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. 3384 _____ Пьянков М.Ф.
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2024

Цель работы

Изучить работу шаблонных классов. Реализовать новые модули для выполнения поставленной задачи.

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.

e) Примечание:

f) 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

g) После считывания клавиши, считанный символ должен сразу обрабатываться, и далее работа должна проводить с сущностью, которая представляет команду.

h) Для представления команды можно разработать системы классов или использовать перечисление enum.

i) 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”

j) 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

Архитектура новых модулей представлена на рис. 1.

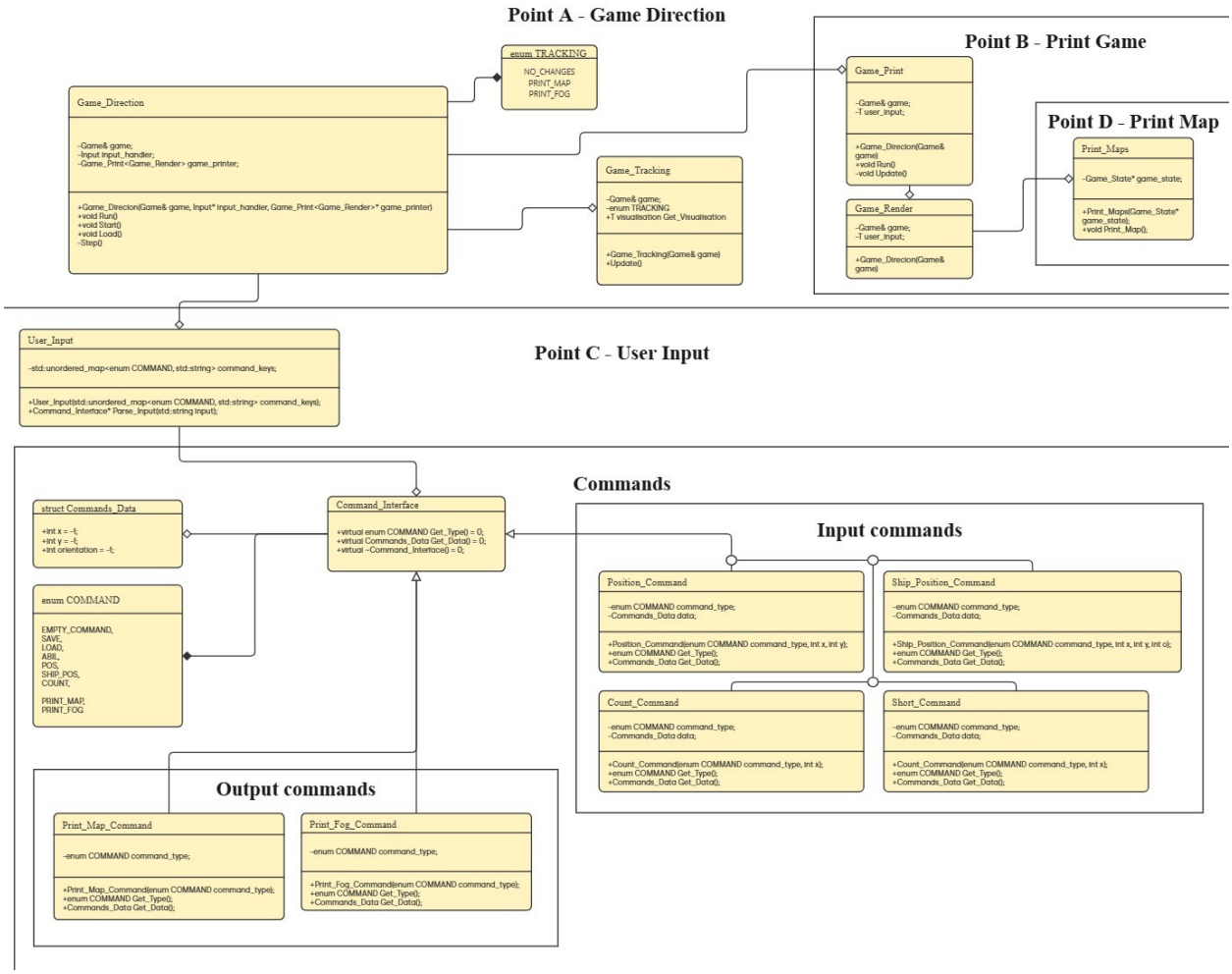

Рисунок 1 — Архитектура новых модулей программы (UML-диаграмма классов)

класс Game_Direction – основной класс.

Методы: void Run() - метод, обеспечивающий запуск игры, void Start() - метод, отвечающий за старт игры и расстановку кораблей, void Load() - метод, обеспечивающий загрузку игры void Step() - метод, созданный для упрощения работы программы, выполняет повторяющуюся часть цикла игры. Поля: Game* game – указатель на игру, Game_Print<Game_Render> game_printer — объект отрисовки игры, Input input_handler — шаблонный объект, отвечающий за считывание и обработку команд.

Класс Command_Interface – класс-интерфейс для команд.

Классы Position_Command, Ship_Position_Command, Count_Command, Short_Command – классы команд, наследующиеся от класс-интерфейса команд. Поля: enum COMMAND command_type. Commands_Data data. Методы: enum COMMAND Get_Type() - возвращает тип команды, Commands_Data Get_Data() - возвращает данные, полученные из команды.

Класс User_Input – класс, создающий команду на основе считанной и обработанной строки. Поля: std::unordered_map<enum COMMAND, std::string> command_keys — поле задающее соответствие командам и символам или строкам. Методы: Command_Interface* Parse_Input(std::string input) — считывание, обработка и возврат объекта класса команда.

Класс Print_Maps – класс отрисовки поля. Поля: Game_State* game_state — состояние игры. Методы: void Print_Map() - вывод поля игрока, void Print_Fog_of_War() - вывод поля компьютера.

Класс Game_Render – класс отрисовки игры. В общем случае он может отрисовывать что угодно, но в данном случае используется отрисовка поля

классом Print_Maps. Поля: Game* game — указатель на игру, Print_Maps* printer — указатель на отрисовыватель карты. Методы: void Render(enum TRACKING action) — отрисовка поля в соответствии с командой, переданной в метод.

Класс Game_Print – шаблонный класс вывода игры. Класс принимает в качестве параметра какой-нибудь отрисовыватель. Поля: Game* game — указатель на игру, Render* render — указатель на отрисовыватель игры. Методы: void Use(enum TRACKING action) — метод, применяющий отрисовыватель игры.

Класс Game_Tracking – класс, хранящий и сравнивающий состояние игры, изменяет параметры отрисовки enum TRACKING.

Выводы

Успешно изучены шаблонные классы, необходимые модули реализованы.