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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Объектно-ориентированное программирование»
Тема: Интерфейсы, динамический полиморфизм.

Студент гр. 1384 _____ Тапеха В. А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2022

Цель работы.

Изучить понятие принципы полиморфизма, научиться реализовывать классы, которые в иногда являются интерфейсом, и осуществлять межклассовые отношения соблюдая полиморфизм.

Задание.

Реализовать систему событий. Событие - сущность, которая срабатывает при взаимодействии с игроком. Должен быть разработан класс интерфейс общий для всех событий, поддерживающий взаимодействие с игроком. Необходимо создать несколько групп разных событий реализуя унаследованные от интерфейса события (например, враг, который проверяет условие, будет ли воздействовать на игрока или нет; ловушка, которая безусловно воздействует на игрока; событие , которое меняет карту; и.т.д.). Для каждой группы реализовать конкретные события, которые по разному воздействуют на игрока (например, какое-то событие заставляет передвинуться игрока в определенную сторону, а другое меняет характеристики игрока). Также, необходимо предусмотреть событие “Победа/Выход”, которое срабатывает при соблюдении определенного набора условий.

Реализовать ситуацию проигрыша (например, потери всего здоровья игрока) и выигрыша игрока (добрался и активировал событие “Победа/Выход”)

Требования:

- Разработан интерфейс события с необходимым описанием методов
- Реализовано минимум 2 группы событий (2 абстрактных класса наследников события)
- Для каждой группы реализовано минимум 2 конкретных события (наследники от группы события)
- Реализовано минимум одно условное и безусловное событие (условное - проверяет выполнение условий, безусловное - не проверяет).

- Реализовано минимум одно событие, которое меняет карту (меняет события на клетках или открывает расположение выхода или делает какие-то клетки проходимыми (на них необходимо добавить события) или не непроходимыми
- Игрок в гарантированно имеет возможность дойти до выхода

Примечания:

- Классы событий не должны хранить никакой информации о типе события (никаких переменных и функций дающие информации о типе события)
- Для создания события можно применять абстрактную фабрику/прототип/строитель

Выполнение работы.

Для выполнения лабораторной работы были созданы классы, отвечающие за игрока, создание клетки поля, создания поля, их вывод и взаимодействие пользователя с программой.

Новые классы:

1) Был создан интерфейс CellType, от которого будут наследоваться все типы клеток (CellBox, CellHeal, CellKey, CellNewLevel, CellStandard, CellTrap, CellWall). Интерфейс содержит метод get_event, который возвращает указатель на событие. В случае когда наследуемый класс не содержит внутри себя событие возвращается nullptr.

2) Был создан интерфейс Event, от которого будут наследоваться все интерфейсы групп событий (EventPlayer, EventField).

Интерфейс EventPlayer содержит в себе в себе метод execute(Player& player), который, исходя из того, какой класс от него наследуется, изменяет поля игрока. От него наследуются классы Box, Heal, Key,Trap, которые соответственно увеличивает поле xp игрока (если соблюдается условие, повышает увеличивает lvl), увеличивает поле health игрока, увеличивает поле num_of_keys, уменьшает поле health игрока на случайно количество единиц.

Интерфейс EventField содержит в себе в себе метод execute(Field& field), который, исходя из того, какой класс от него наследуется, изменяет класс типа Field. От него наследуется класс NewLevel, при вызове метода execute чистит поле и заново его генерирует.

3) Был написан класс PlayerView, который выводит все поля игрока.

4) Были написан интерфейс Observer, который содержит содержит метод update и от которого будут наследоваться PlayerView и FieldView. Метод update выводит либо статистику игрока, либо поле.

5) Создан класс Observable, который хранит в себе все наблюдаемые объекты. Есть методы attach, detach, notify, которые добавляет в вектор наблюдаемых объектов, удаляет объект из этого вектора и вызывает update у всех элементов.

Измененные классы.

1) В классе Field был переписан метод make_field под новую реализацию. Этот метод теперь полностью инициализирует двумерные массив из клеток. Он же вызывается в конструкторе. Так было сделано для того, чтобы в будущем воспользоваться этим методом при полном изменении поля. Также был написан метод clear_field, который чистит память. Он же вызывается в деструкторе поля. Метод change_player_pos теперь возвращает событие, которое произошло на конкретной клетке. Это нужно чтобы поднять событие на уровень класса Controller.

2) В класс Controller были добавлены методы end_game() и win_game(), которые проверяют конец игры. Был добавлен метод check_event, который проверяет к какой группе событий относится событие.

3) В класс CommandReader были добавлены методы print_end_game() и print_win_game(), которые проверяют конец игры.

Тестирование программы.

