

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: Логическое разделение классов

Студент гр. 8303

Удод М.Н.

Преподаватель

Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Научиться логическому разделению классов

Задание.

Разработать и реализовать набора классов для взаимодействия пользователя с юнитами и базой. Основные требования:

- Должен быть реализован функционал управления юнитами
- Должен быть реализован функционал управления базой

Ход выполнения работы.

1. Был реализован функционал управления юнитами. Так, с помощью команд `show unit`, `move unit`, `create unit`, `attack unit` можно показывать информацию, перемещать, создавать или атаковать юнитами.
2. Была добавлена команда `show base` для просмотра информации о базе.
3. Был добавлен класс `GameFacade`, который реализует простой функционал управления игрой. Так, функция `nextTurn` производит всю работу по вводу команды, ее распознавания и исполнения.
4. Так как управление происходит через команды, то им требуется узнавать информацию о игровом поле и изменять его. Именно по этому в качестве посредника между объектами на поле и командами выступает класс `GameField`. Так, команда `attack unit 0 0 1 1` получает юнит, расположенный по координатам (0, 0), сообщает ему, что он атакует юнит, расположенный по координатам (1, 1) с помощью метода `attack`. В методе `attack` данная информация передается полю и оно обрабатывает это действие.
5. Команды реализованы в классах, наследующихся от интерфейса `Command`, содержащий единственный метод для выполнения команд.
6. Паттерн «Цепочка обязанностей» реализован в классах, наследующихся от интерфейса `CommandHandler`. На вход метода `handle` таких классов подается команда в виде строк, разделенных по пробелу. Так, `CommandHandler` по первому слову команды определяет, к какому типу команд относится обрабатываемая, и передает команду обработчику следующего уровня.

Примеры.

```

Now player: 0
|. | |. | |. |
|. | |. | |. |
|. | |. | |. |

create base 0 0
Now player: 1
|B| |. | |. |
|. | |. | |. |
|. | |. | |. |

show base 0 0
Base info:
    HP: 100
    Armor: Armor( Damage Absorption: 3 )
    Max Objects Count: 5
Now player: 0
|B| |. | |. |
|. | |. | |. |
|. | |. | |. |

```

```

Now player: 0
|. | |. | |. |
|. | |. | |. |
|. | |. | |. |

create base 0 0
Now player: 1
|B| |. | |. |
|. | |. | |. |
|. | |. | |. |

create base 2 2
Now player: 0
|B| |. | |. |
|. | |. | |. |
|. | |. | |B|

create unit 0 1 1
Now player: 1
|B| |. | |. |
|W| |. | |. |
|. | |. | |B|

show unit 0 1
Unit info:
    HP: 100
    Weapon: Weapon( Damage: 10 )
    Armor: Armor( Damage Absorption: 1 )

```

```

Now player: 0
|. | |. | |. |
|. | |. | |. |
|. | |. | |. |

create base 0 0
Now player: 1
|B| |. | |. |
|. | |. | |. |
|. | |. | |. |

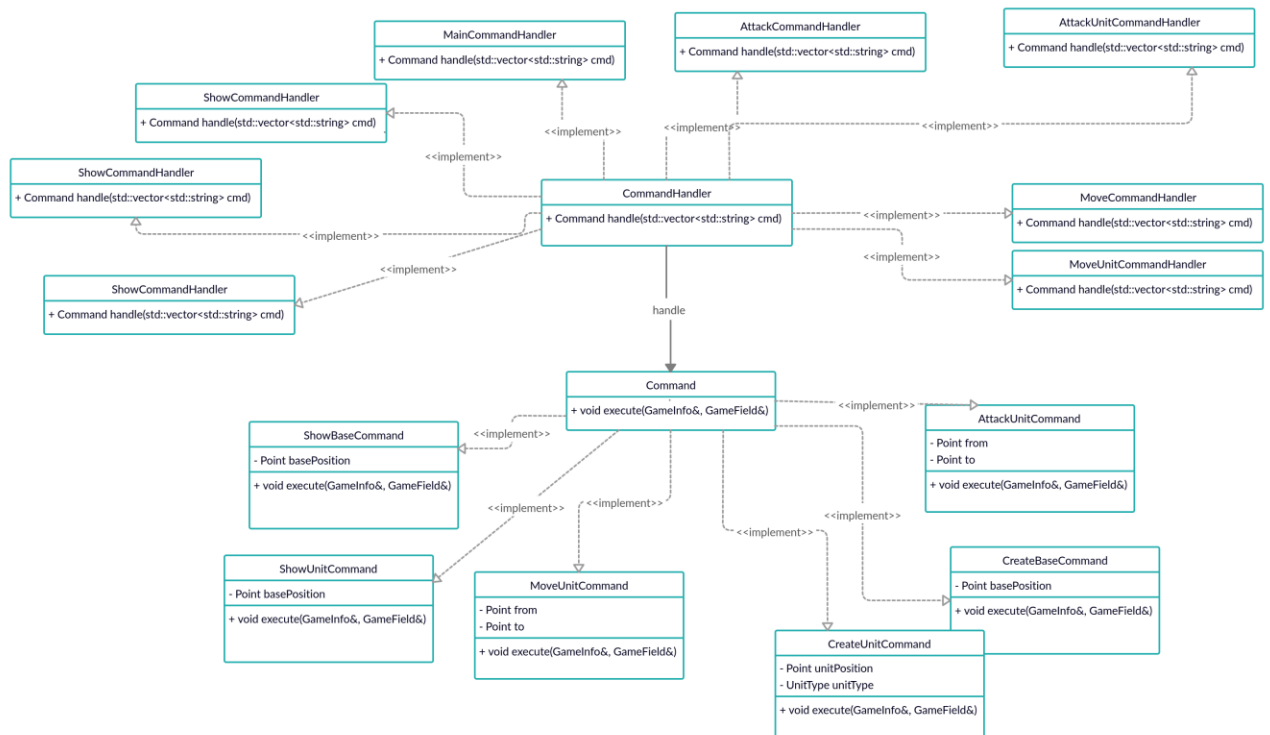
create base 2 2
Now player: 0
|B| |. | |. |
|. | |. | |. |
|. | |. | |B|

create unit 1 0 1
Now player: 1
|B| |W| |. |
|. | |. | |. |
|. | |. | |B|

move unit 1 0 0 1
Base: Unit 0x562338151cc0 moving
Now player: 0
|B| |. | |. |
|W| |. | |. |
|. | |. | |B|

