МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Объектно-ориентированное программирование »

Тема: Полиморфизм

Студент гр. 3384 _____ Пьянков М.Ф.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы

Изучить виртуальные методы, научится создавать класс-интерфейс, создать классы-ошибки и реализовать обработку исключений. Реализовать новые модули программы, удовлетворяющие условию лабораторной работы.

Основные теоретические положения

В лабораторной работе применялись виртуальные методы, функции библиотеки random, контейнер queue.

Использовался стандартный класс ошибок std::invalid_argument для создания базового класса ошибок в программе, а также методы try, throw и catch для обработки ошибок.

Программа разделена на модули, в каждом модуле — один класс.

Задание

a) Создать класс-интерфейс способности, которую игрок может применять. Через наследование создать 3 разные способности:

   i. Двойной урон - следующая атак при попадании по кораблю нанесет сразу 2 урона (уничтожит сегмент).

   ii. Сканер - позволяет проверить участок поля 2x2 клетки и узнать, есть ли там сегмент корабля. Клетки не меняют свой статус.

   iii. Обстрел - наносит 1 урон случайному сегменту случайного корабля. Клетки не меняют свой статус.

b) Создать класс менеджер-способностей. Который хранит очередь способностей, изначально игроку доступно по 1 способности в случайном порядке. Реализовать метод применения способности.

c) Реализовать функционал получения одной случайной способности при уничтожении вражеского корабля.

d) Реализуйте набор классов-исключений и их обработку для следующих ситуаций (можно добавить собственные):

   i. Попытка применить способность, когда их нет

   ii. Размещение корабля вплотную или на пересечении с другим кораблем

   iii. Атака за границы поля

Примечания: Интерфейс события должен быть унифицирован, чтобы их можно было единообразно использовать через интерфейс. Не должно быть явных проверок на тип данных

Выполнение работы

Архитектура способностей представлена на рис. 1.