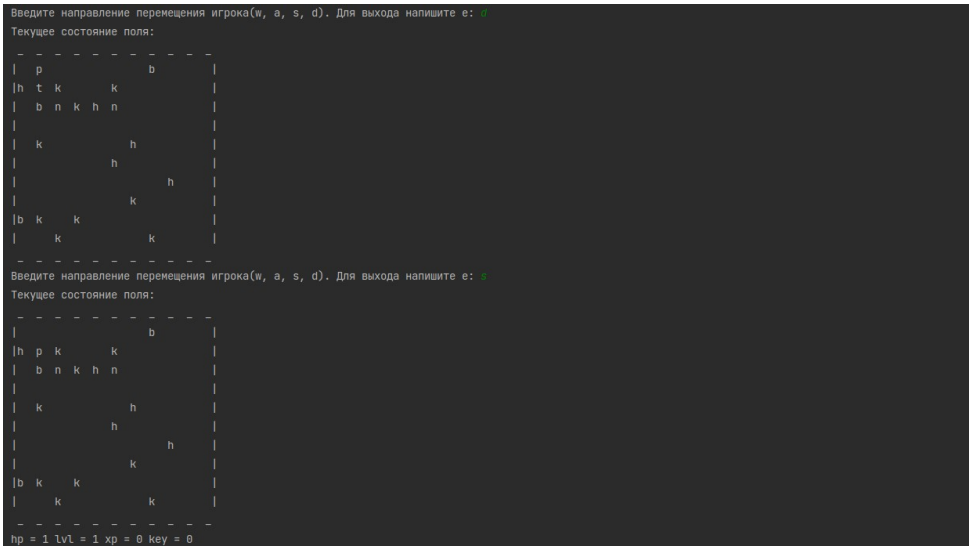

Рис. 1. Тестирование программы

Рис. 2. Тестирование программы

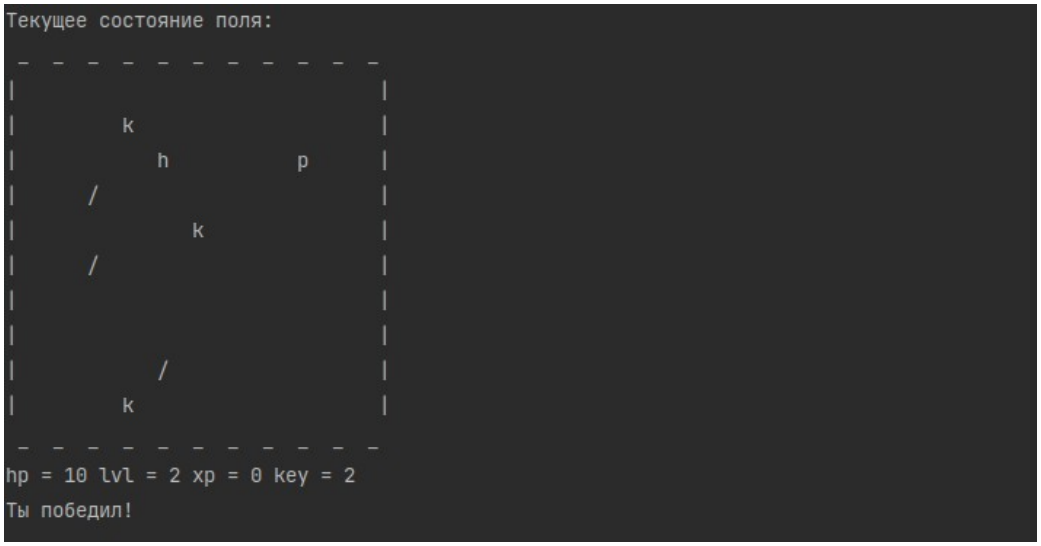

Рис. 3. Тестирование программы.

Рис. 4. Тестирование программы

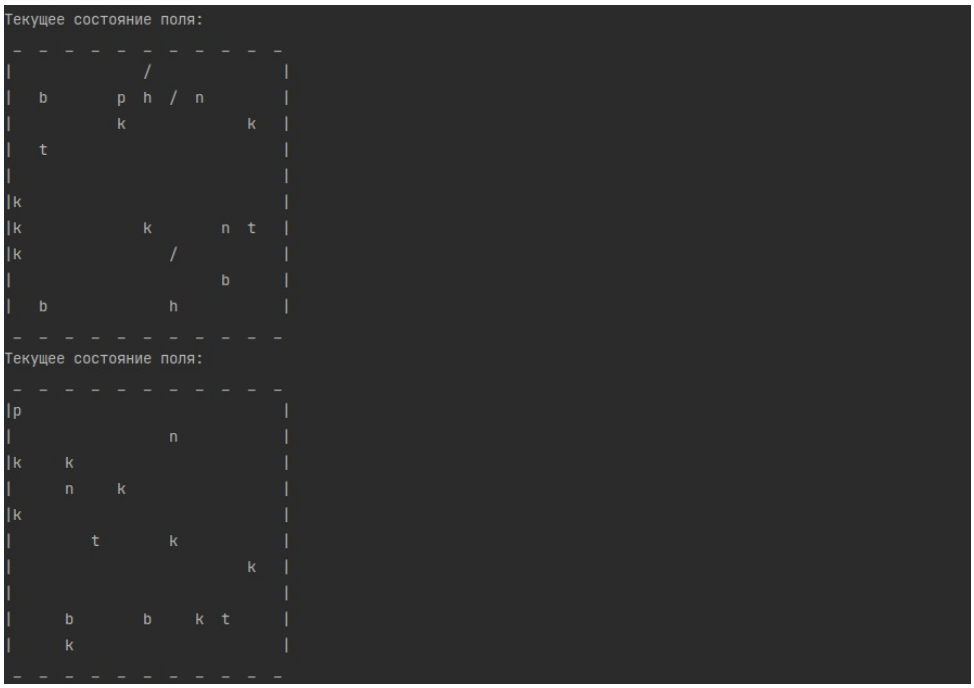

Рис. 5. Тестирование программы.

UML-диаграмма межклассовых отношений.

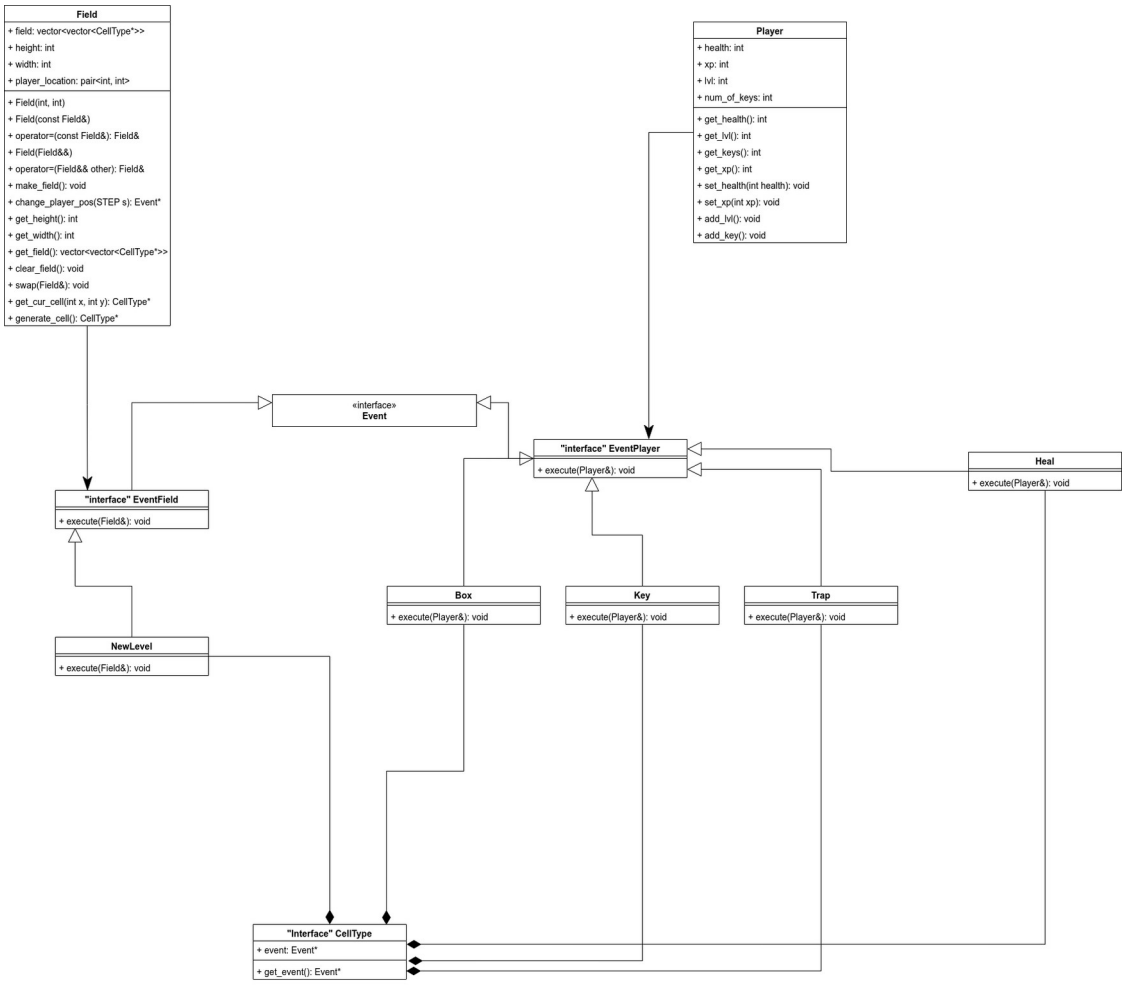

Рис. 6 UML-диаграмма.

Выводы.

Я изучил понятие принципы полиморфизма, научился реализовывать классы, которые иногда являются интерфейсом, и осуществлять межклассовые отношения, соблюдая полиморфизм.