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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «ООП»

Тема: Полиморфизм

Студент гр. 3384 _____ Баяндин Д.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы.

Изучить основные принципы полиморфизма на языке C++, написать полиморфные классы, которые реализуют способности для игры морской бой.

Задание.

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

- Двойной урон - следующая атак при попадании по кораблю нанесет сразу 2 урона (уничтожит сегмент).
- Сканер - позволяет проверить участок поля 2x2 клетки и узнать, есть ли там сегмент корабля. Клетки не меняют свой статус.
- Обстрел - наносит 1 урон случайному сегменту случайного корабля. Клетки не меняют свой статус.

2. Создать класс менеджер-способностей. Который хранит очередь способностей, изначально игроку доступно по 1 способности в случайном порядке. Реализовать метод применения способности.
3. Реализовать функционал получения одной случайной способности при уничтожении вражеского корабля.
4. Реализуйте набор классов-исключений и их обработку для следующих ситуаций (можно добавить собственные):

- Попытка применить способность, когда их нет
- Размещение корабля вплотную или на пересечении с другим кораблем
- 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.

1.) Исправления предыдущей лабораторной работы:

Метод *create_field()* теперь только создает поле, расстановкой кораблей занимается метод *put_ship()*, в котором реализовано “отлавливание” неверных координат и их перезапись с помощью конструкции *try-catch*. Так же в методе *attack_segme*nt была реализована конструкция *try-catch*. Созданы два метода, которые работают со способностями bombing – реализует туман войны, а check_cell – реализует проверку определенного сегмента поля для сканера.

2.) Лабораторная работа:

Класс *iAbility* – абстрактный базовый класс, который определяет три виртуальных метода для классов способностей: use_ability, setcoordinates, getname.

Класс *DoubleDamage*, наносит двойной урон по заданной пользователем клеткой, в методе *use_ability()* вызывается метод атаки поля *attack_segment()* с флагом true, что означает, что удар будет двойным. В методе *setcoordinates()* инициализируются координаты, куда будет произведена атака. Этот метод был сделан для того, чтобы можно было инициализировать координаты в момент применения способности, а не когда она будет только записана в вектор. Метод *getname()* возвращает названия метода – это сделано для того, чтобы можно узнать запрашивать ли координаты у пользователя или нет.

Класс *Scanner*, сообщает находится ли в заданном пользователем квадрате 2 на 2 корабль, в методе *use_ability()* перебирается клетки нужного квадрата и вызывается метод поля *check_cell,* который возвращает true, если в текущей клетке есть корабль. Метод *setcoordinates* и *getname()* аналогичны предыдущему классу.

Класс *Bombardment* атакует один случайный сегмент корабля, при это не отображая пользователю изменения на карте, в методе *use_ability()* с помощью функции *rand()* сначала определяется случайный корабль, а затем случайный сегмент этого корабля. С помощью метода *bombing()* на поле меняется состояние клетки на *fog_war* или *dead_ship*, *fog_war* означает, что при выводе поля, оно не будет показываться, как подбитый корабль. Метод *setcoordinates()* здесь пуст,

потому что класс не принимает координаты, а метод *getname()* аналогичен предыдущим.

Класс *AbilityManager* в конструкторе инициализирует вектор *queue_abilities* и добавляет в него по одному объекту классов способностей, который с помощью функции shuffle перемешиваются. Метод *add_ability()* получает случайное число от 0 до 2, и в зависимости от этого числа добавляет какую-то способность в вектор *queue_abilities.* Метод *is_empty()* выводит true, если способностей нет, и false, если способности еще остались. Метод *apply_abilities()* достает первую способность из очереди и вызывает метод *getname()*, чтобы проверить нужен ли для применения способности ввод координат, если да, то принимает координаты и с помощью конструкции try-catch проверяет их на корректность, а если они не корректны запрашивает заново. Назначает с помощью *set_coordinates(x, y),* полученные координаты. Применяет способность с помощью *use_ability* и удаляет её из очереди способностей.

Класс *Exception* наследуется от *invalid_argument* и будет базовым классов для всех остальных классов ошибок.

Класс *IncorrectCoordinatesException* означает, что ошибка случилась при вводе координат.

Класс PlaceShipException означает, что ошибка возникла при расстановке кораблей, методы *getxerror* и *getyerror* вернут координаты, на которых или рядом с которыми стоит корабль.