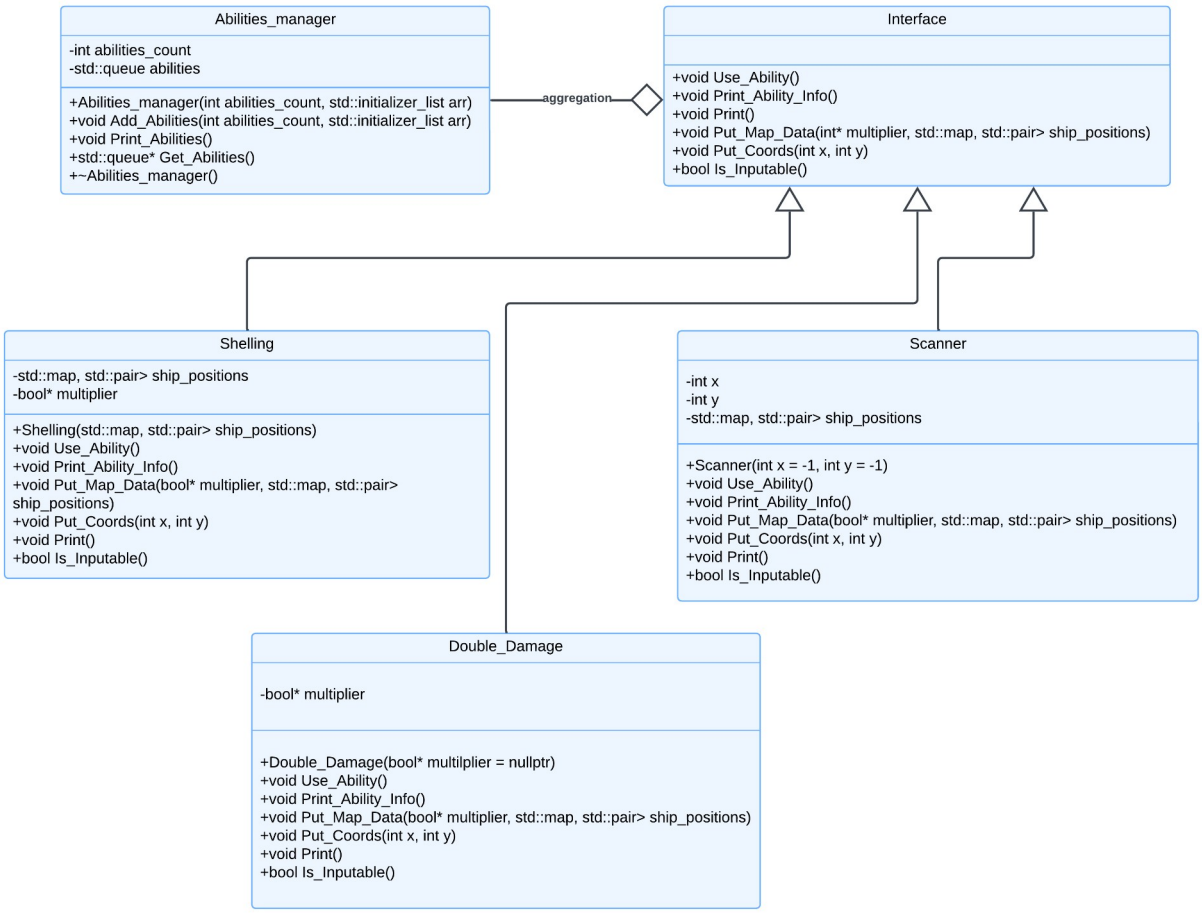

Рисунок 1 — Архитектура новых модулей программы (UML-диаграмма классов)

Архитектура классов-ошибок представлена на рис. 2.

