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

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Объектно ориентированное программирование»
Тема: Шаблонные классы

Студент гр. 8382 _____ Мирончик П.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ

Разработка и реализация набора классов правил игры. Основные требования:

- Правила игры должны определять начальное состояние игры
- Правила игры должны определять условия выигрыша игроков
- Правила игры должны определять очередность ходов игрока
- Должна быть возможность начать новую игру

Выполнены основные требования	5 баллов
Реализован шаблонный класс игры, в качестве параметра шаблона передаются конкретные правила	3 балла
Должно быть реализовано минимум 2 правил игры	2 балла
**Класс игры в шаблоне поддерживает кол-во игроков. И для определенного кол-ва должен быть специализирован отдельно*	*3 балла*
**Передача хода между игроками реализована при помощи паттерна "Состояние"*	*4 балла*
**Класс игры один единственный и создается паттерном "Синглтон"*	*3 балла*
Кол-во баллов за основные требования	**10 баллов**
Максимальное кол-во баллов за лаб. работу	**20 баллов**

ХОД РАБОТЫ

Описание основных классов.

Описание из лабораторной работы №2:

GameBoard – корень приложения. Хранит информацию о клетках доски (*Cell*), привязанных к доске объектах (*GameObject*), подписчиках на изменения поля (*BoardListener*). К экземпляру *GameBoard* привязывается *ObjectsController* и *MouseTracker*. GameBoard отвечает за рассылку уведомлений об изменении игрового поля (перемещение/добавление/удаление юнитов), передачу действий пользователя (мышь и клавиатура) игровым объектам, обработку корректного удаления/добавления объектов, отрисовку поля и вызов функций отрисовки у подписанных объектов. Добавление и удаление объектов возможно только через *ObjectsController*.

Cell – элемент сетки игры, клетка. Содержит информацию о ландшафте в клетке, положении клетки а также объектах, находящихся в данной клетке.

ObjectsController – мост между доской и объектами. Содержит методы для создания объектов поля (юнитов и нейтральных объектов), добавления и удаления элементов с поля (вызывая затем соответствующие методы в *GameBoard*, если вызов корректен: например, при добавлении элемента необходимо убедиться, что в целевой клетке отсутствует объект). При необходимости взаимодействия объектов поля между собой (например нанесение урона) действие также проходит через *ObjectsController*.

MouseTracker – как следует из названия, класс предназначен для отслеживания действий пользователя при помощи мыши. На текущий момент единственным классом, использующим *MouseTracker*, является *GameBoard*. Данный класс позволяет отслеживать перемещения мыши в удобном формате, отслеживая смещения мыши относительно последней позиции и нажатия левой клавишей мыши.

GameObject – базовый класс для всех объектов поля. Отвечает за хранение своего состояния (привязан ли к доске) и позиции ячейки, в которой он находится в данный момент. *GameObject* предоставляет ряд полезных интерфейсов (*BoardListener*, слушатели состояния привязки) и обязательных к реализации абстрактных методов (отрисовка, обработка нажатий клавиатуры и мыши).

Unit – базовый класс для юнитов: объектов, которыми может манипулировать пользователь. Обладает такими характеристиками, как: здоровье, скорость, атака. Может перемещаться по полю.

Neutral – базовый класс для нейтральных юнитов. Пользователь не может влиять на нейтральные юниты. Каждый *Neutral* обладает радиусом действия. Если *Unit* попадает в зону действия, на него накладывается определенный эффект, который наследуется от *NeutralEffect*.

Terrain – класс ландшафта. Каждой клетке поля (*Cell*) устанавливается определенный тип ландшафта. Terrain обладает следующими возможностями: отрисовка, возможность накладывать эффекты на объекты типа *Unit*.

Effect – эффект, который накладывается на объекты типа *Unit*. Имеет возможность изменять любые свойства объекта. По сути эффекты – основной способ взаимодействия с юнитами.

TerrainEffect – класс, являющийся наследником *Effect*. По большей части это вспомогательный класс для других эффектов ландшафта. Он отслеживает положение *Unit*-а, к которому привязан, и, если нет нейтральных объектов подходящего типа, в радиус действия которых попадает целевой юнит, то эффект снимается.

Классы, дополнительно затронутые в лабораторной работе №4:

GameController – базовый абстрактный класс для контроллера игры, где хранится состояние хода и игры (окончена или нет).

Rules – базовый абстрактный класс для правил игры. Требует реализовать создание игрового поля, работу с состояниями (ходами) а также проверку окончания игры.

