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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Объектно-ориентированное программирование»

Тема: Шаблонные классы, управление

Студент гр. 0383 _____ Коротков А.В.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2021

Цель работы.

Изучить материал по теме «Шаблонные классы» и, на основе полученных знаний, реализовать шаблонные классы правил игры, в которых параметр шаблона должен определить конкретные значения в правилах, и шаблонный класс игры, который параметризуется конкретными правилами, обеспечив выполнение необходимых требований.

Задание.

Необходимо определить набор правил для игры в виде классов (например, какие задачи необходимо выполнить, чтобы он мог выйти с поля; какое кол-во врагов и вещей должно быть на поле, и т.д.). Затем определить класс игры, которое параметризуется правилами. Класс игры должен быть прослойком между бизнес-логикой и командами управления, то есть непосредственное изменение состояния игрой должно проходить через этот класс.

Требуется:

- Созданы шаблонные классы правил игры. В данном случае параметр шаблона должен определить конкретные значения в правилах. Должна быть возможность записывания логов в файл, в консоль или одновременно в файл и консоль.
- Создан шаблонный класс игры, который параметризуется конкретными правилами. Класс игры должен проводить управление врагами, передачей хода, передавать информацию куда переместить игрока, и т.д.

Основные теоретические положения:

Потенциальные паттерны проектирования, которые можно использовать:

- Компоновщик (*Composite*) - выстраивание иерархии правил
- Фасад (*Facade*) - предоставления единого интерфейса игры для команд управления
- Цепочка обязанностей (*Chain of Responsibility*) - обработка поступающих команд управления
- Состояние (*State*) - отслеживание состояние хода / передача хода от игрока к врагам
- Посредник (*Mediator*) - организация взаимодействия элементов бизнес-логики

Выполнение работы.

В ходе лабораторной работы были реализованы следующие классы:

class Rules – является дружественным с классом Game и предназначен для хранения параметров игры, таких как характеристики существ, количество врагов и предметов, скорость передвижения и так далее. Этот класс является шаблонным. В качестве параметра шаблона принимаются несколько целочисленных значений, описывающих различные параметры игры. Объекты этих классов хранят эти значения в своих *private* полях и благодаря дружественности класса *Game*, доступ к этим параметрам доступен в классе игры. Для удобства хранения параметров игры были реализованы вспомогательные структуры *HeroStats, EnemyStats, EnemiesCount, ItemsValues, ItemsCount, WinConditions.*

class *Game* – шаблонный класс игры, в котором реализованы необходимые методы с классом *Rules* в качестве параметра шаблона. Методы класса *Game* при инициализации и в игровом процессе используют в своих методах

параметры из объекта класса *Rules*, хранящегося в приватном поле класса *Game.*

Логика игры была изменена. Одними из параметров класса *Rules* являются параметры *necessaryExit* и *allEnemiesMustBeBeaten.* Первый из которых отвечает за то, необходим ли выход для победы, либо же достаточно победить врагов. Второй же параметр определяет, необходимо ли персонажу одержать победу над всеми врагами, либо же только над половиной из них.

Для инициализации значений характеристик персонажа, врагов, а также количества врагов, количества предметов и значений поднятия показателей персонажа класс *Game* передает необходимые структуры в *FieldBuilder,* который располагает на поле врагов и предметы в соответствии с конкретными правилами.

class HeroMover – класс, обеспечивающий передвижение персонажа на протяжении выполнения программы.

class EnemyMover – класс, обеспечивающий передвижение врагов на протяжении выполнения программы. Данный класс хранит в себе указатель на класс Field и в режиме реального времени способен отслеживать происходящие изменения.

Также был реализован простейший интерфейс для отображения характеристик персонажа.

UML-диаграмму классов смотреть в приложении А.

