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

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Объектно-ориентированное программирование»
Тема: Уровни абстракции, управление игроком

Студент гр. 1381 _____ Смирнов Д. Ю.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2022

Цель работы.

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

Задание.

Реализовать набор классов отвечающих за считывание команд пользователя, обрабатывающих их и изменяющих состояния программы (начать новую игру, завершить игру, сохраниться, управление игроком, и т. д.). Команды/клавиши, определяющие управление должны считываться из файла.

Требования:

- Реализован класс/набор классов обрабатывающие команды.
- Управление задается из файла (определяет какая команда/нажатие клавиши отвечает за управление, например, w - вверх, s - вниз, и.т.д).
- Реализованные классы позволяют добавить новый способ ввода команд без изменения существующего кода (например, получать команды из файла или по сети). По умолчанию, управление из терминала или через GUI, другие способы реализовывать не надо, но должна быть такая возможность.
- Из метода, считывающего команду, не должно быть “прямого” управления игроком.

Примечания:

- Для реализации управления можно использовать цепочку обязанностей, команду, посредника, декоратор, мост, фасад.

Выполнение работы.

Для выполнения лабораторной работы создан набор классов отвечающих за считывание команд пользователя, обрабатывающих их и изменяющих состояния программы.

Новые классы:

КОНФИГУРАЦИЯ: Абстрактный класс *IConfig* хранит в себе объекта типа *map*, один отвечает за стандартные клавиши команд, другой пользовательскую конфигурацию клавиш, заданных пользователем. Есть методы, которые устанавливает пользовательскую конфигурацию клавиш команд на стандартную и проверяет пользовательскую конфигурацию клавиш. От данного класса можно наследовать разные классы, считывающие конфигурацию (например из текстового файла, JSON, из интернета и т. д) в таком случае не нужно менять остальные классы при добавлении нового способа задания конфигурации. Класс *FileConfig* является наследником *IConfig'a* и отвечает за считывание конфигурации из файла.

УПРАВЛЕНИЕ: Интерфейс *IControler* необходим, для того чтобы была возможность создавать классы, отвечающие за новый способ ввода команд без изменения существующего кода. Интерфейс имеет один чистый виртуальный метод *get_command(),* который будет реализован в классах наследниках. Класс *ConsoleControler,* отвечает за считывание из стандартного консольного потока ввода.

МОСТ УПРАВЛЕНИЯ: Создан *ControlBridge* – наследник *MediatorObject*, конструктор класса принимает в качестве аргументов объект тип *map*, отвечающий за интерпретацию команд пользователя и указатель на *IControler*. Метод класса запрашивает у пользователя следующую команду интерпретирует её и сообщает медиатору, чтобы тот забрал команду. Объект данного класса хранится в поле класса *Mediator'a*.

UML диаграмма классов представлена в приложении А.

Выводы.

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

ПРИЛОЖЕНИЕ А

UML диаграмма классов