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

ОТЧЕТ

по практической работе №2
по дисциплине «Объектно-ориентированное программирование»
Тема: Интерфейсы классов, взаимодействие классов, перегрузка операций.

Студент гр. 8383 _____ Федоров И.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение интерфейсов классов, реализация перегрузки операторов и взаимодействия классов с помощью паттернов проетирования.

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.
- *Для хранения информации о юнитах в классе базы используется паттерн "Компановщик"/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.

Был создан класс ***Base***, предназначенный для создания юнитов. **UML** диаграмма для базы представлена на рис. 1. Данный класс содержит в себе следующие ключевые поля и методы:

1) Указатель на абстрактную фабрику создания юнитов *AFactory* (была реализована в 1-й лабораторной работе, содержит в себе три метода для создания юнитов разных типов). Метод по созданию юнитов базы обращается к соответствующему методу фабрики.
2) Вектор указатель на юнитов, которые были создана данной базой.
3) Число созданных юнитов, а также допустимый лимит создания.

4) Очки здоровье *hitPoint*. Координаты расположения базы содержатся в классе GameObj (класс из 1-й работы), от которого унаследован класс **Base**.
5) Методы добавления и удаления юнитов из списка.
6) Метод поиска места появления юнита (база создает юнита и помещает его рядом с базой в радиусе клетки, если есть место).

