**sМИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №3**

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

Тема: **Логическое разделение классов**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Мололкин К.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

**Цель работы.**

Продолжить работу над игрой, изучить паттерны проектирования, необходимые для логического разделения классов.

**Постановка задачи.**

Разработать и реализовать набора классов для взаимодействия пользователя с юнитами и базой. Основные требования:

* Должен быть реализован функционал управления юнитами
* Должен быть реализован функционал управления базой

**Выполнение работы.**

На протяжении двух предыдущих лабораторных было уже реализован некоторый функционал, необходимый для данной лабораторной работы, поэтому необходимо разделить классы, которые реализуют интерфейс работы, и классы, непосредственно выполняющие те или иные функции.

**Управление юнитами и базой**

Class Game реализуют паттерн «Фасад», где пользователь управляет программой.

Поля:

1. private GameBoard\* board – указатель на поле;

Методы:

1. public Game() – конструктор класса;
2. public void startGame() – в данном методе происходит запуск игры, то есть запускается цикл считывания команд пользователя.

Список команд пользователя:

1. Перезагрузка игры
2. Вывод игрового поля
3. Передвижение юнита
4. Вывод информации о базе
5. Создание юнита на базе
6. Вывод информации о юните
7. Атака юнита
8. Завершение игры

**class Command**

Интерфейс команд, реализует паттерн «Команда».

Поля наследников:

1. GameBoard\*\* board – указатель на указатель на поле(указатель на указатель используется для того чтобы исполнить команду перезапуска поля).

Методы:

1. bool execute() – метод исполнения команды, если команда не выполнена, то возвращается false.

Классы наследники:

1. UnitAttackCommand (исполняет команду атаки юнитом);
2. RestartCommand (исполняет команду перезагрузки игры);
3. PrintBoardCommand (исполняет команду печати поля);
4. MoveUnitCommand (исполняет команду передвижения юнита);
5. AboutBaseCommand (исполняет команду вывода информации о базе);
6. CreateUnitFromBaseCommand (исполняет команду генерации юнита на базе);
7. AboutUnitCommand (исполняет команду вывода информации о юните)

**class Mediator**

Реализует паттерн «Посредник».

Используется для исполнения команды атаки.

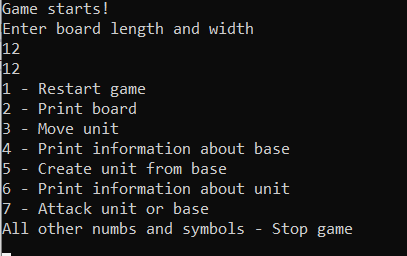
Методы:

1. attackBase(int attackPower, Base\* base) – атака базы;
2. attackUnit(int attackPower, IUnit\* unit) – атака юнита;

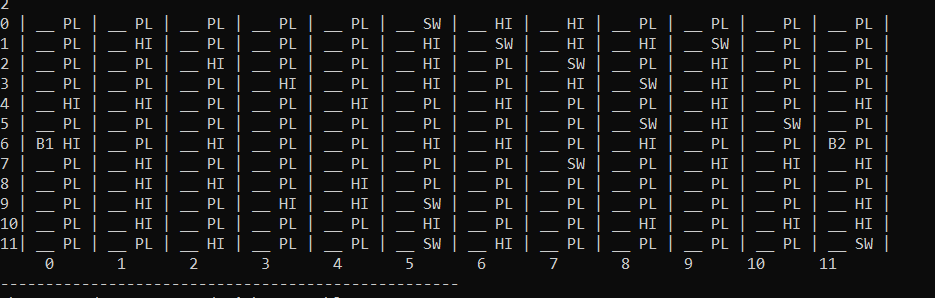
Все созданные классы, и уже имеющиеся, реализуют паттерн «Цепочка обязанностей». Для создания юнита класс CreateUnitFromBaseCommand обращается к методы createUnit класса GameBoard, из поле в свою очередь обращается к базе, база к фабрике юнитов, затем созданный юнит ставится на клетку. Класс AboutUnitCommand обращаются к полю, а поле к клетке, клетка к юниту для вывода информации о нем. Класс AboutBaseCommand обращается к полю, а поле к базе. MoveUnitCommand так же обращается к полю, а поле к клетке. Таким образом реализуется данный паттерн

