**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №6**

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

Тема: Шаблонные классы

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Почаев Н.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

## Цель работы.

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;
* Должно быть реализовано минимум 2 правил игры.

Дополнительные требования:

* Класс игры в шаблоне поддерживает кол-во игроков. И для определенного кол-ва должен быть специализирован отдельно;
* Передача хода между игроками реализована при помощи паттерна “Состояние”;
* Класс игры один единственный и создается паттерном “Синглтон”.

## 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. Для компиляции текущей версии программы под Windows необходим MinGW 8.10 (для более полноценной поддержки C++17) и Qt версии 14.10 и выше.

## 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** |
| IGameProcess  (./Game/GameProcess) | Интерфейс класса процесса игры. Описывает методы для паттерна *“Одиночка”* и в некотором роде является интерфейсом паттерна *“Прокси”* для класса игры, созданного в предыдущих ЛР. |
| SignalSlotGameProcess  (./Game/GameProcess) | Для взаимодействия с UI класс должен наследоваться от QObject, однако, это невозможно сделать в шаблонном классе. Решением является задание абстрактного нешаблонного класса с вынесенной функциональностью взаимодействия с интерфейсом. |
| GameProcess  (./Game/GameProcess) | Класс, реализующий интерфейсы, описанные выше. Реализует паттерн Singleton, а также является машиной состояний в паттерне *“Состояние”*.  Является шаблонным и поддерживает разные виды правил игры и кол-во игроков (тип данных их хранения).  Для каждого правила и кол-ва специализирован отдельно. |
| IGamersState  (./Game/GameProcess) | Интерфейс класса состояния (описан ниже). |
| PlayerGameState  (./Game/GameProcess) | Класс реализует интерфейс, описанный выше, и является частью паттерна *“Состояние”*. Каждое состояние описывает конкретного игрока, ход передаётся по кругу от большего к меньшему.  В зависимости от текущего состояния класса процесса игры осуществляется добавление баз, а также проверка на завершение игры по указанным правилам. |
| IGameRule  (./Game/GameProcess) | Интерфейс, описывающий класс правил игры. |
| AbstractGameRule:  oneToOneRule;  twoByTwoRule  (./Game/GameProcess) | Абстрактный класс правил игры, а также его конкретные реализации. Определяют время на ход, кол-во игроков, размер поля, а также условия завершения игры (победы). Подробное описание выводится в интерфейсе программы. |

## Выводы.

В ходе выполнения лабораторной работы были написаны требуемые классы, а также реализована поддержка нескольких видов правил игры, а также функциональность передачи хода между несколькими игроками.

# 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[])

{

std::shared\_ptr<UIFacade> game = std::make\_shared<UIFacade>(argc, argv);

game->start();

return 0;

}