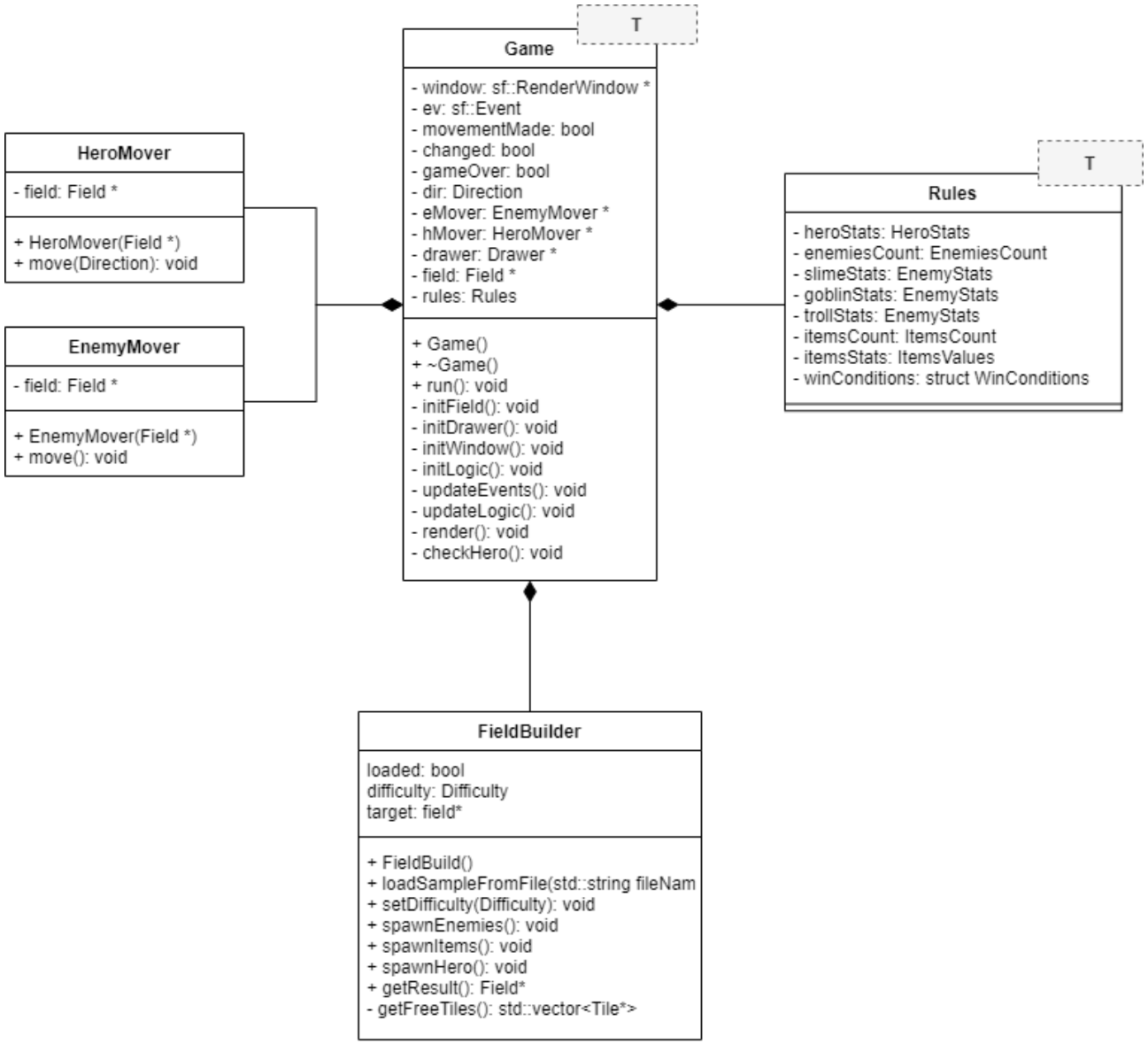
Тестирование программы и процесса логирования смотреть в приложении В.

Выводы.

В ходе работы был изучен материал по теме «Шаблонные классы». Был реализован набор классов и методов, позволяющий задавать все параметры игры из одного класса, не меняя при этом остальной код, создана UML-диаграмма классов. Результатом выполнения лабораторной работы стал класс, из которого можно непосредственно управлять параметрами игры.

Приложение А

UML-диаграмма



Приложение В

Тестирование и запуск программы

Тестирование.

Пример запуска программы: