

**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**

Студент гр. 3342

Корниенко А.Е.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2024

## **Цель работы**

Создать классы для управления игрой и считывания ввода пользователем, для отрисовки поля игры.

## 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

Класс `Commands` хранит команды, доступные пользователю.

Методы:

`std::map<std::string, std::string> get_command_map()` – геттер для поля

`std::map<std::string, std::string> command_map`

`std::vector<std::string> get_long_commands()` – геттер для поля

`std::vector<std::string> long_commands`

Поля:

`std::map<std::string, std::string> command_map` – сопоставляет короткую версию команды с их полной версией

`std::vector<std::string> long_commands` – хранит команды.

Класс `ConsoleInputer` обрабатывает ввод пользователя.

Методы:

`std::string getInput()` – обрабатывает ввод пользователя

`std::string getCommand(const std::string& input)` – проверяет на корректность команде, введенную строку

`std::string getString()` – методы для ввода строки

Поля:

`Input input` – класс, определяющий метод ввода.

`Commands& commands` – команды.

Класс `ConsolePrinterMessage` выводит сообщения в консоль

Методы:

`void print(std::string string)` – определяет вывод.

Класс `ConsoleShow_Table` определяет вывод поля в консоль

Методы:

`void showTable(Table& table, bool flag)` – выводит поле

`void showShips(ManagerShips& ship_manager)` – выводит состояния корабли

Класс `Controller` преобразует ввод пользователя в команды и выполняет при помощи их. Наследуется от `IController`.

Методы:

`virtual int setMode()` – используется для того, чтобы задать начало игры ( новая или загрузка игры), и конца игры ( новая игра или закончить )

`std::string setFilename()` – задаёт имя файла для сохранения

`std::vector<int> setPlaceship(int len_ship)` – задаёт расположение корабля

`Coord setCoord()` – задаёт координату для атаки или для способности ( при необходимости )

`void Endgame()` – конец игры

`void run()` – начало игры

`void print_message(std::string string)` – вывод сообщения через класс, который выводит информацию в консоль

Поля:

`Commands list_commands` - команды

`Game &game` - игра

`Inputer inputer` – класс, который обрабатывает ввод пользователя

`View<Printer_Table, Printer_Message> view` – классы вывода информации об игре

`std::map<std::string, std::function<void()>> commands` – сопоставляет команды с функциями, которые должны выполняться при их вводе

`bool isGameRunning` – хранит `true`, когда игра находится в цикле, `false` при конце игры

Класс `Input` определяет ввод пользователя

Методы:

`std::string input()` – определяет ввод пользователя

Класс `Printer_Message` определяет, отвечает за вывод сообщений

Методы:

`void attackMessage(Game &game)` – сообщение об итоге атаки

`void useSkill(Game &game)` – сообщение о применении способности

`void nextSkill(Game &game)` – сообщение об информации о следующей доступной способности

`void help()` – сообщение об информации о доступных командах

`void print_message(std::string message)` – вывод сообщения

Поля:

ConsolePrinterMessage printer – выводит сообщения

Commands& commands – команды

Класс Printer\_Table определяет вывод, выводит информацию о полях

Методы:

void drawGameTable(Game &game) – рисует поле игрока и бота

void showShips(Game &game, bool flag) – выводит состояния кораблей

void showTableShips(Game &game, bool flag) – выводит поле и состояния кораблей

void showTable(Game &game, bool flag) – выводит поле

Поля:

ConsoleShow\_Table show – выводит поля

Класс View принимает команды от контроллера и выводит информацию об игре.

