МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Создание классов, конструкторов и методов

Студентка гр.1381 _____ Рымарь М.И.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

Цель работы.

Изучить основы объектно-ориентированного программирования. Научиться создавать простые классы и работать с ними, а также с конструкторами и методами. Реализовать основу консольной игры: прямоугольное игровое поле, состоящее из клеток, событие и игрока.

Задание.

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.
- Игровое поле должно быть зациклено по вертикали и горизонтали.

Примечания:

- При написании конструкторов учитывайте, что события должны храниться по указателю для соблюдения полиморфизма
- Для управления игроком можно использовать медиатор, команду, цепочку обязанностей

Выполнение работы.

В ходе выполнения лабораторной работы для реализации всех подзадач были созданы такие классы, как: класс клетки; класс поля; класс игрока; классы, отвечающие за взаимодействие пользователя с программой.

Класс клетки – *Cell* – имеет три поля (*possibility*, *playerOnCell* и *event*), которые имеют приватный модификатор доступа, сеттеры (*setPossibility*, *setPlayerOnCell* и *setEvent*), геттеры (*getPossibility*, *getPlayerOnCell* и *getEvent*), метод *update* и конструктор *Cell*.

Класс поля – *Field* - отвечает за создание поля, в том числе выделение памяти под него. Также в этом классе находится функция, реализующая перемещение игрока по полю.

1. Приватные поля: *width*, *height*, *cells* (двумерный массив – матрица игрового поля), *playerCoordinates* (координаты игрока). Также приватный модификатор доступа имеет метод swap, который был создан для конструкторов копирования и перемещения, и соответствующих им операторов присваивания.

2. Конструктор *Field*, определённый стандартными значениями длины и ширины – 10 и 10, соответственно.

3. Конструкторы копирования и перемещения и соответствующие им операторы присваивания.

4. Два метода - *makeField()* и *playerMove()*, первый из них отвечает за создание поля, второй – за передвижение игрока. Во втором методе находится оператор множественного выбора *switch*, с помощью которого реализовано перемещение игрока по полю.

5. Геттеры - *getWidth*, *getHeight*, *getCells*.

Класс игрока – *Player* – хранит в себе поля, содержащие характеристики игрока (*health*, *keys*, *protection*), геттеры и сеттеры, соответствующие этим характеристикам, конструктор, заполняющий характеристики игрока, абстрактный класс *enum Step* («шаги» игрока).

Абстрактный класс события *Event* (интерфейс) – в нём нет полей, он создан в качестве класса, от которого будут наследоваться другие события. В этом классе есть метод *execute*, выполняющий определённое действие, зависящее от конкретного события. Также в классе есть деструктор.

Дополнительный класс для вывода поля *fieldView* содержит в себе одно поле *field*, один метод *printField* и конструктор.

Класс, отвечающий за считывание данных, *commandReader* – содержит поля, хранящие размеры игрового поля – *width*, *height*; символ *c*, в котором хранится направление перемещения игрока (вводится пользователем), и переменная *step*, которая будет меняться в зависимости от введённого символа, отвечает за перемещение. В классе есть геттеры, возвращающие значения width, *height*, *char* и *step*; методы, отвечающие за считывание *readSize*, *readStep*,

readChar; метод, проверяющий на корректность введённые данные *checkCommand*.

Класс Controller, который задаёт параметры игры и контролирует её, содержит два поля – *field*, *viewField* – само игровое поле и его отображение. Далее реализован конструктор, задающий эти параметры, и четыре метода *setField* (создаёт поле с параметрами, заданными пользователем), *setDefaultField* (если пользователь не ввёл параметры, либо решил создать стандартное поле, то этот метод сгенерирует стандартное поле), *setStep* (метод меняет поле в зависимости от движения игрока), *showField* (функция выводит само поле).

Класс Mediator, который является связующим звеном между двумя предыдущими классами и пользователем, содержит два поля: *input* (тип *commandReader*) и *game* (тип *Controller*). Также класс содержит метод *start*, который начинает игру.

