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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

Тема: управление, разделение на уровни абстракции

Студентка гр. 0382 _____ Андрющенко К.С

Преподаватель _____ Жангиров Т. Р

Санкт-Петербург

2021

Цель работы.

Создание классов, осуществляющих взаимодействие между пользовательским заданием команд и их дальнейшем исполнением в классе, управляющем процессом игры.

Задание.

Необходимо организовать управление игрой (номинально через CLI). При управлении игрой с клавиатуры должна считываться нажатая клавиша, после чего происходит перемещение игрок или его взаимодействия с другими элементами поля.

Требования.

- Реализовать управление игрой. Считывание нажатий клавиш не должно происходить в классе игры, а должно происходить в отдельном наборе классов.
- Клавиши управления не должны жестко определяться в коде. Например, это можно определить в отдельном классе.
- Классы управления игрой не должны напрямую взаимодействовать с элементами игры (поле, клетки, элементы на клетках)
- Игру можно запустить и пройти.

Выполнение работы.

В ходе работы реализуется паттерн проектирования Команда.

UML-диаграмма для данной лабораторной работы см. Рисунок 1.

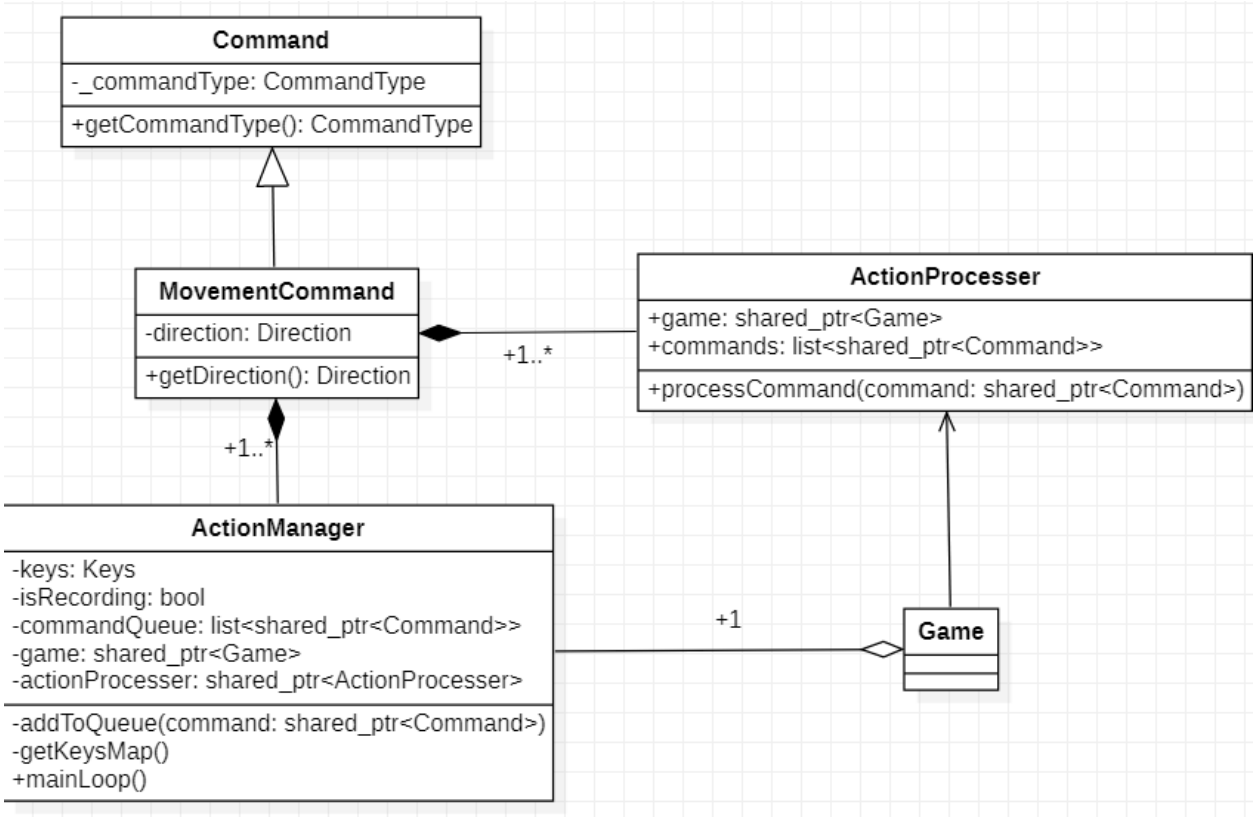

Рисунок 1 – UML-диаграмма лабораторной работы №5

Описание диаграммы.

Описание диаграммы.

Классы, созданные и модифицированные в данной лабораторной работе,

их методы и свойства.

Класс Command:

Заголовочный файл см. ПРИЛОЖЕНИЕ А.

Класс Command, является общим интерфейсом для команды.

Свойства класса.

CommandType _commandType – тип команды.

Методы класса.

virtual CommandType getCommandType() – функция, возвращающая тип команды ;

Перечисление enum CommandType содержит типы комманд.

Movement– Команды движения.

Abort–Завершение процесса игры.

Классы MovementCommand: Заголовочный файл см. Ошибка! Источник ссылки не найден..

