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.0382 _____ Литягин С.М.

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.

В ходе работы были созданы или изменены следующие классы:

1. В классе Cell было добавлено поле Object* object, а также добавлены следующие методы:
   - метод GetObjectType() возвращает тип объекта, находящего на ней
   - метод GetObject() возвращает указатель на объект, который хранится в поле object клетки
   - метод SetObject(Object* object) устанавливает в поле object клетки указатель на объект
2. Класс интерфейса элемента клетки Object был изменен. Объявлены следующие методы:
   - виртуальный метод SetCoord(int x, int y) для присваивания координат объекту (метод реализуется в классах-наследниках)
   - виртуальный метод GetCoord() возвращает указатель на массив с координатами объекта (метод реализуется в классах-наследниках)
   - виртуальный метод SetType(ObjectType value) устанавливает тип объекту (метод реализуется в классах-наследниках)

- виртуальный метод SetHealth(int health) устанавливает здоровье объекту (метод реализуется в классах-наследниках)
- виртуальный метод GetHealth() возвращает значение здоровья объекта (метод реализуется в классах-наследниках)
- виртуальный метод Interaction(Object* unit) проводит взаимодействие с объектом (метод реализуется в классах-наследниках)
- виртуальный метод GetType() возвращает тип объекта (метод реализуется в классах-наследниках)
- виртуальный метод GetForce() возвращает значение урона, наносимого объектом (метод реализуется в классах-наследниках)
- виртуальный метод GetData() возвращает значение поля данных объекта (метод реализуется в классах-наследниках)

3. Класс интерфейса юнитов Unit наследуется от интерфейса Object. Объявлены следующие методы:

- виртуальный метод SetForce(int damage) для установки урона, наносимого юнитом (метод реализуется в классах-наследниках)
- виртуальный метод GetForce() для получения урона, наносимого юнитом (метод реализуется в классах-наследниках)
- виртуальный метод SetMaxHealth(int maxHealth) для установки максимального здоровья (метод реализуется в классах-наследниках)
- виртуальный метод IsAlive() для проверки, активен ли объект
- виртуальный метод Move(Field* field, int x, int y) для перемещения юнита (метод реализуется в классах-наследниках)

4. Классы юнитов Ent, Eye, Spider наследуются от класса интерфейса Unit. Описание классов в файлах Ent.cpp, Eye.cpp, Spider.cpp, определение в Ent.h, Eye.h, Spider.h. Имеют поля int health, int force, bool is_alive, ObjectType type, int coord[2]. Содержат следующие методы:

- переопределенный метод SetCoord(int x, int y); устанавливает переданные координаты в coord[0] и coord[1] соответственно (т.е. устанавливает объекту координаты, на которых он находится)
- переопределенный метод GetCoord() возвращает указатель на целочисленный массив coord (поле объекта класса), содержащий координаты объекта
- переопределенный метод SetType(ObjectType value) устанавливает в поле type переданное значение value (т.е. устанавливает тип)
- переопределенный метод SetHealth(int health) устанавливает полю health переданное значение health (т.е. устанавливает количество здоровья); также проверяет, если установленное значение <= 0, то устанавливает полю is_alive значение false
- переопределенный метод GetHealth() возвращает значение из поля health (т.е. возвращает количество здоровья)
- переопределенный метод Interaction(Object* unit) проводит взаимодействие с объектом; вызывается метод SetHealth() объекта unit для установки здоровья, равного разности его здоровья до взаимодействия и урона, наносимого объектом, метод взаимодействия которого был вызван к данному объекту; метод вызывается при определенном условии, которое проверяется при вызове метода Move () (т.е. метода передвижения объекта)
- переопределенный метод GetType() возвращает значение поля type (т.е. возвращает тип объекта)
- переопределенный метод SetForce(int damage) устанавливает в поле force значение damage (т.е. устанавливает урон, наносимый юнитом)
- переопределенный метод IsAlive() возвращает значение поля is_alive (т.е. возвращает true, если юнит все еще жив; false, если юнит уже не жив)

- переопределенный метод Move(Field* field, int x, int y) реализует перемещение юнита; выполняется проверка: если значения x и y >0 и <Size-1, а также клетка по координате (x,y) доступна для передвижения (метод клетки IsMovable() возвращает true, если тип клетки подходит для передвижения) (данное условие не проверяется при передвижении юнитов с типом spider), то если на клетке по координате (x,y) нет никакого объекта (т.е. метод клетки GetObjectType() вернет empty), то для клетки с координатой, на которой находится юнит, в поле object методом SetObject() устанавливается нулевой указатель, затем объекту устанавливается новая координата (x,y) методом SetCoord(), а клетке по этой же координате в поле object методом SetObject() устанавливается указатель на данный объект; если же на клетке по координате (x,y) есть объект с типом hero (т.е. игрок), то вызывается метод юнита Interaction() к объекту, находящемуся на клетке с данной координатой (таким образом, юниты взаимодействуют только с игроком)

