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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «ООП»

Тема: Добавления игрока и элементов для поля

Студент гр. 9304 _____ Тиняков С.А.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Создать игрока и элементы игрового поля. Создать механизм взаимодейсвия между ними.

Задание.

Создан класс игрока, которым управляет пользователь. Объект класса игрока может перемещаться по полю, а также взаимодействовать с элементами поля. Для элементов поля должен быть создан общий интерфейс и должны быть реализованы 3 разных класса элементов, которые по разному взаимодействуют с игроком. Для взаимодействия игрока с элементом должен использоваться перегруженный оператор (Например, оператор +). Элементы поля могут добавлять очки игроку/замедлять передвижения/и.т.д.

Обязательные требования:

- Реализован класс игрока
- Реализованы три класса элементов поля
- Объект класса игрока появляется на клетке со входом
- Уровень считается пройденным, когда объект класса игрока оказывается на клетке с выходом (и при определенных условиях: например, набрано необходимое кол-во очков)
- Взаимодействие с элементами происходит через общий интерфейс
- Взаимодействие игрока с элементами происходит через перегруженный оператор

Дополнительные требования:

Для создания элементов используется паттерн Фабричный метод/Абстрактная фабрика

Реализовано динамическое изменение взаимодействия игрока с элементами через паттерн Стратегия. Например, при взаимодействии с определенным количеством элементов, игрок не может больше с ними взаимодействовать

Выполнение работы.