BaseRules – базовый класс для большинства правил, наследующийся от *Rules* и реализующий логику обработки состояний и проверки окончания игры.

PlayerState – состояние, хранящее *SQUAD*_ID текущего игрока.

Rules2, *Rules3* – классы правил для 2 и трех игроков соответственно.

MapConstructor – вспомогательный класс для заполнения карты ландшафтом.

ПУТИ К КЛАССАМ

BaseUnitAttackBehaviour -
\include\GAME\engine\behaviour\BaseUnitAttackBehaviour.hpp

BaseUnitClickBehaviour -
\include\GAME\engine\behaviour\BaseUnitClickBehaviour.hpp

BaseUnitMoveBehaviour -
\include\GAME\engine\behaviour\BaseUnitMoveBehaviour.hpp

BlackHole - \include\GAME\engine\units\BlackHole.hpp

BlackHoleEffect - \include\GAME\engine\units\BlackHole.hpp

BoardListener - \include\GAME\engine\BoardListener.hpp

BoardView - \include\GAME\engine\graphics\BoardView.hpp

Cell - \include\GAME\engine\Cell.hpp

CellClickBehaviour - \include\GAME\engine\behaviour\CellClickBehaviour.hpp

CellDrawer - \include\GAME\engine\graphics\CellDrawer.hpp

Chancel - \include\GAME\engine\units\Chancel.hpp

ChancelEffect - \include\GAME\engine\units\Chancel.hpp

ConsoleLogAdapter - \include\GAME\log\ConsoleLogAdapter.hpp

Effect - \include\GAME\engine\Effect.hpp

EffectsComparator - \include\GAME\engine\Effect.hpp

EffectsSet - \include\GAME\engine\Effect.hpp

FileLogAdapter - \include\GAME\log\FileLogAdapter.hpp

FileSession - \include\GAME\log\FileSession.hpp

FileSigner - \include\GAME\serialize\FileSigner.hpp

GameBoard - \include\GAME\engine\GameBoard.hpp

ObjectsController - \include\GAME\engine\ ObjectsController.hpp

GameObject - \include\GAME\engine\GameObject.hpp

GameSerializer - \include\GAME\serialize\GameSerializer.hpp

GridDrawer - \include\GAME\engine\graphics\GridDrawer.hpp

GroundTerrain - \include\GAME\engine\terrains\GroundTerrain.hpp

Heal - \include\GAME\engine\units\Heal.hpp

HealthDrawer - \include\GAME\engine\graphics\HealthDrawer.hpp

Home - \include\GAME\engine\units\Home.hpp

InObjectsTable - \include\GAME\serialize\InObjetsTable.hpp

LavaTerrain - \include\GAME\engine\terrains\LavaTerrain.hpp

Log - \include\GAME\log\Log.hpp

LogAdapter - \include\GAME\log\LogAdapter.hpp

Loggable - \include\GAME\log\Log.hpp

LogInfo - \include\GAME\log\LogInfo.hpp

MouseTracker - \include\GAME\engine\MouseTracker.hpp

Neutral - \include\GAME\engine\Neutral.hpp

NeutralEffect - \include\GAME\engine\NeutralEffect.hpp

ObjectInfo - \include\GAME\serialize\ObjectInfo.hpp

OutObjectsTable - \include\GAME\serialize\OutObjetsTable.hpp

SeaTerrain - \include\GAME\engine\terrains\SeaTerrain.hpp

Serializer - \include\GAME\serialize\Serializer.hpp

ShapeDrawer - \include\GAME\engine\graphics\ShapeDrawer.hpp

Stone - \include\GAME\engine\units\Stone.hpp

Terrain - \include\GAME\engine\Terrain.hpp

Unit - \include\GAME\engine\Unit.hpp

UnitAttachBehaviour -
\include\GAME\engine\behaviour\UnitAttachBehaviour.hpp

UnitMoveBehaviour -
\include\GAME\engine\behaviour\UnitMoveBehaviour.hpp

Viewport - \include\GAME\engine\graphics\Viewport.hpp

ЗАПУСК ПРИЛОЖЕНИЯ

Проект собирается при помощи VisualStudio2017 и, насколько я знаю, не требует дополнительных разрешений/установки библиотек. Для запуска можно использовать дебажную сборку, находящуюся в ${ProjectRoot}/Debug/SimpleGame.exe. Программа использует дополнительные библиотеки (SFML), однако они находятся внутри проекта, так что приложение должно запуститься корректно.

ВЫВОД

При выполнении лабораторной работы были изучены различные паттерны проектирования, изучены основные способы работы с шаблонами классов, их написание, использование и специализация при необходимости.