**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «ООП»**

**Тема: Полиморфизм**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Песчатский С. Д |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2024

## 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

Было добавлено 6 новых классов, а также были изменены старые.

Класс Spell – класс интерфейс для способностей

Методы:

* 1. void cast(Map\*, Manager\*) – использование способности
  2. void saySpell() – выводит название способности

Класс SpellManager – менеджер способностей

Поля:

* 1. std::vector<std::unique\_ptr<Spell>> queue – очередь способностей

Методы:

* 1. void addSpell() – добавляет способность в очередь
  2. void showSpells() – выводит очередь способностей
  3. void nextSpell() – выводит название следующей способности
  4. void useSpell(Map\* cords, Manager\* forThisToWorkProperly) – использование следующей способности в очереди
  5. void regShot(Map\* cords, Manager\* forThisToWorkProperly) – метод для обычного выстрела

Класс DoubleDamage – способность двойного урона, наследована от Spell

Методы:

* 1. void cast(Map\*, Manager\*) – использование способности
  2. void saySpell() – выводит название способности

Класс Scanner – способность сканнера, наследована от Spell

Методы:

* 1. void cast(Map\*, Manager\*) – использование способности
  2. void saySpell() – выводит название способности

Класс Barrage – способность случайной атаки, наследована от Spell

Методы:

* 1. void cast(Map\*, Manager\*) – использование способности
  2. void saySpell() – выводит название способности

Класс NoSpells – наследован от std::exception, используется когда в очереди нет заклинаний

Поля:

* 1. std::string Error – сообщение об ошибке

Методы:

1) explicit NoSpells() – конструктор класса

* 1. const char\* what() const noexcept override – возвращает сообщение об ошибке

Класс ShotOOB – наследован от std::exception, используется когда выстрел выходит за границы поля

Поля:

* 1. std::string Error – сообщение об ошибке

Методы:

1) explicit NoSpells() – конструктор класса

* 1. const char\* what() const noexcept override – возвращает сообщение об ошибке

Класс ImproperShipPlacement – наследован от std::exception, используется когда положение корабля недопустимо

Поля:

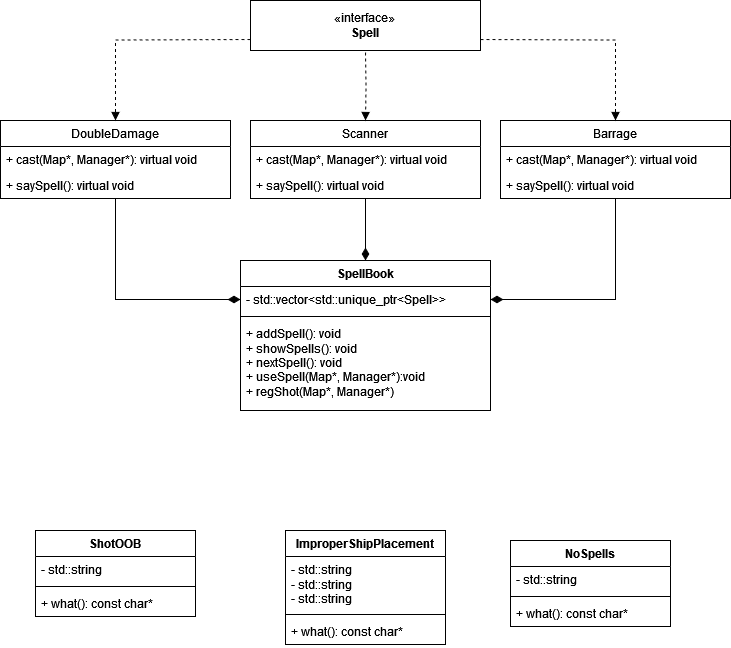
* 1. std::string Error – сообщение об ошибке
  2. std::string first – первая часть сообщения
  3. std::string second – вторая часть сообщения

Методы:

1) explicit NoSpells() – конструктор класса

* 1. const char\* what() const noexcept override – возвращает сообщение об ошибке

int main содержится только создания объектов каждого класса и вызов методов для проверки правильности работы программы.



## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| Проверяемые методы | Входные данные | Выходные данные |
| Инициализация поля | 10 5 | wwwwwwwwww  wwwwwwwwww  wwwwwwwwww  wwwwwwwwww  wwwwwwwwww |
| Расстановка двух кораблей, один из которых заведомо расположен неверно | 2 4 r 0 3  3 d 3 0 | Ship number: 2 has incorrect position  wwwwwwwwww  wwwwwwwwww  wwwwwwwwww  1111wwwwww  wwwwwwwwww |
| Использование способности | 0 0 | AREA IS SCANNED  SHIP DETECTED |
| Выстрел в пустую ячейку | 5 2 | wwwwwwwwww  wwwwwwwwww  wwwww0wwww  1111wwwwww  wwwwwwwwww |

## Выводы

Были изучены способы использования классов, а также их методов и полей. Были созданы Ship, Manager, Map, три класса, наследующиеся от класса-интерфейса Spell: DoubleDamage, Scanner, Barrage; и три класса, наследующиеся от класса исключения: NoSpells, ImproperShipPlacement, ShotOOB. Так же были реализованы способы применения способностей и взаимодействия классов между собой.

# Приложение А Исходный код программы

Название файла: main.cpp

#include <iostream>

#include "SpellBook.h"

int main(){

    Manager manage;

    SpellBook Skills = SpellBook();

    Skills.showSpells();

    int size[2];

    for(int i=0; i<2; i++){std::cin>>size[i];}

    Map area(size);

    area.Show();

    std::cout<<"number//len//ori//pos\n";

    manage.get\_Fleet();

    area.PlaceFleet(&manage);

    area.Show();

    manage.show\_Fleet();

    Skills.nextSpell();

    Skills.useSpell(&area, &manage);

    Skills.regShot(&area, &manage);

    area.Show();

}

Название файла: Ship.cpp

#include "Ship.h"

Ship::Ship(int len){

    for (int i = 0; i<len; i++){

        segHp.push\_back(untouched);

    }

}

bool Ship::damage(int index){

        if(segHp.at(index)==untouched){segHp.at(index)=damaged;}

        if(segHp.at(index)==damaged){segHp.at(index)=destroyed;}

        for(int i=0; i<segHp.size(); i++){

            if(segHp.at(i)!=destroyed){

                return 0;

            }

        }

        if(cond==1){return 0;}

        cond=1;

        return cond;

    }

void Ship::showShip(){

    if (cond){std::cout<<"alive"<<" len-"<<std::to\_string(segHp.size())<<" position-";}

    else{std::cout<<"dead"<<" len-"<<std::to\_string(segHp.size())<<" position-";}

    }

int Ship::getSize(){

        return segHp.size();

    }

bool Ship::isDead(){return cond;}

Ship::~Ship(){}

Название файла: Manager.cpp

#include "Manager.h"

Manager::Manager(){}

void Manager::get\_Fleet(){

    int number\_of\_ships; std::cin>>number\_of\_ships;

    for(int i=0; i<number\_of\_ships; i++){

        int tempLen; char tempCord; std::cin>>tempLen>>tempCord;

        int tempPos[2];

        for(int j=0; j<2; j++){std::cin>>tempPos[j];}

            ShipInfo info(tempPos[0], tempPos[1]);

            if(tempCord=='d'){

                info.Set\_ori(0);

            }

            else if (tempCord=='r')

            {

                info.Set\_ori(1);

            }else{

                std::cout<<"ERROR";

            }

            Report.push\_back(info);

            Ship newborn=Ship(tempLen);

            Fleet.push\_back(newborn);

    }

}

void Manager::show\_Fleet(){

    for(int i=0; i<Fleet.size(); i++){

        std::cout<<"Stats of ship number "<<i+1<<": ";Fleet.at(i).showShip();

        std::cout<<Report[i].x<<"."<<Report[i].y<<" orientation-";

        if(Report[i].ori==0){std::cout<<"vertical";}

        else{std::cout<<"horizontal";}

        std::cout<<'\n';

    }

}

bool Manager::Hit(int \* coords){

    for(int i=0; i<Fleet.size(); i++){

        for(int j=0; j<Fleet.at(i).getSize(); j++){

            if(Report[i].ori){

                if(Report[i].x+j==coords[0] and Report[i].y==coords[1]){

                    bool tmp = Fleet.at(i).damage(j);

                    if(Fleet.at(i).isDead() and tmp){

                        return 1;