Методы:

void solve(std::string command) – принимает команду и выводит, что необходимо

void print\_message(std::string string) – выводит сообщение

Поля:

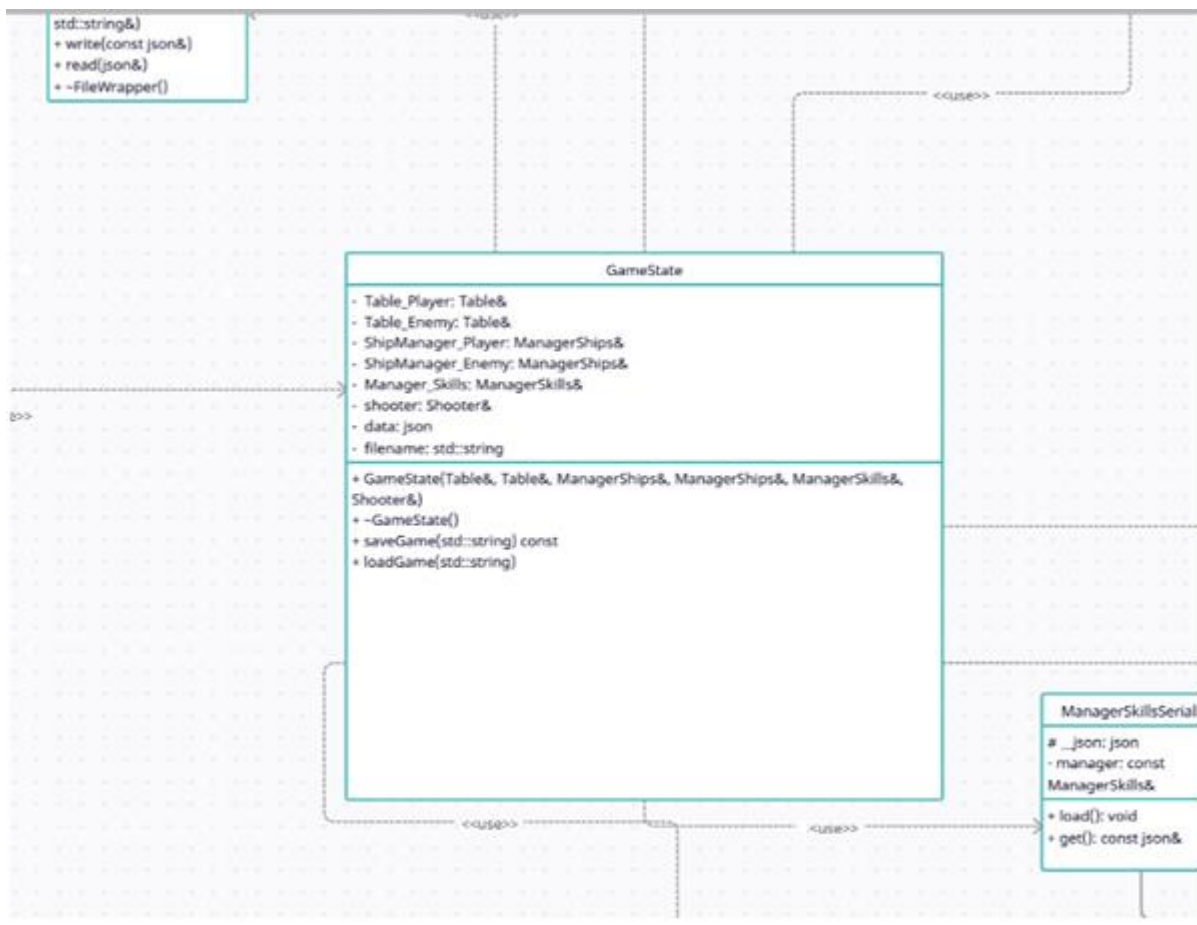
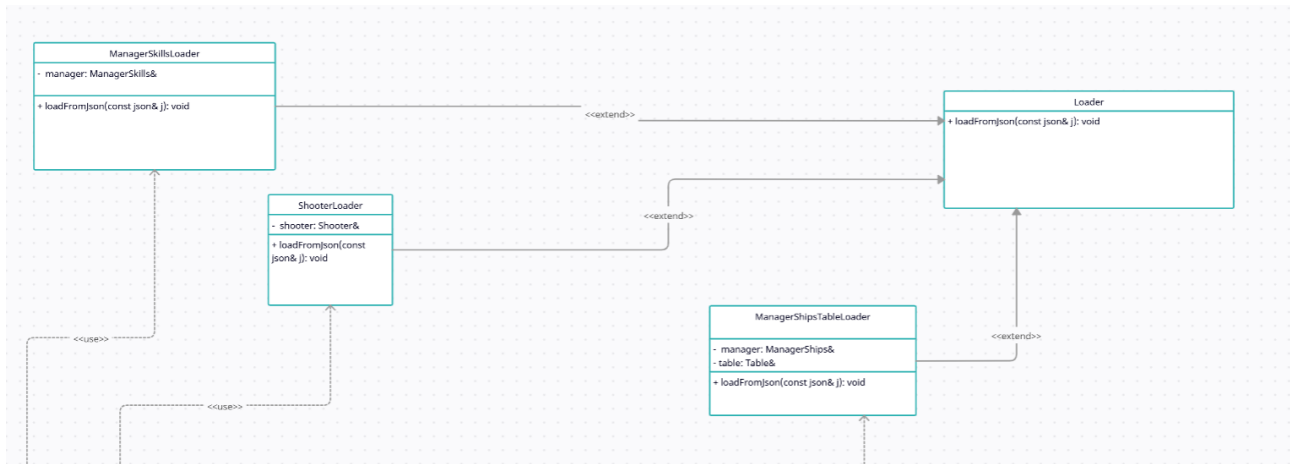
Commands& list\_commands - команды

