**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. 3382 |  | Яковлев Д. С. |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2024

## Цель работы

Научиться использовать принципы ООП на практике: реализовать класс-интерфейс: *Spell*, и через наследование создать 3 способности: двойной урон, сканер и обстрел, соответственно: *DoubleDamage*, *Scan*, *Bombing*. Также создать для них менеджер: *SpellsManager* с очередью способностей и набор классов исключений.

## Задание

## 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 2х2 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.

## Создать класс менеджер-способностей. Который хранит очередь способностей, изначально игроку доступно по 1 способности в случайном порядке. Реализовать метод применения способности.

## Реализовать функционал получения одной случайной способности при уничтожении вражеского корабля.

## 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

Описание архитектуры лабораторной работы: (см. приложение A, полная архитектура в приложении C).

В начале был создан класс способности с виртуальным методом *cast\_spell* для последующего переопределения (создания дочерних классов). Были созданы способности (двойной урон производит атаку с уроном 2, сканирование ищет ячейки кораблей в области 2x2 вниз и направо от переданных координат, обстрел атакует случайную живую ячейку случайного корабля с уроном 1). После этого создадим менеджер способностей. Он хранит в себе очередь имеющихся способностей.

Методы класса *SpellsManager*:

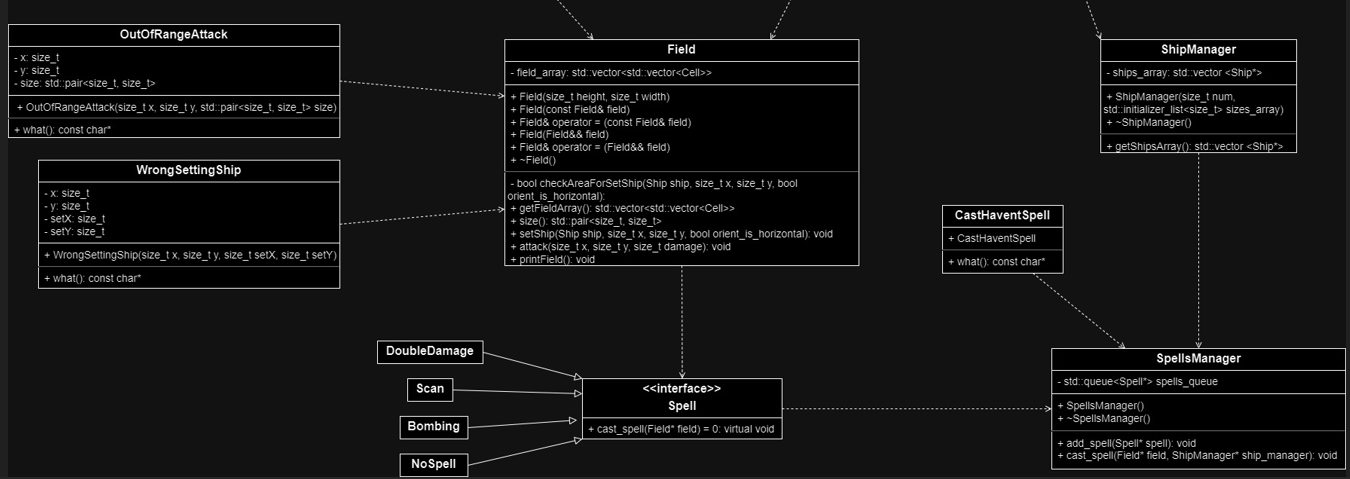
1. Конструктор: с помощью *rand()* случайно создаёт очередь из 3 существующих способностей.
2. *cast\_spell*: спрашивает у пользователя: хочет ли он использовать способность, если да, и в очереди есть способность, то сначала применяет способность, а потом производит атаку, если нет, то просто производит атаку.
3. *add\_spell*: добавляет переданную способность в очередь.
4. Деструктор: очищает очередь способностей, если она не пуста.

Далее были созданы классы исключения. Они выбрасываются согласно условиям в *Field* и *SpellsManager* соответственно.

## Выводы

Была разработана описанная выше архитектура. На практике была усвоена работа с полиморфизмом и классами-исключениями. Написанный исходный код можно посмотреть в приложении B.

