**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

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

Тема: Полиморфизм

|  |  |  |
| --- | --- | --- |
| Студент гр. 2381 |  | Двиков Д.В. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

**Цель работы.**

Изучить основы парадигмы объектно-ориентированного программирования. Реализовать интерфейс игрового события, создать класс для создания поля.

**Задание**

а) Создать интерфейс игрового события. Интерфейс должен обеспечивать срабатывание события когда игрок наступает на клетку.

б) Реализовать интерфейс игрового события тремя конкретными событиями. Одно событие должно положительно влиять на характеристики игрока, второе должно негативно влиять на характеристики игрока, третье изменять координаты игрока на поле. При желании можно реализовать больше событий и/или события меняющие само поле (например, делать из непроходимой клетки проходимую).

в) В классе управления игроком добавить проверку на наличие события на клетке, если событие присутствует, то оно должно сработать. Срабатывание должно происходить через интерфейс события, и не должно быть никаких проверок на тип события (реализация через динамический полиморфизм)

г) Создать класс создающий поле. Предусмотреть возможность создания 2 разных уровней. По желанию можно сделать случайную генерацию уровней. Должно гарантироваться, что игрок может дойти от входа до выхода.

Примечания:

События должны быть такими, чтобы был сценарий проигрыша игрока.

В событиях и клетках не должно быть полей сообщающих информацию о типе события

**Выполнение работы**

1. Определяется класс-интерфейс события *Event*, от которого наследуются классы-группы событий.

Реализуются виртуальные методы класса с модификатором доступа *public*:

* *virtual void truggerEvent()= 0;* – метод срабатывания события.

1. Определяется абстрактный класс *PlayerEvent*, от которого наследуются классы событий, связанныe с изменением игрока.

Реализуются методы класса с модификатором доступа *public*, аналогичные классу Event.

Поля класса с модифиактором доступа protected:

* *Player& player* – ссылка на объект класса игрока.

1. Определяется абстрактный класс *FieldEvent*, от которого наследуются классы событий, связанныe с изменением поля.

Реализуются методы класса с модификатором доступа *public*, аналогичные классу Event.

Поля класса с модифиактором доступа protected:

* *Player& player* – ссылка на объект класса игрока.
* *Field& field* – ссылка на объект класса поля.
* *Playercontroller& cnt* – ссылка на объект класса контроллера

1. Определяется класс *HealEvent*, объектом которого является cобытие, увеличивающее здоровье игрока.

Реализуются методы класса с модификатором доступа *public*:

* *void triggerEvent() override; -* метод, в который передаётся ссылка на объект класса игрока и который добавляет к здоровью игрока +1.
* *HealEvent(Player& player)* – конструктор класса.

1. Определяется класс *Score*, объектом которого является cобытие увеличивающее количество монет игрока.

Реализуются методы класса с модификатором доступа *public*:

* *void triggerEvent() override;*– метод, который добавляет к кол-ву монет игрока +1.
* *Score(Player& player)* – конструктор класса, в который передается ссылка на объект игрока

1. Определяется класс *Enemy*, объектом которого является событие столкновения с врагом.

Реализуются методы класса с модификатором доступа *public*:

* *void triggerEvent() override;*– метод, который отнимает от здоровья игрока -5 единиц и прибавляет 1 очко игроку,
* *Enemy(Player\* player) –* конструктор класса, в который передается ссылка на объект игрока

1. Определяется класс Teleport объектом которого является событие связанное с телепортацией игрока на поле.

Реализуются методы класса с модификатором доступа *public*:

*void triggerEvent() override;*– метод который делает проверку на проходимость клеток, которые будут использованы для телепортирования, и телепортирует игрока

1. Определяется класс EventBuilder создающий поле.

Реализуется конструктор класса с модификатором доступа *public*:

FieldBuilder(Gamefield& field, Player& player, Playercontroller& cnt), принимающий ссылку объект поля, игрока и контроллера. Данный класс создает поле и задает на нём события.

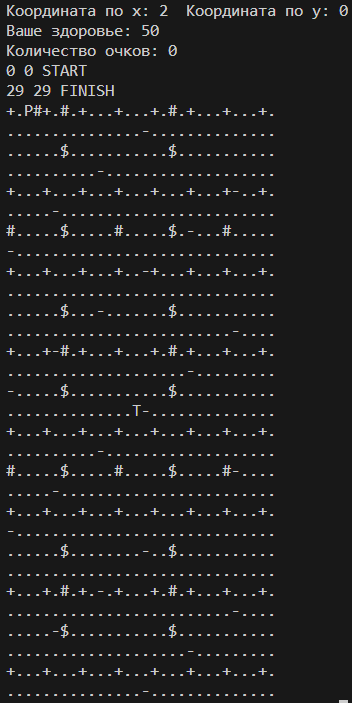
Поля с модификатором доступа *private:*

* *Player& player* – ссылка на объект класса игрока.
* *Gamefield& field* – ссылка на объект класса поля
* *Playercontroller& cnt* – ссылка на объект класса контроллера.

