**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

1. Создать класс-интерфейс способности, которую игрок может применять. Через наследование создать 3 разные способности:
   1. Двойной урон - следующая атак при попадании по кораблю нанесет сразу 2 урона (уничтожит сегмент).
   2. Сканер - позволяет проверить участок поля 2х2 клетки и узнать, есть ли там сегмент корабля. Клетки не меняют свой статус.
   3. 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):
   1. Попытка применить способность, когда их нет
   2. Размещение корабля вплотную или на пересечении с другим кораблем
   3. 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.

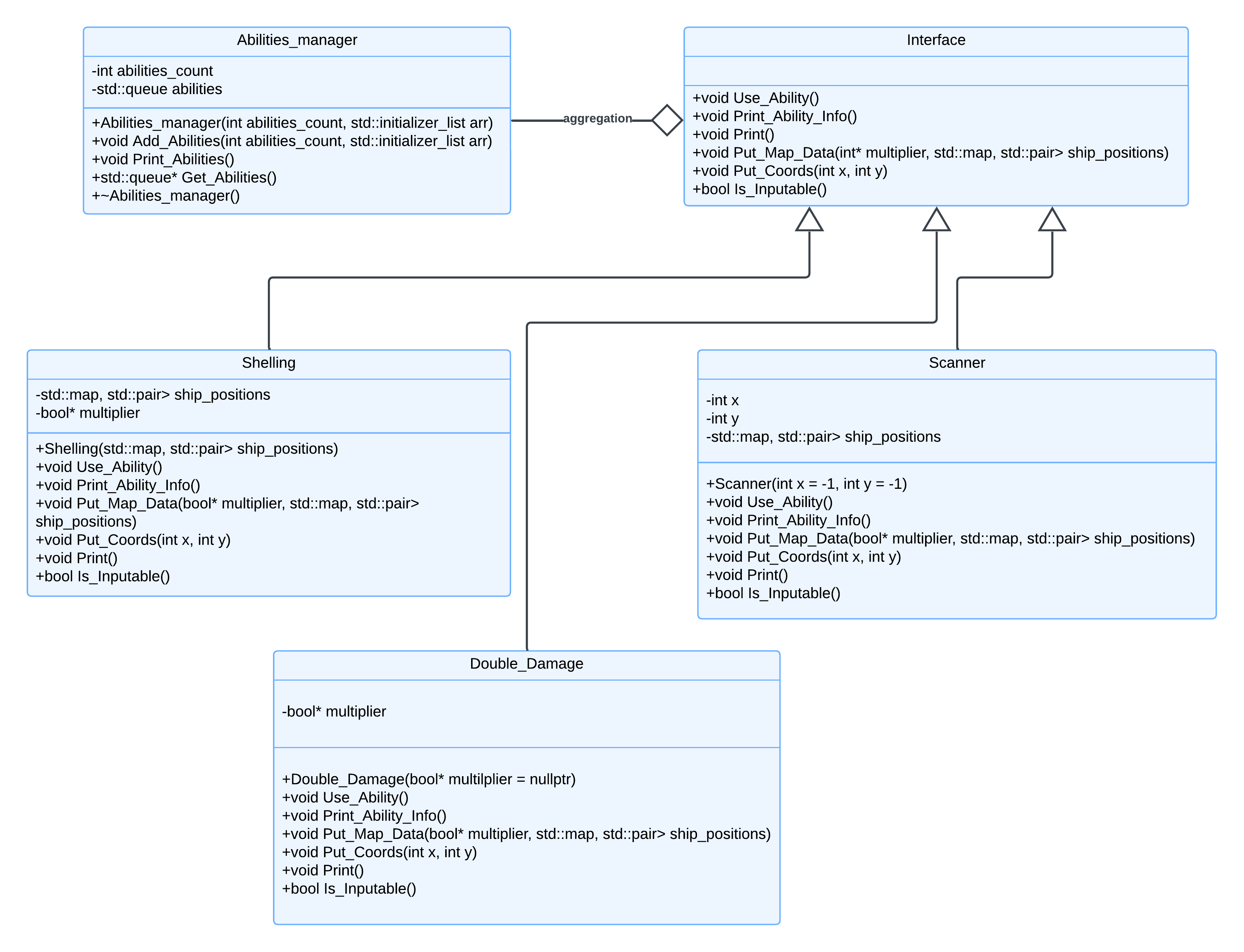
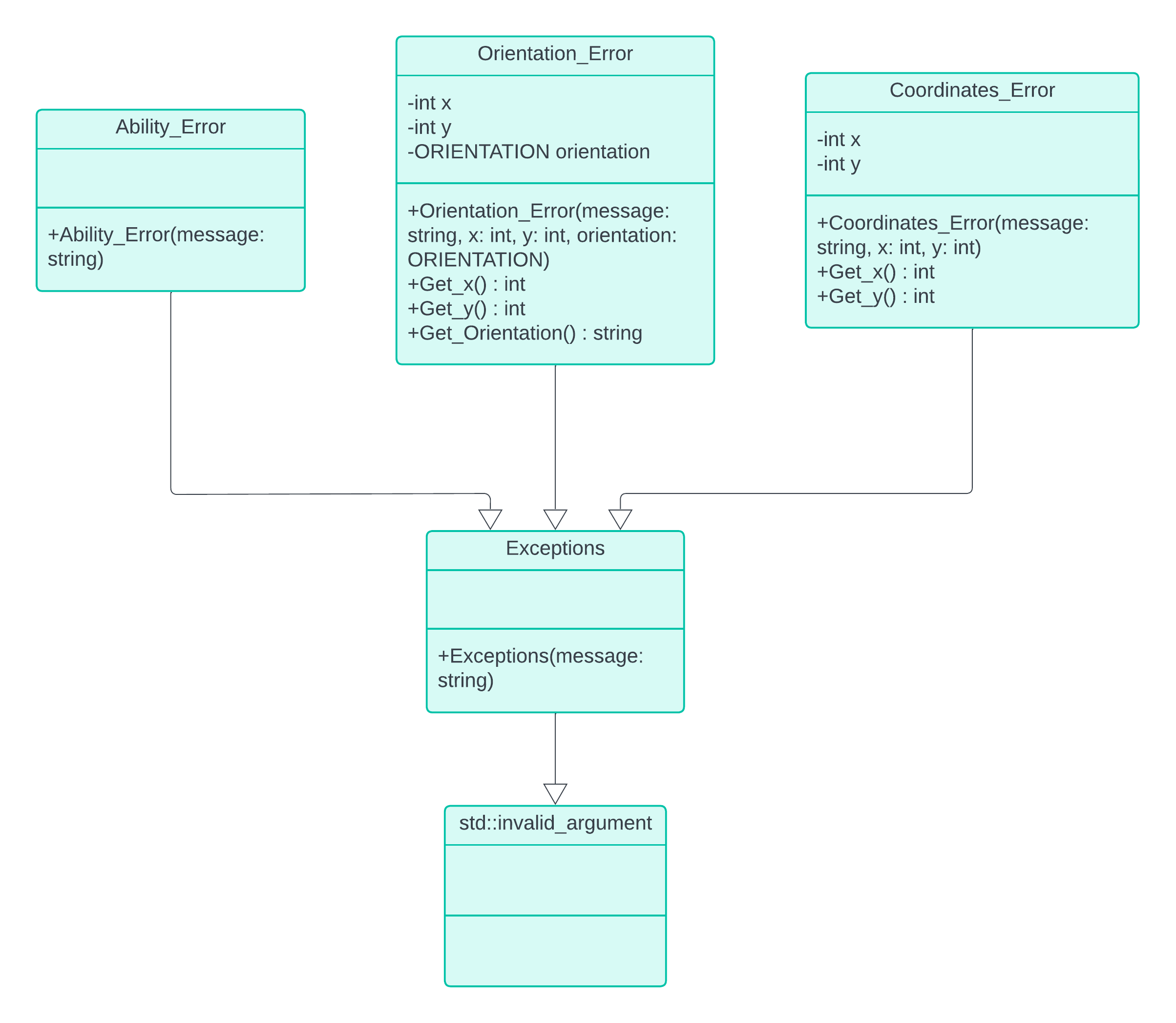


Рисунок 1 — Архитектура новых модулей программы (UML-диаграмма классов)

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.