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(ка) гр. 1381 _____ Денисова О.К.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы

Изучить основные понятия объектно-ориентированного программирования, научиться писать классы, конструкторы для них и методы.

Общая формулировка задачи

- 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.

Выполнение работы

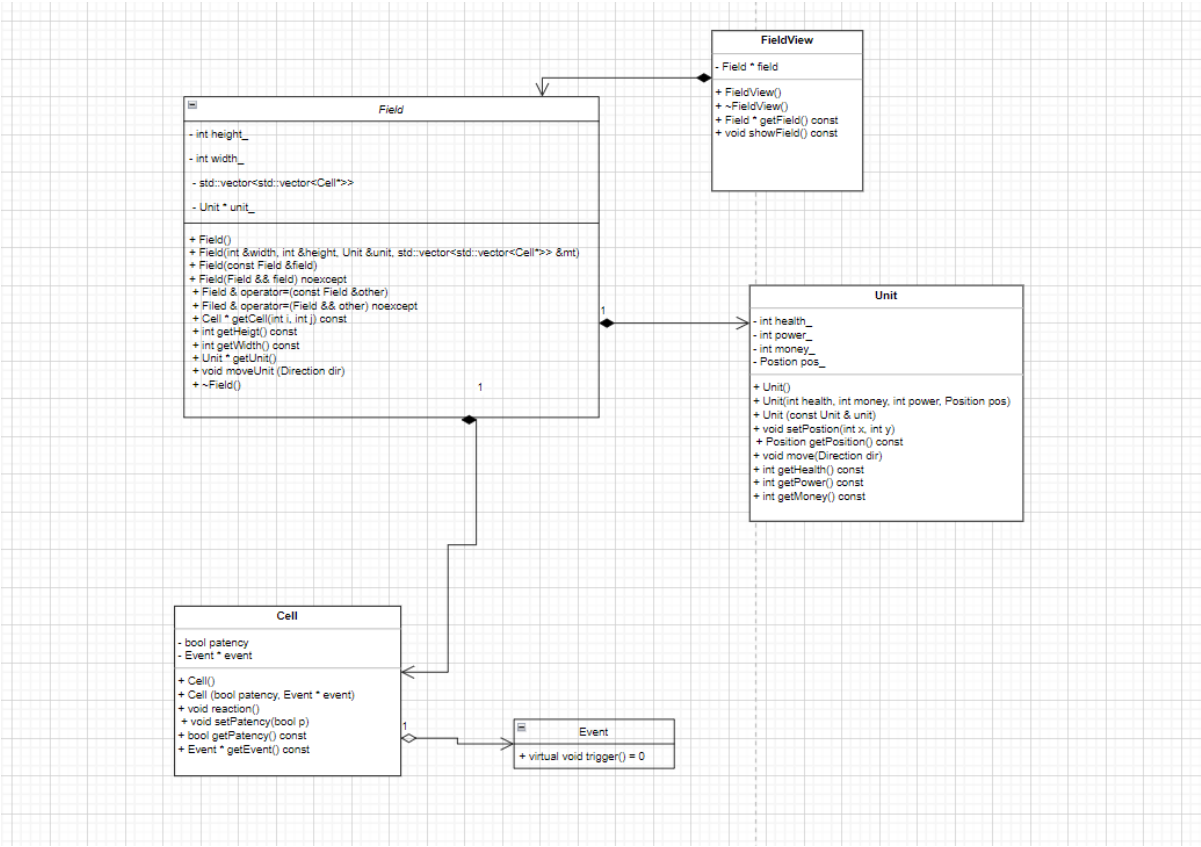
В ходе выполнения лабораторной работы были разработаны следующие классы: Cell (клетка), Field (поле), FieldView (вывод поля), Unit (игрок), Event (событе).

1) Класс Cell имеет 2 характеристики: проходимость клетки и событие, содержащееся на ней. Для двух данных характеристик были реализованы сеттеры и геттеры по необходимости. Были реализованы конструктор по умолчанию и конструктор с параметрами. Также у класса Cell есть метод reaction, который срабатывает, когда игрок становится на клетку.
2) Класс Field имеет 4 характеристики: высота и ширина поля, поле (реализовано на векторах (заголовочный файл <vector>)) и игрок. Для класса поля реализованы 4 конструктора: по умолчанию, с параметрами, конструкторы копирования и перемещения. Для двух последних также есть соответствующие им операторы присваивания. Помимо этого есть геттеры для полей и метод moveUnit, который

отвечает за передвижение игрока по полю (с проверкой проходимости клеток и зацикливанием).