**Демонстрация работы.**

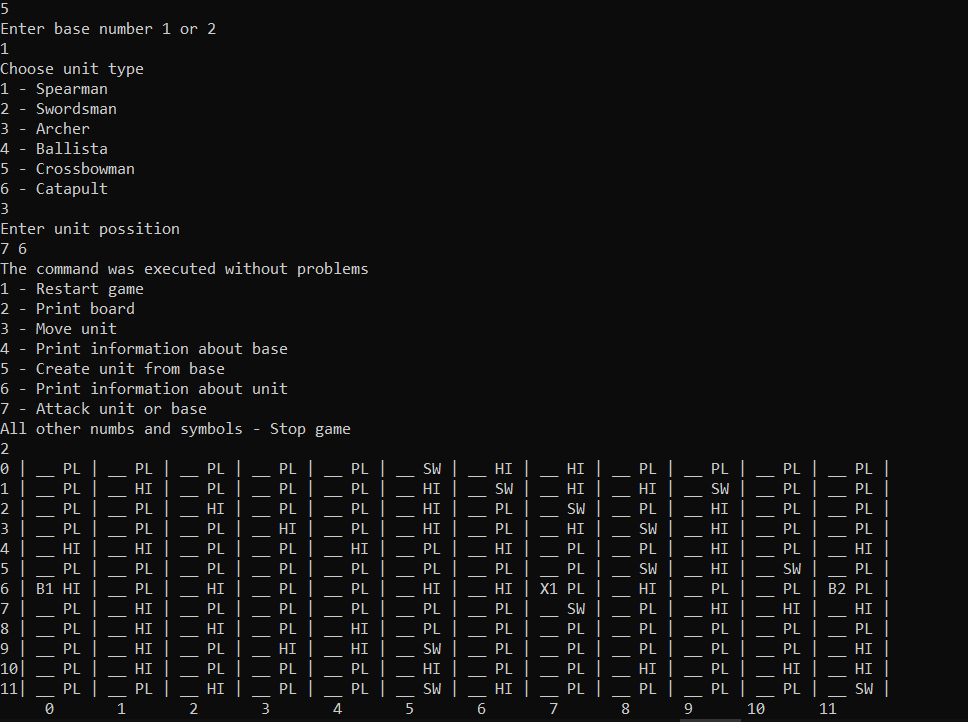
1. При запуске игры считываются размеры поля, а затем оно создается и выводится список команд простейший интерфейс с командами.



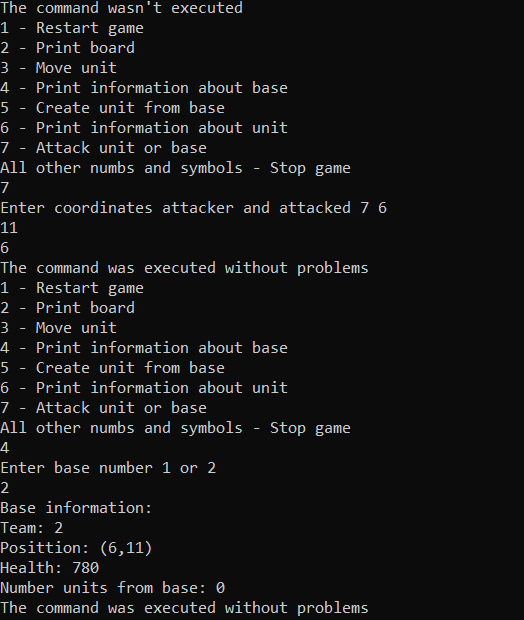
1. Вызовем команду печати поля



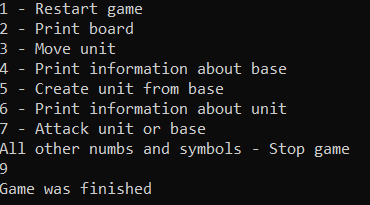
1. Создадим лучника и выведем поле:



1. Вызовем команду атаки юнита на базу и команду вывода свойств базы.



1. Закончим игру.



**Выводы.**

В ходе выполнения данной лабораторной работы были изучены паттерны, необходимые для логического разделения классов.