**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «****Объектно-ориентированное программирование»**

Тема: Шаблонные классы, генерация карты

|  |  |  |
| --- | --- | --- |
| Студент гр. 1381 |  | Смирнов Д. Ю. |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2022

**Цель работы.**

Необходимо реализовать шаблонный класс, генерирующий игровое поле. Этот класс должен задаваться правилами генерации. Также необходимо реализовать набор шаблонных правил.

**Задание.**

Реализовать шаблонный класс, генерирующий игровое поле. Данный класс должен параметризироваться правилами генерации (расстановка непроходимых клеток, как и в каком количестве размещаются события, расположение стартовой позиции игрока и выхода, условия победы, и. т. д.). Также реализовать набор шаблонных правил (например, событие встречи с врагом размещается случайно в заданном в шаблоне параметре, отвечающим за количество событий)

*Требования:*

* Реализован шаблонный класс генератор поля. Данный класс должен поддерживать любое количество правил, то есть должен быть *variadic template*.
* Класс генератор создает поле, а не принимает его.
* Класс генератор не должен принимать объекты классов правил в каком-либо методе, а должен сам создавать (в зависимости от реализации) объекты правил из шаблона.
* Реализовано не менее 6 шаблонных классов правил
* Классы правила должны быть независимыми и не иметь общего класса-интерфейса
* При запуске программы есть возможность выбрать уровень (не менее 2) из заранее заготовленных шаблонов
* Классы правила не должны быть только “хранилищем” для данных.
* Так как используются шаблонные классы, то в генераторе не должны быть *dynamic\_cast*

*Примечания:*

* Для задания способа генерации можно использовать стратегию, компоновщик, прототип
* Не рекомендуется делать *static* методы в классах правилах

Выполнение работы.

Для выполнения лабораторной работы создан набор шаблонных классов-правил и шаблонный класс, генерирующий игровое поле согласно правилам.

*Новые классы:*

ГЕНЕРАТОР ИГРОВОГО ПОЛЯ: Шаблонный класс FieldGen, шаблоном которого является какое-то неопределенное количество правил (используется *variadic template*). В нем есть только один метод *Field\* execute()*, который создает поле, применяет классы-правила к нему и используя распаковку шаблона, и возвращает указатель на него.

ПРАВИЛА ГЕНЕРАЦИИ: Созданы шаблонные классы-правила. В них переопределен оператор (), то есть данные классы являются функторами. Каждый класс правил при вызове оператора () принимает указатель на игровое поле *Field\** и при использовании оператора () изменяют поле.

КЛАССЫ-ПРАВИЛА:

* R\_Field\_Size — задает размеры поля. Размеры поля задаются через шаблон. Создает двумерный вектор объектов класса Cell с заданным размером, а не стандартным (10, 10). Так сделано, чтобы, если не было выбрано данное правило, создавалось поле со стандартным размером.
* R\_Rand\_Events — создает заданное количество событий, с конкретным генератором событий. Данный класс устанавливает в случайные клетки поля события, сгенерированные генератором событий.
* R\_Rand\_Walls — устанавливает заданное количество стен на игровое поле. Данный класс делает случайные клетки не проходимыми.
* R\_Player\_Spawn — задает позицию игрока. Координаты задаются через параметры шаблонна. Игрок ставится на заданную позицию, иначе на координаты (0, 0).
* R\_Column\_Walls — меняет клетки на не проходимые в столбце. Индекс и длинна столбца задается через параметры шаблонна.
* R\_Row\_Walls — меняет клетки на не проходимые в строке. Индекс и длинна строки задается через параметры шаблонна.
* R\_Win\_Cell — устанавливает в клетку событие победы. Координаты клетки задаются через параметры шаблонна.

Также были вынесены в функции некоторые части кода, которая часто использовалась в классах-правил.

При запуске программы у пользователя есть возможность выбрать один из уровней из заранее заготовленных шаблонов. Данные уровни хранятся в контейнере map (в поле класса *Game*) (ключ: номер уровня, значение: лямбда выражение генерирующее уровень). Номер уровня запрашивается у пользователя в классе *IOCommander*, после чего передается в класс *Game* через класс посредника *Mediator*. Если такой уровень есть в *map*, то он сгенерируется, иначе запустится первый уровень.

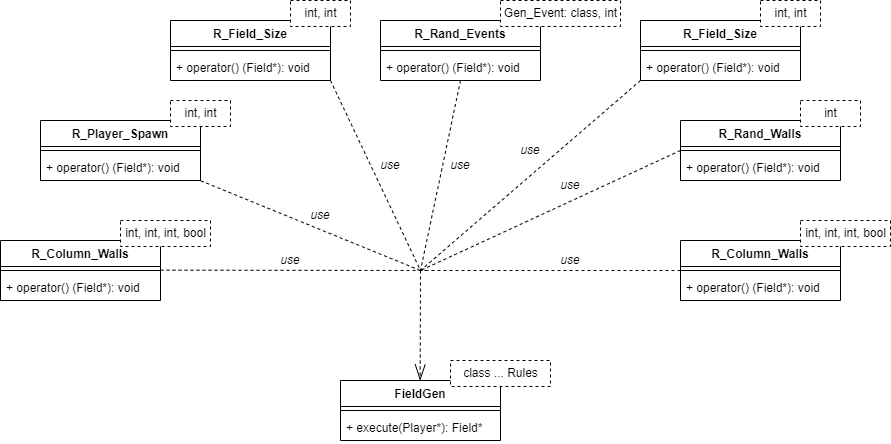
UML диаграмма классов представлена в приложении A.

**Выводы.**

Реализован шаблонный класс, генерирующий игровое поле. Данный класс параметризируется правилами генерации. Также реализован набор шаблонных правил.

**ПРИЛОЖЕНИЕ А**

**UML диаграмма классов**

****