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. 3388 _____ Глебова В.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы

Разработать классы для управления и отображения состояния игры с использованием шаблонов. Создать шаблонный класс управления, обрабатывающий пользовательский ввод и вызывающий методы игры. Тип ввода и преобразования в команды задаётся параметром шаблона. Также разработать шаблонный класс отображения, отслеживающий изменения в игре и визуализирующий её состояние, с возможностью замены способа отрисовки.

Реализовать класс для обработки ввода из терминала, который сопоставляет команды клавишам, конфигурируемым через файл или по умолчанию. Обеспечить проверку корректности конфигурации, исключая дублирование клавиш или команд.

Создать класс отрисовки игрового поля, поддерживающий замену текстового интерфейса на графический без изменений в логике. Система должна быть модульной и обеспечивать разделение логики управления, обработки ввода и визуализации.

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.

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 GameControl

Класс GameControl отвечает за управление текстовыми командами игры и их сопоставление с клавишами управления. Он предоставляет возможность загружать команды из конфигурационного файла, задавать команды по умолчанию и преобразовывать команды в их текстовое или символьное представление.

Поля класса GameControl:

- map<string, char> commands — отображение текстовых команд на соответствующие клавиши управления.
- map<char, string> reverse_commands — обратное отображение, связывающее клавиши управления с текстовыми командами.

Методы класса GameControl:

- GameControl() — конструктор, который вызывает метод setDefault для установки стандартных команд управления.
- void setDefault() — задаёт стандартные команды управления: help, load, save, и attack с соответствующими клавишами.
- char getCommandKey(const string& command) — возвращает клавишу, связанную с переданной текстовой командой. Выбрасывает исключение, если команда не найдена.
- vector<string> split(const string& s, char delim) — разделяет строку s на элементы, используя символ delim в качестве разделителя.
- void load(const string& filename) — загружает команды из файла с указанным именем. Проверяет корректность данных: уникальность команд и клавиш, наличие всех обязательных команд (help, load, save, attack). Выбрасывает исключения при ошибках.

- string parseCommand(char command) — возвращает текстовую команду, связанную с переданной клавишей. Выбрасывает исключение, если клавиша не зарегистрирована.

Класс GameManager

Класс GameManager реализует управление игровым процессом для игры "Морской бой". Он обрабатывает пользовательский ввод, управляет состоянием игры и взаимодействует с её основными компонентами, такими как печать состояния, чтение команд и контроль команд управления.

Поля класса GameManager:

- Game& m_game — ссылка на объект игры, представляющий текущее состояние игрового процесса.
- PrinterT m_game_printer — объект шаблонного типа для вывода состояния игры (игровое поле, статус игроков и т.д.).
- ReaderT m_game_read — объект шаблонного типа для чтения пользовательского ввода, включая команды, размеры поля, корабли и координаты.
- GameControlT m_game_control — объект шаблонного типа для управления командами и их ассоциацией с клавишами.

Методы класса GameManager:

- GameManager(Game& m_game, string control_filename) - Инициализирует игровой менеджер с объектом игры и загружает команды из указанного файла. Если файл недоступен или содержит ошибки, устанавливаются настройки управления по умолчанию.

Публичные методы

- void play() - Основной игровой цикл, включающий начальную настройку игры, вывод справки, обработку команд игрока (атака, помощь, загрузка, сохранение) и управление игровыми состояниями (победа, поражение, ничья).

Приватные методы

- void setNewEnemy() - Создаёт нового противника с рандомным размещением кораблей на игровом поле. Использует генератор случайных чисел для выбора координат и ориентации кораблей.
- void setNewGame() - Настраивает новую игру, создавая поля и корабли для игрока и противника. Проверяет корректность размеров поля и количества кораблей.
- void printHelp() - Выводит справочную информацию с текущими командами и их клавишами.

Шаблонные параметры

- PrinterT — отвечает за вывод информации об игре.
- ReaderT — отвечает за обработку пользовательского ввода.
- GameControlT — отвечает за контроль команд управления и их сопоставление с клавишами.

Класс Reader

Класс Reader предназначен для обработки пользовательского ввода, включая координаты, размеры игрового поля, команды, количество кораблей и имя файла.