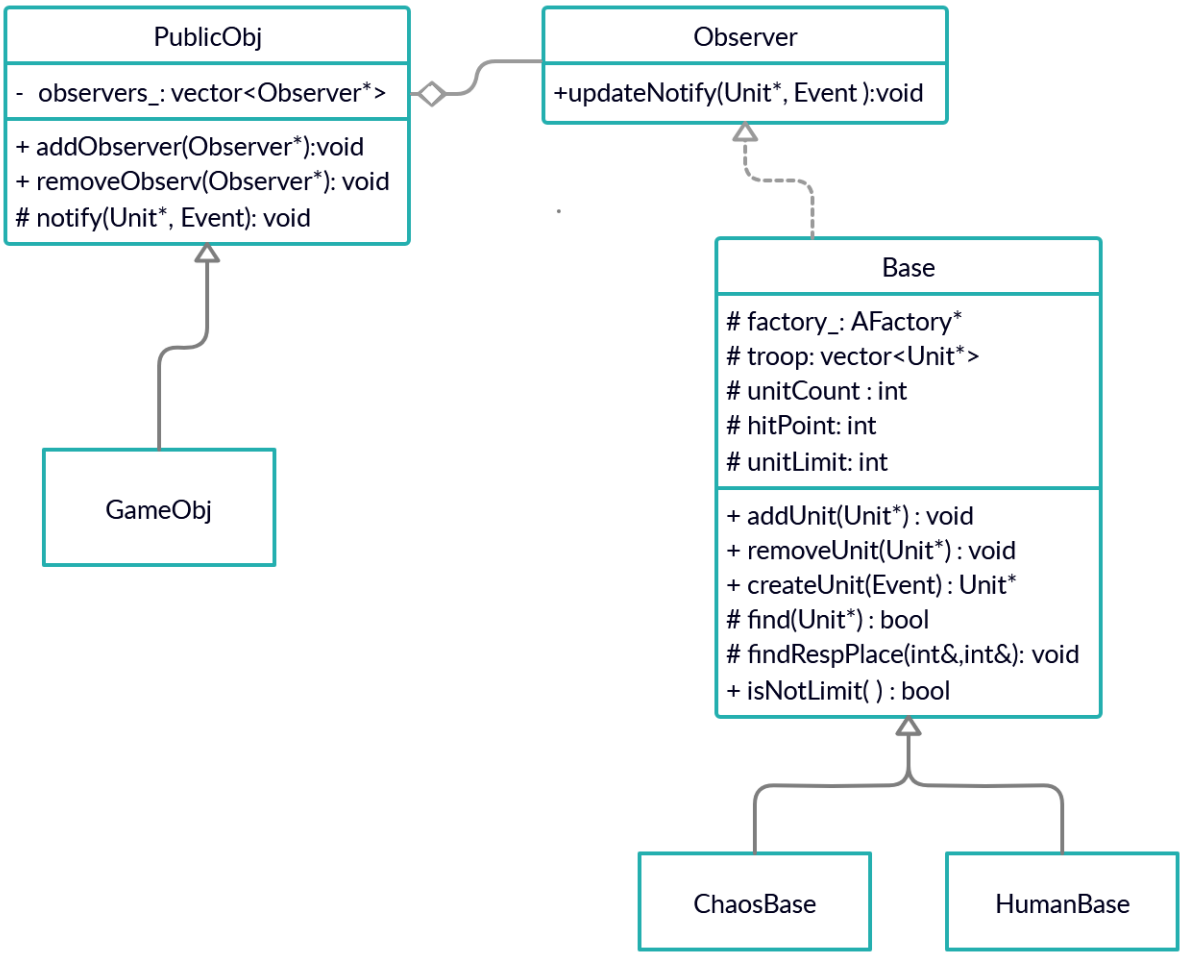

Рисунок 1 - UML диаграмма класса базы

Для наблюдения над юнитами в классе **Base** используется паттерн "Наблюдатель". Для этого был создан абстрактный класс **Observer**, который содержит метод *updateNotify*, принимающий в качестве параметров указатель на юнита, а так же специальный класс *Event*, который отвечает за передачу события.

Так же был реализован класс *PublicObj* (от которого теперь наследуется класс GameObj). Данный класс хранит в себе вектор "наблюдателей", методы

добавления новых и удаления старых наблюдателей, а так же метод оповещения *notify*, который просто проходится по всем вектору и вызывает у "наблюдателей" метод *updateNotify*, передавая им принятое событие. В итоге, при смерти/удалении юнита, тот оповещает базу об этом, и та "вычеркивает" его и списка и уменьшает счетчик созданных юнитов. Аналогично с созданием юнита.

От класса **Base** унаследованы два класса HumanBase и ChaosBase, которые создают соответственно юнитов "людей" или юнитов "хаоса".

Был создан набор классов ландшафтов. UML диаграмма ландшафтов представлена на рис. 2. Для взаимодействия ландшафта с юнитами был использован паттерн "Прокси". Для этого был создан класс-интерфейс *ITerrain,* от которого наследуются классы TerrainProxy Terrain. Интерфейс содержит в себе следующие ключевые абстрактные методы:

1) Метод "открытия" ландшафта. Так, изначально игровое поле является черным (неисследованным). Ландшафт открывается только после того, как юнит зайдет на него (+ в радиусе одной клетки рядом). Поэтому, поле хранит в себе *прокси-класс* ландшафтов, которые создают *настоящие* ландшафты только после их открытия.
2) Метод-геттер, который возвращает стоимость перемещения по данному типу ландшафта в очках. В данном случае если класс закрыт, то прокси-ландшафт просто возвращает значение без участия реального ландшафта, иначе вызывает этот же метод у него.
3) Булевый метод, который отвечает, можно ли пройти по данному ландшафту. Прокси-ландшафт действует аналогично предыдущему методу.

4) Методы добавления и удаления юнита. В данном случае прокси-ландшафт создает реальный ландшафт, и вызывает соответствующий метод у него.
5) Метод воздействия. Определяет воздействие ландшафта на находящегося на нем юнита.