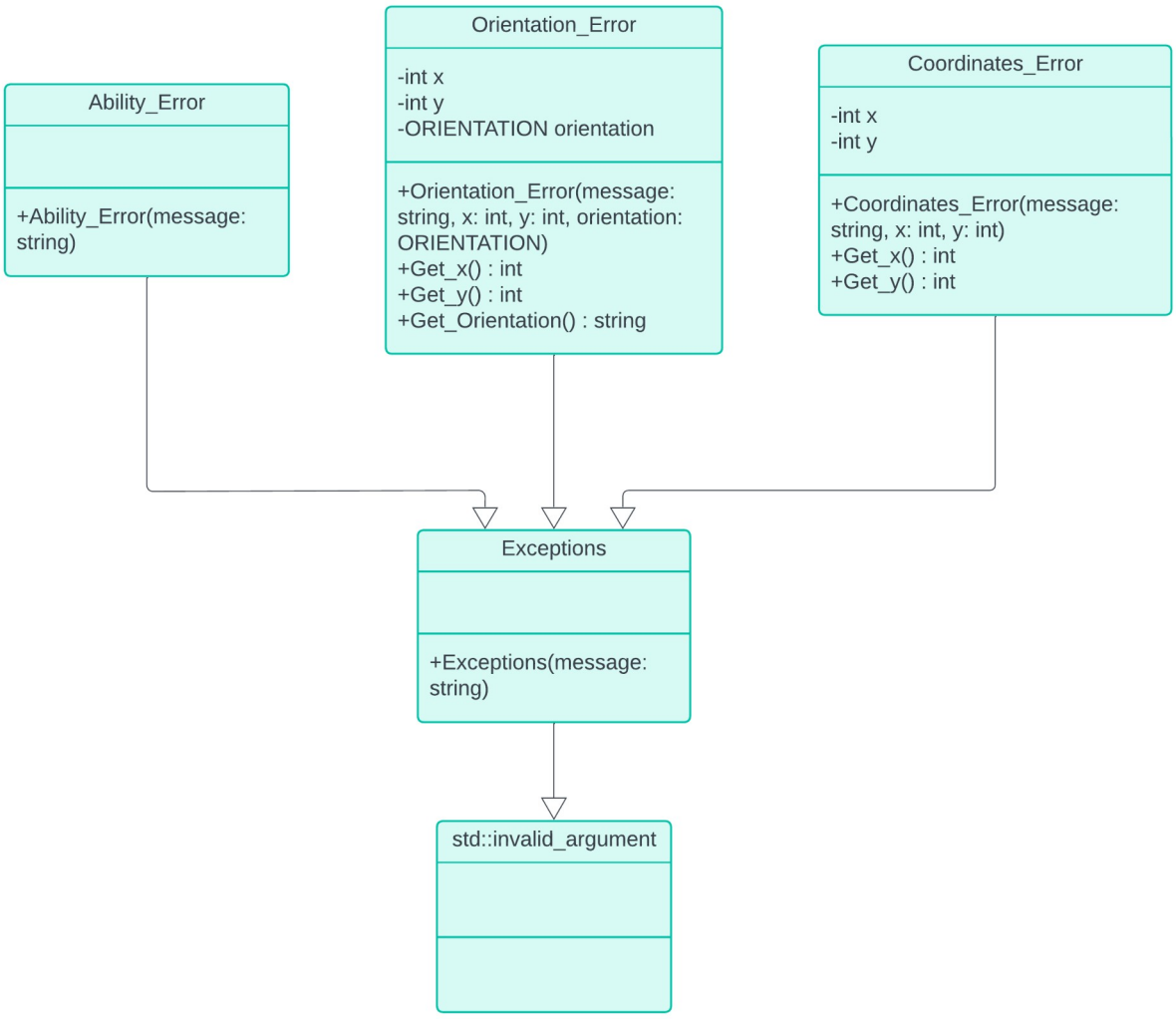

Рисунок 2 — UML-диаграмма классов-ошибок

Основные элементы: интерфейс, способности, менеджер способностей, карта, классы-ошибки.

Взаимодействие элементов: менеджер способностей хранит очередь способностей. Карта принимает очередь способностей и использует её.

Класс Interface – класс-интерфейс для всех способностей. Все методы виртуальные.

Поля: bool inputable – поле, показывающее необходимо ли для данной способности считывать данные с потока ввода.

Методы: virtual void Use_Ability() = 0 — метод, для применения способности, virtual void Print_Ability_Info() = 0 — метод вывода информации о способности, virtual void Print() = 0 — метод вывода названия способности, virtual void Put_Map_Data(int* multiplier, std::map< std::pair<int, int>, std::pair<Ship*, enum ORIENTATION> >ship_positions) = 0 — метод для получения данных из объекта класса Map, virtual void Put_Coords(int x, int y) = 0 – метод для установки координат применения способности.

Классы Double_Damage, Scanner, Shelling – являются наследниками класса Interface и представляют реализацию методов, описанных в нём.

Метод Use_Ability():

Double_Damage – изменяет по указателю множитель урона mutiplier из класса Map.

Scanner – итерируется по ship_positions из класса Map и выводит ответ.

Shellig – с помощью библиотеки ctime выбирается случайный целый корабль и в нём выбирается не уничтоженный сегмент. Итерирование происходит по ship_positions из класса Map.

Класс Ability_Manager – класс необходимый для хранения способностей.

Поля: int abilities_count — количество способностей, std::queue<Interface*>abilities — очередь способностей.

Методы: void Add_Abilities(int abilities_count, std::initializer_list<enum ABILITY> arr) — метод добавления новой способности в очередь, void Print_Abilities() - метод вывода имён всех способностей в очереди, std::queue<Interface*>* Get_Abilities() - метод, передающий очередь способностей в объект класса Map.

Класс Map – покажем новые поля и методы.

Поля: int multiplier — множитель урона, std::queue<Interface*>* - очередь способностей.

Методы: void Add_Abilities(std::queue<Interface*>*abilities) — метод получения очереди способностей от объекта класса Ability_Manager, void Use_Ability() - метод применения способности из очереди.

Также было изменено следующие: теперь сегмент корабля это структура cell с полями segment и user_action – для проверки, что ячейка атакована пользователем, а не способностью. Также были переработаны методы Make_Shoot() в классе Map с учётом выше описанного.

Класс Exceptions – базовый класс ошибок, наследуемый от стандартного класса ошибок std::invalid_argument.

Классы Ability_Error, Coordinates_Error, Orientation_Error – классы наследующиеся от базового класса ошибок - Exceptions(), выводящие соответствующие сообщения об ошибке с дополнительной информацией.

Выводы

Успешно реализованы классы способностей и интерфейс для них, реализован класс ошибок, а также метода их обработки. Изучены виртуальные методы, контейнер queue, библиотека random.