Printer\_Table printer\_table – выводит информацию о поле

Printer\_Message printer\_message – выводит сообщения

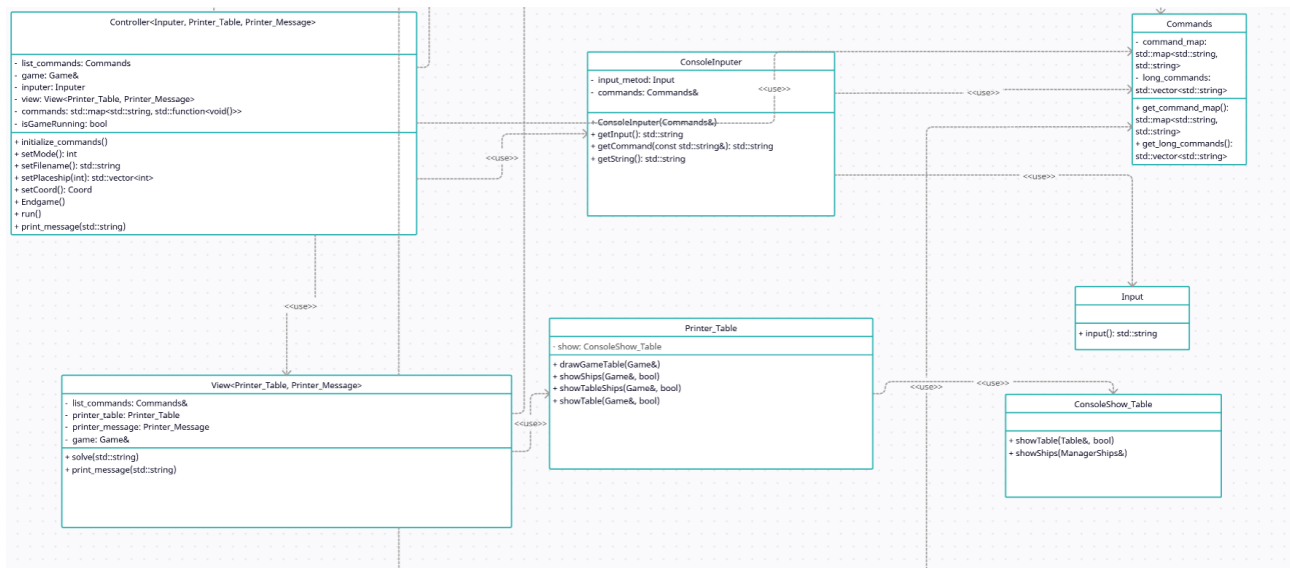
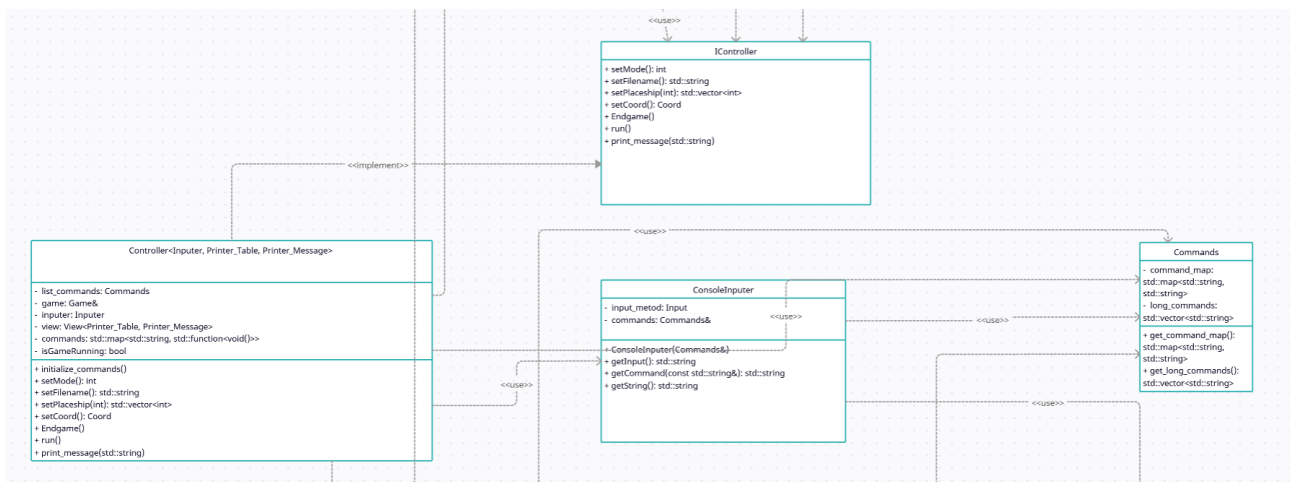
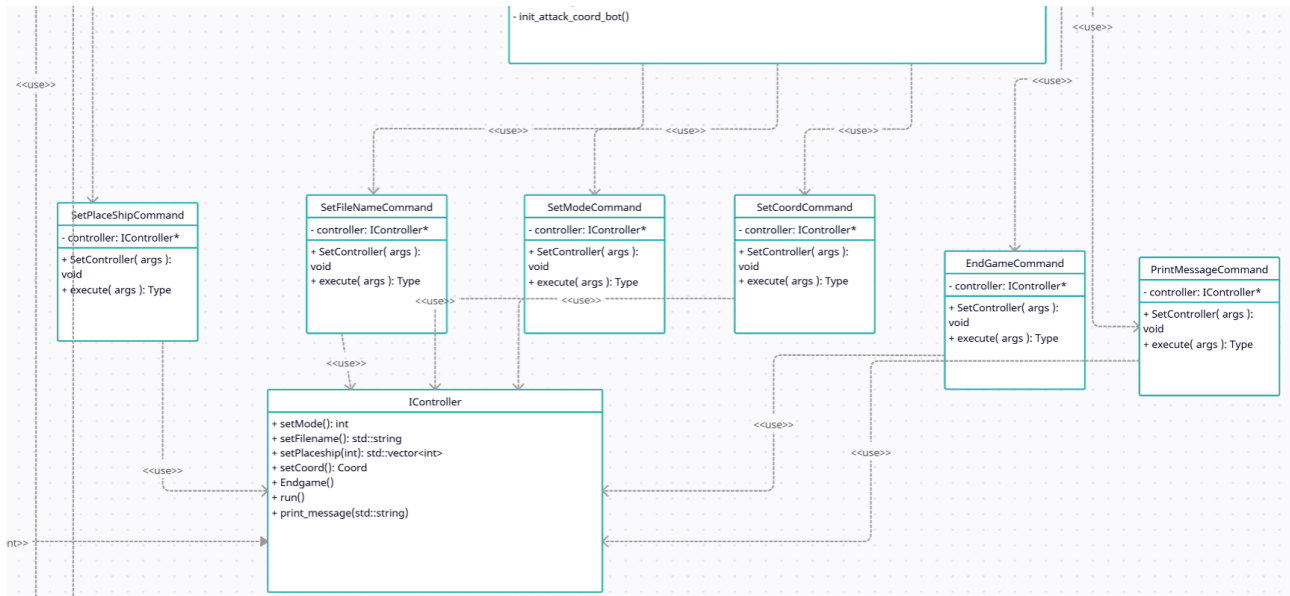
Game& game - игра

