**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе № 2**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: «Полиморфизм»



|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2024

# Цель работы

Изучить основы полиморфизма классов в ООП C++, создать класс-интерфейс способности и три способности на его основе, класс менеджера способностей, реализовать набор классов-исключений для различных исключительных ситуаций.

# 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

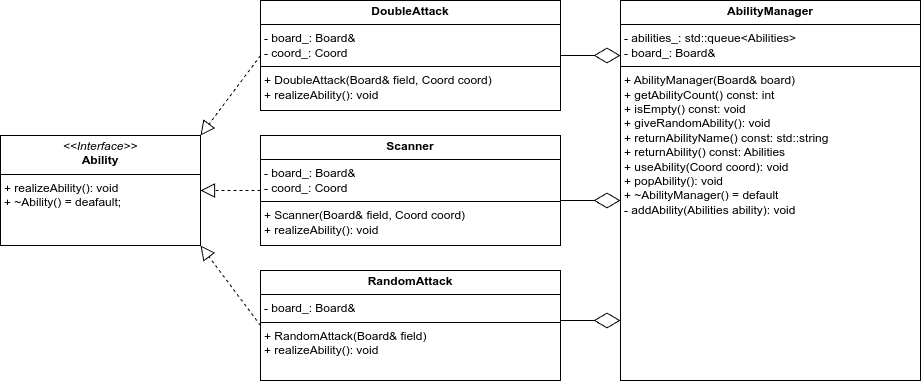


Рисунок 1 – UML-диаграмма классов

Код программы содержит реализацию классов: *Ability*, *DoubleAttack*, *Scanner*, *RandomAttack*, и *AbilityManager* согласно заданию.

Помимо обозначенных классов, реализованы и интегрированы в код следующие 4 класса-исключений:

* Неправильный размер корабля
* Атака за пределы поля
* Попытка применить способность, когда её их нет
* Размещение корабля на уже существующем корабле или вплотную, возле него.

*Ability* является классом-интерфейсом для способностей. Он имеет следующие виртуальные методы:

* *virtual void realizeAbility() = 0* – виртуальный метод для применения способности.
* *virtual ~Ability() {}* – виртуальный деструктор класса.

Класс *DoubleDamage* является реализацией способности двойного урона. Он имеет следующие поля:

* *Board& board* – ссылка на поле.
* *Coord coord* – координаты для применения способности.

И следующие методы:

* *DoubleDamage(Board& board, Coord coord)* – конструктор класса.
* *void realizeAbility() override* – производится двойная атака по находящимся в поле класса координатам. Если координаты выходят за область поля, то вызывается Attack*OutOfRangeException*.

Класс *Scanner* является реализацией способности сканера, который смотрит участок поля 2x2 на наличие кораблей в нём. Он имеет следующие поля:

* *Board& board* – ссылка на поле.
* *Coord coord* – координаты для применения способности.

И следующие методы:

* *Scanner(Board& board, Coord coord)* – конструктор класса.
* *void realizeAbility() override* – производится сканирование поля в области 2x2 по координатам из поля класса, где поданная координата обозначает левый верхний угол.

Класс *RandomAttack* является реализацией способности случайного выстрела по случайному сегменту корабля. Он имеет следующие поля:

* *Board& board* – ссылка на поле.

Он имеет следующие методы:

* *RandomAttack(Board& board)* – конструктор класса.
* *void realizeAbility() override* – производится выстрел по случайному сегменту, не изменяя состояние клетки.

Класс *AbilityManager* отвечает за контроль над способностями, он хранит в очереди названия способностей, которые используются для создателей способностей. Он имеет следующие поля:

* *queue<Abilities> abilities* – очередь имён способностей.
* *Board& board* – ссылка на поле.

И следующие методы:

* *AbilityManager(Board& board)* – конструктор класса.
* *int getAbilityCount() const* – возвращает размер очереди.
* *Abilities returnAbility() const* – возвращает элемент в начале очереди.
* *void isEmpty()* – проверяет очередь на наличие способностей. Если их нет, то вызывает исключение *NoAbilitiesAvaibleException*.
* *void addAbility(Abilities ability)* – добавляет новую способность в очередь.
* *void giveRandomAbility()* – добавляет случайную способность в очередь.
* *void useAbility(Coordinate coordinate={-1,-1})* – применяет способность из очереди. Если в метод не поданы координаты, то используется RandomAttack, иначе DoubleAttack или Scanner. Создаётся объект способности и он сразу же применяется. После выполнения способность убирается из очереди, а сам объект уничтожается.
* *void popAbility()* – удаляет способность из очереди.

# Тестирование:

В *game.cpp* файле происходит симуляция игры с одним игроком и его полем, В начале хода игрок может использовать одну случайную способность или сразу перейти к атаке вражеского поля.

Если применяется способность, то сначала проверяется её наличие, в обратном случае вызывается исключение *NoAbilitiesAvailableException* и ход обнуляется. Затем, если попалась способность двойного урона или сканера, необходимо подать координаты. Если поданные координаты являются некорректными, то вызывается исключение *OutOfRangeException* и ход обнуляется. Затем способность используется.

Иначе игрок атакует поле и при выстреле открывает соответствующую ячейку. При уничтожении корабля выдаётся одна случайная способность. При уничтожении всех кораблей игра заканчивается.

# Выводы

Во время выполнения лабораторной работы были изучены основы объектно-ориентированного программирования, реализованы классы на языке С++, написан первый прототип программы для игры в морской бой.

Во время выполнения данной лабораторной работы были изучены основы полиморфизма классов в ООП C++, был класс-интерфейс способности и три способности на его основе: двойной урон, сканер и обстрел, класс менеджера способностей для использования данных способностей, реализован набор классов-исключений для различных ситуаций, первый прототип игры.

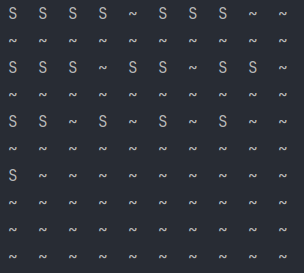


Рисунок 2 – Начало игры. Расстановка кораблей.

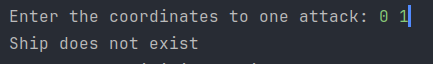


Рисунок 3 – Сообщение при промахе