

**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»**  
**Тема: Создание классов, конструкторов и методов классов**

Студент гр.0382

Литягин С.М.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

## **Цель работы.**

Научиться создавать классы, их конструкторы и методы.

## **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 необходимыми параметрами. Например, предварительно задать кол-во непроходимых клеток и алгоритм их расположения

## **Выполнение работы.**

Для реализации графического интерфейса используется SFML библиотека.

В ходе работы были созданы следующие классы:

1. Класс `Cell` создан для работы с клетками поля. Описание класса в файле `Cell.cpp`, определение в `Cell.h`. Его поля: `sf::Texture texture`, `sf::Sprite`

sprite, int type (текстура клетки, спрайт клетки и ее тип соответственно).

Для класса созданы следующие методы:

- метод `SetTexture(const std::string& filename)` написан для установки текстуры на клетку
- метод `SetPosition(int x, int y)` устанавливает клетку по соответствующим координатам
- метод `SetType(int value)` устанавливает тип клетки (0 – вход, 1 – пол, 2 – стена, 3 - выход)
- метод `GetSprite()` возвращает sprite
- метод `IsMovable()` возвращает true, если по клетке можно двигаться, иначе – false

2. Класс `Field` создан для работы с полем. Описание класса в файле `Field.cpp`, определение в `Field.h`. Его поле: `Cell** cells` (двумерный массив клеток). Согласно условию, реализованы конструкторы копирования и перемещения, а также соответствующие операторы. Для класса созданы следующие методы:

- метод `DrawCells(sf::RenderWindow* window)` рисует клетки поля в окне

3. Класс интерфейса строителя `MapBuilder`. Описание класса в файле `MapBuilder.cpp`, определение в `MapBuilder.h`. Содержит следующие виртуальные методы:

- метод `BuildEnter()` для создания входа на игровом поле
- метод `BuildExit()` для создания выхода на игровом поле
- метод `BuildWalls()` для создания стен на игровом поле
- метод `BuildFloor()` для создания пола на игровом поле

4. Класс строителя `MapBuilderOne`, наследуется от класса интерфейса `MapBuilder`. Описание класса в файле `MapBuilderOne.cpp`, определение в `MapBuilderOne.h`. Имеет поле `Field* field`. Содержит следующие методы:

- переопределенный метод BuildEnter() для создания входа на игровом поле и установке клетке тип входа
- переопределенный метод BuildExit() для создания выхода на игровом поле и установке клетке тип выхода
- переопределенный метод BuildWalls() для создания стен на игровом поле и установке клеткам тип стены
- переопределенный метод BuildFloor() для создания пола на игровом поле и установке клеткам тип пола
- метод Resert() для помещения нового объекта типа Field в поле field
- метод ReturnField() возвращает игровое поле field

5. Класс директора MapDirector. Описание класса в файле MapDirector.cpp, определение в MapDirector.h. Имеет поле MapBuilder\* builder. Содержит следующие методы:

- метод ChangeBuilder(MapBuilder\* build) для смены используемого строителя
- метод ConstructMap() для вызова методов строителя по заполнению игрового поля field клетками

6. Класс Game. Описание класса в файле Game.cpp, определение в Game.h. Содержит следующий метод:

