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.

Студент гр. 0383 Подопригора И.П.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2021

Задание.

Могут быть три типа элементов, располагающихся на клетках:

Игрок - объект, которым непосредственно происходит управление. На поле может быть только один игрок. Игрок может взаимодействовать с врагом (сражение) и вещами (подобрать).

Враг - объект, который самостоятельно перемещается по полю. На поле врагов может быть больше одного. Враг может взаимодействовать с игроком (сражение).

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

Выполнение работы.

Реализован базовый класс Entity, от которого наследуются класс вещей Item и класс существ (враги и игрок) Creature, которые будут обладать общими методами: получить координаты, получить информацию, существует ли объект на поле, изменить позицию объекта на поле, перестать существовать. Всё это предотвращает дублирование кода.

Все вещи наследуются от класса Item, который имеет единственный виртуальный метод воздействия на игрока. Каждый класс вещи реализует этот метод по своему, соблюдается принцип полиморфизма: если мы имеем вещь на поле(указатель на базовый класс) , нам не нужно знать какую именно вещь, мы в любом случае заставляем вещь повлиять на игрока. Всего реализовано 4 вещи – Medkit – увеличивает здоровье игрока на 5 очков, Grindstone – увеличивает значение атаки игрока на 1, Trap – уменьшает здоровье игрока на 1, Toolkit – увеличивает дальность атаки игрока на 1.

Класс Creature имеет общие для всех существ поля и методы. Существа имеют очки здоровья (health_points), очки атаки (attack_points) и дальность атаки (attack_range). Также у них есть общие методы: получить урон(get_damage(int dmg)), который уменьшает здоровье существа и помечает его убитым, если здоровье <= 0.

Класс игрока Player наследуется от Creature и реализует интерфейс IMovable – игрок определяет метод move, который получает на вход направление движения и пытается сдвинуть игрока в этом направлении на одну клетку. Если происходит попытка сдвинуть игрока в непроходимую клетку или врага, то игрок остается на месте, если же на пути у игрока предмет, то игрок взаимодействует с ним, если игрок идет в клетку выхода, то его поле bool level_passed становится true.

Все враги реализуют интерфейс IEnemy с методами next_action – сделать следующее действие, is_player_reachable – можно ли достать атакой до игрока, так выполняется принцип полиморфизма, далее при реализации проекта можно будет хранить массив всех врагов находящихся на поле и не узнавая, какие

конкретно это враги, заставлять их выполнять следующее действие в цикле игры. Враги Fatty и Jumper реализуют интерфейс IMovable и наследуются от базового класса Patrolling, который содержит схему патрулирования в виде массива направлений, в которых нужно идти. Враг MagicTower не является движимым и бьет игрока в радиусе своей атаки.

UML диаграмму классов см. в приложении А.

Выводы.

В ходе выполнения данной лабораторной работы были изучены принцип полиморфизма и создание интерфейсов.

ПРИЛОЖЕНИЕ А

UML ДИАГРАММА КЛАССОВ