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

Студент гр. 1303 _____ Коренев Д.А.

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 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 созданы шаблонный класс, генерирующий игровое поле, и набор шаблонных классов-правил, которыми задается генерация поля.

Шаблонный класс-генератор FieldGenerator, шаблоном которого является любое количество правил, имеет единственный метод generate, возвращающий указатель на созданное поле. В этом методе сначала создается пустое поле, далее каждый класс-правило, используя распаковку шаблона, изменяет его. После применения всех правил генерации метод возвращает указатель на получившееся поле.

Классы правила:

1) GenerationFieldSize задает размеры поля, которые задаются через шаблоны. Переопределенный опреатор () принимает ссылку на поле. Далее идет проверка, было ли поле сгенерировано ранее, и если было, завершает свою работу, иначе вызывает метод setSize у поля и присваивает туда корректные значения (размеры поля не могут быть меньше 10). Далее используются два цикла for для наполнения поля клетками. Для этого вектор, в котором хранятся клетки, расширяется, а у игрового поля вызывается метод setCell, который создает клетку нужного типа (в данном случае CellGrass) на

необходимые координаты. Если это невозможно, прокидывает ошибку некорректного аргумента, она обрабатывается и логируется.

2) GenerationPlayerSpawn задает начальную позицию игрока, которая задается через шаблоны. Переопределенный опреатор () принимает ссылку на поле. Идет проверка на то, было ли для игрока уже установлено начальное положение. И если нет, вызывает у поля методы setCell с аргументами позиции и названием типа клетки (Player) и setPlayerPosition с аргументом позиции. Все возможные ошибки обрабатываются в этих методах и при некорректных данных прокидывают ошибку некорректного аргумента, она обрабатывается и логируется.

3) GenerationFinishSpawn задает позицию выигрыша, которая задается через шаблоны. Переопределенный оператор () принимает ссылку на поле и вызывает у него метод setFinishPosition с позицией полученной из шаблонов. Если это сделать нельзя, то ошибка обрабатывается и логируется.

4) GenerationCell создает клетки нужного типа в определенных координатах, данные задаются через шаблоны. Переопределенный оператор () принимает ссылку на поле, с помощью двух циклов for по заданным координатам у поля вызывается метод changeCell с аргументами координаты и экземпляром клетки, которую необходимо установить. Если метод changeCell не может установить клетку, он прокидывает ошибку некорректного аргумента, которая обрабатывается в описываемом методе и логируется.

5) GenerationEventLine устанавливает события разных типов по заданным координатам, данные задаются из шаблонов. Переопределенный оператор () принимает ссылку на поле, с помощью двух циклов for по заданным координатам у поля вызывается метод setEvent с аргументами координаты и экземпляром события, которую необходимо установить. Если

метод setEvent не может установить клетку, он прокидывает ошибку некорректного аргумента, которая обрабатывается в описываемом методе и логируется.

6) GenerationEventExtendedLine аналогичен GenerationEventLine, однако принимает в шаблоне на два значения больше. Их он передает в конструктор создания Событий.

Для того, чтобы задать определенную генерацию уровня, создан интерфейс LevelStrategy с чистым виртуальным методом Field* generate(), который в классах-наследниках генерирует поле по определенным правилам и возвращает указатель на него. От этого класса наследуются классы FirstLevel и SecondLevel, в которых определяется метод Field* generate(), где генерируется поле с разными правилами.

Также был создан LevelContext, в полях которого находится указатель на поле и levelStrategy, который содержит наследника LevelStrategy. В конструкторе принимает уровень через конструктор, а также предоставляет сеттер для его изменения во время выполнения. Был создан геттер для поля. Также был реализован метод setLevel(), который у levelStrategy вызывает вызывает метод generate(), генерирущий поле по определенным правилам, и записывает это поле в поля класса.

В классах отвечающих за считывание данных были созданы методы отвечающие за считывание у пользователя какой уровень необходимо генерировать. В классе Field был удален метод createField так как за создание поле теперь отвечает другой класс.

Тестирование.

Тестирование программы: пример генерации второго уровня представлен на рисунке 1.

