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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Объектно-ориентированное программирование»
Тема: Интерфейсы классов; взаимодействие классов; перегрузка операций

Студент гр. 8304 _____ Мухин А. М.

Преподаватель _____ Размочаева Н. В.

Санкт-Петербург

Цель работы.

Разработать и реализовать набор классов:

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

Задание.

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

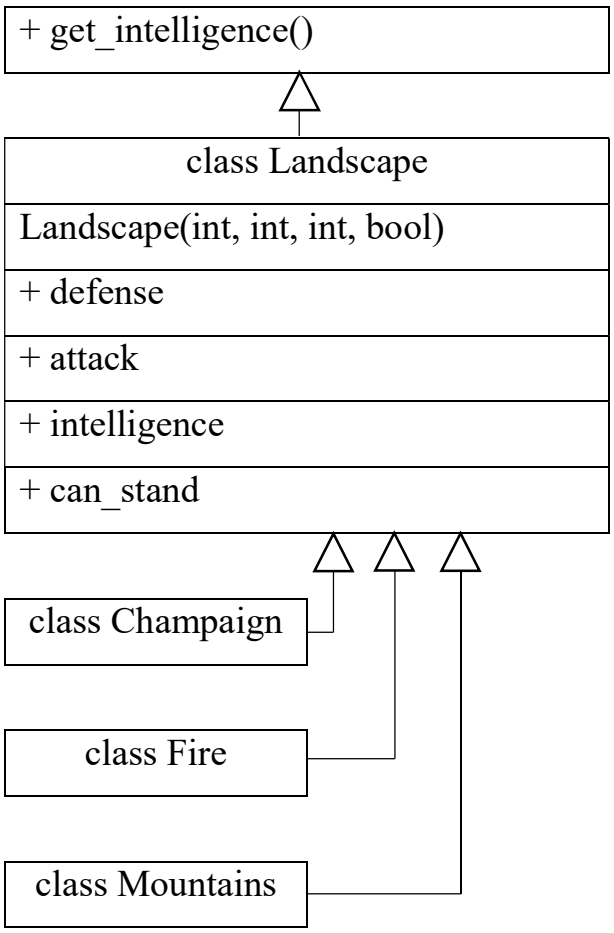
Выполнение работы.

Для того, чтобы понять, как реализованы необходимые классы рассмотрим по очереди их UML диаграммы.

class Base
- max_size
- units
+ x
+ y
Base(int)
+ add_unit(std::shared_ptr<Unit>&)
+ get_base()
+ defense
+ attack
+ current_size

Как видно из диаграммы класс базы содержит в себе текущий и максимальный размеры базы, координаты её расположения, метод добавления юнита и метод просмотра базы, а также массив из умных указателей на юниты базы.

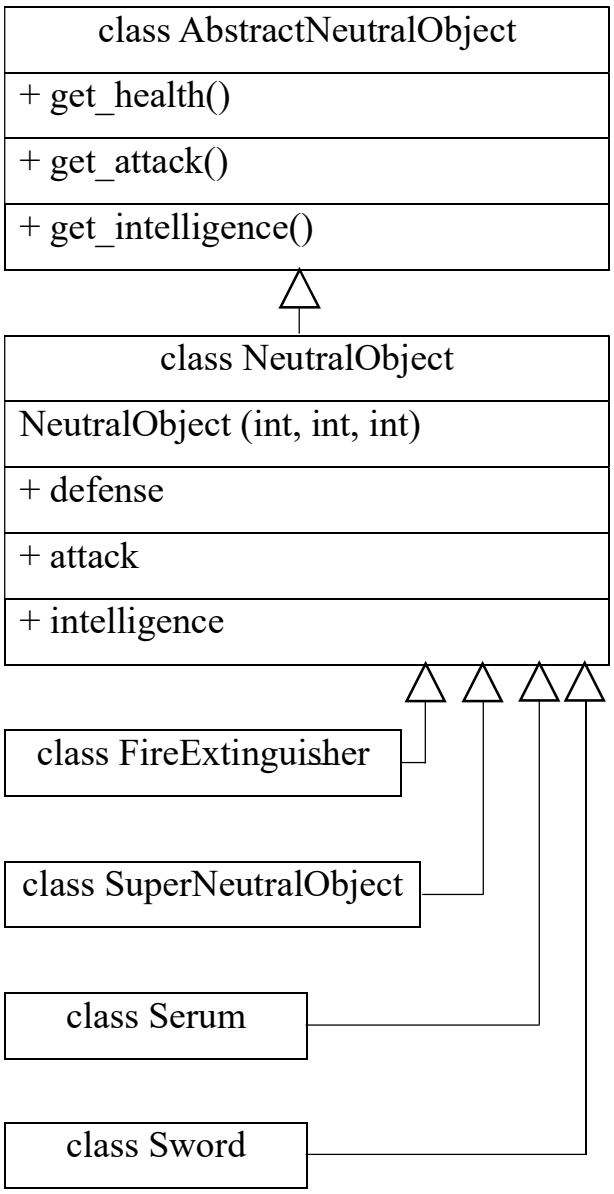
class AbstractLandscape
+ get_health()
+ get_attack()



Класс ландшафта содержит в себе 4 основных параметры, которые заполняются при создании наследуемого класса: defense, attack, intelligence, показывает, может ли элемент стоять на клетки с таким ландшафтом соответственно.

Для лучшего понимая, давайте рассмотрим один из наследуемых классов Mountain, единственный публичный метод которого создаёт класс родителя с необходимым конструктором. В случае с классом Mountain он выглядит так: `Mountains() : Landscape(0, 0, 0, false) {}`. То есть, при каком-либо действии юнита на клетке с таким ландшафтом, к здоровью юнита, защите и интеллекту ничего не прибавится, однако наступить на эту клетку он не сможет.

Кроме класса Mountain наследниками у класса Landscape выступают класс Fire и Champaign с соответствующими конструкторами: `Fire() : Landscape(-10, 0, 0, true) {}` **и** `Champaign() : Landscape(0, 0, 0, true) {}`.



Класс нейтрального объекта полностью идентичен классу ландшафта, однако имеет две отличия:

1) После взаимодействия юнита и нейтрального объекта, нейтральный объект уничтожается с поля.
2) Не существует нейтральных объектов, запрещающих стоять юнитам с ними на одной клетке.

Также, для лучшего понимая рассмотрим одного из наследников класса NeutralObject, класс FireExtinguisher(огнетушитель). В противовес классу ландшафта Fire, который отнимает 10 единиц здоровья у юнита, который встанет с ним на одну клетку, огнетушитель наоборот, даёт 10 единиц здоровья и его единственный публичный метод – конструктор, вызываемый в конкретными

параметрами здоровья, брони и интеллекта выглядит следующим образом:

Кроме огнетушителя, в роли нейтральных объектов выступают ещё меч, супер нейтральный объект и сыворотка, с соответствующими конструкторами:

```
Sword() : NeutralObject(0, 10, 0) {}
S
```

Тестирование.

Тестирование проводилось с помощью юнит тестов и заголовочного файла Юнит тесты можно найти в файле, который располагается по следующему пути:

Выводы.

В ходе данный лабораторной работ, мы научились использовать полиморфизм и наследование, узнали в каком порядке создаются экземпляры наследуемых классов и как явно вызвать нужный нам конструктор для базового класса. А также грамотно подходить к архитектурным решениям, при написании кода программы. Использовали новую метод тестирования, через catch, а не через google_test, как в предыдущей лабораторной работе.