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 Прошичев А.В.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2022

Цель работы.

Изучить и освоить на практике логирование. Научиться перегружать операторы, а также грамотно расставлять логи для качественного отслеживания ошибок (дебага).

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 инициализировано 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 отслеживания изменений в различных классах использовался паттерн «Наблюдатель». Его представителями являются классы Observer и Subject, а также конкретные наблюдатели в виде LogGame, LogError, LogStatus.

Observer –абстрактный класс, который через переопределённые функции которого происходит общение с Subject (наблюдаемым объектом). Его функция Update() является методом реакции на изменения в отслеживаемой сущности.

Subject – класс, от которого наследуются все сущности, за которыми нужно наблюдать. Тем самым, каждому своему потомку он вносит ряд методов: для подписки на наблюдение за ним (Attach), для отписки от наблюдения за ним (Detach), для озвучивания произошедших изменений в отслеживаемом классе (Notify). Ещё класс хранит массив на наблюдателей, именно проходя по нему, он будет оповещать всех о своём изменении.

LogGame – потомок Observer. Класс, наблюдающий за этапами игры такими, как установка размеров поля, запуска прогресса шагов, выполнения команд, завершения игры, поражение или победа. Имеет перегруженный оператор вывода в поток, который по принятому сообщению Message выбирает тип выводимых данных и их структуру.

LogError – потомок Observer. Класс, наблюдающий за возможными ошибками в игре. Именно он уведомит о неправильном вводе, некорректных

значениях длины поля или о попытке встать на стену. Имееи аналогичный перегруженный оператор.

LogStatus – потомок Observer. Класс, наблюдающий за игроком, полем, клетками и событиями. Такой логгер будет уведомлять о изменениях характеристик игрока, уничтожения, создания или активации события, а также изменений поля.

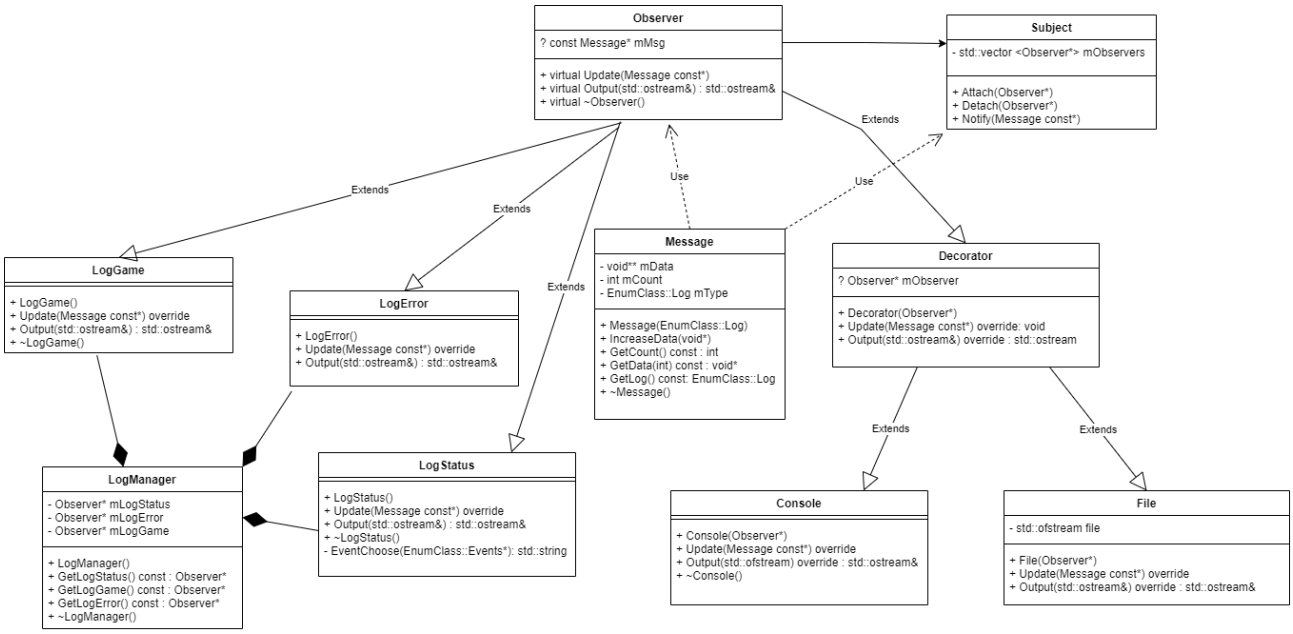
Передача информации в такой системе происходит благодаря классу Message. Его экземпляры передаются у наблюдаемых объектов в Notify и принимаются из метода Update у наблюдателей. Такой класс хранит в себе массив указателей на void и расширяется динамически. Сделано это для возможности записать совершенно любые данные в этот массив. Распаковка данных далее производится по опознаванию типа лога (элемент перечисления log).

