МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный

электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)

Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Объектно-ориентированное программирование»

Тема: Интерфейсы классов, взаимодействие классов, перегрузка операций

Студент гр. 8382 _____ Терехов А.Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Создать класс базы и набор классов ландшафта и нейтральных элементов.

Задание.

Разработать и реализовать набор классов:

- Класс базы
- Набор классов ландшафта карты
- Набор классов нейтральных объектов поля

Класс базы должен отвечать за создание юнитов, а также учитывать юнитов, относящихся к текущей базе. Основные требования к классу база:

- База должна размещаться на поле
- Методы для создания юнитов
- Учет юнитов, и реакция на их уничтожение и создание
- База должна обладать характеристиками такими, как здоровье, максимальное количество юнитов, которые могут быть одновременно созданы на базе, и.т.д.

Набор классов ландшафта определяют вид поля. Основные требования к классам ландшафта:

- Должно быть создано минимум 3 типа ландшафта
- Все классы ландшафта должны иметь как минимум один интерфейс
- Ландшафт должен влиять на юнитов (например, возможно пройти по клетке с определенным ландшафтом или запрет для атаки определенного типа юнитов)
- На каждой клетке поля должен быть определенный тип ландшафта

Набор классов нейтральных объектов представляют объекты, располагаемые на поле и с которыми могут взаимодействие юнитов. Основные требования к классам нейтральных объектов поля:

- Создано не менее 4 типов нейтральных объектов
- Взаимодействие юнитов с нейтральными объектами, должно быть

реализовано в виде перегрузки операций

• Классы нейтральных объектов должны иметь как минимум один общий интерфейс

Ход работы.

Для базы реализован класс Base, выглядящий следующим образом.

```
class Base : public AbstractObject{
    int health;
    std::vector<int> unitsID;
public:
    int getHealth() const;
    explicit Base(std::vector<int> unitsID);
    int takeDamage(int dam);
    void addEnemy(int id);
    void killEnemy(int id);
    std::vector<int> getID();
    void printID();
};
```

Данный класс наследуется от абстрактного объекта, который уже хранит графическое представление и метод вывода этого представления. У базы есть здоровье и сведения о принадлежащих ей юнитах. База может получать урон, реагировать на убийства своих юнитов, а также создавать юнитов при критическом уровне здоровья.

Для ландшафта был создан набор классов унаследованный от абстрактного объекта. Класс ClosedCells объединяет классы, представляющие непроходимые клетки. Класс Road описывает клетки доступные для прохождения.

```
class ClosedCells : public AbstractObject {
public:
    ClosedCells(char pict);
};

class Tree : public ClosedCells {
public:
    Tree();
};

class Rock : public ClosedCells {
public:
    Rock();
};

class Wall : public ClosedCells {
```

```
public:
    Wall();
};

class Road : public AbstractObject {
public:
    Road(char pict = '_');
};
```

Для нейтральных объектов были созданы классы.

```
class NeutralObject : public AbstractObject{
public:
    explicit NeutralObject(char pict);
};

class HealthBox : public NeutralObject{
public:
    HealthBox();
};

class ArmorBox : public NeutralObject{
public:
    ArmorBox();
};

class RandomBox : public NeutralObject{
public:
    RandomBox();
};

class RareBox : public NeutralObject{
public:
    RareBox();
};
```

Класс NeutralObject представлят собой абстракцию, объединяющую виды коробок. HealthBox – принесет от 10 до 19 очков здоровья. ArmorBox – прибавит 1 к броне. RandomBox – принесет либо 1 очко к атаке либо нанесет урон. RareBox – дает юниту по 5 очков к броне и атаке, но появляется на карте крайне редко.

Для взаимодействия юнитов с нейтральными объектами был переопределен оператор +=.

```
Unit &Unit::operator+=(char n) {
    std::cout << n << std::endl;
    switch (n) {
        case '+':
```

```
                this->health += std::rand() % 10 + 10;
                break;
            case 'o':
                this->armor += 1;
                break;
            case 'X':
                this->armor += 5;
                this->damage += 5;
                break;
            case '?':
                switch (std::rand() % 2) {
                    case 0:
                        this->takeDamage(50);
                        break;
                    case 1:
                        this->damage += 1;
                        break;
                    default:
                        break;
                }
                break;
            default:
                break;
        }
        return *this;
    }
```

Вывод.

В ходе работы были получены навыки в перегрузке операторов, и создании интерфейсов.