**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

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

Тема: Уровни абстракции

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

Санкт-Петербург

2023

## Цель работы

Изучить понятия и уровни абстракции в ООП, а также реализовать новый набор классов для запуска игры и считывания ввода пользователя согласно условиям поставленной задачи.

## Задание

а) Создать класс игры. Класс игры отвечает за запуск игры (в начале и во время текущей сессии), выход из игры, выбор уровня, а также инициализирующего перемещение игрока. Также класс должен проводить проверку проигрыша или выигрыша игрока и давать возможность начать новую игру или завершить работу программы.

б) Создать класс (или набор классов) считывания ввода пользователя. Данный класс(ы) должен сопоставить ввод пользователя с командой, которую необходимо выполнить. И передать эту информацию в класс игры. Клавиши управления (то на какую клавишу назначено определенное действия) должны считываться из файла. Класс(ы) должен быть разработан так, чтобы:

* потенциально можно было масштабировать управление с минимальным количеством изменений в коде
* можно было заменить схему управления, например, заменить ввод команду через терминал на чтение команд из файла или из сети с другого устройства

Примечания:

* После считывания клавиши, считанный символ должен сразу обрабатываться, и далее работа должна проводить с сущностью, которая представляет команду.
* Для представления команды можно разработать системы классов или использовать перечисление enum.
* Хорошей практикой является создание “прослойки” между считыванием/обработкой команды и классом игры, которая сопоставляет команду и вызываемым методом игры. Существуют альтернативные решения без явной “прослойки”
* При считывании управления необходимо делать проверку, что на все команды назначена клавиша, что на одну клавишу не назначено две команды, что на одну команду не назначено две клавиши.
* При работе с файлом используйте идиому RAII.

## Выполнение работы

В ходе выполнения данной лабораторной работы были реализованы следующие классы: класс *Game*, класс *Interlayer*, класс *Input\_Interface*, а также абстрактный класс *Input\_from\_keyboard*.

Класс *Game*:

В данном классе были реализованы следующие методы: void Choose\_lvl\_of\_map(), void Play(), void The\_end(), Game(Interlayer& Value\_layer), void Begin(), а также поля: Field field, Interlayer& layer, FieldCreator create\_field.

void Choose\_lvl\_of\_map() – Данный метод для создания поля.

void Play() – Данный метод для управления и запуска самой игры, в нем действует цикл while, который отвечает за ходьбу игрока и кроме того проверяет победу или прогоришь.

void The\_end() – Данный метод для завершения игры, а также для ее рестарта.

Game(Interlayer& Value\_layer) – Конструктор класса, инициализирующий обработчик.

void Begin() – Данный метод для начала игры и вызова методов для выбора уровня, старта и конца игры.

Класс *Interlayer*:

Action Get\_action() – Данный метод для считывания события из клавиатуры.

Interlayer(Input\_Interface& input, const std::string& file\_for\_control) – Конструктор который обрабатывает файл управления и назначает клавиши на определенные действий.

Action Decoding\_action(const std::string& action) – Данный метод для сопоставления схемы управления с действиями.

Класс *Input\_from\_keyboard*:

В данном классе реализована один метод: *char Input\_function() const override*. Данный метод позволяет считывать клавиши, нажатые пользователем, после чего передавая их в так называемую прослойку. Также важно подчеркнуть, что данный класс наследуется от класса *Input\_Interface,* переопределяя в себе виртуальный метод.

Класс *Input\_Interface*:

В данном классе реализована одна виртуальная функция *virtual char Input\_function() const = 0.* Кроме того, так как данный класс содержит только одну виртуальную функцию, то он является абстрактным.

Разработанную UML-диаграмму см. в приложении А.

## Выводы

В ходе данной лабораторной работы было не только изучены, но и реализованы разные уровни абстракции. Кроме того, были добавлены ряд классов как для запуска игры, так и для считывания ввода пользователя согласно прославленным задачам.

# Приложение А UML-ДИАГРАММА РЕАЛИЗОВАННЫХ КЛАССОВ

