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. 8381 _____ Почаев Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Разработать и реализовать класс базы, набор классов ландшафта, набор классов нейтральных объектов поля.

Задание.

Основные требования.

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.

Дополнительные требования.

- Для хранения информации о юнитах в классе базы используется паттерн “Компоновщик” / Использование “Легковеса” для хранения общих характеристик юнитов.
- Для наблюдения над юнитами в классе база используется паттерн “Наблюдатель”.
- Для взаимодействия ландшафта с юнитами используется паттерн “Прокси”.
- Для взаимодействия одного типа нейтрального объекта с разными типами юнитов используется паттерн “Стратегия”.

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки CLion, для компиляции и отладки использовалась UNIX-подобная среда Cygwi. Были задействованы пакеты GCC, CMake, а также GDB.

Реализованные классы

Классы, добавленные в программу в данной лабораторной работе и их функционал представлены в табл. 1. В ней приведено общее описание классов, отдельные моменты пояснены в комментариях к коду.

Таблица 1 – Основные добавленные классы

Класс	**Назначение**
GameBase (абстрактны класс базы)	Местоположение: ./Bases Класс хранит текущее здоровье базы, фабрику, отвечающую за создание элементов, специальный класс UnitStorekeeper, отвечающий за подсчёт элементов в базе (описан ниже), вектор умных указателей на боевые подразделения: легионы или отряды. Является “подписчиком” создаваемых юнитов через интерфейс UnitObserver и своевременно фиксирует их уничтожение.
UnitStorekeeper (класс параметра другого класса, учёт кол-ва юнитов)	Местоположение: ./Auxiliary functionality Класс хранит в себе словарь, где каждому идентификатору типа юнита (std::string) отвечает структура MaxCurrUnitQuantity, сохраняющая максимальное количество юнитов определённого типа и их кол-во на данных момент. В данном классе реализованы все необходимые методы для добавления указанного количества юнитов, проверки возможности осуществления данного действия и т.д.
BaseBuilder + BaseMaster (дополнительно)	Представляют собой интерфейсы “строителя” и “директора” паттерна Строитель. Используется для последовательного конструирования различных реализаций класса базы с инициализаций шаблонных фабрик необходимыми типами юнитов.
HellBaseBuilder + HumanBaseBuilder	Реализуют указанные выше интерфейсы паттерна Строитель.
HellBase + HumanBase	Наследуют описанный выше абстрактный класс базы.

Landscape (абстрактный класс ландшафта)	Местоположение: ./Landscape Хранит в себе вектора идентификаторов (std::string), обозначающие юниты для которых: запрещён проход по данной местности, перемещение на данный тип ландшафта даёт бонус действия, атака с данного типа ландшафта запрещена.
Champaign + Forest + Mountains	Наследники абстрактного класса ландшафта – представляют соответствующие списки юнитов.
NeutralObject	Местоположение: ./Neutral Интерфейс нейтральных объектов. Является частью паттерна Стратегия.
NeutralObjectContext	Класс контекста для паттерна Стратегия, хранящий в себе указатель на объект и позволяющий подставлять необходимый нейтральный элемент при взаимодействии с помощью оператора +=.
Enchanted Robe + EnergyPotion + Legendary Weapon + Poison	Реализация соответствующего типа перегрузки оператора += от интерфейса NeutralObject. Вызывает изменение определённых параметром юнита (через его методы)
UnitMeleeAttackMediator	Местоположение: ./Auxiliary functionality Класс медиатор для взаимодействия (атаки) двух юнитов. Связывает между собой Unit и GameFieldProxy (описан далее) для корректной проверки всех параметров при запросе атаки от одного юнита другого: на этапе proxy осуществляется проверка на ландшафт, на этапе GameField – на расстояние и наличие юнита.

CompositeUnit	Реализация паттерна Компоновщик: позволяет через общий абстрактный класс Unit с другими одиночными реализациями создавать подразделения, состоящие из юнитов одиночного типа. Legion – набор разных типов юнитов (фикс.), Squad – отряд юнитов выбираемого типа и количества.
GameFieldProxy	Реализация паттерна Прокси в двух ипостасях: cache и protection. Первая выносит хранение ландшафта и нейтральных объектов вне логики основного класса поля. Обусловлена это неизменяемостью данных объектов и отсутствием необходимости плодить экземпляры данных классов. Таким образом в классе хранится словарь данных объектов типа: идентификатор – объект и словарь их размещения на поле: координаты – ссылка на объект из предыдущего словаря. При необходимости выполнения определённой проверки или применения эффекта нейтрального объекта идёт обращения по ссылкам к их единственным экземплярам. Защитная составляющая состоит в дополнительных проверках на ландшафт при перемещении и атаке, а также применении нейтрального объекта при его наличии, когда юнит переходит на определённую клетку. Является “подписчиком” размещаемых юнитов, при сообщении об их смерти зачищает клетку, на которой они находились.

Выводы.

В ходе выполнения лабораторной работы были написаны требуемые класс, а также реализовано необходимое взаимодействие между юнитами, базой и юнитами, юнитами и нейтральными объектами поля и т.д.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ. MAIN.CPP

```
#include <iostream>
#include "Tests/examples.h"

int main()
{
    // fieldBasedTest();
    // ObserverDeathTest();
    // landscapeTest();
    unitInteractionTest();

    return 0;
}
```