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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Объектно-ориентированное программирование»

Тема: Интерфейсы классов, взаимодействие классов, перегрузка операций

Студент гр. 8382 _____ Янкин Д.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Получить навыки реализации интерфейсов, перегрузки операций и взаимодействий классов.

Постановка задачи.

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

Выполнение работы.

Реализован класс базы. В качестве параметров здоровья брони и атаки в нем используются те же классы, что и у юнитов. База может создавать юнитов и ведет их учет. При уничтожении базы все юниты, прописанные на ней, уничтожаются. Смотреть директорию /Base.

Были реализованы несколько классов ландшафта, имеющий общий интерфейс. По некоторым типа ландшафта нельзя ходить, другие различаются стоимостью перемещения. Смотреть директорию /Landscape.

Были реализованы несколько нейтральный объектов, влияющие на характеристики юнитов. Воздействие имеет перманентный эффект и срабатывает в тот момент, когда юнит наступает на клетку с нейтральным объектом (даже если эта клетка была промежуточной в маршруте юнита и графически его попадание на клетку не отображалось). Чтобы получить возможность модификации характеристик юнита, методы нейтральных объектов объявлены дружественными к юниту. Смотреть /Neutrals/NeutralObject.

В реализованной программе есть ужасный, но терпимый графический интерфейс, который можно использовать для генерации примеров на ходу.

Взаимодействие нейтральный объектов реализовано через перегрузку операторов.

Паттерн Компоновщик. Был реализован класс UnitGroup, имеющий общий интерфейс с Unit и хранящий в себе набор юнитов. Он хранит именно юниты, а не их интерфейс, чтобы не допустить возможность хранения группы в группе. С точки зрения логики игры это нужно. В обычном состоянии за существование юнита отвечает класс базы. Когда юнит входит в состав группы, он убирает себя из подписчиков базы и уведомляет класс Game об уничтожении (на самом деле

уничтожения не происходит), после этого время жизни юнита полностью зависит от времени жизни группы. Это сделано с целью того, чтобы множество юнитов, состоящее в группе, считалось за одного юнита в базе и в поле. Объединять можно только юнитов одного типа. Смотреть /Unit/UnitGroup.h.

Паттерн Наблюдатель. Класс базы имеет в себе контейнер, хранящий множество всех прописанных в ней юнитов и предоставляет методы, чтобы добавлять в этот контейнер объекты и удалять. При уничтожении база уничтожает всех прописанных на ней юнитов. Таким образом, юниты являются подписчиками, а база – издателем. Смотреть Base/Base.h, Unit/Unit.h, /Unit/UnitGroup.h.

Паттерн Прокси. Был реализован класс LandscapeProxy. Он имеет тот же интерфейс, что и обычный ландшафт. При обращении к нему, LandscapeProxy переадресовывает вызов настоящему ландшафту, при этом вызывая свой приватный метод log(). В данный момент этот метод ничего не делает, однако в будущем его можно использовать для логирования обращений к ландшафту. Таким образом, в данный момент этот прокси является абсолютно бесполезным – дело в том, что никаких осмысленных применений прокси в данном контексте придумать не удалось. Смотреть Landscape/LandscapeProxy.h.

Паттерн Стратегия. Реализованные ранее нейтральные объекты были превращены в стратегии. Отдельно создан класс контекста, размещаемый на поле (раньше на поле рамещались именно нейтральные объекты, которые теперь стратегии). Стратегия выбирается в зависимости от того, какого типа юнит проходит по клетке. Смотреть Neutrals/NeutralContext.h.

Выводы.

Были успешно получены навыки реализации интерфейсов, перегрузки операций и взаимодействий классов.