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 _____ Мучник М.К.

Преподаватель _____ Первицкий А.Ю.

Санкт-Петербург

2024

Цель работы

Изучить шаблоны. Разработать шаблонный класс, отвечающий за управление игрой. Данный класс будет определять способы отображения игры и считывания команд.

Задание

1. 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.
2. 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.
3. 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.
4. 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

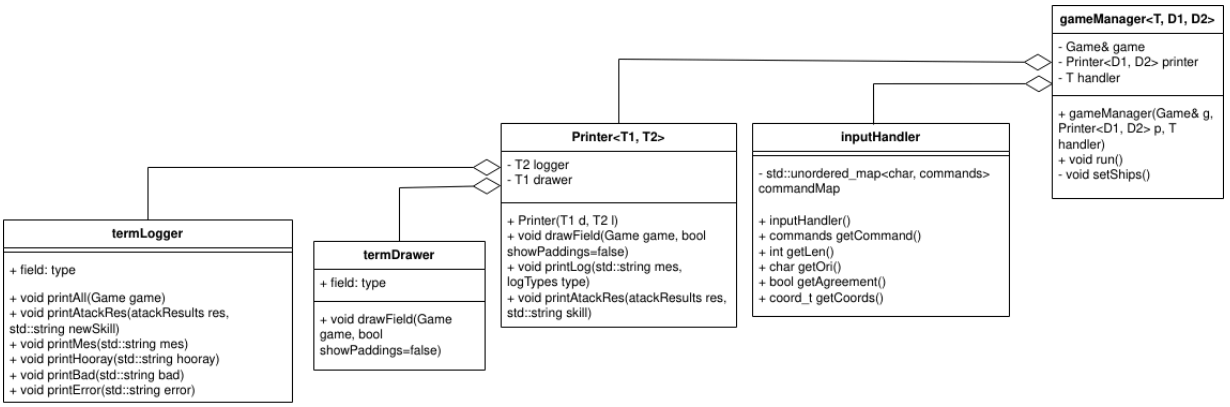

Рисунок 1 – UML-диаграмма реализованных классов

Реализация шаблонного класса управления игрой.

Для управления игрой был создан класс **gameManager**. В качестве приватных полей он содержит ссылку на класс игры, а также классы, отвечающие за отрисовку поля и считывания команд – данные классы передаются в качестве параметров шаблона. Класс содержит следующие методы:

- void setShips() – инициализирует пользовательское поле. Метод запрашивает у пользователя длину и ориентацию корабля, проверяет корректность введенных данных и размещает корабли на поле, пока все корабли не будут размещены. В случае возникновения ошибки пользователю придётся ввести данные повторно;
- конструктор GameManager – инициализирует приватные поля. Определяет ссылку на класс игры, а также класс считывания и отрисовки;
- метод run() – осуществляет основной цикл обработки команд. До тех пор, пока раунд не завершён считывает из терминала команду, после чего вызывает соответствующий метод класса игры. В процессе

считывания команд отрисовывается поле с изменениями. В процессе игры также осуществляется ход противника.

Реализация шаблонного отображения игры.

В качестве параметров шаблона класс отображения игры принимает класс-отрисовщик и вызывает у того соответствующий метод отрисовки поля, а также класс-логгер, выводящий пользователю какие-то оповещения, сообщения и ошибки. При выводе ошибки также очищаются все ошибки потока std::cin и игнорируются все введённые символы.

Реализация класса отвечающего за ввод

- Конструктор inputHandler – инициализирует объект inputHandler, загружая команды из файла commands.txt. Если файл не может быть прочитан или команды не соответствуют ожидаемым, устанавливаются значения по умолчанию для команд;
- Метод getCommand – запрашивает у пользователя ввод команды и проверяет, существует ли такая команда. Если команда недействительна, выбрасывается исключение;
- Метод getLen – запрашивает у пользователя длину корабля. Если ввод некорректен, выбрасывается исключение impoperLenException. Возвращает введённую длину;
- Метод getOri – запрашивает у пользователя ориентацию корабля (вертикальная или горизонтальная). Если ввод некорректен, выбрасывается исключение impoperOriException. Возвращает введённую ориентацию;
- Метод getAgreement – запрашивает у пользователя согласие (да/нет). Если ввод некорректен, выбрасывается исключение improperInputException;

- Метод getCoords – запрашивает у пользователя координаты в формате "буква цифра". Если ввод некорректен, выбрасывается исключение. Возвращает введённые координаты в виде структуры coord_t.

Выводы

Дописана игра Морской бой. Реализован шаблонный класс управления игрой, позволяющий использовать разные способы отображения игры и обрабатывать пользовательские команды, обеспечивая интерактивность и удобство.