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, полиморфизм

Студентка гр. 0382 _____ Ильин Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Изучить применение интерфейсов, полиморфизм.

Задание.

Могут быть три типа элементов располагающихся на клетках:

1. Игрок - объект, которым непосредственно происходит управление. На поле может быть только один игрок. Игрок может взаимодействовать с врагом (сражение) и вещами (подобрать).
2. Враг - объект, который самостоятельно перемещается по полю. На поле врагов может быть больше одного. Враг может взаимодействовать с игроком (сражение).
3. Вещь - объект, который просто располагается на поле и не перемещается. Вещей на поле может быть больше одной.

Требования:

- Реализовать класс игрока. Игрок должен обладать собственными характеристиками, которые могут изменяться в ходе игры. У игрока должна быть прописана логика сражения и подбора вещей. Должно быть реализовано взаимодействие с клеткой выхода.
- Реализовать три разных типа врагов. Враги должны обладать собственными характеристиками (например, количество жизней, значение атаки и защиты, и т. д.; желательно, чтобы у врагов были разные наборы характеристик). Реализовать логику перемещения для каждого типа врага. В случае смерти врага он должен исчезнуть с поля. Все враги должны быть объединены своим собственным интерфейсом.
- Реализовать три разных типа вещей. Каждая вещь должна обладать собственным взаимодействием на ход игры при подборе *(например, лечение игрока)*. При подборе, вещь должна исчезнуть с поля. Все вещи должны быть объединены своим собственным интерфейсом.
- Должен соблюдаться принцип полиморфизма

Потенциальные паттерны проектирования, которые можно использовать:

- *Шаблонный метод (Template Method) - определение шаблона поведения врагов*
- *Стратегия (Strategy) - динамическое изменение поведения врагов*
- *Легковес (Flyweight) - вынесение общих характеристик врагов и/или для оптимизации*
- *Абстрактная Фабрика/Фабричный Метод (Abstract Factory/Factory Method) - создание врагов/вещей разного типа в runtime*
- *Прототип (Prototype) - создание врагов/вещей на основе "заготовок"*

Выполнение работы.

Для реализации графического интерфейса используется SFML библиотека.

Создан интерфейс класса элемента клетки Elem от него наследуются интерфейсы IPerson и IItem, отвечающие за живых существ и вещи соответственно. От IPerson в свою очередь наследуются Player и Enemy, отвечающие за игрока и врагов соответственно.

Для всех живых существ(персон) создан вектор VecOfPerson с ссылками на них, для упрощения дальнейшей работы.

Создан класс Actions, который отвечает за передвижения и атаку персон.

Также реализован класс графики.

Далее подробнее о каждом классе.

Class Elem:

Это интерфейс класс элемента клетки, служит для хранения ссылки на любой объект, который может лежать на поле.

Class IPerson:

Это интерфейс живых существ, в нём объявлены методы, которые нужны всем персонам, такие как: вернуть наносимый урон(или: позицию, скорость, передвижение, индивидуальный номер, здоровье, шаг), получить урон(или: позицию, вещь, шаг).

Дополнительно в классе реализовано нумерование PAction отвечающее за активное состояние.

Class IItem:

Это интерфейс вещей, в нём объявлен метод возвращающий тип вещи.

Class Player:

Это класс игрока, он содержит следующие поля: характеристики игрока, сторону, в которую он будет идти при следующем шаге, индивидуальный номер. Также в данном классе реализованы методы из родительского класса IPerson, помимо методов родительского класса реализованы методы работающие с максимальным количеством жизней.

Class Enemy:

Это класс врага, он содержит следующие поля: характеристики врага, индивидуальный номер. Также в данном классе реализованы методы из родительского класса IPerson.

Class Wizard:

Это класс мага, он содержит поле, являющееся флагом, для проверки можно ли двигаться в данный момент или нет. Также в данном классе переопределены методы из родительского класса Enemy, а точнее это: конструктор, метод отвечающий за передвижение и метод возвращающий тип.

Class Solder:

Это класс солдата. В данном классе переопределены методы из родительского класса Enemy, а точнее это: конструктор, метод отвечающий за передвижение и метод возвращающий тип.

Class Tank:

Это класс танка. В данном классе переопределены методы из родительского класса Enemy, а точнее это: конструктор, метод отвечающий за передвижение и метод возвращающий тип.

Class ATC:

Это класс вещи увеличивающей урон, он содержит поля типа и позиции. В данном классе реализованы методы из родительского класса IItem.