3) Класс FieldView отвечает за вывод поля, соответственно, он имеет одну характеристику: поле, за которое он отвечает. Реализован конструктор по умолчанию и метод showField, который выводит поле на экран.
4) Класс Unit имеет 4 характеристики: здоровье, сила и деньги игрока, а также актуальная позиция игрока на поле (структура, содержащая координаты x и y). Реализованы конструкторы: по умолчанию, с параметрами и конструктор копирования. Есть геттеры для полей и метод move, который обновляет актуальную позицию игрока.
5) Класс Event (интерфейс) не имеет приватных полей, а в его публичной секции объявлен 1 чисто виртуальный метод trigger, который будет срабатывать при активации события.

UML-диаграмма:

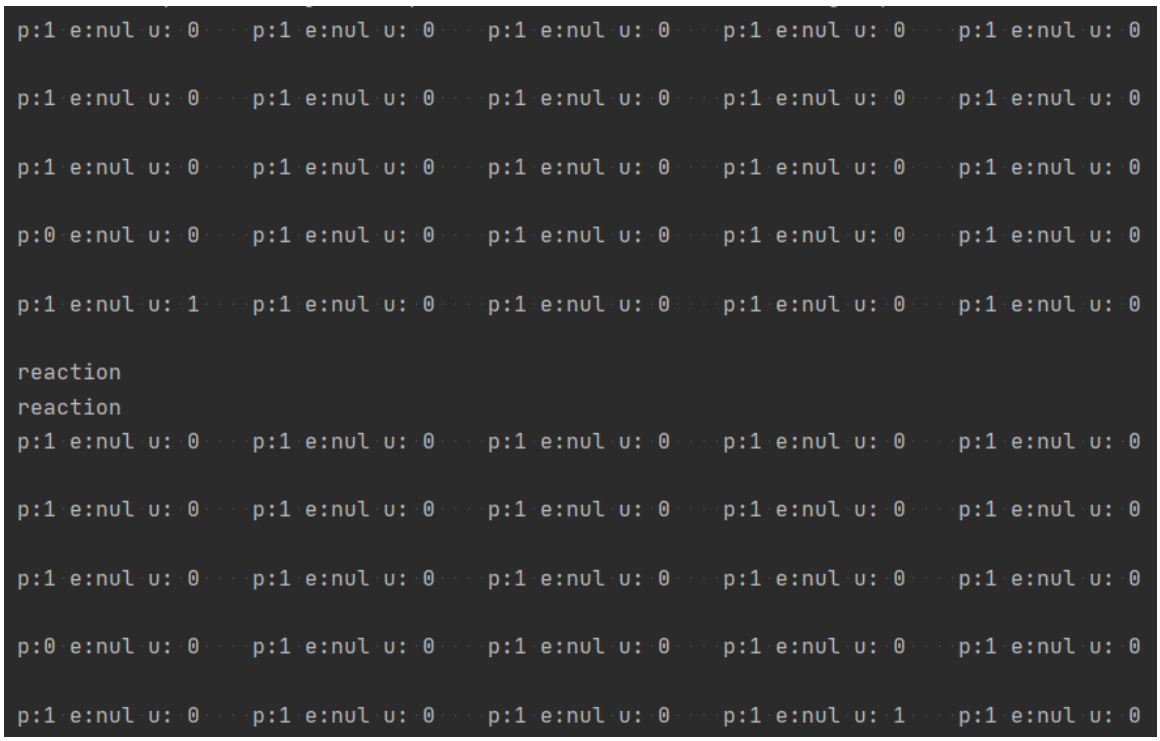


Строение системы классов:

1) Между классами Filed и Cell композиция, т.к. клетка не может существовать без поля, и поле полностью контролирует ее цикл жизни.
2) Между классами Field и Unit композиция, т.к. игрок не может находиться вне поля и существовать без него, игрок по умолчанию появляется на поле при создании поля, т.е. поле контролирует цикл жизни игрока
3) Между классами Cell и Event агрегация, т.к. событие может существовать до клетки и лишь потом быть к ней привязано
4) Между классами FieldView и Field композиция, т.к. нам необходимо существование поля, чтобы показывать его, поэтому FieldView контролирует цикл жизни Field

Пример работы программы: имеется поле 5x5, все клетки проходимые, кроме первой клетки второй снизу строки (клетка 1,0). Параметр p: проходимость, e: событие (в данной работе нет ни одного события), u: находится ли игрок в данный момент на этой клетке.

Игрок пытается походить 7 раз вверх и 2 раза влево. Походить вверх не может, т.к. клетка над ним не проходима, после чего шагает 2 раза влево (2 раза срабатывает реакция клетки от шага игрока), после чего оказывается в клетке 0,3 (нижняя строка, 4-й столбец).



Выводы

В ходе выполнения лабораторной работы были приобретены знания об объектно-ориентированном программировании и опыт работы с ним на примере практической задачи на C++. Были приобретены навыки написания классов, конструкторов и методов.