                    }

                }

            }

            else{

                if(Report[i].y+j==coords[1] and Report[i].x==coords[0]){

                    bool tmp = Fleet.at(i).damage(j);

                    if(Fleet.at(i).isDead() and tmp){

                        return 1;

                    }

                }

            }

        }

    }

    return 0;

}

std::vector <Ship>& Manager::send\_Fleet(){

    return Fleet;

}

std::vector <ShipInfo>& Manager::send\_Coordinates(){

    return Report;

}

void Manager::delete\_ship(int index){

    Fleet.erase(Fleet.begin()+index);

    Report.erase(Report.begin()+index);

}

Manager::~Manager(){}

Название файла: Map.cpp

#include "Map.h"

        Map::Map(const int \* cors){

            size=new int[2];

            size[0]=cors[0];

            size[1]=cors[1];

            std::vector<Sector> tmp;

            for(int j=0; j < cors[0]; j++){

                for(int i = 0; i < cors[1]; i++){

                    tmp.push\_back(unknown);

                }

                field.push\_back(tmp);

                tmp.clear();

            }

        }

        void Map::Show(){

            for(int i=0; i<size[1]; i++){

                for(int j=0; j<size[0]; j++){

                    if(field.at(j).at(i)==unknown){std::cout<<" \*";}

                    if(field.at(j).at(i)==ship){std::cout<<" 1";}

                    if(field.at(j).at(i)==blank){std::cout<<" 0";}

                }

                std::cout<<'\n';

            }

        }

        bool Map::CheckShip(ShipInfo boat, int length){

            int X=boat.x;

            int Y=boat.y;

            for(int i=0; i<length; i++){

                if(boat.ori){

                    if (i+X>=size[0] or i+X<0 or Y>=size[1] or Y<0){

                        return false;

                    }

                    if (0<=X+1+i and X+1+i<size[0]){

                        if(field[X+1+i][Y]==ship){return false;}}

                    if (i==0){

                        if(X-1>=0){

                            if(field[X-1][Y]==ship){return false;}}}

                    if (0<=X-1 and X-1<size[0]){

                        if(field[X+i][Y-1]==ship){return false;}}

                    if (0<=X+1 and X+1<size[0]){

                        if(field[X+i][Y+1]==ship){return false;}}

                }

                else{

                    if (i+Y>=size[1] or i+Y<0 or X>=size[0] or X<0){

                        return false;

                    }

                    if (0<=Y+1+i and Y+1+i<size[1]){

                        if(field.at(X).at(Y+i+1)==ship){return false;}}

                    if (i==0){

                        if(X-1>=0){

                            if(field.at(X)[Y-1]==ship){return false;}}}

                    if (0<=X-1 and X-1<size[0]){

                        if(field[X-1][Y+i]==ship){return false;}}

                    if (0<=X+1 and X+1<size[0]){

                        if(field[X+1][Y+i]==ship){return false;}}

                }

            }

            return true;

        }

        void Map::PlaceFleet(Manager\* manage){

            std::vector <Ship>& Fleet = manage->send\_Fleet();

            std::vector <ShipInfo>& Report = manage->send\_Coordinates();

            int k =0;

            for(int i=0; i<Fleet.size();i++ ){

                k++;

                try{

                if(CheckShip(Report[i], Fleet.at(i).getSize())){

                if(Report[i].ori){

                    int j=Report[i].y;

                    for(int k=0; k<Fleet.at(i).getSize() and 0<=k<size[0] and 0<=j<size[1]; k++){

                        field[k+Report[i].x][j]=ship;

                    }

                }

                else{

                    int j=Report[i].x;

                    for(int k=0; k<Fleet.at(i).getSize() and 0<=k<size[0] and 0<=j<size[1]; k++){

                        field[j][k+Report[i].y]=ship;

                    }

                }

                }

                else{

                    manage->delete\_ship(i);

                    i--;

                    throw ImproperShipPlacement(k);

                }

                }

                catch (const ImproperShipPlacement& e){

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

                }

            }

        }

        bool Map::SectorChecker(int \*cords){

            if (field[cords[0]][cords[1]]==ship){

                return true;

            }

            else{

                return false;

            }

        }

        void Map::ActivateDD(){

            DD=1;