Class HP:

Это класс вещи увеличивающей здоровье, он содержит поля типа и позиции. В данном классе реализованы методы из родительского класса IItem.

Class MP:

Это класс вещи увеличивающей манну, он содержит поля типа и позиции. В данном классе реализованы методы из родительского класса IItem.

Class Actions:

Это класс отвечающий за все действия.

В классе объявлены публичные поля:

- *VecOfPerson& _vec* — указатель на вектор персон, которые и будут производить действия
- *Floor& _floor* — указатель на поле

Реализован конструктор который получает указатели на вектор живых существ и на поле.

Методы класса:

- *PAction move(PAction side)* — даёт персонам возможность перемещаться, атаковать, поднимать предметы. Возвращает тип с которым завершилась игра, если игра не завершилась возвращает действие подразумевающее ничего.
- void movePerson(Coords new_pos, int num_of_per) — передвигает персон по игровому полю.
- *Coords newPos(Coords now_pos, int side_of_per)* — преобразует рандомное число, враги они создают в направление.

Class VecOfPerson:

Это класс хранящий всех живых существ.

В классе объявлено приватное поле:

- *int _quantity* — количество живых существ
- *const Floor &_floor* – указатель на поле
- std::vector <IPerson*> persons – вектор живых существ

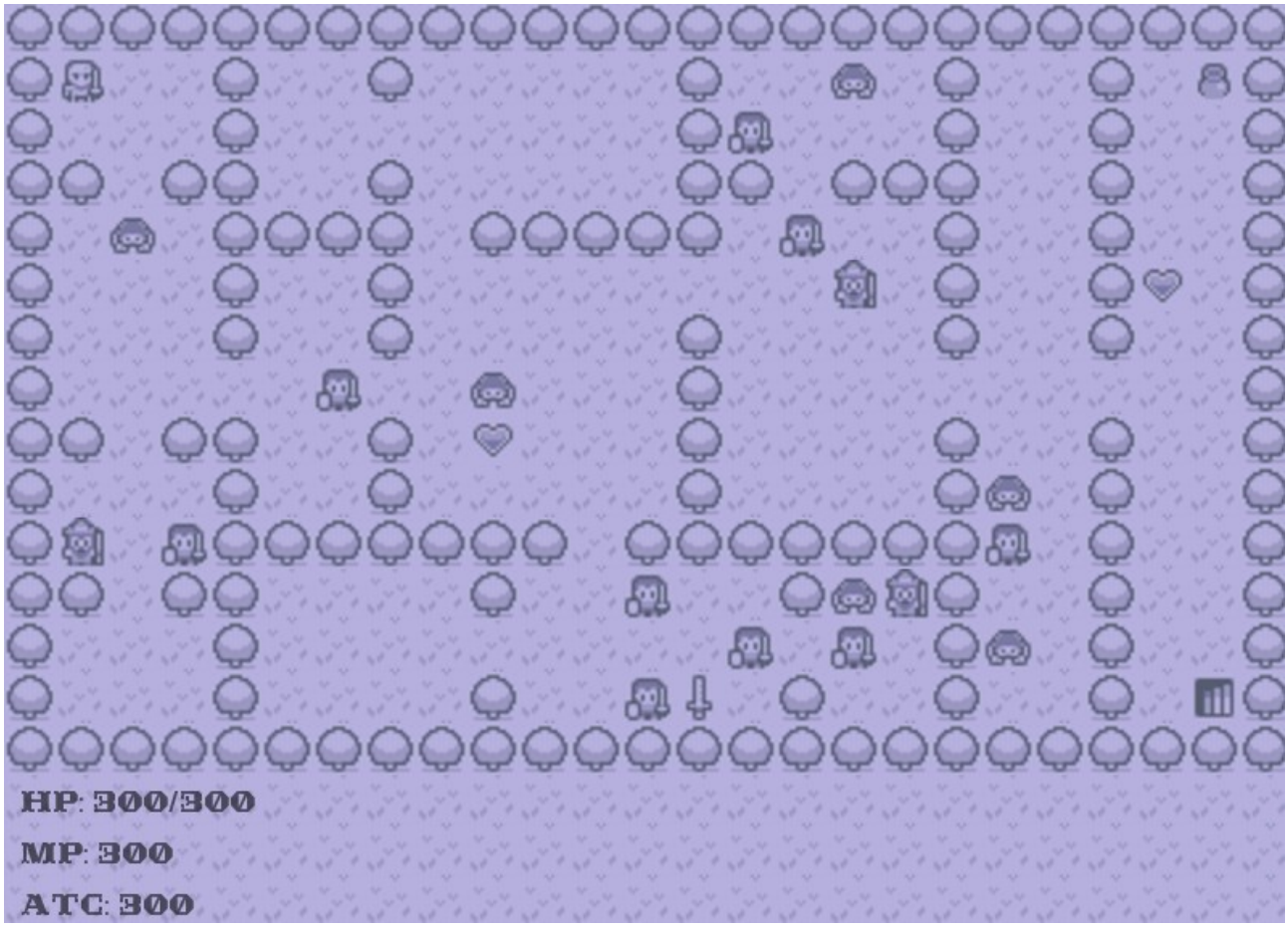
Реализован конструктор, который по полю собирает всех персон, считает их, даёт им свои индивидуальные номера и заполняет вектор живых существ.