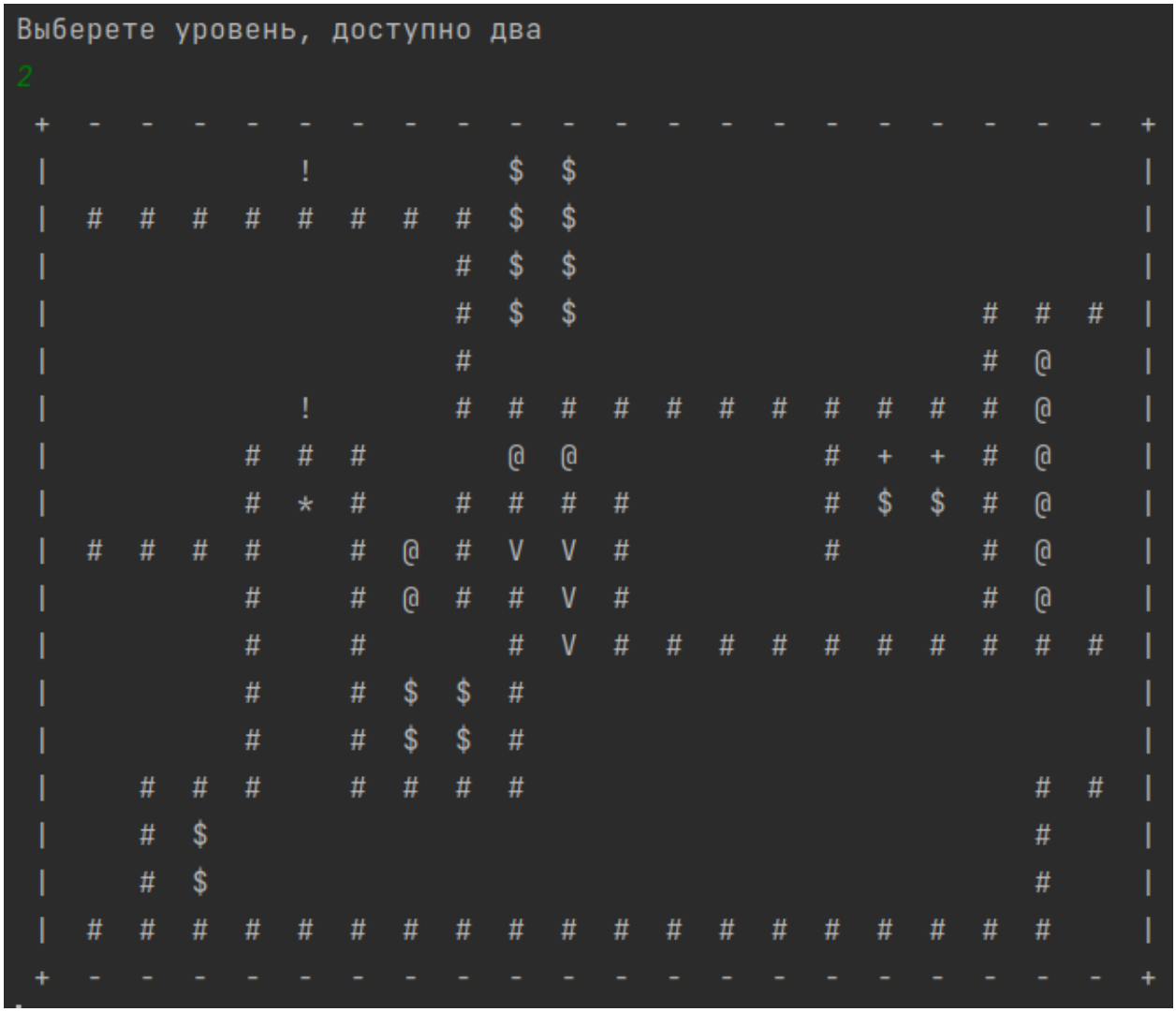

Рис. 1 Второй уровень

UML диаграмма межклассовых отношений.

UML диаграмма межклассовых отношений представлена на рисунке 2.

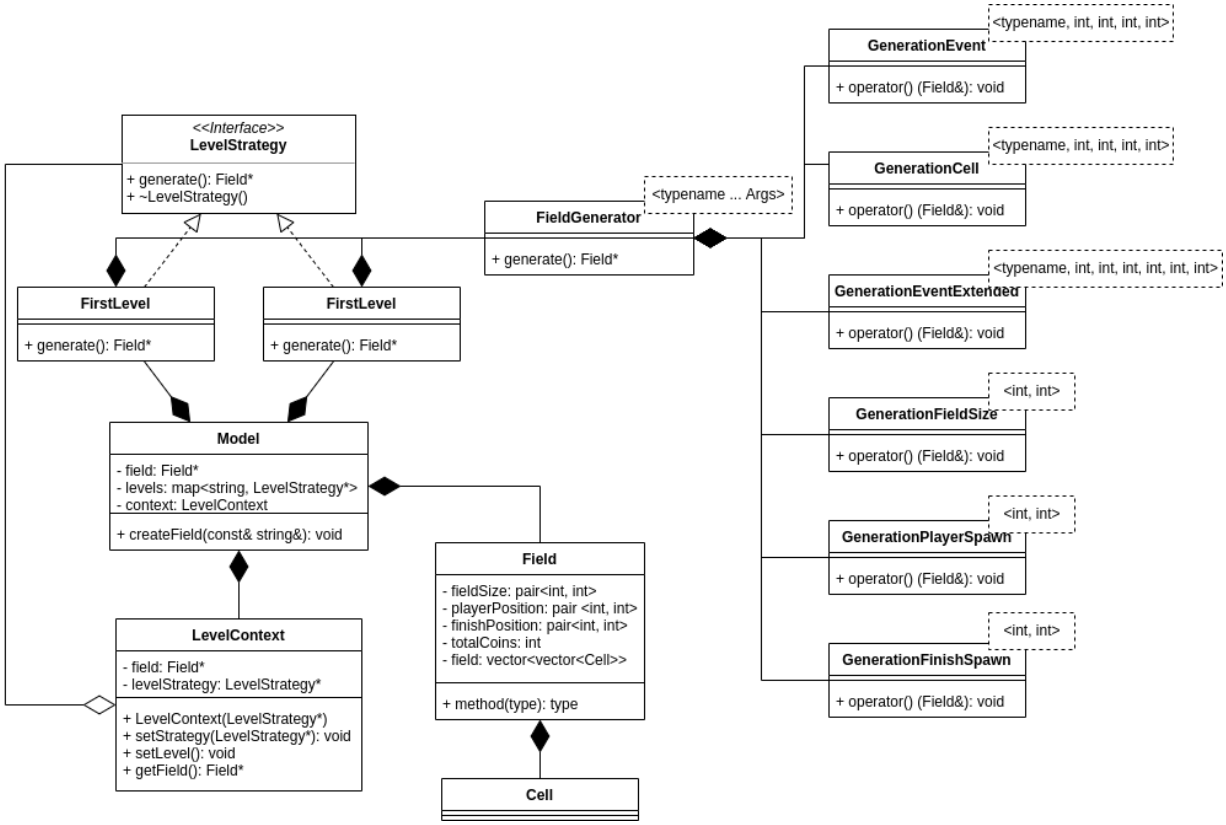

Рис. 2 Межклассовые отношения

Выводы.

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