```

UML-диаграмма.



Вывод.

В ходе выполнения лабораторной работы было изучено логическое разделения классов с помощью реализации таких паттернов, как «Команда», «Посредник», «Фасад» и «Цепочка обязанностей».

Приложение А. Исходный код программы

1. Main.cpp

```
#include <iostream>
#include "GameFacade.h"

int main() {

    auto unit = new FireMage;

    GameFacade game(2, 3, 3);

    while (!game.isOver()){

        std::cout << game;
        game.nextTurn();

    }

    return 0;
}
```

2. GameFacade.h

```
#ifndef UNTITLED13_GAMEFACADE_H
#define UNTITLED13_GAMEFACADE_H

#include <sstream>
#include "GameInfo.h"
#include "UI/Commands/Command.h"
#include "UI/Commands/MainCommand.h"

class GameFacade: public GameInfo {

public:

    GameFacade(int playersCount, int fieldWidth, int fieldHeight):
    GameInfo(playersCount, fieldWidth, fieldWidth){}
    void nextTurn(){

        std::string commandString;
        std::getline(std::cin, commandString);

        std::vector <std::string> commandSplitted;
        std::stringstream ss(commandString);
        std::string commandWord;
        while (ss >> commandWord)
            commandSplitted.push_back(commandWord);

        CommandPtr command = MainCommandHandler().handle(commandSplitted);
        command->execute(*this, gameField);

        nextUser();

    }

}
```

```

void nextUser(){
    nowPlayerIndex++;
    if (nowPlayerIndex >= playersBases.size())
        nowPlayerIndex = 0;
}

friend std::ostream &operator<<(std::ostream &stream, const GameFacade
&game){

    stream << "Now player: " << game.nowPlayerIndex << std::endl;
    stream << game.gameField << std::endl;
    return stream;

}

};

#endif //UNTITLED13_GAMEFACADE_H

```

3. GameInfo.h

```

#ifndef UNTITLED13_GAMEINFO_H
#define UNTITLED13_GAMEINFO_H

#include "GameField/GameField.h"

class GameInfo {
protected:
    GameField gameField;
    std::vector<Base*> playersBases;
    int nowPlayerIndex;

public:
    GameInfo(int playersCount, int fieldWidth, int fieldHeight):
        gameField(fieldHeight, fieldWidth),
        playersBases(playersCount, nullptr),
        nowPlayerIndex(0)
    {}

    Base *getNowPlayerBase(){ return playersBases[nowPlayerIndex]; }
    bool setNowPlayerBase(Base *base){
        if (playersBases[nowPlayerIndex]){
            return false;
        } else{
            playersBases[nowPlayerIndex] = base;
        }
    }
}

```

```

    }
    bool isOver() {
        return false;
    }
};

#endif //UNTITLED13_GAMEINFO_H

```

4. Command.h

```

#ifndef UNTITLED13_COMMAND_H
#define UNTITLED13_COMMAND_H

#include <string>
#include <memory>
#include "../GameField/GameField.h"
#include "../GameInfo.h"

class Command {
public:
    virtual void execute(GameInfo &gameInfo, GameField &gameField){}
};

typedef std::unique_ptr<Command> CommandPtr;

class CommandHandler{
private:
    virtual CommandPtr handle(std::vector<std::string> &cmd)=0;
};

#endif //UNTITLED13_COMMAND_H

```