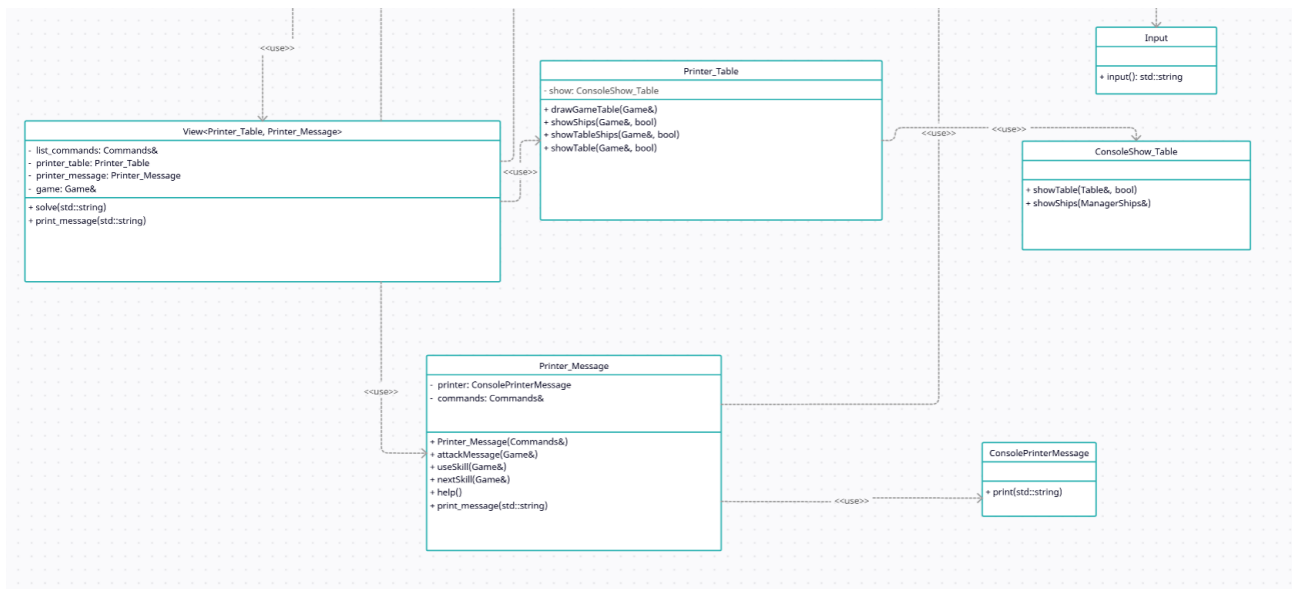
## UML-диаграммы классов:











## **Выводы**

В ходе выполнения работы были разработаны классы для управления игрой и считывания ввода пользователем, для отрисовки поля игры.