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

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Объекто-ориентированное программирование»
Тема: Перегрузка операторов / Логирование

Студент гр. 2381 _____ Двиков Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2023

Цель работы

Изучить методы работы с классами. Программу в стиле ООП. .

Задание

а) Реализовать набор классов “сообщений” с общим интерфейсом, который будут срабатывать в определенные моменты и хранить информацию о событии(не путать с классом игрового события из лаб. 3), но не должны хранить сообщение в виде строки. Должны быть реализованы класс для следующих событий:

Игрок выиграл. Хранится информация о характеристиках игрока

Игрок проиграл. Хранится информация о координатах клетки на которой событие произошло

Была запущена новая игра. Хранится информация о размерах поля и стартовой позиции игрока.

Была введена клавиша и сработала команда. Информация о введенном символе и какая команда сработала.

Была введена клавиша, но никакая команда не сработала. Информация о введенном символе.

б) Для сообщений перегрузить оператор вывода в поток. Таким образом можно выводить сообщение в различные потоки (cout, файл). При выводе в поток сообщения, должна формироваться строка и подставляться хранимая информация.

в) Разработать систему классов, которые отслеживают сообщения и выводят их в файл и/или консоль. Куда выводить запрашивается у пользователя при запуске программы: никуда, в файл, в консоль, в файл и консоль. Классы, в которых происходит отслеживаемое событие, должны только отправлять сообщение, но не знать куда, то есть только создают сообщение, инициализируя его информацию, и отправляют.

Примечания:

Система отслеживания должна масшт абируема для новых потоков вывода без изменения кода. Для этого вывод в файл и терминал можно обернуть в отдельные классы с общим интерфейсом.

Для записи в файл придерживайтесь идиомы RAII

Отслеживаемые сущности не должны знать о том, кто и как их логирует.

Выполнение работы

Message класс:

Это абстрактный базовый класс, определяющий интерфейс для сообщений.

Он содержит виртуальный деструктор и чисто виртуальную функцию print, которая должна быть реализована в производных классах.

Определен friend оператор перегрузки <<, который позволяет выводить сообщения в std::ostream.

MessageHandler класс:

Этот класс отслеживает и отправляет сообщения.

Он хранит указатели на объекты Message в векторе m_messages.

Метод trackMessage добавляет сообщение в вектор для последующей отправки.

sendMessages метод отправляет сообщения из вектора, выводя их в std::ostream, после чего освобождает память, выделяемую для сообщений.

PlayerWon класс:

Этот класс является производным от Message и представляет сообщение о победе игрока.

Он содержит информацию о здоровье и счете игрока.

Метод print выводит сообщение о победе игрока в std::ostream.

PlayerLost класс:

Этот класс также наследуется от Message и представляет сообщение о поражении игрока.

Содержит информацию о позиции, на которой игрок проиграл.

Метод print выводит сообщение о поражении игрока в std::ostream.

ValidCommandPressed класс:

Еще один производный класс от Message, представляющий сообщение о нажатии корректной команды.

Содержит информацию о нажатой клавише и направлении команды.

Метод print выводит сообщение о нажатии корректной команды в std::ostream.

InvalidCommandPressed класс:

Этот класс также наследуется от Message и представляет сообщение о нажатии некорректной команды.

Содержит информацию о нажатой некорректной клавише.

Метод print выводит сообщение о нажатии некорректной команды в std::ostream.

GameStarted класс:

Класс GameStarred – это производный класс от Message, представляющий сообщение о начале игры.

Содержит информацию о размере поля и начальной позиции игрока.
Метод print выводит сообщение о начале игры в std::ostream

void GameManager::run():

Запрашивает у пользователя режим ведения журнала (File, Console, Both, Nowhere) и создает соответствующие MessageHandler.

Создает игровые объекты (игрока, поле, контроллер, вид, наблюдателя и т.д.).

Запрашивает у пользователя уровень (1 или 2) и создает поле соответствующего уровня.

Создает сообщение о начале игры и отслеживает его.

Игровой цикл, в котором игрок вводит команды для перемещения, обновляет игровое состояние и отслеживает изменения.

Отправляет сообщения о поражении или победе игрока и запрашивает у пользователя желание сыграть снова.

void GameManager::trackMessage(Message* msg):

Метод для отслеживания сообщений.

Добавляет сообщение msg в соответствующие MessageHandler (консоль или файл), если они доступны.

void GameManager::sendMessage():

Метод для отправки отслеженных сообщений.

Отправляет накопленные сообщения из MessageHandler (консоль или файл), если они доступны, используя метод sendMessages().