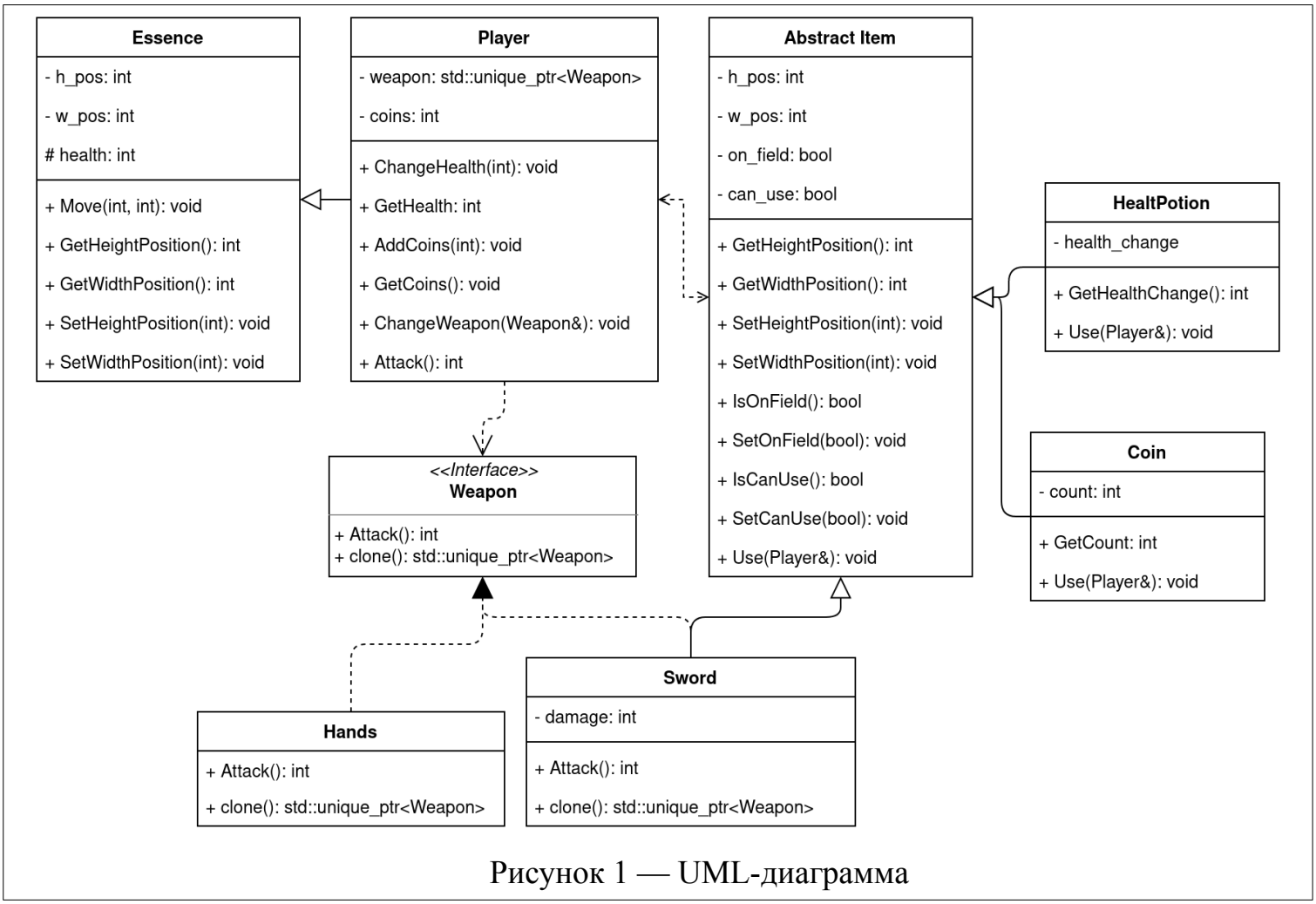

Рисунок 1 — UML-диаграмма

Класс *Essence* является реализацей любого объекта, который может передвигаться. Поля *h_pos* и *w_pos* хранят координаты соответственно по высоте и длине поля. Поле *health* обозначение количество жизней у сущности. Данное поле объявленно с модификатором доступа *protected*, так что об управлением количеством жизней должны позаботится классы-наследники. Метод *Move* изменяет координаты положения сущности на поле. Методы *GetHeightPosition, GetWidthPosition, SetHeightPosition* и

SetWidthPosition отвечают за получение и установку значений полей *h_pos* и *w_pos*.

Класс *Player* наследуется от класса *Essence*. Поле *weapon* отвечает за текущее оружие у игрока, поле *coins* — за количество монет. Методы *ChangeHealth* и *GetHealth* отвечают соответственно за изменение и получение текущего количества жизней. Аналогично *AddCoins* и *GetCoins* для количества монет. Метод *ChangeWeapon* меняет оружие игрока. Метод *Attack* возвращает силу атаки игрока. Для взаимодействия с классом *Item* был перегружен оператор +.

Для предметов был создан абстрактный класс *Item*. В нём определён чисто виртуальный метод *Use*, при помощи которого предмет взаимодействует с игроком. Поля *h_pos* и *w_pos* отвечают за положение на игровом поле. Для взаимодействия с ними были реализованны методы *GetHeightPosition, GetWidthPosition, SetHeightPosition* и *SetWidthPosition*. Поле *on_field* отвечает за то, находится ли предмет на игровом поле или нет. Поле *can_use* означает, можно ли использовать предмет или нет. Для взаимодействия с последними двумя полями были реализованы методы *IsOnField, SetOnField, IsCanUse, SetCanUse*. Также были реализованны конструктор и оператор копирования.

Класс *Coin* является реализацей предмета монетка. Данный класс наследуется от *Item*. Поле *count* отвечает за количество монеток. Для того, чтобы узнать значение *count* был реализован метод *GetCount*. Также был сделан метод *Use*, который вызывает у игрока метод *AddCoins.* Также были сделаны конструктор и оператор копирования.

Класс *HealthPotion* является реализацией предмет зелье здоровья(или яда). Поле *health_change* отвечает за количество добавленных(отнятых) жизней. Метод *GetHealthChange* возвращает значение *health_change*. Был сделан метод *Use*, который вызывает у игрока метод *ChangeHealth.* Также были сделаны конструктор и оператор копирования.

Интерфейс *Weapon* является интерфейсом оружия. Метод *Attack* возвращает количество урона оружия. Метод *clone* возвращает умный указаель на *Weapon*.

Класс *Hands* реализует интерейс *Weapon*. Метод *Attack* возвращает ноль. Метод *clone* создаёт умный указатель на *Hands*, который затем преобразовывается в умный указатель на *Weapon*.

Класс *Sword* реализует интерфейс *Weapon* и наследуется от класса *Item*. Поле *damage* отвечает за наносимый урон. Метод *Attack* возвращает значение *damage*. Метод *clone* создаёт умный указатель на *Sword*, который затем преобразовывается в умный указатель на *Weapon*. Метод *Use* вызывает у игрока метод *ChangeWeapon*.

Для проверки правильности работы классов были созданы *unit*-тесты.

Выводы.

Были реализованны классы *Essence, Player, Item, Coin, HealthPotion, Weapon, Hands* и *Sword*. При создании предметов использовались шаблоны Абстрактная и Фабричная фабрика. Для взаимодействия игрока с предметами был перегружен оператор +. Также были сделаны *unit*-тесты для проверки правильнности реализации классов.