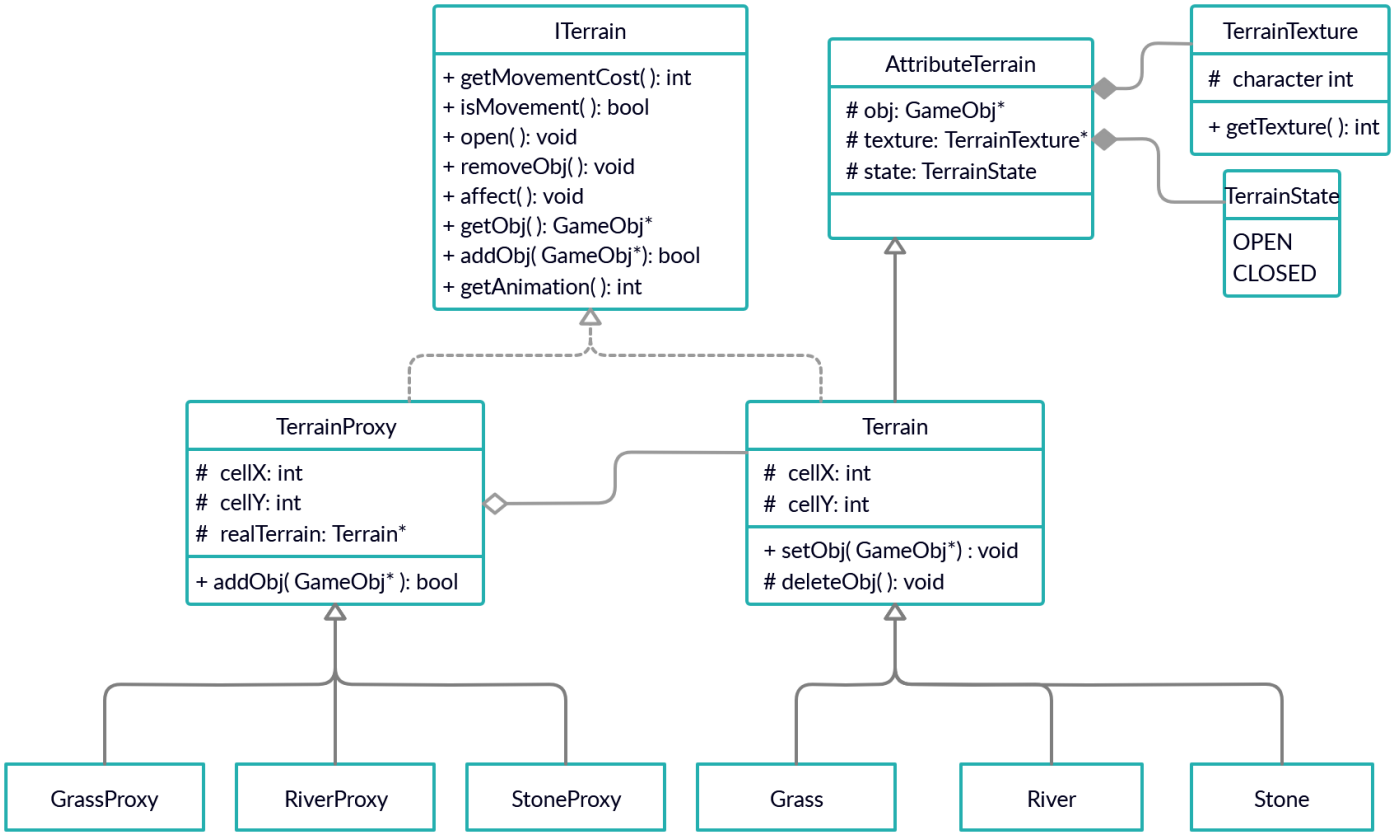

Рисунок 2 - UML диаграмма набора ландшафтов

От класса **Terrain** унаследованы три типа ландшафтов: трава, вода и камень. Трава не оказывает никакого воздействия и является проходимой. Вода блокирует метод атаки у юнита (в воде юнит беззащитен) и является проходимой. Камень является не проходимым ландшафтом.

Был создан набор классов нейтральных объектов. UML диаграмма набора представлена на рис. 3. Был создан абстрактный класс, от которого унаследованы 4 типа нейтральных объектов: мина, аптечка, портал и броня.

Был перегружен оператор += для каждого типа объекта. *Мина* при взаимодействии с юнитом отнимает хп, *аптечка* прибавляет. *Портал* хранит

в себе координаты места, куда перекинет юнита при взаимодействии. *Броня* добавляет очки защиты юниту. Соответствующие перегруженные операторы объявлены как *дружественные* у класс **Unit**.