Наследник класса команды.

Служит для создания команды движения в определенном направлении.

Свойства класса:

Direction direction – Направление движения.

Методы:

Direction getDirection() – Получение направления.

Класс ActionManager:

Менеджер команд игры. Контейнер команд (композиция). Агрегирует класс управления игрой и исполнитель команд.

Свойства:

struct Keys – Поля структуры хранят номера управляющих клавиш.

bool isRecording – Состояние записи команд.

std::list<std::shared_ptr<Command>> commandQueue – очередь команд.

Методы:

getKeysMap() – Установка управления. Занесение номеров клавиш соответствующих команд в структуру.

void addToQueue(std::shared_ptr<Command> command) – Добавление команд в очередь.

Класс ActionProcesser: Заголовочный файл см. Ошибка! Источник ссылки не найден..

Проверяет введенные команды, управляет очередью и передает действие движения в класс управления игрой.

Агрегирует класс управления игрой. Является контейнером команд (композиция).

Свойства:

std::list<std::shared_ptr<Command>> commandQueue – очередь команд.

void processCommand(std::shared_ptr<Command> command) – Передача команды в класс игры, либо завершение игры.

void mainLoop() – Главный цикл игры.

Проверка классов.

Приложим скриншоты тестов, находящихся в main(), результат и при необходимости памяти после выполнения тестов. Проверим выполнение корректной команды и последовательности команд. Проверим установку команд и результат ввода неправильной команды. Выедем результат игры.

Main():

```
ActionManager actionManager;
actionManager.mainLoop();
```

Рисунок 2 – Главный цикл игры.

Output(консоль):

```
2 ent   0       3       0       0
1       0       4       4       0
1       3       0       1       0
0       0       0       0       0 ext

Enter ESC command
Enter UP command
Enter DOWN command
Enter LEFT command
Enter RIGHT command
Enter RECORD command
Enter your command
Non traversable for enemy
0 ent   2       0       4       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Non traversable for enemy
0 ent   0       2       4       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
NON TRAVERSABLE
Non traversable for enemy
0 ent   0       2       4       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
NON TRAVERSABLE
Non traversable for enemy
0 ent   0       2       4       0
0       0       1       0       0
0       4       3       1       0
```

```
0       0       0       0       0 ext

Enter your command
NON TRAVERSABLE
Non traversable for enemy
0 ent   0       2       4       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Non traversable for enemy
0 ent   0       2       4       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Non traversable for enemy
0 ent   2       0       4       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Non traversable for enemy
0 ent   0       2       4       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
MOVEMENT: Player's current health is max

MOVEMENT: Player's current defence: 10

MOVEMENT: Player's current attack: 20

MOVEMENT: Attack: 0
```

```
MOVEMENT: Defense: 5

MOVEMENT: Health: 1

Non traversable for enemy
0 ent   0       0       2       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Non traversable for enemy
0 ent   0       0       2       0
0       0       1       0       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Non traversable for enemy
0 ent   0       0       0       0
0       0       1       2       0
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Enter your command
Non traversable for enemy
0 ent   0       0       0       0
0       0       1       0       2
0       4       3       1       0
0       0       0       0       0 ext

Enter your command
Non traversable for enemy
0 ent   0       0       0       0
0       0       1       0       0
0       4       3       1       2
0       0       0       0       0 ext
```

```
Enter your command
Conditions are not met!
Enemy interaction: 0
Item interaction: 1
```

Вывод.

В ходе работы были написаны классы обработки команд пользователя для данной игры. Реализован паттерн Команда. Описаны крайние случаи.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл Command.hpp

```
#pragma once

enum CommandType {
  Movement,
  Abort
};

class Command
{
private:
  CommandType _commandType;
public:
  virtual CommandType getCommandType();
  Command(CommandType commandType);
  ~Command();
};

class MovementCommand : public Command
{
public:
  MovementCommand(Direction direction) :
Command(CommandType::Movement), direction(direction) { }

  ~MovementCommand() { };

  Direction getDirection()
  {
    return direction;
  }

private:
  Direction direction;
};
```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл ActionManager.hpp

```
#pragma once

#include "ActionProcesser.hpp"
#include "Command.hpp"

struct Keys
{
  char ESC;
  char UP;
  char DOWN;
  char LEFT;
  char RIGHT;
  char RECORD;
};

class ActionProcesser;
class ActionManager
{
private:
  std::shared_ptr<ActionProcesser> actionProcesser;
  std::shared_ptr<Game> game;

  bool isRecording = false;
  void addToQueue(std::shared_ptr<Command> command);
  void getKeysMap();
  std::list<std::shared_ptr<Command>> commandQueue;

  Keys keys;
public:
  ActionManager();
  void mainLoop();

  ~ActionManager();
};
```

ПРИЛОЖЕНИЕ С
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл ActionProcesser.hpp

```
#pragma once
#include "../Game.hpp"
#include "Command.hpp"
class ActionProcesser
{
private:
  /* data */
public:
  ActionProcesser(std::shared_ptr<Game> game);
  std::shared_ptr<Game> game;

  std::list<std::shared_ptr<Command>> commands;
  void processCommand(std::shared_ptr<Command> command);
  ~ActionProcesser();
};
```