Также для получения возможности выводить в консоль, в файл, одновременно по обоим направлениям или никуда, к тому же с лёгкостью вносить виды потоков был введён паттерн «Декоратор», который представляется один основной класс Decorator и его конкретные потомки Console, File.

Decorator – класс, который наследуется от Observer, в следствии чего он мало отличим от классов LogGame, LogError, LogStatus. Но при этом он приобретает возможность хранить у себя указатель на своего отца. Паттерн устроен так, что Decorator называют «обёрткой», которая можно незаметно обрабатывать проходящие через неё данные, при этом не нарушая информационный обмен. В данном случае, Decorator имеет двух потомков, каждый из них надстраивается над конкретным наблюдателем. В течение программы такая «обёртка» передаст при запросе данные по цепочки дальше, чтобы они дошли и обработались наблюдателем, но потом сделает вывод переданных данных через перегруженные у наблюдателей операторы вывода. В связи с чем, программа выводит логи, никак не создавая каких-либо ещё ответвлённых классов. Стоит отметить, что так называемые «обёртки» можно заворачивать сами в себя из-за общих предков, поэтому цепочкам может быть

очень длинной и приэтом каждый учатсник этой цепочки выведет информацию в свой поток.

Класс LogManager является организатором классов-логгером, он выделяет на них память, по данным пользователя оборачивает в нужные декораторы, а далее они могут быть спокойно прикреплены к своим наблюдаемым объектам.



Тестирование.

```
Please, choose one or several log types.
If program take an incorrect input from you, it will set default values.
The list of the types:
Enter 1 -- log type that controls Player, Field and events
Enter 2 -- log type that controls the status of the game
Enter 3 -- log type that controls critical mistakes from a user
if you'd like to use any output type, enter the corresponding number.
Could enter several numbers, spliting their with a space.
```

Изображение №1. Интерфейс при выборе уровней логов

```
Please, choose one or several output types.
The list of the types:
Enter 1 -- output to the console
Enter 2 -- output to the file
if you'd like to use any output type, enter the corresponding number.
Could enter several numbers, spliting their with a space.
```

Изображение №2. Интерфейс при выборе способов вывода логов

```
Incorrect command! Please, check the correct format of the enter!
[ERROR] Attemp to enter incorrect values: н
п
Incorrect command! Please, check the correct format of the enter!
[ERROR] Attemp to enter incorrect values: п
```

Изображение №3. Логи ошибок при неправильном вводе.

```
[STATUS] Created BearBush
[STATUS] Cell triggered from position that has X Coord: 1 Y coord: 10
[STATUS] Created SaintWaterCave
[STATUS] Cell triggered from position that has X Coord: 4 Y coord: 14
[STATUS] Thirst: 19

[LOG] The game did a step. Now 1 steps have been taken
```

Изображение №4. Логи от игрока, поля, событий и игрового шага

```
[LOG] Field got size
[STATUS] Player moved on X Coord: 7 Y coord: 6
[STATUS] Created WaterCave
[STATUS] Cell triggered from position that has X Coord: 11 Y coord: 4
[STATUS] Created WaterCave
[STATUS] Cell triggered from position that has X Coord: 14 Y coord: 6
[STATUS] Created RabbitBush
[STATUS] Cell triggered from position that has X Coord: 13 Y coord: 7
[STATUS] Created RabbitBush
[STATUS] Cell triggered from position that has X Coord: 1 Y coord: 8
[STATUS] Created WolfBush
[STATUS] Cell triggered from position that has X Coord: 4 Y coord: 10
[STATUS] Created WolfBush
[STATUS] Cell triggered from position that has X Coord: 6 Y coord: 10
[STATUS] Created WolfBush
[STATUS] Cell triggered from position that has X Coord: 4 Y coord: 11
[STATUS] Thirst: 19

[LOG] The game did a step. Now 1 steps have been taken
[STATUS] Player moved on X Coord: 6 Y coord: 6
[STATUS] Created WaterCave
[STATUS] Cell triggered from position that has X Coord: 11 Y coord: 3
[STATUS] Created BerryBush
[STATUS] Cell triggered from position that has X Coord: 2 Y coord: 10
[STATUS] Created RabbitBush
[STATUS] Cell triggered from position that has X Coord: 12 Y coord: 10
[STATUS] Created WaterCave
[STATUS] Cell triggered from position that has X Coord: 3 Y coord: 12
[STATUS] Created WaterCave
[STATUS] Cell triggered from position that has X Coord: 1 Y coord: 13
[STATUS] Created EarthQuakeCave
[STATUS] Cell triggered from position that has X Coord: 1 Y coord: 14
[STATUS] Thirst: 18
[STATUS] Hunger: 9
```

Изображение №5. Текстовые файл с выводом логов

Выводы.

Изучил и освоил на практике логирование. Научился перегружать операторы, а также грамотно расставлять логи для качественного отслеживания ошибок (дебага).