В основной функции main вызываем только метод медиатора: *Mediator().start()*.

Тестирование.

Интерфейс командной строки при генерировании поля стандартных размеров представлен на рисунке 1.

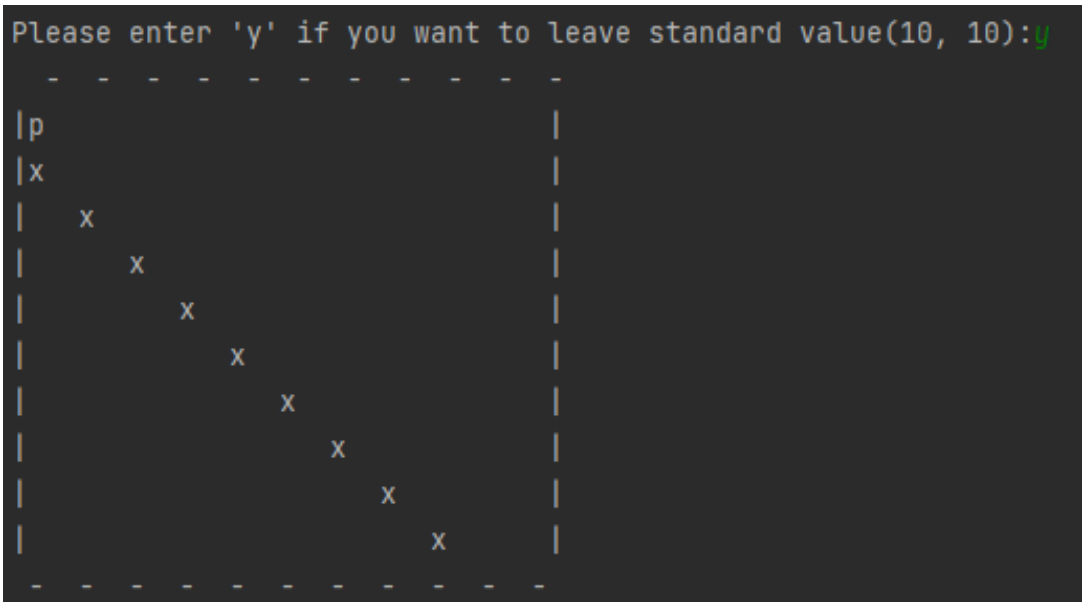

Рисунок 1 – Генерирование стандартного поля

Работа программы при генерировании поля с параметрами, заданными пользователем, представлен на рисунке 2.

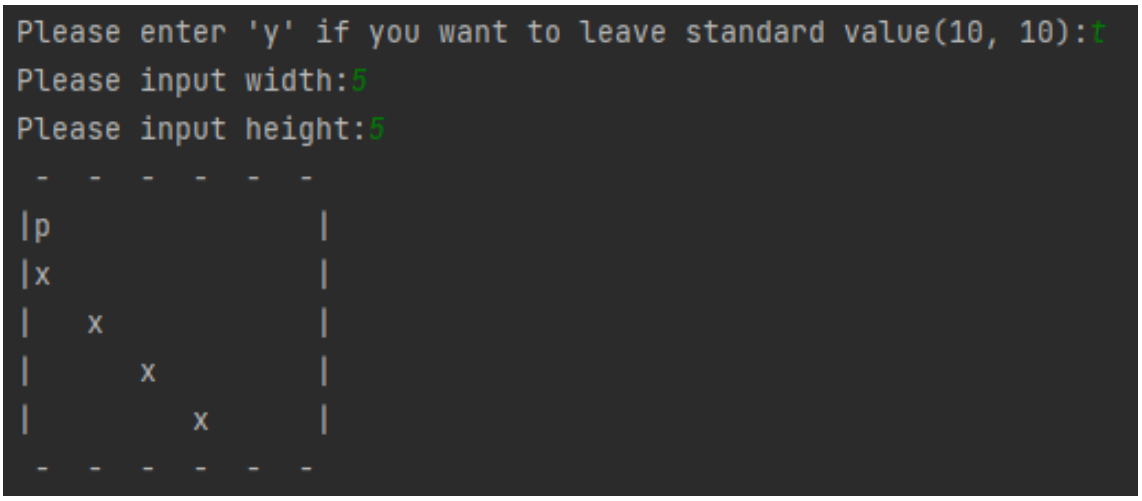

Рисунок 2 – Генерирование поля с пользовательскими параметрами

Перемещение игрока по игровому полю можно увидеть на рисунках 3 и 4.