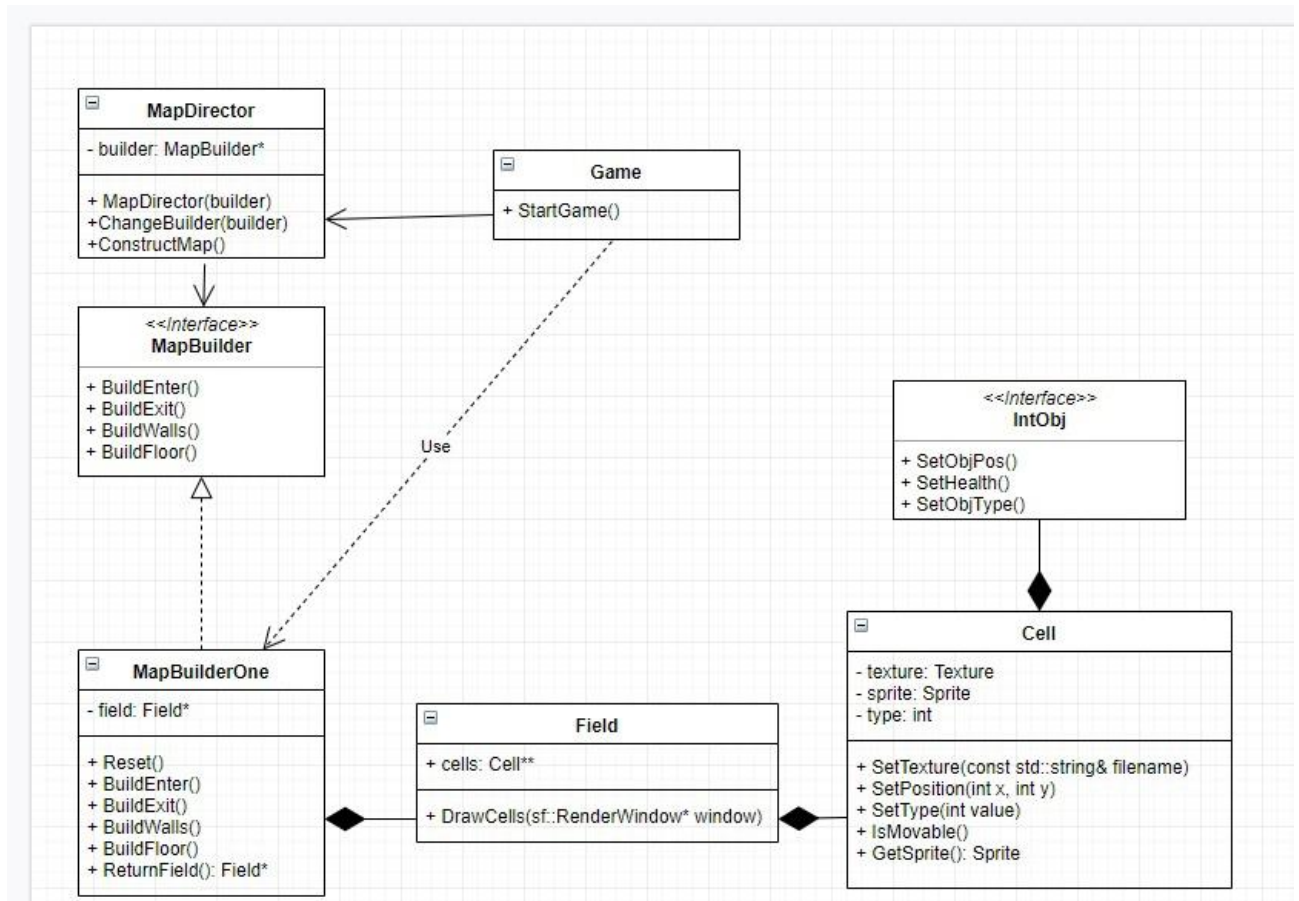
- метод StartGame() для создания игрового поля, а также создания окна

7. Класс интерфейса элемента клетки IntObj. Описание класса в файле IntObj.cpp, определение в IntObj.h. Содержит следующие виртуальные методы:

- метод SetObjPos() для установки элемента на определенную клетку
- метод SetObjType() для установки типа объекта
- метод SetHealth() для установки количества здоровья объекту

UML-диаграмма классов представлена на рис. 1.

Рисунок 1 – UML-диаграмма классов.



## Выводы.

В ходе лабораторной работы научились создавать классы, их конструкторы и методы. Также был реализован класс поля, который хранит набор клеток в виде двумерного массива. Создание такого поля происходит при помощи порождающего паттерна Строителя.