5. Юниты создаются при использовании паттерна Абстрактная фабрика. Для этого был реализован класс интерфейса фабрик ObjectFabric. Описание класса в файле ObjectFabric.cpp, определение в ObjectFabric.h. Объявлен следующий метод:

- виртуальный метод CreateUnit() возвращает указатель на созданный объект (метод реализуется в классах-наследниках)

6. Классы фабрик EntFabric, EyeFabric, SpiderFabric (производят юнитов ent, eye, spider соответственно) наследуются от класса интерфейса ObjectFabric. Описание классов в файлах EntFabric.cpp, EyeFabric.cpp, SpiderFabric.cpp, определение в EntFabric.h, EyeFabric.h, SpiderFabric.h. Содержат следующий метод:

- переопределенный метод CreateUnit() возвращает указатель на созданный объект

7. Класс Game. Добавлено два метода:

- метод CreateEvil(Field* field, int EVIL) для генерации противников на поле; генерируется EVIL штук противников, с помощью функции rand() выбирает число до 9: если оно менее 3, то создается противник типа eye; если оно менее 6, то создается противник типа ent; если оно менее 9, то создается противник типа spider; созданный противник устанавливается на координату (x,y), где x и y меньше Size-2 и больше 1, а также при условии, что на клетке с такой координатой нет объекта (если объект на клетке уже есть, то выбираются другие x и y с помощью функции rand(), пока условие не выполнится)
- метод CreateThing(Field* field, int THING) для генерации вещей на поле; генерируется THING штук вещей, с помощью функции rand() выбирает число до 9: если оно менее 3, то создается вещь типа candy; если оно менее 6, то создается вещь типа axe; если оно менее 9, то создается вещь типа coin; созданная вещь устанавливается на координату (x,y), где x и y меньше Size-2 и больше 1, а также при условии, что на клетке с такой координатой нет объекта (если объект на клетке уже есть, то выбираются другие x и y с помощью функции rand(), пока условие не выполнится)

8. Класс интерфейса вещей Thing. Описание класса в файле Thing.cpp, определение в Thing.h. Объявлен следующий метод:

- виртуальный метод IsAvailable() для проверки, доступен ли объект (метод реализуется в классах-наследниках)

9. Классы вещей, наследуемые от интерфейса Thing, Candy, Coin, Axe. Класс Candy имеет поле int hp, класс Axe имеет поле int damage, класс

Coin имеет поле int value; классы Candy, Coin, Axe также содержат поля int coord[2], ObjectType type, bool is_available. Содержат следующие методы:

- переопределенный метод SetCoord(int x, int y); устанавливает переданные координаты в coord[0] и coord[1] соответственно (т.е. устанавливает объекту координаты, на которых он находится)
- переопределенный метод GetCoord() возвращает указатель на целочисленный массив coord (поле объекта класса), содержащий координаты объекта
- переопределенный метод GetData() возвращает значение поля данных (hp, damage, value соответственно для классов Candy, Coin, Axe) и устанавливает в поле is_ available значение false (поскольку метод вызывается, когда произошло взаимодействие с этим предметом, то мы и делаем его недоступным в дальнейшем)
- переопределенный метод IsAvailable() возвращает значение поля is_ available (т.е. возвращает true, если предмет все еще доступен; false, если предмет уже не доступен)
- переопределенный метод GetType() возвращает значение поля type (т.е. возвращает тип объекта)

10. Класс игрока Hero. Наследуется от класса интерфейса Unit. Имеет поля int health, int force, int coins, bool is_alive, int MaxHealth, ObjectType type, int coord[2]. Содержат следующие методы:

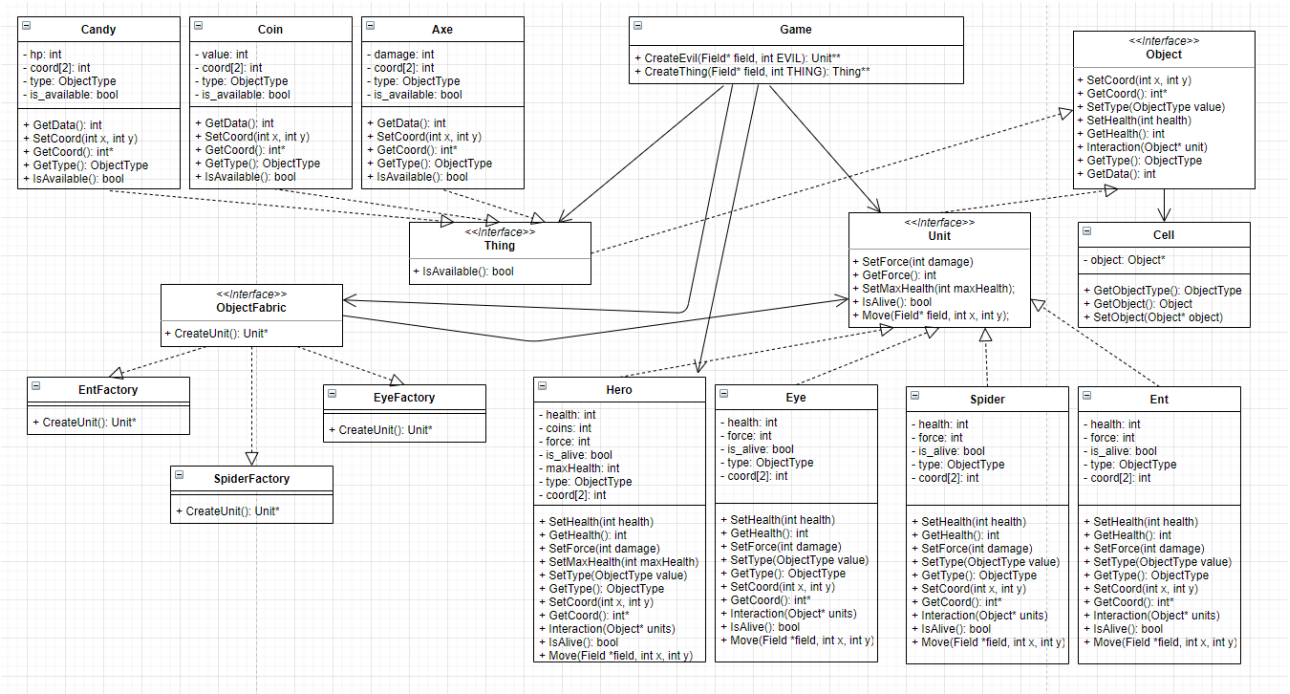
- переопределенный метод SetMaxHealth(int maxHealth) для установки максимального здоровья
- переопределенный метод SetCoord(int x, int y); устанавливает переданные координаты в coord[0] и coord[1] соответственно (т.е. устанавливает объекту координаты, на которых он находится)
- переопределенный метод GetCoord() возвращает указатель на целочисленный массив coord (поле объекта класса), содержащий координаты объекта

- переопределенный метод SetType(ObjectType value) устанавливает в поле type переданное значение value (т.е. устанавливает тип)
- переопределенный метод SetHealth(int health) устанавливает полю health переданное значение health (т.е. устанавливает количество здоровья); также проверяет, если установленное значение <= 0, то устанавливает полю is_alive значение false
- переопределенный метод GetHealth() возвращает значение из поля health (т.е. возвращает количество здоровья)
- переопределенный метод Interaction(Object* unit) проводит взаимодействие с объектом; если тип объекта unit соответствует либо ent, либо eye, либо spider (т.е. противники), то объекту unit устанавливается здоровье, равное разности его здоровья до взаимодействия и урона, наносимого игроком; если тип бъекта unit соответствует axe (т.е. предмет), то игроку устанавливает урон, равный сумме его урона до взаимодействия с предметом и значения поля данных damage предмета; если тип бъекта unit соответствует candy (т.е. предмет), то игроку устанавливает здоровье, равное сумме его здоровья до взаимодействия с предметом и значения поля данных hp предмета; если тип бъекта unit соответствует coin (т.е. предмет), то игроку устанавливает количество монет, равное сумме его монет до взаимодействия с предметом и значения поля данных value предмета
- переопределенный метод GetType() возвращает значение поля type (т.е. возвращает тип объекта)
- переопределенный метод SetForce(int damage) устанавливает в поле force значение damage (т.е. устанавливает урон, наносимый игроком)
- переопределенный метод IsAlive() возвращает значение поля is_alive (т.е. возвращает true, если игрок все еще жив; false, если игрок уже не жив)

- переопределенный метод Move(Field* field, int x, int y) реализует перемещение юнита; выполняется проверка: если значения x и y >0 и <Size-1, а также клетка по координате (x,y) доступна для передвижения (метод клетки IsMovable() возвращает true, если тип клетки подходит для передвижения) (данное условие не проверяется при передвижении юнитов с типом spider), то если на клетке по координате (x,y) нет никакого объекта (т.е. метод клетки GetObjectType() вернет empty), то для клетки с координатой, на которой находится юнит, в поле object методом SetObject() устанавливается нулевой указатель, затем объекту устанавливается новая координата (x,y) методом SetCoord(), а клетке по этой же координате в поле object методом SetObject() устанавливается указатель на данный объект; если же на клетке по координате (x,y) есть объект, тип которого отличен от empty (т.е. какой-то объект), то вызывается метод юнита Interaction() к объекту, находящемуся на клетке с данной координатой (таким образом, игрок взаимодействует с объектами)
- метод SetCoins(int value) добавляет значение value к полю количества монет игрока coins; если их 10 и более, то увеличивает максимальное здоровье на 5, а значение поля coins становится 0
- метод GetCoins() возвращает значение из поля coins

UML-диаграмма классов представлена на рис. 1.

Рисунок 1 – UML-диаграмма классов.



Выводы.

В ходе лабораторной работы было изучено применение интерфейсов, полиморфизм. Также были разработаны классы юнитов и предметов, класс игрока. Взаимодействие юнита и игрока. Взаимодействие игрока и объектов. Генерация объектов на игровом поле.