**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 |  | Шумов О.Д. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

## Цель работы.

Изучить уровни абстракции в ООП, а также реализовать 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 считывания 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

Были созданы 5 классов:

1. Game – класс игры.
2. Input – абстрактный класс считывания ввода пользователя.
3. InputTerminal - класс наследующий Input и реализующий считывание из терминала.
4. Actions – enum class, содержащий действия пользователя для взаимодействия с игрой.
5. Handler – класс обработки ввода пользователя.

Также создан файл scheme в которой написана схема управления игрой.

В классе Game реализованы следующие поля:

1. Field field\_ - объект класса поля.
2. FieldCreator field\_creator\_ - объект класса создания уровня.
3. Handler& handler\_ - ссылка на объект класса обработчика ввода пользователя.

Все перечисленные поля имею модификатор доступа private, что защищает их от нежелательного изменения из вне.

В классе Game реализованы следующие методы:

1. void start() – метод для запуска игры.
2. void choose\_level() – метод для выбора уровня игры.
3. void play() – метод для взаимодействия игры с пользователем.
4. void end() – метод для начала новой игры, либо завершения игры.
5. Game(Handler& handler) – конструктор класса Game. В него передается ссылка на объект класса обработчика ввода пользователя.

Методы choose\_level(), play() и end() имеют модификатор доступа private, что защищает их от нежелательного изменения из вне.

В классе Input реализован следующий метод:

1. virtual char reader() - виртуальный метод считывания ввода пользователя.

Класс InputTerminal наследует класс Input и реализует метод считывания из терминала.

В классе Actions реализовано перечисление enum в котором объявлены действия пользователя для взаимодействия с игрой.

В классе Handler реализованы следующие поля:

1. Input& input\_ - ссылка на объект класса ввода пользователя.
2. std::unordered\_map<char, Actions> actions\_ - словарь символа и объекта класса enum действия пользователя.

Все перечисленные поля имею модификатор доступа private, что защищает их от нежелательного изменения из вне.

В классе Handler реализованы следующие методы:

1. Actions get\_action() – метод для получения действия пользователя.
2. Actions convert\_action(const std::string& action) – метод для перевода схемы управления в действия.
3. Handler(Input& input, const std::string& new\_file) – конструктор класса Handler. В него передается ссылка на объект класса ввода пользователя, а также файл из которого нужно читать схему управления.

Метод convert\_action(const std::string& action) имеет модификатор доступа private, что защищает его от нежелательного изменения из вне.

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, был добавлен ряд классов как для запуска игры, так и для считывания ввода пользователя.

**Приложение А  
UML-диаграмма**