# ПРИЛОЖЕНИЕ A **UML-ДИАГРАММА КЛАССОВ**



# ПРИЛОЖЕНИЕ B **ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: spells.h

#pragma once

#include <iostream>

#include "field.h"

*class* Spell{

*public:*

        virtual *void* cast\_spell(Field\* *field*) = 0;

};

*class* DoubleDamage: *public* Spell{

*public:*

*void* cast\_spell(Field\* *field*) override;

};

*class* Scan: *public* Spell{

*public:*

*void* cast\_spell(Field\* *field*) override;

};

*class* Bombing: *public* Spell{

*public:*

*void* cast\_spell(Field\* *field*) override;

};

*class* NoSpell: *public* Spell{

*public:*

*void* cast\_spell(Field\* *field*) override;

};

Название файла: spells.cpp

#include "spells.h"

*void* *DoubleDamage*::cast\_spell(*Field*\* *field*){

*size\_t* x, y;

    std::cout << "Enter the coordinates of the double attack:\n";

    std::cin >> x >> y;

    std::cout << '\n';

*field*->attack(x, y, 2);

}

*void* *Scan*::cast\_spell(*Field*\* *field*){

*size\_t* x, y;

    std::cout << "Enter the scan coordinates:\n";

    std::cin >> x >> y;

    std::cout << '\n';

    std::*vector*<std::*vector*<*Cell*>> field\_array = *field*->getFieldArray();

    if(x + 1 < *field*->size().second && y + 1 < *field*->size().first &&

    (field\_array[x][y].isShipCell() || field\_array[x+1][y].isShipCell() || field\_array[x][y+1].isShipCell() || field\_array[x+1][y+1].isShipCell())){

        std::cout << "There are ships in this area.\n";

    } else{

        std::cout << "There are no ships in this area.\n";

    }

    std::cout << "Enter the coordinates of the attack:\n";

    std::cin >> x >> y;

    std::cout << '\n';

*field*->attack(x, y, 1);

}

*void* *Bombing*::cast\_spell(*Field*\* *field*){

    std::*vector*<*ShipCell*\*> ship\_cell\_array;

    std::*vector*<std::*vector*<*Cell*>> field\_array = *field*->getFieldArray();

    for(*size\_t* i = 0; i < *field*->size().first; i++){

        for(*size\_t* j = 0; j < *field*->size().second; j++){

            if(field\_array[i][j].isShipCell() && field\_array[i][j].getShipCell()->getHealth() != 0){

                ship\_cell\_array.push\_back(field\_array[i][j].getShipCell());

            }

        }

    }

    if(ship\_cell\_array.size() != 0){

*size\_t* index = rand() % ship\_cell\_array.size();

        ship\_cell\_array[index]->attack(1);

        std::cout << "A random segment of a random ship was attacked.\n";

    }

*size\_t* x, y;

    std::cout << "Enter the coordinates of the attack:\n";

    std::cin >> x >> y;

    std::cout << '\n';

*field*->attack(x, y, 1);

}

*void* *NoSpell*::cast\_spell(*Field*\* *field*){

*size\_t* x, y;

    std::cout << "Enter the coordinates of the attack:\n";

    std::cin >> x >> y;

    std::cout << '\n';

*field*->attack(x, y, 1);

}

Название файла: spellsManager.h

#pragma once

#include <iostream>

#include <queue>

#include "spells.h"

#include "shipManager.h"

#include "spellsManagerExceptions.h"

*class* *SpellsManager*{

*private:*

        std::*queue*<*Spell*\*> spells\_queue;

*public:*

        SpellsManager();

*void* add\_spell(*Spell*\* *spell*);

*void* cast\_spell(*Field*\* *field*, *ShipManager*\* *ship\_manager*);

        ~SpellsManager();

};

Название файла: spellsManager.cpp

#include "spellsManager.h"

*SpellsManager*::SpellsManager(){

*size\_t* n = rand();

    if(n%3 == 1){

        add\_spell(new *Scan*);

        if(n%2 == 1){

            add\_spell(new *Bombing*);

            add\_spell(new *DoubleDamage*);

        } else{

            add\_spell(new *DoubleDamage*);

            add\_spell(new *Bombing*);

        }

    } else if(n%3 == 2){

        add\_spell(new *Bombing*);

        if(n%2 == 1){

            add\_spell(new *Scan*);

            add\_spell(new *DoubleDamage*);

        } else{

            add\_spell(new *DoubleDamage*);

            add\_spell(new *Scan*);

        }

    } else{

        add\_spell(new *DoubleDamage*);

        if(n%2 == 1){

            add\_spell(new *Bombing*);

            add\_spell(new *Scan*);

        } else{

            add\_spell(new *Scan*);

            add\_spell(new *Bombing*);

        }

    }

}

*bool* isChanged(std::*vector*<*bool*> *a*, std::*vector*<*bool*> *b*){

        for(*size\_t* i = 0; i < *a*.size(); i++){

            if(*a*[i] != *b*[i]){

                return true;

            }

        }

        return false;

    }

*void* *SpellsManager*::cast\_spell(*Field*\* *field*, *ShipManager*\* *ship\_manager*){

*char* flag = '1';

    std::*vector*<*bool*> start\_array, finish\_array;

    for(*size\_t* i = 0; i < *ship\_manager*->getShipsArray().size(); i++){

        start\_array.push\_back(*ship\_manager*->getShipsArray()[i]->isShipDestroyed());

    }

    while(flag != 'n' && flag != 'y'){

        std::cout << "Do you want to use the ability when attacking? (y - yes/ n - no)\n";

        std::cin >> flag;

        std::cout << '\n';

    }

    if(flag == 'n'){

*Spell*\* temp = new *NoSpell*;

        temp->cast\_spell(*field*);

        for(*size\_t* i = 0; i < *ship\_manager*->getShipsArray().size(); i++){

            finish\_array.push\_back(*ship\_manager*->getShipsArray()[i]->isShipDestroyed());

        }

        if(isChanged(start\_array, finish\_array)){

*size\_t* n = rand();

            if(n%3 == 1){

                add\_spell(new *Scan*);

            } else if(n%3 == 2){

                add\_spell(new *Bombing*);

            } else{

                add\_spell(new *DoubleDamage*);

            }

        }

    } else if(flag == 'y' && !spells\_queue.empty()){

        spells\_queue.front()->cast\_spell(*field*);

        spells\_queue.pop();

        for(*size\_t* i = 0; i < *ship\_manager*->getShipsArray().size(); i++){

            finish\_array.push\_back(*ship\_manager*->getShipsArray()[i]->isShipDestroyed());

        }

        if(isChanged(start\_array, finish\_array)){

*size\_t* n = rand();

            if(n%3 == 1){

                add\_spell(new *Scan*);

            } else if(n%3 == 2){

                add\_spell(new *Bombing*);

            } else{

                add\_spell(new *DoubleDamage*);

            }

        }

    } else{

        throw *CastHaventSpell*();

    }

}

*void* *SpellsManager*::add\_spell(*Spell*\* *new\_spell*){

    spells\_queue.push(*new\_spell*);

}

*SpellsManager*::~SpellsManager(){

    while(!spells\_queue.empty()){

        spells\_queue.pop();

    }

}

Название файла: fieldExceptions.h

#pragma once

#include <iostream>

*class* *OutOfRangeAttack*{

*private:*

*size\_t* x, y;

    std::*pair*<*size\_t*, *size\_t*> size;

*public:*

    OutOfRangeAttack(*size\_t* *x*, *size\_t* *y*, std::*pair*<*size\_t*, *size\_t*> *size*);

    const *char*\* what();

};

*class* *WrongSettingShip*{

*private:*

*size\_t* x, y, setX, setY;

*public:*

    WrongSettingShip(*size\_t* *x*, *size\_t* *y*, *size\_t* *setX*, *size\_t* *setY*);

    const *char*\* what();

};

Название файла: fieldExceptions.cpp

#include "fieldExceptions.h"

*OutOfRangeAttack*::OutOfRangeAttack(*size\_t* *x*, *size\_t* *y*, std::*pair*<*size\_t*, *size\_t*> *size*)

:x(*x*), y(*y*), size(*size*){}

const *char*\* *OutOfRangeAttack*::what(){

    std::*string* message = "Size of field is " + std::to\_string(size.second) + ":" + std::to\_string(size.first) + ".\n";

    message += "The x coordinate must be less than " + std::to\_string(size.second) + ", now: " + std::to\_string(x) + ".\n";

    message += "The y coordinate must be less than " + std::to\_string(size.first) + ", now: " + std::to\_string(y) + ".\n";

    return message.c\_str();

}

*WrongSettingShip*::WrongSettingShip(*size\_t* *x*, *size\_t* *y*, *size\_t* *setX*, *size\_t* *setY*)

:x(*x*), y(*y*), setX(*setX*), setY(*setY*){}

const *char*\* *WrongSettingShip*::what(){

    std::*string* message = "It is impossible to place the ship according to the coordinates (" + std::to\_string(setX) + ", " + std::to\_string(setY) + ").\n";

    message += "Another ship is already located at the coordinates (" + std::to\_string(setX) + ", " + std::to\_string(setY) + ").\n";

    return message.c\_str();

}

Название файла: spellsManagerExceptions.h

#pragma once

#include <iostream>

*class* *CastHaventSpell*{

*public:*

    CastHaventSpell();

    const *char*\* what();

};

Название файла: spellsManagerExceptions.cpp

#include "spellsManagerExceptions.h"

*CastHaventSpell*::CastHaventSpell(){}

const *char*\* *CastHaventSpell*::what(){

    std::*string* message = "You don't have any spell now.\n";

    return message.c\_str();

}

# ПРИЛОЖЕНИЕ C **UML-ДИАГРАММА КЛАССОВ**