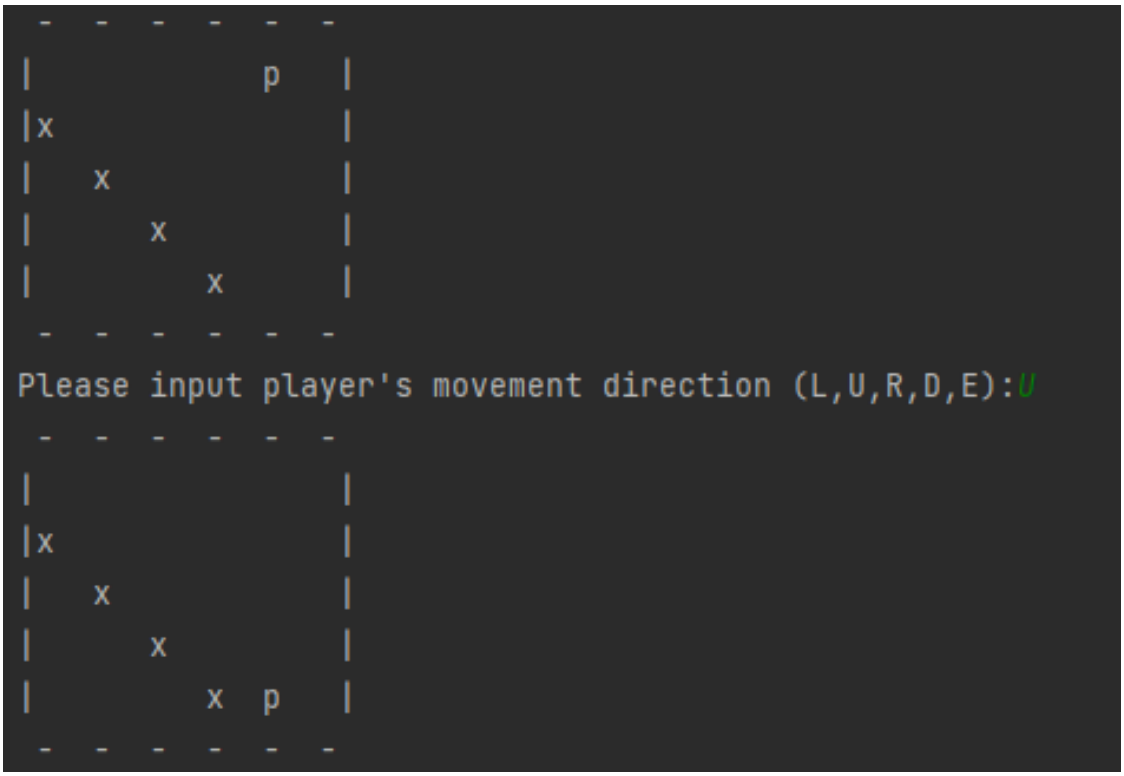

Рисунок 3 – Перемещение игрока вверх

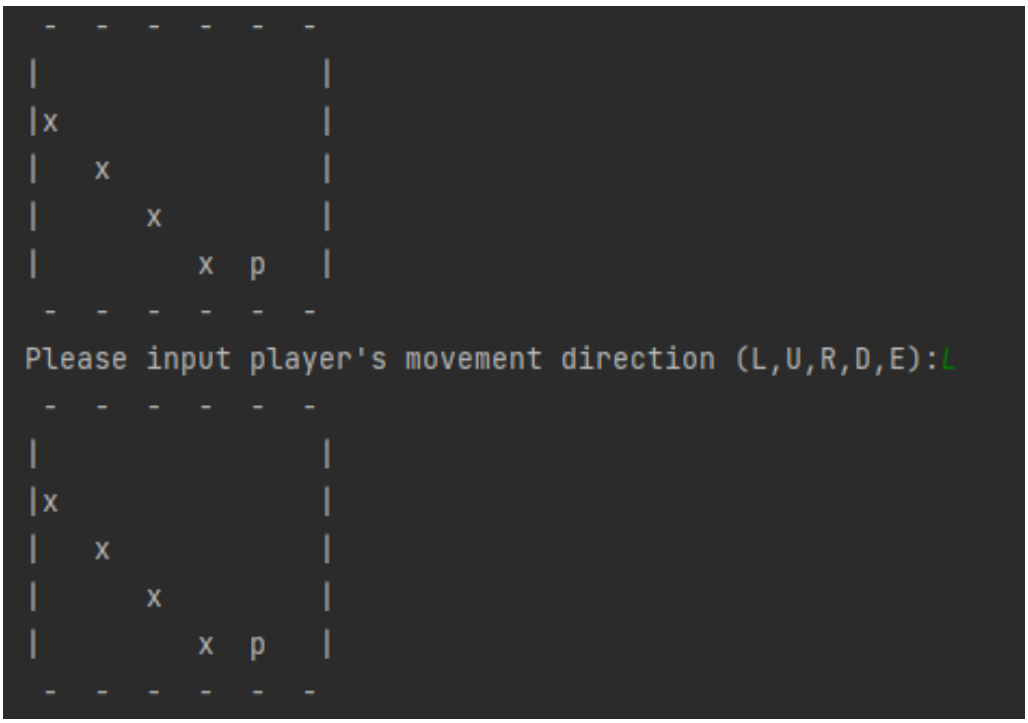

Рисунок 4 – Попытка перемещения игрока на клетку со «стеной»

Обработка некорректных данных, введённых пользователем, показана на рисунке 5.

