**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №1**

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

**Тема: Создание классов, конструкторов и методов**

| Студент гр. 1303 |  | Чубан Д.В. |
| --- | --- | --- |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2022

## **Цель работы.**

Реализовать прямоугольное игровое поле, состоящее из клеток. Клетка - элемент поля, которая может быть проходима или нет (определяет, куда может стать игрок), а также содержит какое-либо событие, которое срабатывает, когда игрок становится на клетку. Для игрового поля при создании должна быть возможность установить размер (количество клеток по вертикали и горизонтали). Игровое поле должно быть зациклено по вертикали и горизонтали, то есть если игрок находится на правой границе и идет вправо, то он оказывается на левой границе (аналогично для всех краев поля).

Реализовать класс игрока. Игрок – сущность, контролируемая пользователем.Игрок должен иметь свой набор характеристик и различный набор действий(например, разные способы перемещения, попытка избежать событие, и так далее).

**Требования.**

Реализован класс игрового поля

Для игрового поля реализован конструктор с возможностью задать размер и

конструктор по умолчанию (то есть конструктор, который можно вызвать без

аргументов).

Реализован класс интерфейс события (в данной лабораторной это может быть

пустой абстрактный класс).

Реализован класс клетки с конструктором, позволяющим задать ей начальные

параметры.

Для клетки реализованы методы реагирования на то, что игрок перешел на

клетку.

Для клетки реализованы методы, позволяющие заменять событие. (То есть

клетка в ходе игры может динамически меняться).

Реализованы конструкторы копирования и перемещения, и соответствующие

им операторы присваивания для игрового поля и при необходимости клетки.

Реализован класс игрока минимум с 3 характеристиками. И соответствующие

ему конструкторы.

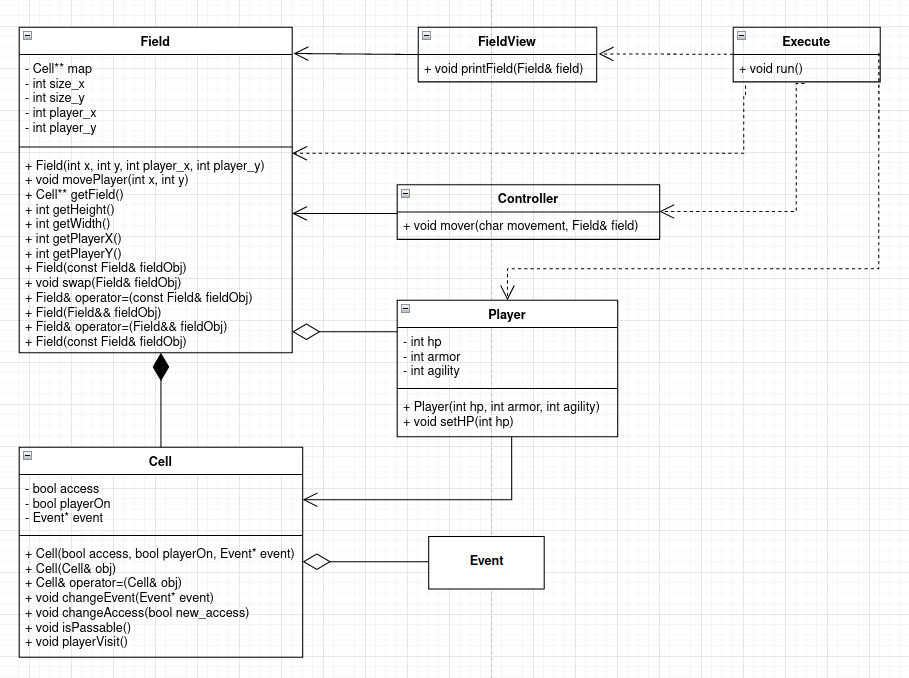
Реализовано перемещение игрока по полю с проверкой допустимости на

переход по клеткам.

## **Описание архитектурных решений и классов.**

* 1. Работа написана с использованием классов для облегчения взаимодействия многих
  2. компонентов, составляющих игру. Помимо этого, код, написанный в стиле ООП, легче обслуживать и модифицировать.
  3. Рассмотрим краткое содержание классов и то, как они связаны между собой:

1. Класс *Execute*: класс имеет только метод *run()*, который считывает входные данные, с помощью которых инициализируются поля игрока *Player* и игрового поля *Field.* Также создаются интерфейсы *Controller* и *FieldView.* Объект данного класса обеспечивает общую работу программы.
2. Класс *Cell*: элемент данного класса является основным элементом взаимодействия игрока с игрой. Класс имеет поля координат *x,y,* ссылку на возможное событие на этой клетке *event*, и флаги взаимодействия *access* (отвечает за возможность игрока перейти в эту клетку) и *playerOn* (отображает находится ли игрок на этой клетке). Класс имеет два конструктора: по умолчанию и с возможностью полностью задать ей данные. Также имеются следующие методы:
   1. *void changeEvent(Event\* event) -* позволяет заменить событие на данной клетке
   2. *void changeAccess() -* позволяет изменить доступность нахождения игрока на клетке
   3. *void isPassable() -* позволяет узнать, может ли игрок перейти в данную клетку
   4. *void playerVisit() -* изменяет состояние нахождения игрока на клетке (вошел/вышел)
3. Класс *Field:* класс, означающий игровое поле, по которому передвигается игрок. Имеет следующие поля: двумерный массив клеток *Cell\* map* поля размеров поля *size\_x, size\_y,* поля координат игрока на поле *player\_x, player\_y.* Также класс имеет конструктор по умолчанию и с возможностью задать все данные, конструктор копирования и конструктор перемещения. Методы класса:
   1. *void movePlayer(int x, int y) -* изменяет координаты игрока на поле
   2. *Cell\*\* getField()* - позволяет получить доступ к игровому полю
   3. *int getHeight(), int getWidth()* - позволяют узнать размеры поля
   4. *int getPlayerX(), int getPlayerY()* - позволяют узнать координаты игрока на поле
4. Класс *FieldView:* имеет только метод *void printField(Field& field)*, который выводит на экран игровое поле.
5. Класс *Controller:* имеет только метод *void mover(char movement, Field& field)*, который преобразует введенные команды передвижения в изменения координат и вызывает метод *movePlayer(in x, int y)* объекта *field.*
6. Класс *Event:* на данный момент является абстрактным пустым классом. Его реализация будет произведена в следующих лабораторных работах.
7. Класс *Player:* класс игрока, имеет поля *int hp, int armor, int agility,* конструктор по умолчанию и с возможностью внесения собственных данных и метод *void setHP(int hp)* для изменения количества здоровья игрока.

****

UML-диаграмма отношения классов

**Вывод.**

Реализовано игровое поле, состоящее из клеток. Поле имеет зацикленную

структуру, игрок может передвигаться по нему с помощью клавиш WASD. Клетки могут быть проходимые или нет, а также они реагируют на присутствие игрока.

Была проведена работа с использованием классов при помощи языка C++,

и принципы составления UML-диаграмм.