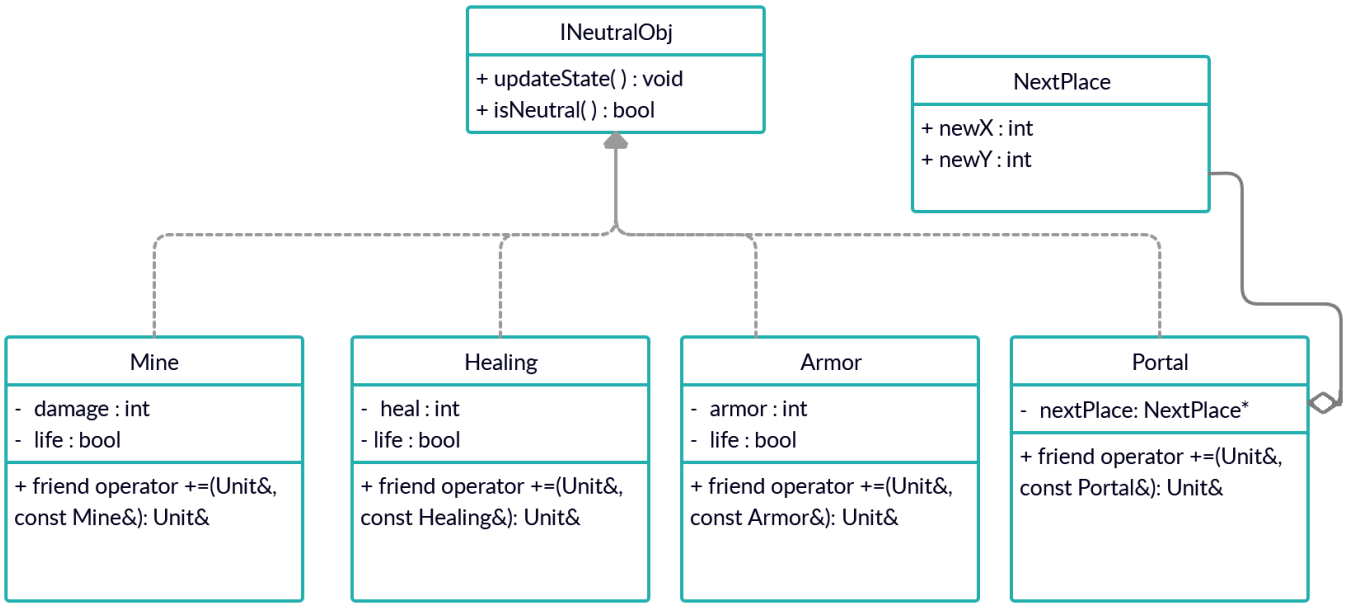

Рисунок 3 - UML диаграмма набора нейтральных объектов

На рисунках 4-8 приведены некоторые примеры работы: открытие карты по мере продвижения, сбор аптечек и мин, размещение базы на поле. Отображения состояния юнита (прибавка к хп или же получени урона) пока не реализована.

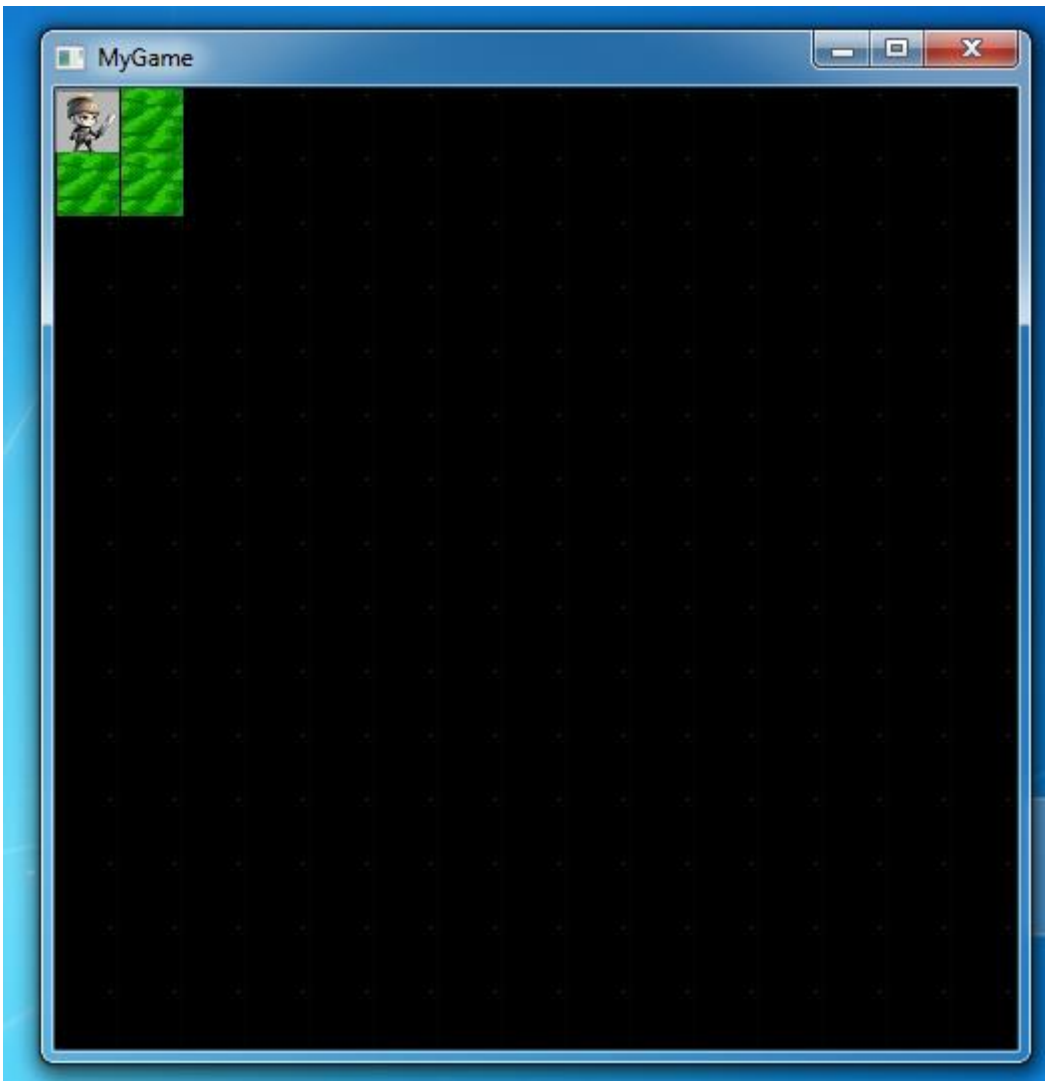

Рисунок 4 - Поле неисследовано.

