МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «ООП»

Тема: Уровни абстракции, управление игроком

Студентка гр. 1383 Седова Э.А

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Реализовать набор классов отвечающих за считывание команд пользователя, обрабатывающих их и изменяющих состояния программы.

Задание.

- Реализован класс/набор классов обрабатывающие команды
- Управление задается из файла
- Реализованные классы позволяют добавить новый способ ввода команд без изменения существующего кода (например, получать команды из файла или по сети).
- Из метода считывающего команду не должно быть “прямого” управления игроком

Выполнение работы.

Создан класс *ReadConfig*, содержащий поле *default_command()* со значениями класса map, в котором ключами выступают символы, с помощью которых осуществляется управление, а значениями – названия команд, поле Prefix, содержащее список строк с названиями всех доступных команд и метод *get_defaut()*, возвращающий дефолтные значения управления. Класс *ReadConfig* наследует класс *ReadFileConfig*. В методе *readConfig()* происходит построчное считывание и обработка команд из файла. В поле error заносится значение true, если программе не удаётся открыть файл, название команды не содержится в списке Prefix, введено неправильное количество команд или в значении команды указано больше одного символа. В методе *getConfig()* возвращается обработанное значение *config* класса map из файла, либо дефолтное значение *config* класса map с помощью метода *get_default()*, если при вводе команд в файл пользователем была допущена ошибка и полю error было присвоено значение true.

Далее, в конструкторе класса *Controller*, создаётся объект *config* класса *ReadFileConfig*. У него поочерёдно вызываются методы *readConfig()* и

getConfig(). Игрок передвигается по полю, если введённый в консоль символ является подходящим ключом к одной из команд, перечисленных в списке *Prefix*.

Тестирование программы.

