

**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.0382

Литягин С.М.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

### **Цель работы.**

Изучить применение шаблонов. Написать класс игры, параметризуемый правилами; данный класс изменяет состояние игры.

### **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 прослойкой между бизнес-логикой и командами управления, то есть непосредственное изменение состояния игрой должно проходить через этот класс.

### **Требование:**

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

### *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.**

Чтобы наша игра параметризовалась правилами, были написаны шаблонные классы для них. Было решено написать всего 3 правила: правило, задающее строителя игрового поля; правило, задающее количество противников на игровом поле; правило, задающее количество вещей на игровом поле.

Первому из них соответствует шаблонный класс `template<class T> class RuleBuilder`. Для дальнейшей работы в классе игры мы будем сохранять в поле `int builder` значение, полученное из конструктора. По этому значению и будет выбираться строитель. Собственно, для получения значения данного поля также был написан метод `GetBuilderID()`.

Второму из них соответствует шаблонный класс `template<class T> class RuleEnemy`. Для дальнейшей работы в классе игры мы будем сохранять в поле `int value` значение, полученное из конструктора. По этому значению и будет определяться количество противников на поле. Для получения значения данного поля был написан метод `GetValue()`. На случай, если установленное пользователем значения нас как-то не устроят в дальнейшем, написан метод `SetValue(int value)`, который, собственно, установит в поле нужное нам значение.

Третьему из них соответствует шаблонный класс `template<class T> class RuleThing`. Для дальнейшей работы в классе игры мы будем сохранять в поле `int value` значение, полученное из конструктора. По этому значению и будет определяться количество вещей на поле. Для получения значения данного поля был написан метод `GetValue()`. На случай, если установленное пользователем значения нас как-то не устроят в дальнейшем, написан метод `SetValue(int value)`, который, собственно, установит в поле нужное нам значение.

Собственно, данные правила должны параметризовать шаблонный класс игры, в котором еще и происходит управление изменением состояния игры. Для этого мы переделаем класс `Game`. Теперь он является шаблонным классом `template<typename R1, typename R3, typename R3> class Game`. Предполагается,