Методы класса:

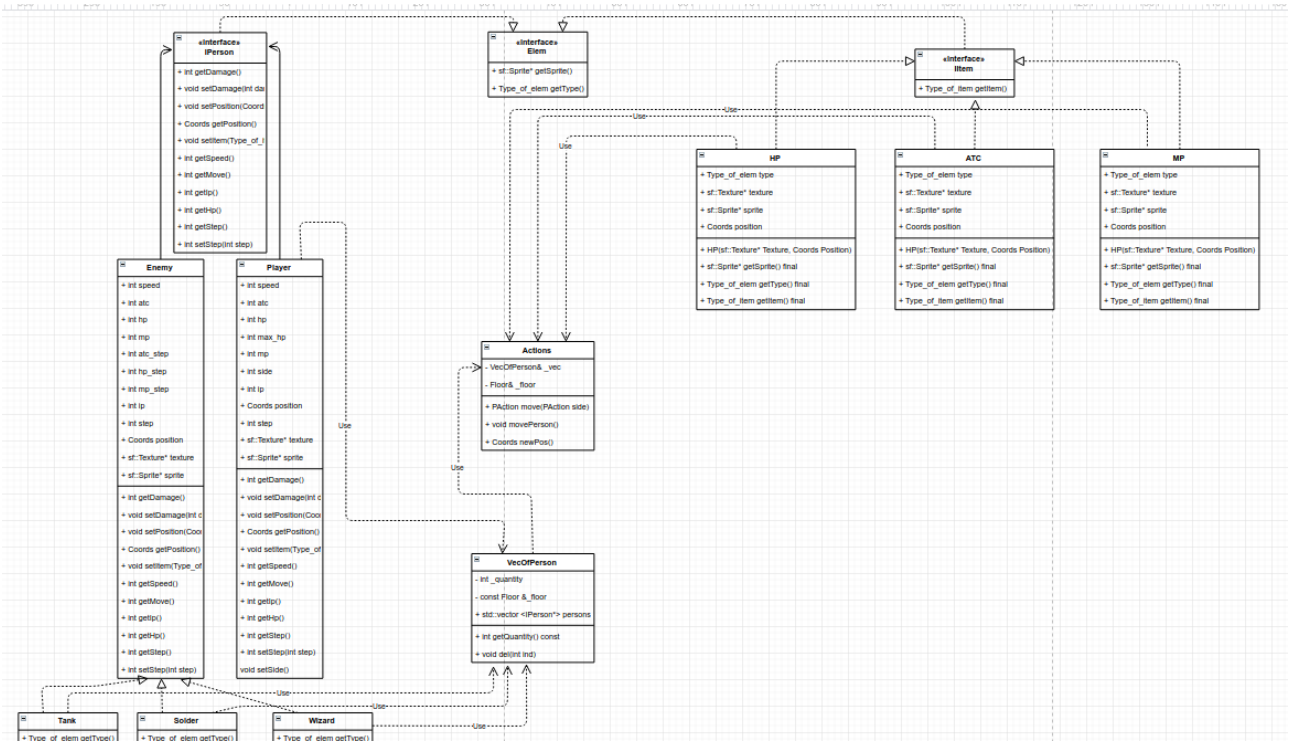
- *int getQuantity() const* — возвращяет количество живых существ.
- *void del(int ind)* — удаляет определённую персону.

Тестирование.

Результат работы программы:



UML диаграмма



Выводы.

Было исследовано создание интерфейсов, наследованию от них, работе с интерфейсами. Разработана программа, которая создает на игровом поле персон и вещи и отрисовывает их в окне.