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. 8383 _____ Бессуднов Г. И.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

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

Постановка задачи.

- Должен быть реализован функционал управления юнитами
- Должен быть реализован функционал управления базой

Ход работы.

В условии лабораторной работы требовалось реализовать паттерн «Посредник». Данный паттерн был реализован еще в первой лабораторной работы. В качестве посредника выступает класс `GameManager`, который обрабатывает все команды пользователя и осуществляет связь между компонентами программы. Выбор базы для управления, создание определенного типа юнита базой, выбор юнита, перемещение выбранного юнита и выход из программы были реализованы ранее посредством оператора `switch`.

В классе `GameManager` содержится класс `ActionManager`, который выполняет роль фасада. Он содержит три класса: `BaseCommander` `UnitCommander` `GameFlowCommander`, которые выполняют определенные действия и манипулируют базами, юнитами и игрой в целом соответственно. Методы этих классов вызываются из одного метода класса `ActionManager - int action(char action)`. Таким образом достаточно вызвать всего лишь этот метод, передать ему номер команды и она будет исполнена.

Классы `BaseCommander` и `UnitCommander` содержат в себе поле объекта-команды. Это поле необходимо для реализации паттерна «Команда». Эти поля являются указателями на дочерний объект класса Command. Данный класс имеет всего одну виртуальную функцию `virtual void execute()`, которая необходима для исполнения команды. У класса есть два наследника `BaseCommands` и `UnitCommands`, который нужны для команд базы и команд юнитов соответственно. Указатели именно на эти объекты содержатся в

BaseCommander и UnitCommander. Далее у BaseCommands есть два наследника: NextBaseCommand, отвечающий за выбор баз, и SpawnUnitCommand, отвечающий за создание юнитов. У UnitCommands есть три наследника: NextUnitCommand, отвечающий за выбор юнита, MoveUnitCommand, отвечающий за передвижение юнита, ActionUnitCommand, отвечающий за действие, которое выполнит юнит (пока только атака). Принцип работы: UnitCommander получил код команды передвижения юнитов. Его переменной UnitCommands *command присваивается новое значение command = new MoveUnitCommand(unitsContainer, Vector2D(0, -1)). Далее у переменной command вызывается метод execute() и команда исполняется.

Выводы.

В ходе выполнения лабораторной работы были реализованы паттерны поведения, которые позволили сделать код программы более гибким и удобным.