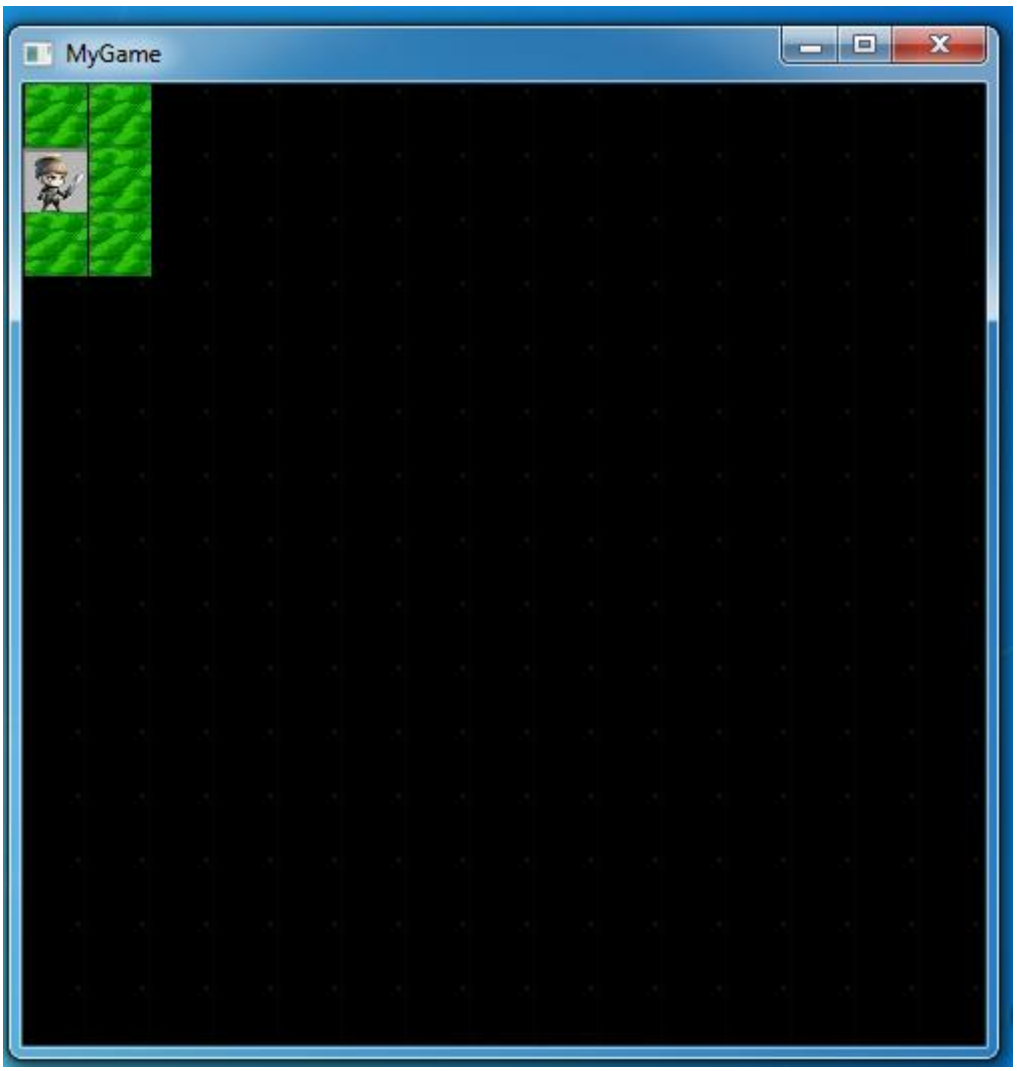

Рисунок 5 - Поле открывается по мере продвижения

Рисунок 6 - База и нейтральные объекты

Выводы.

В ходе выполнения лабораторной работы были реализованы класс базы, наборы классов ландшафтов и нейтральных объектов, примитивная версия паттерном Наблюдатель и Прокси.