        }

        bool Map::Hit(int \*coords, Manager\* manage){

            try{

            if((0<=coords[0] and coords[0]<size[0]) and (0<=coords[1] and coords[1]<size[1])){

                if(field[coords[0]][coords[1]]==ship){

                    if(manage->Hit(coords)){

                        if (DD){manage->Hit(coords);DD=0;}

                        return 1;

                    }

                }

                if(field[coords[0]][coords[1]]==unknown){

                    field[coords[0]][coords[1]]=blank;

                }

            }

            else{throw ShotOOB();}

            }

            catch(const ShotOOB& e){

                std::cerr<<e.what()<<std::endl;

            }

            return 0;

        }

        Map& Map::operator=(const Map& source){

            if(this==&source){

                return \*this;

            }

            size=source.size;

            if (source.field.empty()){

                std::vector<std::vector<Sector>> field;

                std::vector<Sector> tmp;

                for(int j=0; j < size[0]; j++){

                    for(int i = 0; i < size[1]; i++){

                        tmp.push\_back(source.field[j][i]);

                    }

                    field.push\_back(tmp);

                }

            }

    return \*this;

}

int \* Map::getSize(){

    return size;

}

Map& Map::operator=(Map&& moved){

    if(this==&moved){

        return \*this;

    }

    field=moved.field;

    size=moved.size;

    return \*this;

}/\*

Map::Map(const Map& source) : size(source.size), field(source.field) {}

Map::Map(Map&& moved) noexcept: size(moved.size), field(moved.field) {}\*/

Map::~Map(){}

Название файла: DoubleDamage.cpp:

#include "DoubleDamage.h"

void DoubleDamage::cast(Map\* cords, Manager\* forThisToWorkProperly){

        std::cout<<"DAMAGE INCREASED";

        cords->ActivateDD();

    }

void DoubleDamage::saySpell(){std::cout<<"DoubleDamage\n";}

Название файла: Scanner.cpp:

#include "Scanner.h"

void Scanner::cast(Map\* cords, Manager\* forThisToWorkProperly){

    int shot[2];std::cin>>shot[0]>>shot[1];

    int \*tmp = cords->getSize();

    if(shot[0]<0){shot[0]=0;}

    if(shot[0]>=tmp[0]){shot[0]=tmp[0]-2;}

    if(shot[1]<0){shot[1]=0;}

    if(shot[1]>tmp[1]){shot[1]=tmp[1]-2;}

    int tmp1[2]; tmp1[0]=shot[0]+1;

    int tmp2[2]; tmp2[1]=shot[1]+1;

    int tmp3[2]; tmp3[0]=shot[0]+1; tmp3[1]=shot[1]+1;

    std::cout<<"AREA IS SCANNED\n";

    if(cords->SectorChecker(shot) || cords->SectorChecker(tmp1) || cords->SectorChecker(tmp2) || cords->SectorChecker(tmp3)){

        std::cout<<"SHIP DETECTED\n";

    }

    else{

        std::cout<<"NO SHIP";

    }

}

void Scanner::saySpell(){std::cout<<"Scanner\n";}

Название файла: Barrage.cpp:

#include "Barrage.h"

void Barrage::cast(Map\* cords, Manager\* forThisToWorkProperly){

    std::vector<Ship> tmp = forThisToWorkProperly->send\_Fleet();

    if(tmp.size()>0){

        int a = rand()%tmp.size(); int b = rand()%tmp.at(a).getSize();

        tmp.at(a).damage(b);

    }

    std::cout<<"RANDOM DAMAGE\n";

}

void Barrage::saySpell(){std::cout<<"Barrage\n";}

Название файла: NoSpells.cpp:

#include "NoSpells.h"

    NoSpells::NoSpells(){}

    const char\* NoSpells::what() const noexcept{

        return Error.c\_str();

    }

Название файла: ImproperShipPlacement.cpp:

#include "ImproperShipPlacement.h"

        ImproperShipPlacement::ImproperShipPlacement(int index){

        Error.append(first);

        Error.append(std::to\_string(index));

        Error.append(second);

        }

    const char\* ImproperShipPlacement::what() const noexcept{

        return Error.c\_str();

    }

Название файла: ShotOOB.cpp:

#include "ShotOOB.h"

    ShotOOB::ShotOOB(){}

    const char\* ShotOOB::what() const noexcept{

        return Error.c\_str();

    }