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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Создание классов, конструкторов и методов классов

Студент гр. 0383 _____ Девятериков И.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Реализовать несколько классов, которые позволят изучить особенности взаимодействия классов друг с другом в рамках языка программирования C++. А так же ознакомиться с особенностями построения UML диаграмм.

Задание.

Игровое поле представляет из себя прямоугольную плоскость, разбитую на клетки. На поле на клетках в дальнейшем будут располагаться игрок, враги, элементы взаимодействия. Клетка может быть проходимой или непроходимой, в случае непроходимой клетки, на ней ничего не может располагаться. На поле должны быть две особые клетки: вход и выход. В дальнейшем игрок будет появляться на клетке входа, а затем выполнив определенный набор задач дойти до выхода.

При реализации класса поля запрещено использовать контейнеры из stl

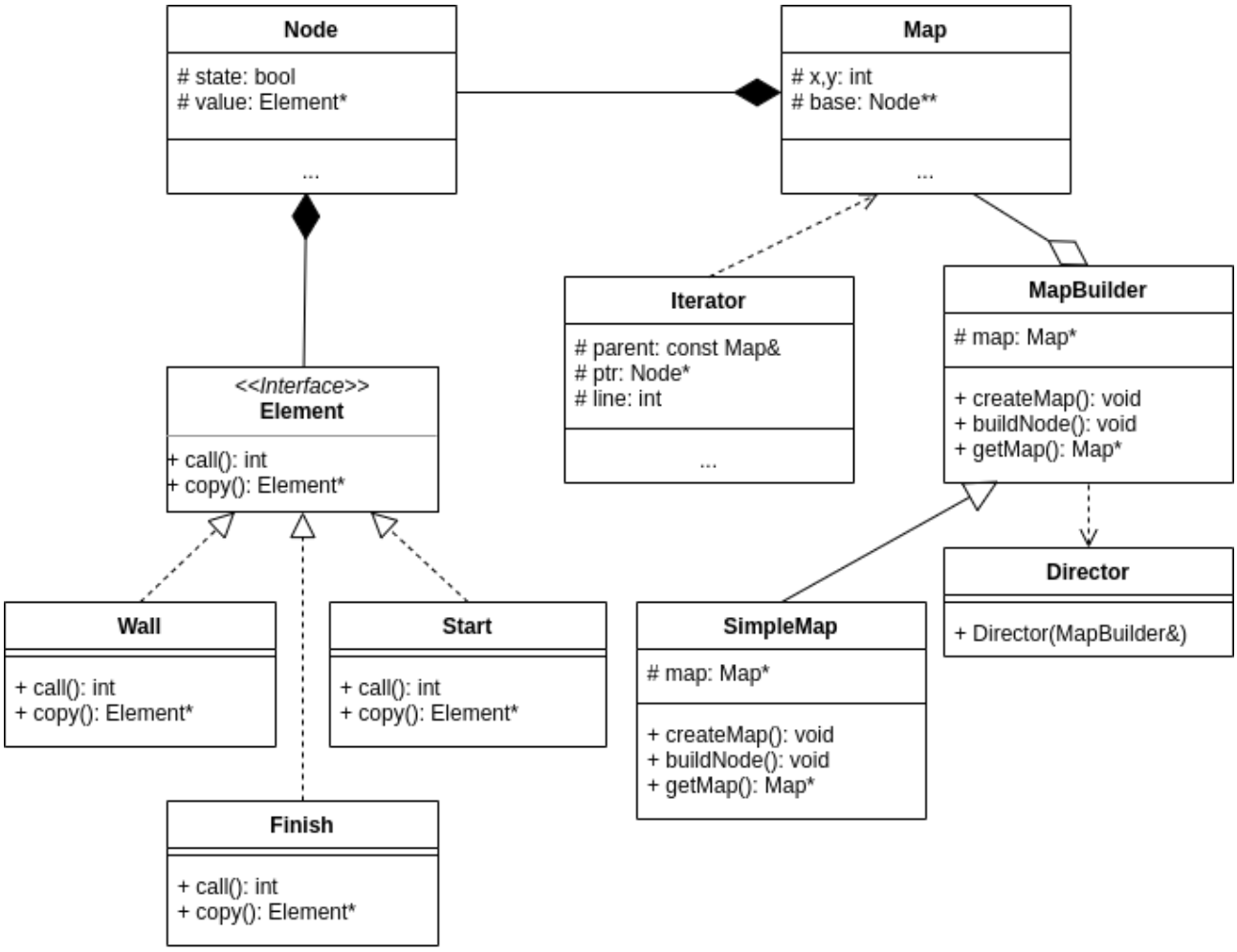
Требования:

- Реализовать класс поля, который хранит набор клеток в виде двумерного массива.
- Реализовать класс клетки, которая хранит информацию о ее состоянии, а также того, что на ней находится.
- Создать интерфейс элемента клетки.
- Обеспечить появление клеток входа и выхода на поле. Данные клетки не должны быть появляться рядом.
- Для класса поля реализовать конструкторы копирования и перемещения, а также соответствующие операторы.
- Гарантировать отсутствие утечки памяти.

Потенциальные паттерны проектирования, которые можно использовать:

- *Итератор (Iterator) - обход поля по клеткам и получение косвенного доступа к ним*
- *Строитель (Builder) - предварительное конструирование поля с необходимым параметрами. Например, предварительно задать кол-во непроходимых клеток и алгоритм их расположения*

Выполнение работы.



Class Map:

Класс имеет три protected поля – ширину, высоту и указатель на двумерный массив типа Node.

Конструктор по умолчанию создаёт поле 1x1, если не переданы другие его размеры. Выделяется память для массив Node* размера x+1 для корректной работы итератора, после чего в каждой ячейке получившегося массива, кроме последней, выделяется память на y элементов Node.

Конструктор, который принимает rvalue& Map, просто производит std::swap всеми полями класса.

Конструктор, который принимает lvalue& Map, поэлементно копирует каждую ячейку переданного класса, чтобы избежать совместного владения одними и теми же полями через разные классы.

Оператор=, который принимает rvalue& Map, просто производит std::swap всеми полями класса.

Оператор=, который принимает lvalue& Map, сначала, очищает текущие показатели Map, а затем, поэлементно копирует каждую ячейку переданного класса, чтобы избежать совместного владения одними и теми же полями через разные классы.

Метод setValue() позволяет переданный элемент по данным координатам.

Особенности реализации функций end(), begin() будут подробно рассмотрены при объяснении принципа работы итератора.

Class Iterator:

Конструктор принимает const& на объект, из которого был вызван, и индекс линии, к которой которой привязан текущий элемент. Знание о внешнем класса необходимо для того чтобы переходить от элементов одного столбца к элементам другого не выходя за рамки матрицы.

В самом классе реализованы операторы !=, ==, *, ++, – (последние имеют префиксную и постфиксную реализации).

Class Node:

Класс имеет два protected поля, state — указывает является ли поле пустым, value хранит указатель на интерфейс взаимодействия с различными видами содержимого на клетках.

Конструктор, который принимает rvalue& Node, просто производит std::swap всеми полями класса.

Конструктор, который принимает lvalue& Node, вызывает метод copy() соответствующего элемента.

Оператор=, который принимает rvalue& Node, просто производит std::swap всеми полями класса.

Оператор=, который принимает lvalue& Node, сначала (если необходимо), очищает текущее значение value, а затем, вызывает метод copy() соответствующего элемента.

Class Element:

Интерфейс имеющий три virtual метода (copy(), call(), ~Element()).

Class Wall, Start, Finish:

Классы реализации интерфейса Element. Последние два в дальнейшем, скорее всего, будут расширены в своей реализации.

Class MapBuilder:

Абстрактный класс для создания карты (класса Map).

Class Director:

Класс осуществляющий вызов методов класса MapBuilder в определённом порядке.

Class SimpleMap:

Одна конкретная простая реализация класса MapBuilder.

Выводы.

В ходе выполнения лабораторной работы произошло ознакомление с построением UML диаграмм. Был реализован собственный контейнер, имеющий возможность итеративного перебора его элементов. Реализован интерфейс, который позволит взаимодействовать с различными элементами.