**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Логическое разделение классов

|  |  |  |
| --- | --- | --- |
| Студент гр. 8303 |  | Логинов Е.А. |
| Преподаватель |  | Филатов А.Ю. |

Санкт-Петербург

2020

**Цель работы.**

Реализация функционала управления юнитами и базой.

**Ход работы.**

1. Был реализован функционал управления юнитами. Так, при помощи команд createUnit, moveUnit можно создавать и перемещать юнитов.
2. Был реализован функционал управления базой. Так, при помощи команд setBase, baseInfo и можно создавать базу и выводить информацию о ней(здоровье, текущее количество юнитов на базе, максимально возможное кол-во юнитов).
3. Управление программой реализовано через паттерн “Фасад” в класее Façade (файлы Façade.h/.cpp). Класс имеет два метода:

void Facade::createField() – метод создает поле и выводит его на экран. Также указателю на класс паттерна «Цепочка обязанностей» присваиваются обработчики различных команд, которые образуют цепочку команд.

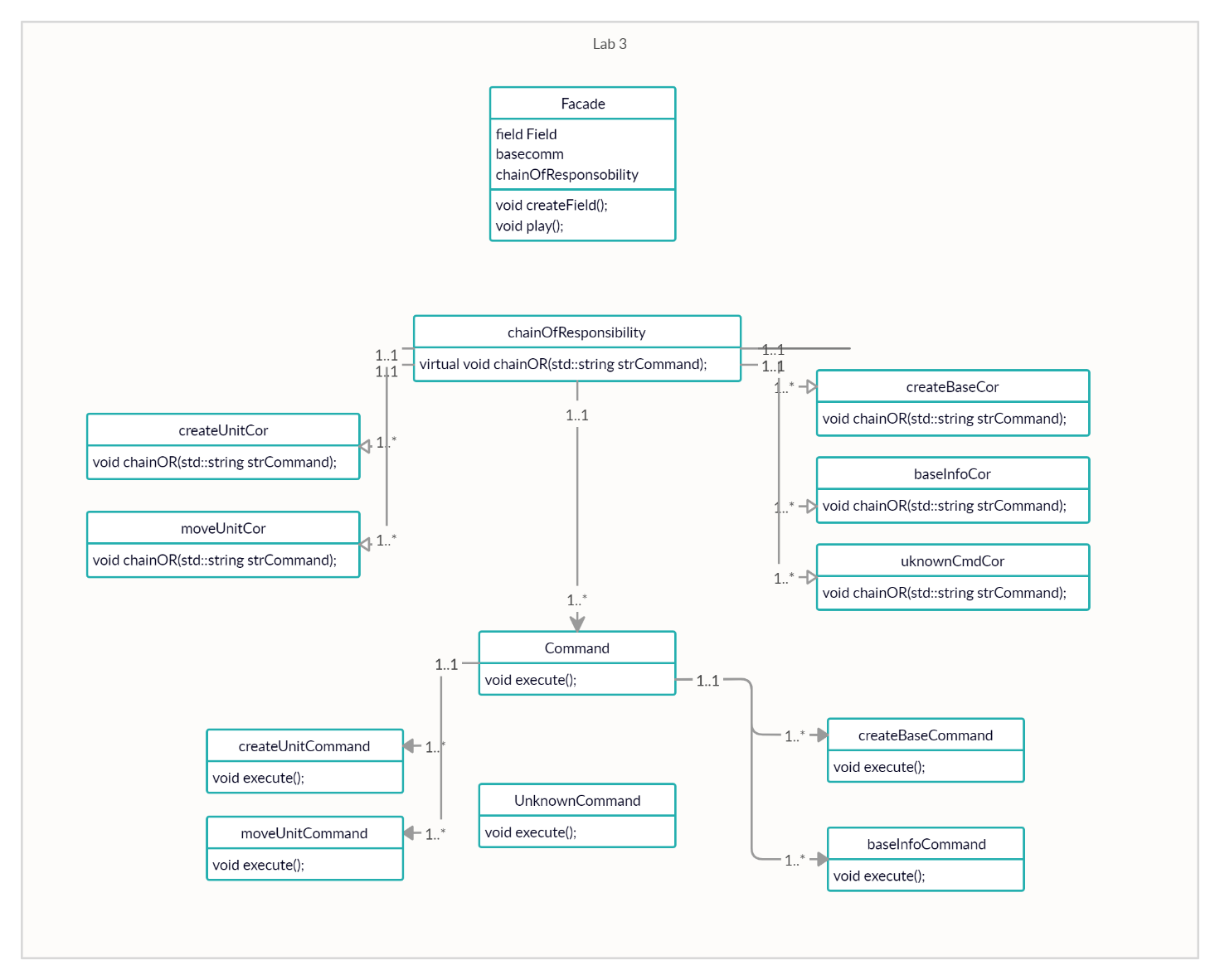
void Facade::play() – метод предназначен для приема команд( createUnit, moveUnit, setBase, baseInfo) от пользователя. Команда exit завершает программу.

1. Для приёма команд используется паттерн «Цепочка обязанностей», реализована в классе chainOfResponsibility (файл chainOfResponsibility.h/.cpp). Создается набор объектов разных классов, при этом каждый объект содержит указатель на другой объект, которому он может передать свою команду, если он не может ее обработать.

*virtual* void *chainOR* (std::string commandStr) – в каждом обработчике сообщения создается объекта класса паттерна команды и переопределяем для него метод void execute().

1. Для передачи команд используется паттерн «Команда». Реализован в классе Command (файлы command.h). Все реализованные команды наследуются от класса Command.

**UML-диаграмма.**



**Описание доработок для ЛР1 и ЛР2.**

1. Лр2 - демонстрация работы в функции exec2() в main.cpp:
   1. Для хранения информации о юнитах был использован паттерн «Кмпоновщик», реализованный в классе UnitsGroup (файлы UnitsGroup.h.cpp). Несколько юнитов могут быть объединены в одну группу, для этого используются методы addUnit() и removeUnit().
   2. Для наблюдения над юнитами в классе был использован паттерн «Наблюдатель», реализованный в классах Observer(файлы Observer.h) и SubjectObserver(файлы SubjectObserver.h/.cpp).

Абстрактный класс Observer имеет только один метод void update(SubjectObserve\* subject), который всегда вызывается из предмета, который выполняет наблюдение(класс SubjectObserve).

В классе SubjectObserve есть массив наблюдателей std::vector<Observer\*> masObserver; для предмета, который наблюдает. Если в объекте что-то меняется, то метод notify() оповещает об этом всех наблюдателей.

В конкретной работе «Наблюдатель» сообщает базе о смерти юнита.

* 1. Для взаимодействия одного типа нейтрального объекта с разными типами юнитов использовался паттерн “Стратегия” (файлы restoreArmour.h, restoreFuell.h, restoreHealth.h). У каждого нейтрального объекта есть базовый класс, от которого наследуются две стратегии: улучшение определенной характеристики юнита в дневное и ночное время. Например, в нейтральном объекте restoreArmour количество брони для юнита пополняется больше именно в ночное время.

1. ЛР 1 – демонстрация работы в функции exec1() в main.cpp

2.1) Был создан итератор для поля (файл iterator.h/.cpp), который перебирает все юниты на поле и выводит характеристики полей юнитов(здоровье, броня, топливо и др.)

2.2) Был изменен конструктор перемещения (Filed.cpp)

**Выводы.**

В ходе лабораторной работы были разработаны и реализованы наборы классов для взаимодействия пользователя с юнитами и базой. Получены умения по логическому разделению классов при помощи паттернов