**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. 3344 | Анахин Е.Д. |
| Преподаватель | Жангиров Т.Р. |

Санкт-Петербург

2024

## Цель работы.

Освоить использование принципа «полиморфизм» в языке cpp.

**Задание**.

1. Создать класс-интерфейс способности, которую игрок может применять. Через наследование создать 3 разные способности:
2. Двойной урон - следующая атак при попадании по кораблю нанесет сразу 2 урона (уничтожит сегмент).
3. Сканер - позволяет проверить участок поля 2х2 клетки и узнать, есть ли там сегмент корабля. Клетки не меняют свой статус.
4. Обстрел - наносит 1 урон случайному сегменту случайного корабля. Клетки не меняют свой статус.
5. Создать класс менеджер-способностей. Который хранит очередь способностей, изначально игроку доступно по 1 способности в случайном порядке. Реализовать метод применения способности.
6. Реализовать функционал получения одной случайной способности при уничтожении вражеского корабля.
7. Реализуйте набор классов-исключений и их обработку для следующих ситуаций (можно добавить собственные):
8. Попытка применить способность, когда их нет
9. Размещение корабля вплотную или на пересечении с другим кораблем
10. Атака за границы поля
11. Примечания:
12. Интерфейс события должен быть унифицирован, чтобы их можно было единообразно использовать через интерфейс
13. Не должно быть явных проверок на тип данных

## Выполнение работы.

Был создан класс-интерфейс для способностей. По этому интерфейсу были созданы класс-способности:

* DoubleDamageAbility — наносит двойной урон по клетке
* RandomAttackAbility — наносит урон случайному сегменту корабля
* ScannerAbility — сканирует область 2x2 и выводит, есть ли в том месте корабль

Также был создан менеджер способностей, который хранит их, возвращает (для использования) и добавляет новые.

* **Класс Ability (DoubleDamage, RandomAttack, Scanner)**

**Методы:**

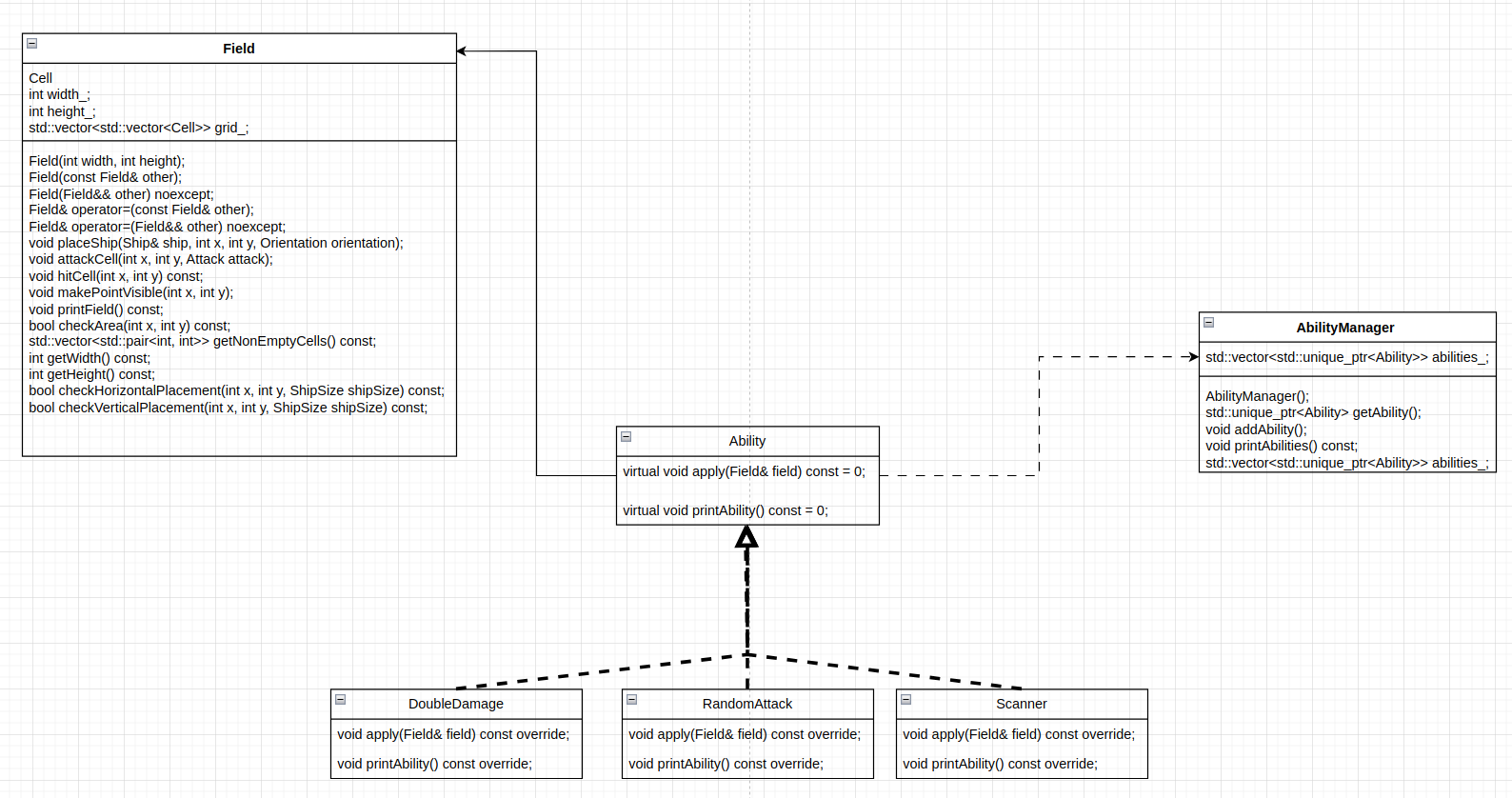
* virtual void apply(Field& field) const- виртуальный метод использования способности
* virtual void printAbility() const — виртуальный метод вывода способности
* **Класс AbilityManager**

**Методы:**

* AbilityManager(); - конструктор
* std::unique\_ptr<Ability> getAbility(); - возвращает абилку для использования
* void addAbility(); - добавляет абилку в стек
* void printAbilities() const; - выводит список абилок

Классы исключения:

1. **AttackOutOfRangeException** — вызывается, если пользователь пытается атаковать за пределы поля
2. **ShipPlacementOutOfRangeException** — вызывается, если пользователь пытается поставить корабль за пределы поля
3. **ShipIncorrectPlacementException** — вызывается, если пользователь пытается поставить один корабль за рядом с другим или за пределы поля
4. **BaseException** — базовый класс исключений
5. **EmptyAbilityException** — вызывается, если пользователь пытается вызывать способность, когда ее нет

****

Исходный код программы см. в Приложении А.

**Вывод:**

В ходе лабораторной работы были созданы классы для способностей, а также созданы исключения и функционал добавления способности при уничтожения корабля.

**ПРИЛОЖЕНИЕ A.**

**ИСХОДНЫЙ КОД ПРОГРАММЫ.**

main.cpp:

*#include <iostream>*

*#include "entities/ship/ship.h"*

*#include "managers/ship\_manager/ship\_manager.h"*

*#include "entities/field/field.h"*

*#include <limits>*

*#include "abilities/base\_ability.h"*

*#include "managers/ability\_manager/ability\_manager.h"*

*#include "managers/ability\_manager/exceptions.h"*

*int main() {*

*ShipManager shipManager({ShipSize::BIG, ShipSize::SMALL, ShipSize::MEDIUM, ShipSize::LARGE});*

*Field field(10, 10);*

*AbilityManager abilityManager;*

*try {*

*field.placeShip(shipManager.getShip(0), 1, 1, Orientation::Horizontal);*

*field.placeShip(shipManager.getShip(2), 8, 1, Orientation::Vertical);*

*} catch(const ShipPlacementOutOfRangeException& e) {*

*std::cerr << e.what() << '\n';*

*} catch(const ShipIncorrectPlacementException& e) {*

*std::cerr << e.what() << '\n';*

*}*

*int x, y;*

*bool gameOver = false;*

*std::string command;*

*while (!gameOver) {*

*if (shipManager.hasDestroyedShipsChanged() == true) {*

*abilityManager.addAbility();*

*}*

*abilityManager.printAbilities();*

*std::cout << "Введите команду (attack x y / ability / Конец): ";*

*std::cin >> command;*

*if (command == "attack") {*

*std::cin >> x >> y;*

*if (std::cin.fail()) {*

*std::cerr << "Неверный ввод" << std::endl;*

*std::cin.clear();*

*std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');*

*continue;*

*}*

*try {*

*field.attackCell(x, y, Attack::Default);*

*field.printField();*

*} catch(const AttackOutOfRangeException& e) {*

*std::cerr << e.what() << '\n';*

*}*

*} else if (command == "ability") {*

*try {*

*std::shared\_ptr<Ability> ability = abilityManager.getAbility();*

*ability->apply(field);*

*field.printField();*

*} catch(const EmptyAbilityException& e) {*

*std::cerr << e.what() << '\n';*

*}*

*} else if (command == "Конец") {*

*gameOver = true;*

*} else {*

*std::cerr << "Неизвестная команда" << std::endl;*

*std::cin.clear();*

*std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');*

*continue;*

*}*

*}*

*std::cout << "Game Over!" << std::endl;*

*field.printField();*

*shipManager.printShips();*

*return 0;*

*}*

base\_ability.h:

*#ifndef BASE\_ABILITY\_H*

*#define BASE\_ABILITY\_H*

*#include <iostream>*

*#include "../entities/field/field.h"*

*class Ability {*

*public:*

*virtual ~Ability() = default;*

*virtual void apply(Field& field) const = 0;*

*virtual void printAbility() const = 0;*

*};*

*#endif*

ship\_manager.h: *#ifndef SHIP\_MANAGER\_H*

*#define SHIP\_MANAGER\_H*

*#include <vector>*

*#include "../../entities/ship/ship.h"*

*class ShipManager {*

*public:*

*ShipManager(const std::vector<ShipSize>& shipSizes);*

*Ship& getShip(int index);*

*int getShipCount() const;*

*void hitShipSegment(Ship& ship, int segmentIndex);*

*private:*

*std::vector<Ship> ships\_;*

*};*

*#endif // SHIP\_MANAGER\_H*

double\_damage.cpp

*#include "./double\_damage\_ability.h"*

*#include <limits>*

*void DoubleDamage::apply(Field& field) const {*

*int x, y;*

*std::cout << "Нужны координаты для абилки\n";*

*std::cin >> x >> y;*

*if (std::cin.fail()) {*

*std::cerr << "Неверный ввод" << std::endl;*

*std::cin.clear();*

*std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');*

*return;*

*}*

*field.attackCell(x, y, Attack::Double);*

*}*

*void DoubleDamage::printAbility() const {*

*std::cout << "Двойной урон\n";*

*}*

random\_damage.cpp

*#include "./random\_attack\_ability.h"*

*#include <algorithm> // для std::shuffle*

*void RandomAttack::apply(Field& field) const {*

*auto shipCells = field.getNonEmptyCells();*

*if (shipCells.empty()) {*

*return;*

*}*

*std::random\_device rd;*

*std::mt19937 gen(rd());*

*std::shuffle(shipCells.begin(), shipCells.end(), gen);*

*auto [randomX, randomY] = shipCells.front();*

*try {*

*field.hitCell(randomX, randomY);*

*} catch (AttackOutOfRangeException& e) {*

*std::cout << e.what() << std::endl;*

*}*

*}*

*void RandomAttack::printAbility() const {*

*std::cout << "Случайная атака по кораблю\n";*

*}*

scanner\_ability.cpp: *#include "./scanner\_ability.h"*

*#include <limits>*

*void Scanner::apply(Field& field) const {*

*int x, y;*

*std::cout << "Нужны координаты для абилки\n";*

*std::cin >> x >> y;*

*if (std::cin.fail()) {*

*std::cerr << "Неверный ввод" << std::endl;*

*std::cin.clear();*

*std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');*

*return;*

*}*

*if (field.checkArea(x, y)) {*

*std::cout << "В заданной области находится целый корабль\n";*

*} else {*

*std::cout << "Данная область пуста\n";*

*}*

*}*

*void Scanner::printAbility() const {*

*std::cout << "Сканирование области\n";*

*}*

ability\_manager.cpp

*#include "./ability\_manager.h"*

*#include <algorithm>*

*#include <random>*

*#include "exceptions.h"*

*AbilityManager::AbilityManager() {*

*abilities\_.push\_back(std::make\_unique<DoubleDamage>());*

*abilities\_.push\_back(std::make\_unique<Scanner>());*

*abilities\_.push\_back(std::make\_unique<RandomAttack>());*

*const unsigned seed = std::chrono::system\_clock::now().time\_since\_epoch().count();*

*std::default\_random\_engine rng(seed);*

*std::shuffle(abilities\_.begin(), abilities\_.end(), rng);*

*}*

*std::unique\_ptr<Ability> AbilityManager::getAbility() {*

*if (abilities\_.empty()) {*

*throw EmptyAbilityException();*

*}*

*std::unique\_ptr<Ability> ability = std::move(abilities\_.front());*

*abilities\_.erase(abilities\_.begin());*

*return ability;*

*}*

*void AbilityManager::addAbility() {*

*std::cout << "Добавляем абилку\n";*

*using AbilityCreator = std::function<std::unique\_ptr<Ability>()>;*

*static const std::vector<AbilityCreator> allAbilities = {*

*[]() { return std::make\_unique<DoubleDamage>(); },*

*[]() { return std::make\_unique<Scanner>(); },*

*[]() { return std::make\_unique<RandomAttack>(); }*

*};*

*std::random\_device rd;*

*std::mt19937 gen(rd());*

*std::uniform\_int\_distribution<> dis(0, allAbilities.size() - 1);*

*int randomIndex = dis(gen);*

*abilities\_.push\_back(allAbilities[randomIndex]());*

*}*

*void AbilityManager::printAbilities() const {*

*if (this->abilities\_.empty()) {*

*std::cout << "Список способностей пуст\n";*

*return;*

*}*

*std::cout << "Список способностей:\n\n";*

*for (int i = 0; i < this->abilities\_.size(); i++) {*

*std::cout << i << ": ";*

*this->abilities\_[i].get()->printAbility();*

*}*

*std::cout << "\n\n";*

*}*

Дерево проекта:

*├── entities*

*│ ├── field*

*│ │ ├── field.cpp*

*│ │ └── field.h*

*│ └── ship*

*│ ├── ship.cpp*

*│ └── ship.h*

*├── main.cpp*

*└── managers*

*└── ship\_manager*

*├── ship\_manager.cpp*

*└── ship\_manager.h*

CmakeLists.txt:

*cmake\_minimum\_required(VERSION 3.8)*

*project(WorldOfShips CXX)*

*set(CMAKE\_CXX\_STANDARD 11)*

*set(CMAKE\_CXX\_STANDARD\_REQUIRED True)*

*set(SOURCES*

*src/entities/ship/ship.cpp*

*src/managers/ship\_manager/ship\_manager.cpp*

*src/entities/field/field.cpp*

*src/main.cpp*

*)*

*add\_executable(Game ${SOURCES})*