МИНОБРНАУКИ 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. 8381 _____ Почаев Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

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

Задание.

- Выполнены все основные требования к взаимодействию
- Добавлен функционал просмотра состояния базы
- Имеется 3+ демонстрационных примера
- Реализован паттерн “Фасад” через который пользователь управляет программой
- Объекты между собой взаимодействуют через паттерн “Посредника”
- Для передачи команд используется паттерн “Команда”
- Для приема команд от пользователя используется паттерн “Цепочка обязанностей”

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки Qt Creator, для компиляции и отладки использовалась UNIX-подобная среда Cygwi и набор адаптированных инструментов MiniGW. Были задействованы пакеты GCC, CMake, а также GDB.

Реализованные классы

Классы, добавленные в программу в данной лабораторной работе и их функционал представлены в табл. 1. В ней приведено общее описание классов, отдельные моменты пояснены в комментариях к коду.

Таблица 1 – Основные добавленные классы

Класс	**Назначение**
Game (./Game)	Класс реализует интерфейс IGame и является классом, инкапсулирующим в себе всё игровое взаимодействие юнитов, баз и т.д., оставляя снаружи только интерфейс передачи желаемых действий. Также хранит в себе информацию об именах заведённых баз, предоставленных пользователю для добавлению юнитов.
MainWindow (./GUI)	Класс главного окна программы – стартовое меню. Даёт возможность выбора количества игроков, размер поля, а также запустить окно с основной игрой. Деактивируется при запуске игровой окна и активируется при его закрытии. GameWindow находится в зависимости от данного окна и в случае его закрытия также скрывается.
GameWindow (./GUI)	Класс окна игрового процесса. Реализуется GUI основных действий игрока и получения информации о действиях, которые хочет выполнить игрок.
UIFacade (./Game)	Класс реализуется паттерн *“Фасад”* и реализует взаимодействие между UI и бизнес-логикой игры. Информация в него приходит через систему сигналов и слотов по пути GameWindow→MainWindow→ текущий класс. На выполнения запрос и данный из UI уже передаются через класс Command (паттерны “Команда” и “Цепочка обязанностей”).
Command (./GUI)	Класс реализует в себе интерфейс ICommand. Основан на паттернах *“Команда”* и *“Цепочка обязанностей”*. От Command наследуются 3 вида команд: GameCommand, BaseCommand, а также FieldCommand, реализующие разные аспекты взаимодействия и получения информации внутри игры.
FacadeMediator	Класс реализует в себе интерфейс IFacadeMediator.

(./Game)	Основан на паттерне *“Посредник”*. От FacadeMediator наследуются addUnitFacadeMediator, unitAttackFacadeMediator и unitMoveFacadeMediator, реализующие посреднечество для разных комплексных операций. Сам FacadeMediator отвечает за передачу и распределения сигнала команды получения информации.
UnitAttackMediator (./AuxiliaryFunctionality/ UnitMediators.h)	Класс реализует паттерн *“Посредник”* между юнитами при их взаимодействии – атаке. Когда один из юнитов хочет атаковать другой, он отправляет соответствующий сигнал, содержащий возможный наносимый урон. Пройдя через проверки прокси поля на возможность проведения данного действия, другой юнит получает и сигнал и урон.

Выводы.

В ходе выполнения лабораторной работы были написаны требуемые класс, а также реализовано взаимодействие между функциональными классами игры и пользователем.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAIN.CPP

```
#include <iostream>

#include <QApplication>
#include <QGridLayout>
#include <QWidget>
#include <QLabel>
#include <QScreen>

#include "Tests/examples.h"
#include "Game/UIFacade.h"

int main(int argc, char *argv[])
{
    UIFacade *game = new UIFacade(argc, argv);
    game->start();

    return 0;
}
```