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

Студентка гр. 3385 Завьялова В.Д.

Преподаватель Жангиров Т. Р.

Санкт-Петербург

2024

Цель работы

Целью данной лабораторной работы является создание:

1. Класса управления игрой, который использует шаблонный параметр для определения класса, обрабатывающего ввод команд.

2. Класса отображения игры, который будет реагировать на изменения в игре и выполнять отрисовку, используя класс, заданный в качестве параметра шаблона.

3. Класса для чтения ввода пользователя, который будет считывать команды из терминала и преобразовывать их в команды для игры. Он будет загружать соответствие команд и символов из файла.

4. Класс для отрисовки поля, который будет отвечать за визуализацию игры и может быть заменён при необходимости (например, для реализации графического интерфейса).

5. Прослойка обработки команд, которая будет обеспечивать валидацию команд и их соответствие с клавишами, что позволит избежать конфликтов ввода.

Задание

a. Создать шаблонный класс управления игрой. Данный класс должен содержать ссылку на игру. В качестве параметра шаблона должен указываться класс, который определяет способ ввода команда, и переводящий введенную информацию в команду. Класс управления игрой, должен получать команду для выполнения, и вызывать соответствующий метод класса игры.

b. Создать шаблонный класс отображения игры. Данный класс реагирует на изменения в игре, и производит отрисовку игры. То, как происходит отрисовка игры определяется классом переданном в качестве параметра шаблона.

c. Реализовать класс считывающий ввод пользователя из терминала и преобразующий ввод в команду. Соответствие команды введенному символу должно задаваться из файла. Если невозможно считать из файла, то управление задается по умолчанию.

d. Реализовать класс, отвечающий за отрисовку поля.

Примечание:

- Класс отслеживания и класс отрисовки рекомендуется делать отдельными сущностями. Таким образом, класс отслеживания инициализирует отрисовку, и при необходимости можно заменить отрисовку (например, на GUI) без изменения самого отслеживания
- После считывания клавиши, считанный символ должен сразу обрабатываться, и далее работа должна проводить с сущностью, которая представляет команду.
- Для представления команды можно разработать системы классов или использовать перечисление enum.
- Хорошей практикой является создание “прослойки” между считыванием/обработкой команды и классом игры, которая сопоставляет команду и вызываемым методом игры. Существуют альтернативные решения без явной “прослойки”

- При считывания управления необходимо делать проверку, что на все команды назначена клавиша, что на одну клавишу не назначено две команды, что на одну команду не назначено две клавиши.