Класс ConsoleCmdR:

Конструктор ConsoleCmdR:

- Принимает два параметра:
    - cmnds - словарь, содержащий отображение строковых команд (std::string) на перечисление Direction.
    - gameInterruption - булевое значение, указывающее, может ли команда прервать игру.

Метод getch():

- Читает один символ с консоли без отображения ввода на экране и возвращает его.
- Возвращает символ, считанный с консоли.

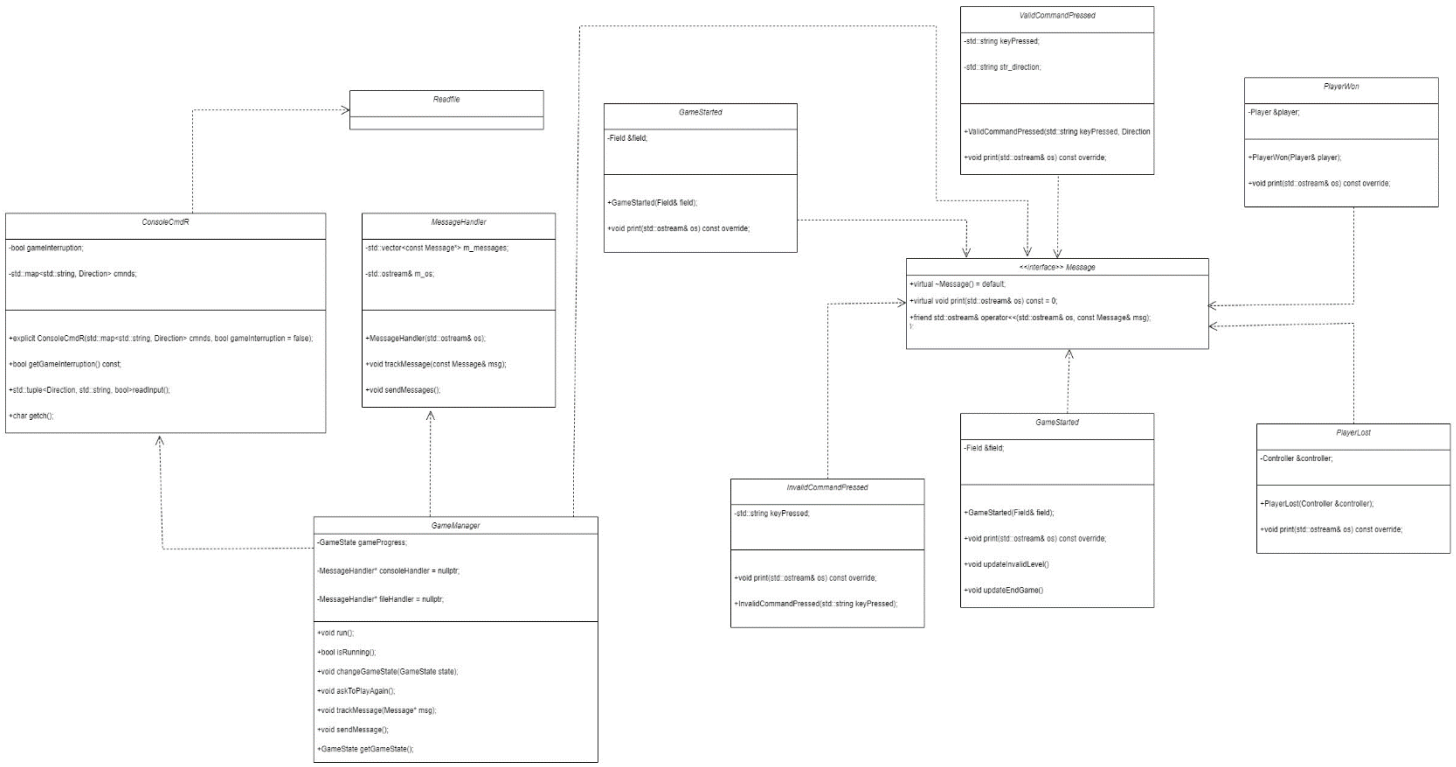
Метод getGameInterruption():

- Возвращает значение переменной gameInterruption, указывающей, может ли команда прервать игру.

Метод readInput():

- Считывает команду из консоли.
- Происходит считывание одного символа с помощью метода getch().
- Добавляет считанный символ в строку команды.

- Проверяет, содержится ли команда в словаре cmnds.
- Если команда не найдена в словаре, устанавливает isValid в false и возвращает кортеж с информацией о недопустимой команде.
- Если найденная команда является командой завершения игры



Выводы

Был реализован класс набор классов “сообщений”, Для сообщений был перегружен оператор вывода в поток, а также Разработана система классов, которые отслеживают сообщения и выводят их в файл и/или консоль.