль с помощью перегруженного оператора вывода в поток. В FileLog в конструкторе FileLog(const std::string&) определяется имя файла, в который будет происходить вывод, а в методе print так же, как выводит сообщение с перегруженным оператором, только выводит информацию в файл.

2) Реализован абстрактный класс Loglevel, от которого наследуются ErrorLog, GameLog, StatusLog, которые наблюдают за конкретными объектами, при этом сами сущности, за которыми они наблюдают ничего не знают о наблюдателях. LogLevel хранит в protected вектор, хранящий в себе всех сущностей, которые выводят сообщение (то есть в нем хранятся элементы типа Logger*). LogLevel содержит внутри чистые виртуальные методы, которые отвечают за вывод сообщения у всех сущностей, которые выводят сообщение (метод update), а другой устанавливает эти сущности (метод set_loggers). В наследниках эти методы реализованы похожим образом. А в своих

конструкторах принимает указатель на subject, от которого наследуются наблюдаемые сущности.

3) Написан абстрактный класс Subject, от которого наследуются все наблюдаемые классы. Он содержит внутри себя поле с вектором из наблюдателей (std::vector<LogLevel*>) и методы attach, который добавляет в вектор нового наблюдателя, detach, который удаляет какого-то конкретного наблюдателя, и notify, который вызывает метод update для каждого наблюдателя.

4) Реализован класс Message, который формирует сообщение исходя из введенного префикса и введенной информации. В этом файле перегружен также оператор вывода.

5) Был написан класс GameStatus для удобства отслеживания изменения статуса игры, который является наблюдаемым объектом.

Измененные классы:

Классы, которые были написаны в прошлых лабораторных работах, почти не было изменены. Только наблюдаемых классах при изменении объекта (Field, Player, GameStatus, CommandReader) вызывается метод класса-родителя notify, который извещает наблюдателей.

В классе, где находится цикл с игрой, хранятся указатели на уровни логирования. Там же пользователя спрашивают, какие уровни должны логироваться и в какие потоки должны выводиться логи. И исходя из этого вызывается у уровней логирования вызывается функция set_loggers, которая заполняет объектами, выводящими логи в определенный поток.

UML-диаграмма межклассовых отношений.

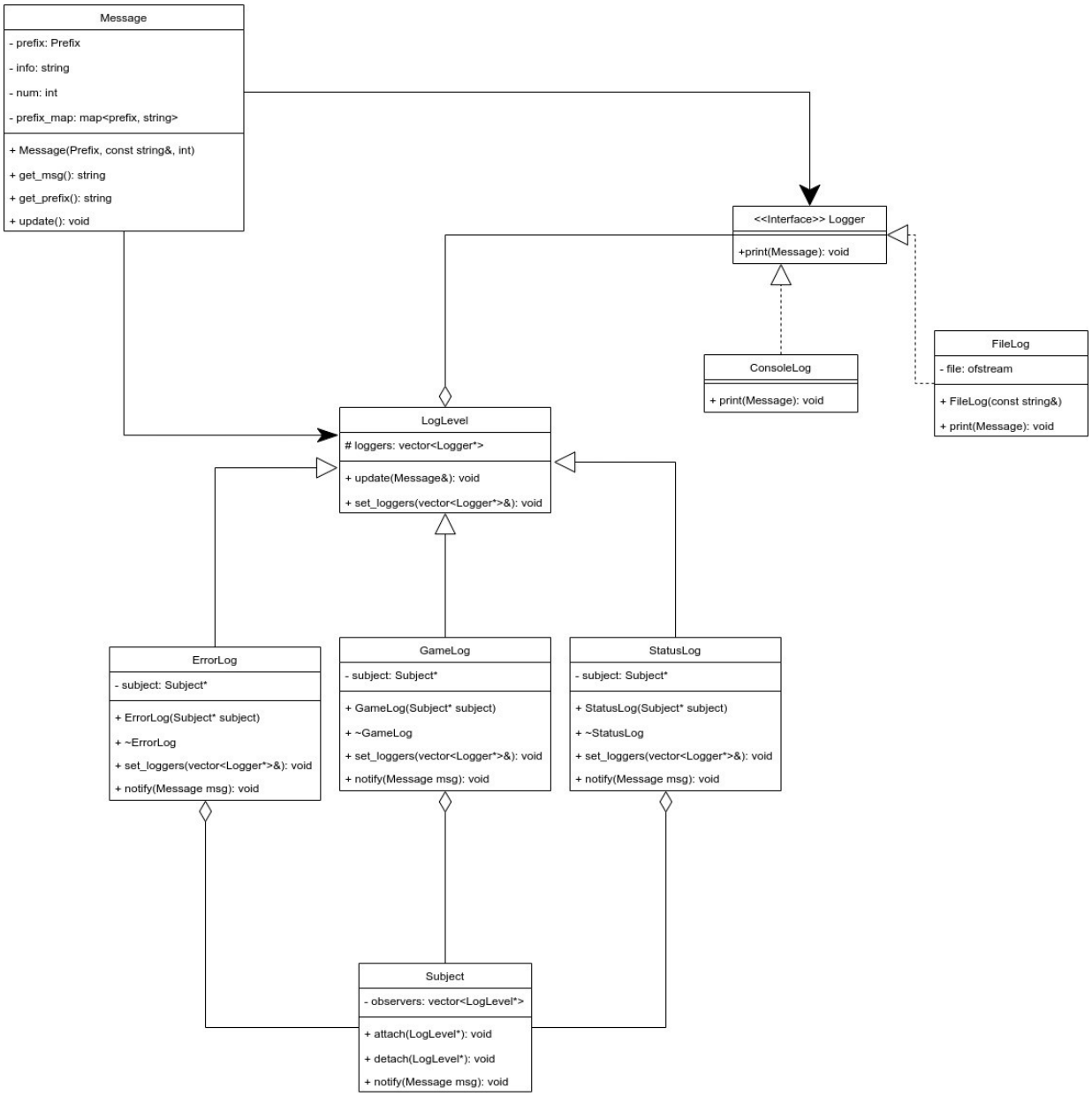

Рис. 1 UML-диаграмма.

Выводы.

Были изучены основы логирования. В ходе лабораторной работы был написан набор классов, отвечающий за логирование игры, были написаны классы, которые выводят информацию в разные потоки, а также был реализован класс с перегруженными оператором вывода в поток.