Вывод:

В ходе выполнения лабораторной работы были изучены основы парадигмы объектно-ориентированного программирования, создан интерфейс игрового события и его реализации. Кроме этого были доработаны функции взаимодействия с игроком, создан класс, генерирующий игровое поле и заполняющий его определенными событиями.

**Результаты работы программы:**



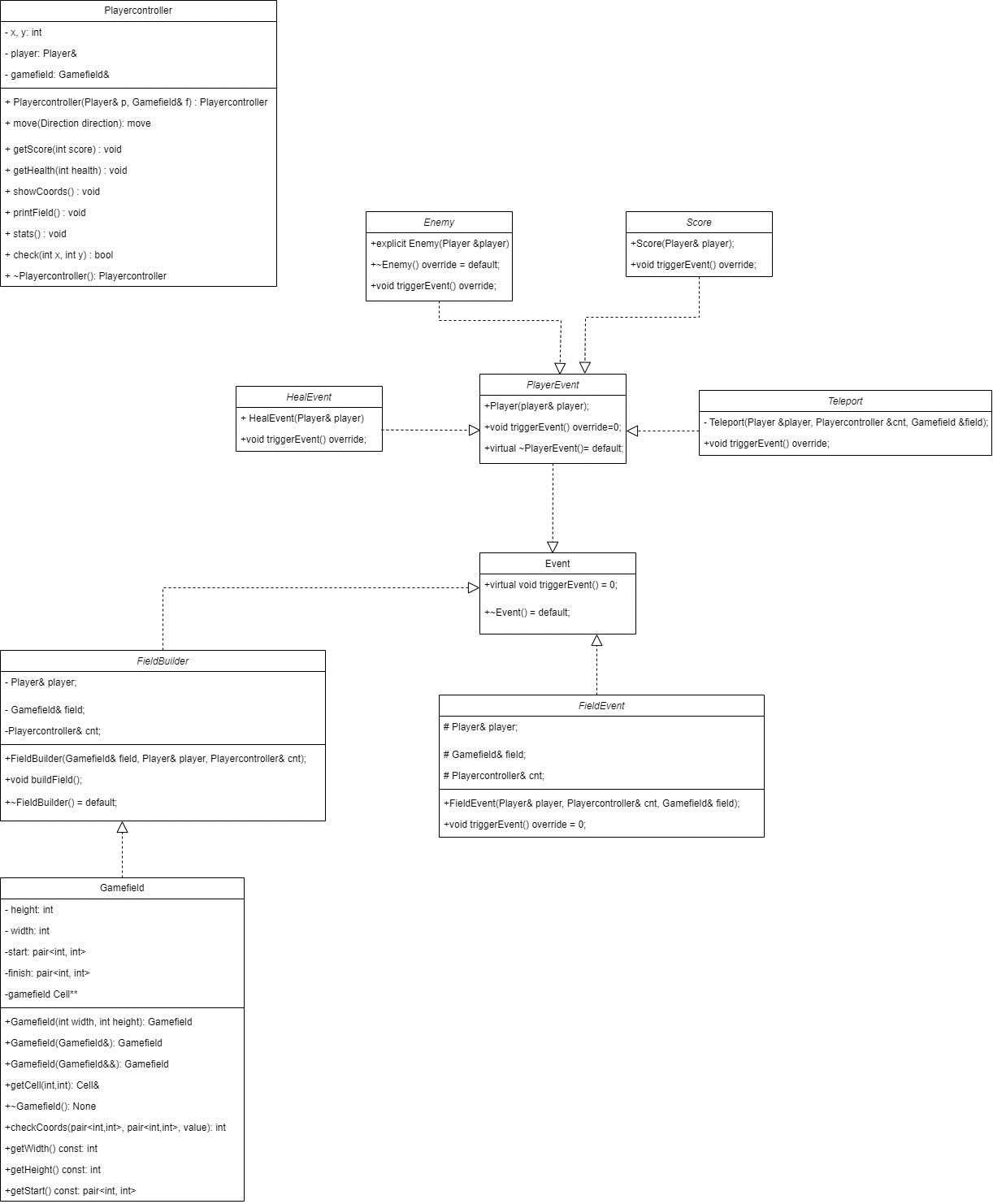
**Приложение В**  


Рисунок 1 – Uml -диаграмма