Выполнение работы

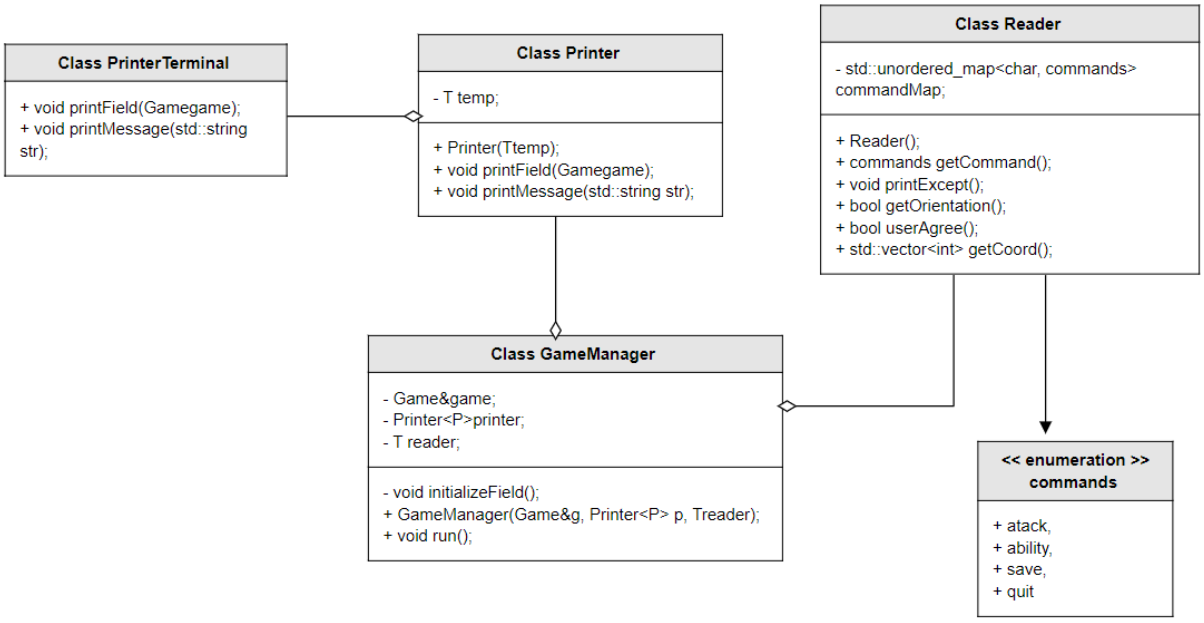

Рисунок 1 – UML-диаграмма классов

Class Reader

1. Содержит таблицу соответствия между символами ввода (*char*) и командами (*commands enum*).
2. Метод *getCommand()* считывает ввод пользователя и возвращает соответствующую команду.
3. Метод *printExcept()* предназначен для вывода сообщений об ошибках ввода.
4. *getOrientation()* получает от пользователя информацию об ориентации (логическое значение).
5. *userAgree()* получает подтверждение от пользователя (логическое значение).
6. *getCoord()* получает от пользователя координаты (вектор целых чисел).

Основные обязанности класса *Reader*:

- Считывание и интерпретация пользовательского ввода.
- Преобразование ввода в логические команды.
- Получение необходимых данных от пользователя для игры.
- Обработка ошибок ввода.

Class Printer

Класс Printer является шаблонным классом, который предназначен для печати информации, связанной с игрой. Он может использоваться для печати поля игры и сообщений. Вот краткое описание того, что делает этот класс:

Члены класса:

1. *temp*: Хранит временный объект типа T, который может быть использован для различных целей, таких как форматирование или хранение состояния.

Методы класса:

2. *Printer(T temp)*: Конструктор, который инициализирует объект Printer с временным объектом temp.
3. *void printField(Game game):* Метод для печати поля игры.
4. *void printMessage(std::string str)*: Метод для печати сообщения.

Class PrinterTerminal

Методы класса:

1. *void printField(Game game)*: Метод для печати поля игры в терминал. Он вызывает методы printField объектов *gameField и enemyField* класса *Game*, чтобы отобразить поле игры и поле противника.
2. *void printMessage(std::string str)*: Метод для печати сообщения в терминал. Он выводит переданное сообщение на экран.

Class GameManager

1. Конструктор:
    - Инициализирует объект *GameManager* с ссылкой на игру (*game*), объектом для печати (*printer*) и объектом для чтения ввода (*reader*).
2. Метод *initializeField*:
    - Инициализирует игровое поле, запрашивая у пользователя координаты и ориентацию кораблей.
    - Создает корабли и размещает их на игровом поле.
    - Обрабатывает исключения и ошибки ввода.

3. Метод *run:*
    - Основной метод, который управляет игровым процессом.
    - Проверяет наличие сохранения и загружает его, если пользователь согласен.
    - Инициализирует игровое поле и поле противника.
    - Обрабатывает команды пользователя и выполняет соответствующие действия (атака, использование способностей, сохранение, выход).
    - Обрабатывает атаки противника и проверяет состояние игры (победа или поражение).

Вывод

В данной лабораторной работе были освоены принципы объектно-ориентированного программирования, такие как абстракция, инкапсуляция и полиморфизм, применительно к созданию гибкой и расширяемой архитектуры игрового движка. Было реализовано разделение логики управления игрой, ввода команд и отрисовки с помощью шаблонных классов, что обеспечило модульность и возможность замены компонентов. Также была отработана работа с файлами конфигурации и обработка исключительных ситуаций.