Методы класса Reader:

Публичные методы

- vector<size_t> getCoords() - Считывает две целые координаты (x, y) из пользовательского ввода и возвращает их в виде вектора.
- vector<size_t> battlegroundSize() - Запрашивает у пользователя размеры игрового поля (ширину и высоту), считывает их и возвращает как вектор. Перед запросом выводит подсказку: "Введите размеры поля: ".

- char getCommand() - Считывает одиночный символ из пользовательского ввода, представляющий команду, и возвращает его.
- vector<size_t> getShips() - Запрашивает количество кораблей разной длины (1, 2, 3, 4) у пользователя. Выводит подсказку: "Введите кол-во кораблей длинны 1 2 3 4: ". Считывает 4 целых числа и возвращает их как вектор.
- string getFilename() - Запрашивает имя файла у пользователя. Перед запросом выводит подсказку: "Введите название файла: ". Возвращает введённую строку.

Класс SkillsManager

Класс SkillsManager отвечает за управление навыками (скиллами) в игре. Он включает функции для создания, добавления, использования и получения информации о доступных навыках. Класс работает с объектами навыков, которые реализуют интерфейс ISkill.

Поля класса

- vector<unique_ptr<ISkill>> skills - Список доступных навыков. Хранит уникальные указатели на объекты, реализующие интерфейс ISkill.

- vector<unique_ptr<ICreator>> creators - Список объектов типа ICreator, отвечающих за создание различных типов навыков (например, DoubleDamageCreator, BombingCreator, ScannerCreator).

Методы класса

- SkillsManager() - Создает объект менеджера навыков. Инициализирует creators списком из объектов для создания навыков (например, DoubleDamageCreator, BombingCreator, ScannerCreator). Создает

навыки с помощью объектов ICreator и перемешивает их в случайном порядке.

- SkillsManager(vector<string> skills_names) - Инициализирует менеджер навыков из списка имен навыков (skills_names). Использует соответствующий ICreator для создания навыков на основе переданных имен ("DoubleDamageSkill", "BombingSkill", "ScannerSkill").

Основные методы

- unique_ptr<ISkill> popSkill() - Удаляет и возвращает первый навык из списка skills. Если список пуст, выбрасывает исключение NoSkillException.
- void addRandomSkill() - Добавляет случайный навык в список skills, используя случайный объект из creators.
- void addRandomSkills(size_t n) - Добавляет n случайных навыков в список skills.
- bool usePopSkill(InfoHolder& info_holder) - Извлекает первый навык из списка skills и применяет его, вызывая метод useSkill() с переданным объектом info_holder.
- size_t size() - Возвращает текущее количество навыков в списке skills.
- size_t size() const - Константная версия метода size(), возвращающая количество навыков.
- vector<string> getPrettySkills() const - Возвращает список имен текущих навыков в человеко-читаемом формате. Определяет тип навыка через dynamic_cast и добавляет соответствующую строку ("DoubleDamageSkill", "BombingSkill", "ScannerSkill") в результирующий вектор.

Приватные методы

- ICreator& getRandomCreator() - Возвращает ссылку на случайный объект из списка creators.

main()

Функция main() реализует запуск основной игровой логики, создавая объекты для управления полем битвы, флотами кораблей и игровыми действиями. Инициализируются объекты игровых полей, менеджеров кораблей для игрока и противника, а также сам игровой процесс через класс Game. Управление вводом/выводом, обработка команд и игровой цикл реализуются с помощью шаблонного класса GameManager. Вся игра запускается с вызовом gm.play() в безопасной среде с обработкой исключений.

UML-диаграмма классов

Выводы

В рамках лабораторной работы был создан набор классов, реализующих управление игрой и отображение её состояния с использованием шаблонов. Разработаны шаблонные классы для обработки пользовательского ввода и отображения игрового процесса. Класс управления отвечает за обработку команд, заданных в конфигурационном файле, и их передачу игровым методам, что обеспечивает гибкость настройки команд и их привязки к клавишам. Визуализация состояния игры спроектирована таким образом, чтобы текстовый интерфейс можно было заменить на графический без изменения игровой логики. Система разделяет ответственность между модулями управления, ввода и вывода, что повышает модульность и расширяемость кода.