Класс IncorrectFieldSize означает, что ошибка возникла при запрашивании размера поля, а IncorrectQuantity означает, что ошибка возникла при запросе количества кораблей.

Разработанный программный код см. в приложении А.

UML диаграмма.

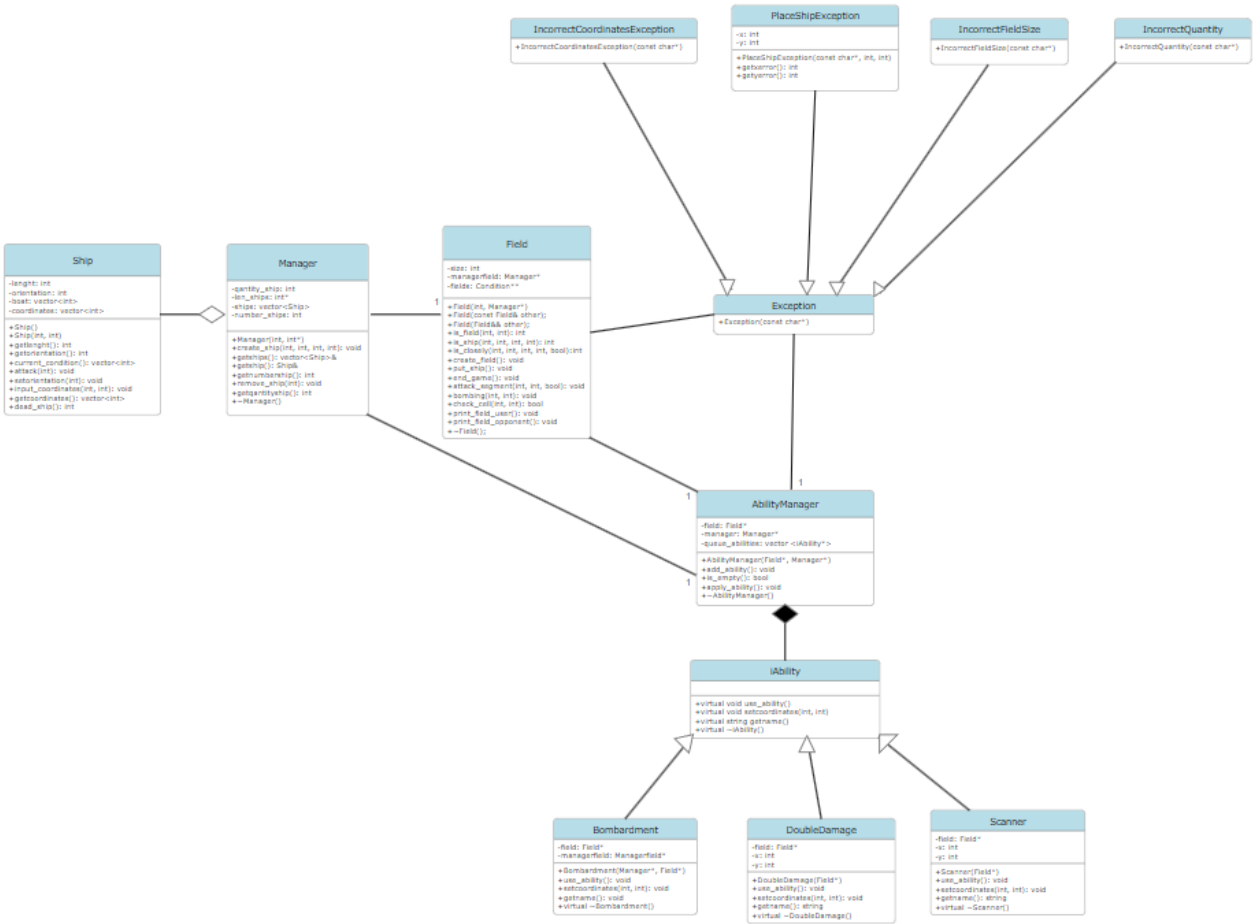

Рисунок 1 - UML Диаграмма

Выводы.

В ходе выполнения лабораторной работы было реализовано полиморфные классы, которые создают и применяют способности. Разработан класс управления способностями, а также классы ошибок.