**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

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

Тема: **Отслеживание изменений**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2300 |  | Гаранин Р.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

## Цель работы

Реализовать дополнительный класс для связи с игрой и отслеживания изменений в ней. Кроме того, данный класс должен отрисовывать игровое поле, согласно условиям задачи.

## Задание

а) Реализовать класс, который связывается с игрой, и отслеживает изменения в игре: перемещение игрока, победа или выигрыш, срабатывание событий. Данный класс должен реагировать на изменения и отрисовывать игровое поле, а также выводить информацию для игрока (например, предлагать начать новую игру).

б) При отрисовке поля должна считываться информация с поля и об игроке, и в зависимости от расположения происходит вывод представления поля в терминал. В представлении поля непроходимые клетки, игрок, события должны отображаться различными символами. Игрок, события, клетки и другие игровые сущности не должны знать ничего о том, каким символом они отрисовываются. За выбор символа отвечает класс выполняющий отрисовку

Примечания:

* Класс отслеживания и класс отрисовки рекомендуется делать отдельными сущностями. Таким образом, класс отслеживания инициализирует отрисовку, и при необходимости можно заменить отрисовку (например, на GUI) без изменения самого отслеживания
* При проверке типов события можно использовать dynamic\_cast / typeinfo, либо сделать связывание модель-представление

## Выполнение работы

В ходе выполнения данной лабораторной работы были реализованы 4 класса: *enum class States\_game, class Interface\_render, class Terminal\_render, class Tracker.*

1. *enum class States\_game*:

Данный класс содержит состояния игры.

1. *class Interface\_render*:

Данный класс создан для вывода определенных данным. В данном классе реализованы только виртуальные метода, в связи с чес можно сказать что данный класс является абстрактным.

1. *class Terminal\_render*:

Данный класс также создан для вывода данных игры в консоль, наследуя и переопределяя методы из класса с*lass Interface\_render*. В данном классе реализованы следующие методы:

void Print\_playing\_field(Player& player, Field& field) const override – Данный метод принимает ссылку на поле и игрока, и выводит актуальное поле в консоль.

void Print\_player(Player& player) const override – Данный метод принимает ссылку на игрока, и также выводит актуальную информацию об игроке в консоль.

void Print\_start() const override – Данный метод выводит строку приветствия.

void Print\_level() const override - Данный метод выводит строку для выбора уровня игры.

void Print\_win() const override - Данный метод выводит строку о том, что игрок победил.

void Print\_lose() const override - Данный метод выводит строку о том, что игрок проиграл.

void Print\_the\_end() const override - Данный метод выводит последнюю строку перед выходом из игры.

void Print\_game() const override - Данный метод выводит строку о том какие клавиши игры отвечают за управление.

void Print\_new\_game() const override - Данный метод выводит строку с вопросом, хочет ли игрок начать игру заново.

1. *class Tracker*:

Данный класс создан для вызова и вывода определенных данных на экран, то есть передавая в метод Checking*\_the\_state* состояние игры, вызывает экземпляр класса *Terminal\_render* и соотвественно выводит определенные сообщения, также содержит следующие поля: Field& field, Player& player, Interface\_render& render, а также следующие методы: *Tracker(Interface\_render& Value\_render, Player& Value\_player, Field& Value\_field), void Checking\_the\_state(States\_game state) const.*

Разработанную UML-диаграмму см. в приложении А.

## Выводы

В ходе данной лабораторной работы был реализован класс не только для отслеживания изменений в игре: перемещение игрока, победа или выигрыш и срабатывания события, но и для отрисовки игрового поля, который считывает информацию с поля и об игроке, и в зависимости от расположения происходит вывод представления поля в терминал.

# Приложение А UML-ДИАГРАММА РЕАЛИЗОВАННЫХ КЛАССОВ