На рисунках 1 - 6 представлено тестирование программы

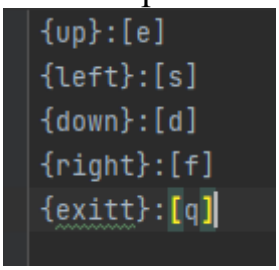

Рисунок 1 – Тестирование программы

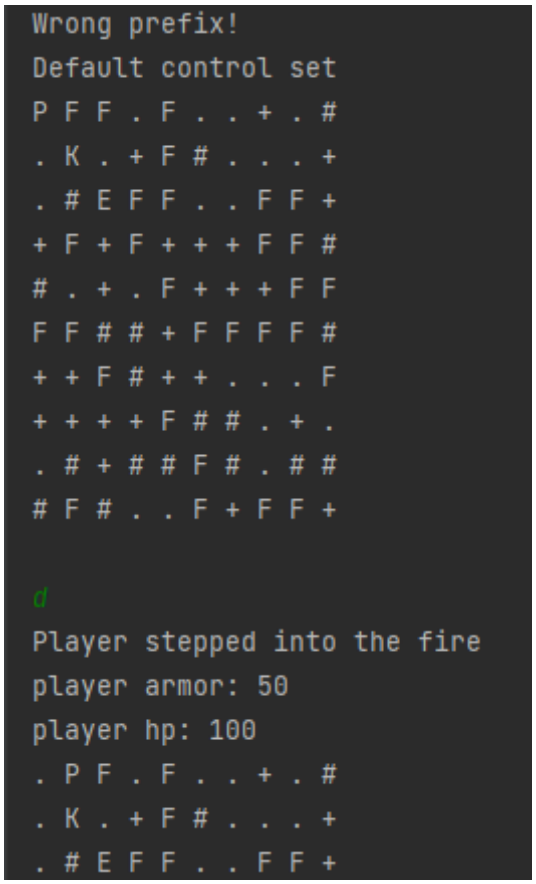

Рисунок 2 – Тестирование программы

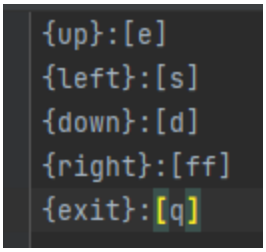

Рисунок 3 – Тестирование программы

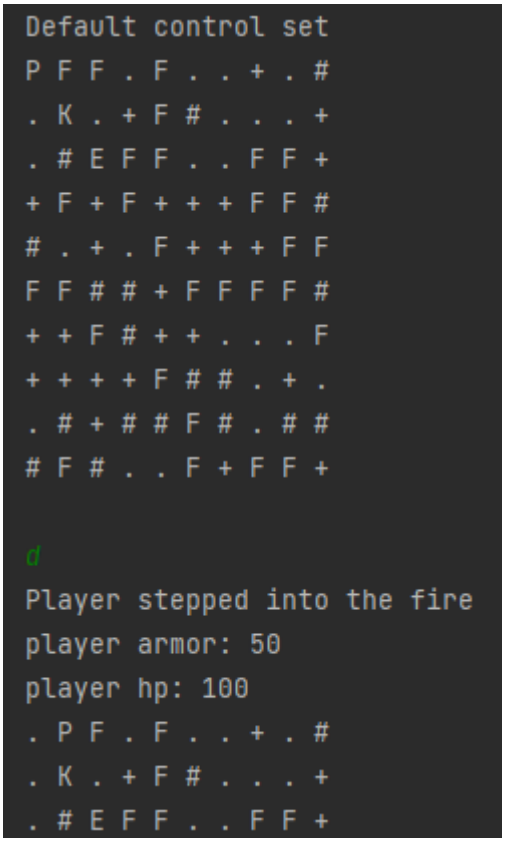

Рисунок 4 – Тестирование программы

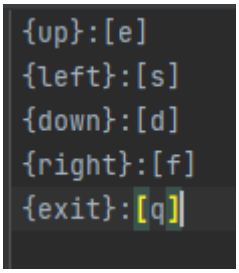

Рисунок 5 – Тестирование программы

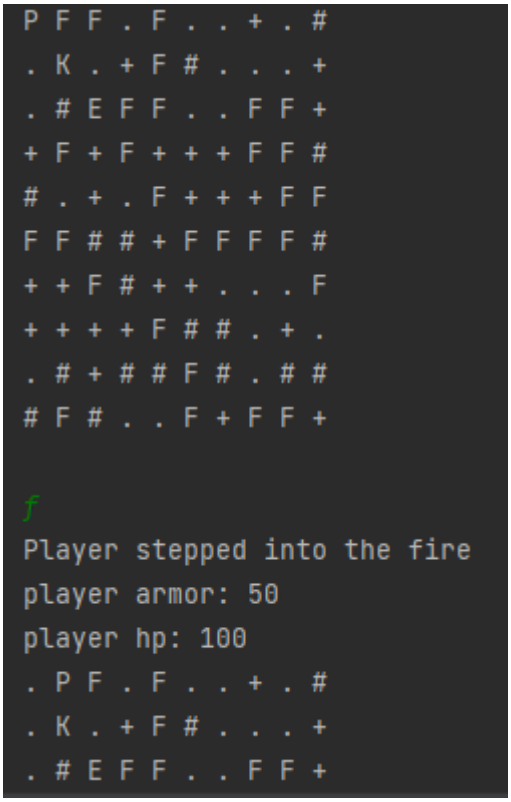

Рисунок 6 – Тестирование программы

UML-диаграмма межклассовых отношений.

На рисунке 7 изображена UML Диаграмма классов

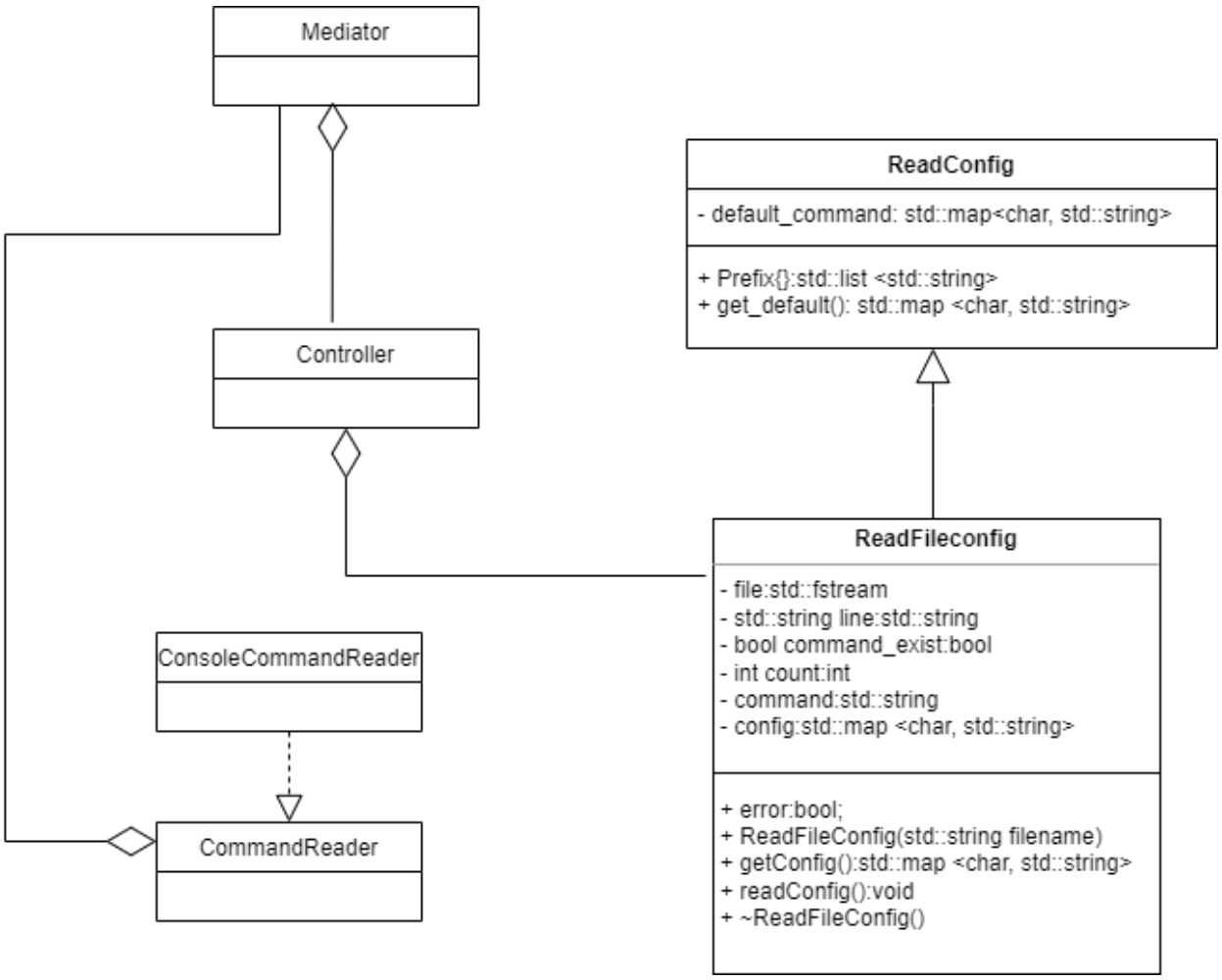

Рисунок 7 – UML Диаграмма классов

Выводы.

В ходе выполнения работы были реализованы классы, отвечающие за считывание команд пользователя, обрабатывающие их и изменяющих состояния программы.