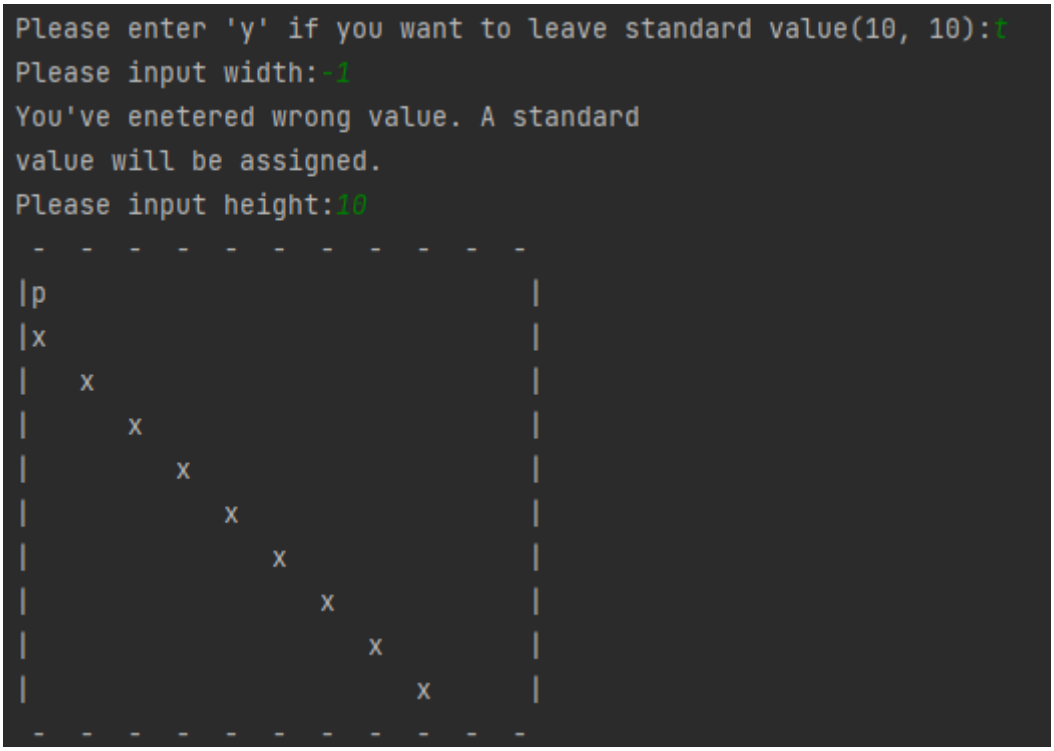

Рисунок 5 – Обработка некорректных данных

UML-диаграмма межклассовых отношений.

На рисунке 6 представлена UML-диаграмма межклассовых отношений.

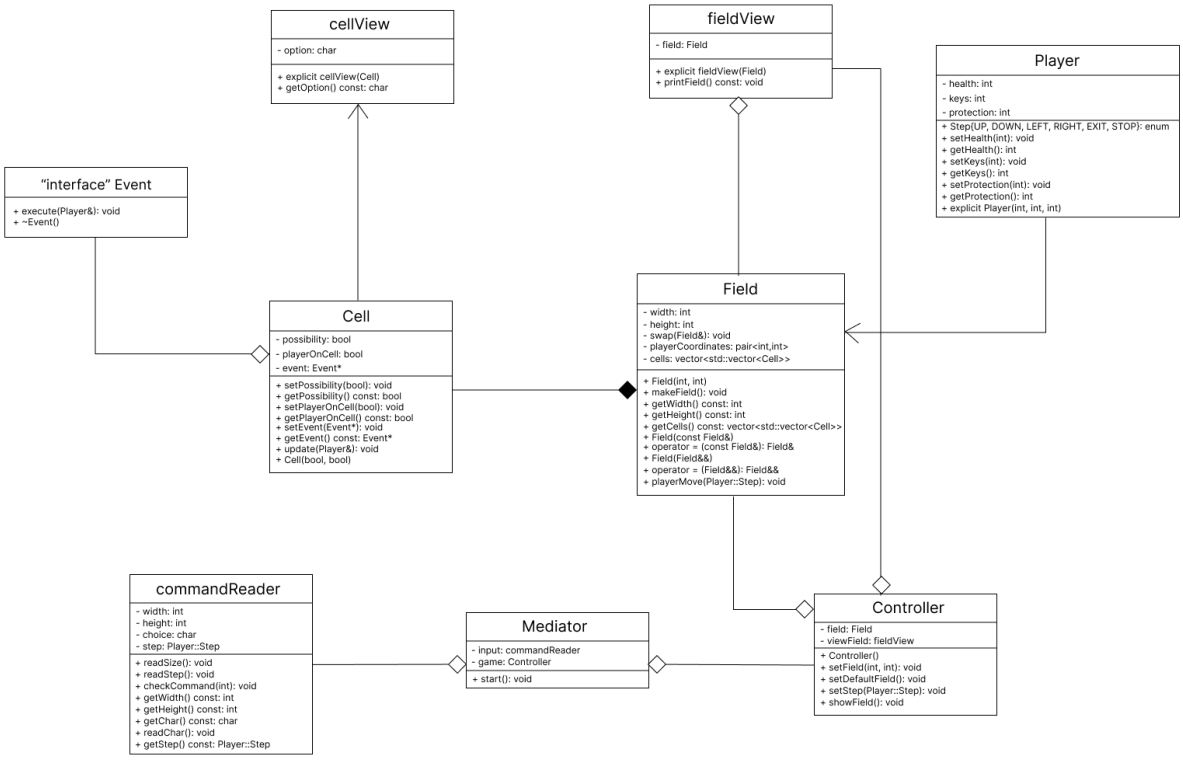

Рисунок 6 – UML-диаграмма

Выводы.

В ходе выполнения лабораторной работы были изучены основы объектно-ориентированного программирования. Реализована «основа» консольной игры: классы клетки, поля и игрока, интерфейс события.