**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

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

Тема: Отслеживание изменений

|  |  |  |
| --- | --- | --- |
| Студент гр. 2382 |  | Ваньков Я.С. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

## Цель работы

Изучить методы работы с классами. Написать простую программу в стиле ООП. Изучить паттерн обсервер.

## **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

**IView — интерфейс класса отрисовщика событий.**

**ConsoleView — наследуется от интерфейса IView.**

Класс выводит в консоль текстовое содержимое игры.

В конструктор класса передается Observer, который следит за изменениями в игре. Также в конструктор класса передается игра, над которой работает отрисовщик. В деструкторе — удаляется observer.

Методы класса:

void viewField() - отрисовывает поле. Каждая клетка проверяется на возможное событие в ней(игрок, стена, пустая клетка и т.д) и выводит нкжное текстовое значение в консоль

void viewPlayer() - выводит характеристики игрока

void update() - сама игра. Выводит характеристики и поле

void updateChooseLevel() - предлагает выбрать уровень

void updateInvalidLevel() - говорит, что уровень выбран некорректно

void updateEndGame() - вывод текст конца игры с нужным исходом

**EnumView — перечисление возможных команд обсервера**

**IObserver — интерфейс обсервера.**

**GameObserver — обсервер отслеживание изменения событий игры**

В конструкторе добавляет обсервер к игре.

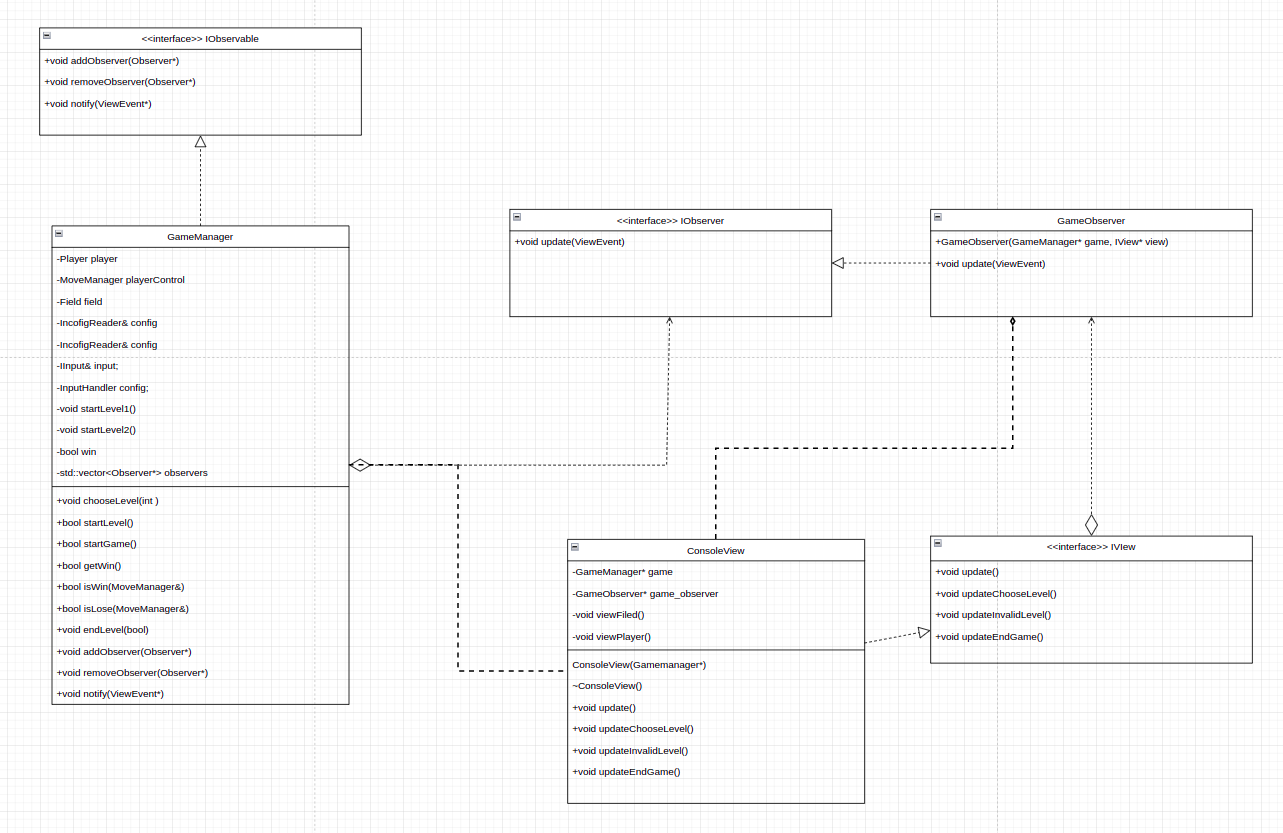
Метод update — вызывает отрисовщик, в зависимости от Enum команды переданной в аргументе

**IObservable — интерфейс класса, за которым идет слежка.**

Теперь GameManager — наследник IObservable

Методы addObserver и removeObserver — добавляют или удаляют обсерверы, следящие за классом наследника

notify — в зависимости от Enum команды переданной в аргументе вызывает обновление обсервера с данной командой



## Выводы

Был реализован класс, отслеживающий все события. Был реализован класс, отрисовывающий изменения в игре, в зависимости от изменения переданных классом обсервера.

# Приложение А Исходный код программы

# https://github.com/DoAlvaro/OOP\_2382