**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Создание классов

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3388 |  | Снигирев А.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2024

# Цель работы

Написать основные классы кораблей, игрового поля и менеджера кораблей и необходимые методы к ним.

# Задание

1. Создать класс корабля, который будет размещаться на игровом поле. Корабль может иметь длину от 1 до 4, а также может быть расположен вертикально или горизонтально. Каждый сегмент корабля может иметь три различных состояния: целый, поврежден, уничтожен. Изначально у корабля все сегменты целые. При нанесении 1 урона по сегменту, он становится поврежденным, а при нанесении 2 урона по сегменту, уничтоженным. Также добавить методы для взаимодействия с кораблем.
2. Создать класс менеджера кораблей, хранящий информацию о кораблях. Данный класс в конструкторе принимает количество кораблей и их размеры, которые нужно расставить на поле.
3. Создать класс игрового поля, которое в конструкторе принимает размеры. У поля должен быть метод, принимающий корабль, координаты, на которые нужно поставить, и его ориентацию на поле. Корабли на поле не могут соприкасаться или пересекаться. Для игрового поля добавить методы для указания того, какая клетка атакуется. При попадании в сегмент корабля изменения должны отображаться в менеджере кораблей.

Каждая клетка игрового поля имеет три статуса:

* неизвестно (изначально вражеское поле полностью неизвестно),
* пустая (если на клетке ничего нет)
* корабль (если в клетке находится один из сегментов корабля).

Для класса игрового поля также необходимо реализовать конструкторы копирования и перемещения, а также соответствующие им операторы присваивания.

Примечания:

Не забывайте для полей и методов определять модификаторы доступа

Для обозначения переменной, которая принимает небольшое ограниченное количество значений, используйте enum

Не используйте глобальные переменные

При реализации копирования нужно выполнять глубокое копирование

При реализации перемещения, не должно быть лишнего копирования

При выделении памяти делайте проверку на переданные значения

У поля не должно быть методов возвращающих указатель на поле в явном виде, так как это небезопасно

## Выполнение работы

**Файл main.сpp**. Здесь лежит функция main() и подключаются заголовочные файлы. В функции происходит создание объектов классов. Также в файле временно расположены вызовы различных методов для проверки.

Файл Ship.h. В этом файле объявлен класс корабля.

Объявлены поля и методы класса Ship:

const int MIN\_LENGTH = 1; - константа, минимальная длина

const int MAX\_LENGTH = 4; - максимальная длина

int length; - фактическая длина

Orientation orientation; - ориентация

std::vector<DamageLevel> segment\_arr; - вектор самого корабля

Методы:

Ship(int length, Orientation orientation); - конструктор

~Ship(); - деструктор

int get\_length() const; - геттер, возвращает длину

Orientation get\_orientation() const; - геттер, возвращает ориеннтацию

SegmentStatus get\_seg\_status(int index) const; - возвращает состояние выбранного сегмента

void decrease\_health(int index, int damage); - уменьшает здоровье выбранного сегмента на нужную величину

bool is\_destroyed() const; - проверяет разрушенность корабля

В файле Ship.cpp эти методы реализованы.

Файл manager.h. Объявление класса ShipManager.

Поля:

const int MIN\_LENGTH = 1;

const int MAX\_LENGTH = 4;

std::vector<std::unique\_ptr<Ship>> ships; - массив указателей на корабли

std::vector<bool> shipSunkStatus; - массив текущих статусов

int shipsRemaining; - текущее число кораблей

Методы реализованы в manager.cpp.

Методы:

ShipManager(const std::vector<int>& shipSizes); - конструктор

const std::unique\_ptr<Ship>& get\_ship(int index) const; - метод, возвращающий корабль соответствующего индекса

void change\_ship\_status(Ship\* ship); - обновляет корабль, указатель на который получает

int get\_start\_length() const; - геттер, возвращает начальное количество кораблей

int get\_ships\_remaining() const; - геттер, возвращает текущее количество кораблей

bool check\_all\_ships() const; - проверяет целостность всего флота

Файл gamefield.hpp. Объявление класса игрового поля.

Поля:

int width; - ширина поля

int height; - высота поля

std::vector<std::vector<CellStatus>> field; - поле

std::vector<std::vector<Ship\*>> ships\_links; - поле из ссылок на корабли

std::vector<std::vector<int>> segmentIndexGrid; - поле индексов соответствующих кораблей

Методы:

bool place\_ship(Ship\* ship, int x, int y, Orientation orientation); - размещение корабля

CellStatus get\_cell\_status(int x, int y) const; - геттер, возвращает точку выбранной координаты

void attack(int x, int y, ShipManager& shipManager); - проводит атаку на выбранную точку.

char get\_field\_point(int x, int y) const; - геттер, возвращает точку в человекопонятном виде

bool is\_correct\_position(int x, int y) const; - проверка

bool can\_place\_ship(Ship\* ship, int x, int y, Orientation orientation) const; - проверка возможности размещения

bool touch\_other\_ship(int x, int y) const; - проверка на касание

void copy\_from(const GameField& other); - копирование

GamerField(); - конструктор, инициализирует нулями все точки.

void set\_ship(Ship& shp) — ставит корабль на поле

void attack(Point pt) — производит атаку на выбранную коррдинату.

void delete\_ships() - обнуляет все поле

void print() - печатает поле

std::vector<std::vector<CellStatus>> get\_field() - возвращает копию поля.

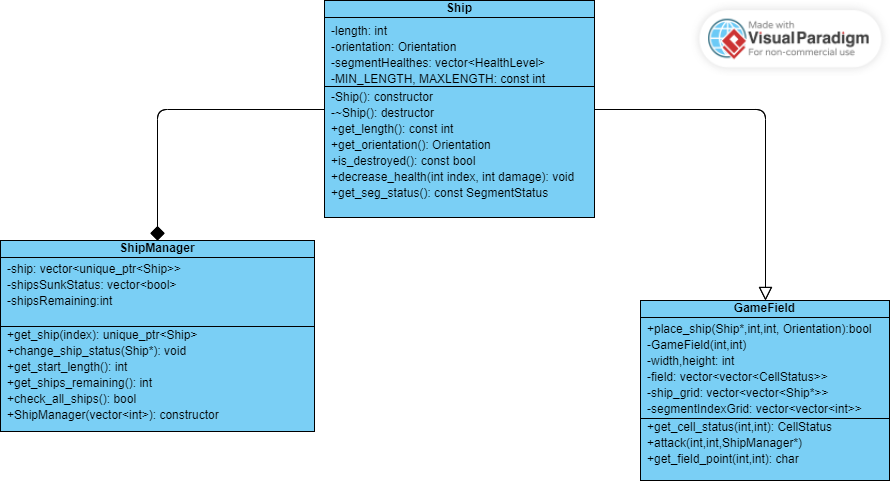
UML-диаграмму классов смотреть в **ПРИЛОЖЕНИИ Б**

# Вывод

В ходе лабораторной работы были реализованы базовые классы для создания игры «Морской бой». Также была прописана логика, примерно намечающая курс дальнейшего способа написания игры.

**ПРИЛОЖЕНИЕ Б**

UML-диаграмма

Рисунок 1. UML-диаграмма