МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Объектно-ориентированное программирование»
Тема: "Добавление врагов"

Студент гр. 9383 _____ Корсунов А.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Написать программу в ООП стиле согласно заданию. Углубить знания об ООП, по возможности изучить предложенные паттерны.

Задание.

Создать шаблонный класс врага. Параметр шаблона должен определять поведение врага (параметров шаблона может быть несколько, например отдельный параметр для политики передвижения и для политики атаки). Класс врага должен препятствовать игроку. Класс игрока должен иметь возможность взаимодействовать с врагом и наоборот.

Обязательные требования:

- Создан шаблонный класс врага
- Создано не менее 3 типа поведения врагов
- Взаимодействие происходит через перегруженный оператор

Дополнительные требования:

- Передача хода между игроком и врагами происходит с использованием паттерна Состояния в классе игры

Выполнение работы.

Для выполнения работы использовалось библиотека SFML, предназначенная для работы с 2D графикой.

Был создан Шаблонный класс Opponents, для которого используются классы поведения (Behavior, Behavior_steal_people, Behavior_steal_alco, Behavior_to_start). Класс Behavior – абстрактный (определена чисто виртуальная функция, в которой перегружается унарный оператор '-');

Предложенные паттерны не используются.

Рисунок 1 — Пример работы программы (рыцарь — игрок, лестница — вход, ворота — выход, кролик — объект, который нужно спасти, грибы повышают здоровья, гоблины — отнимают здоровье, щит — добавляет брони, фляга — добавляет очки опьянения, плиты и колонны соответственно проходимые и непроходимые клетки, маги - враги).

```
Logs:
log: Игрок установлен в: 0, 0

log: Кролики установлены в: 3, 1

log: Кролики установлены в: 28, 1

log: Кролики установлены в: 1, 8

log: Гоблины установлены в: 6, 3
```

Рисунок 2 — Пример вывода логов в файл.

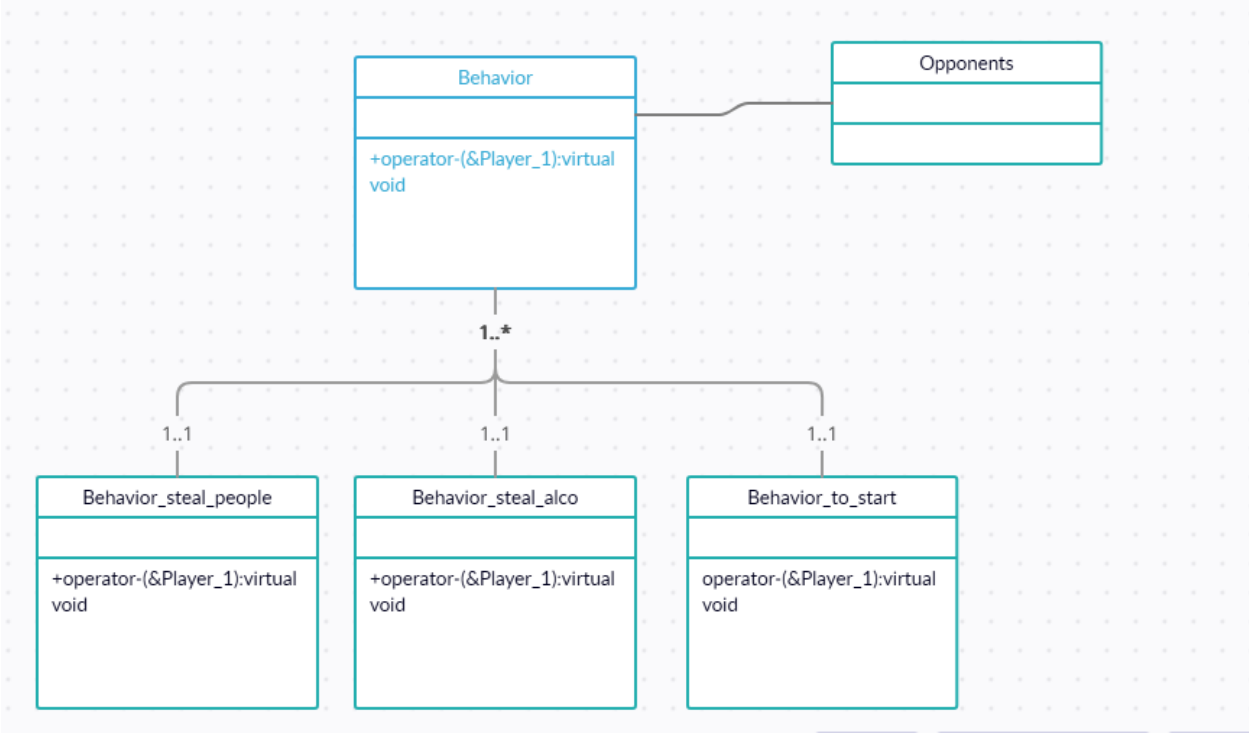

Рисунок 3

На рисунке 3 изображена UML-диаграмма новых классов, реализованных в данной работе.

Выводы.

Написана программа в ООП стиле согласно заданию. Углублены знания об ООП, реализованы некоторые из паттернов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

```
#include "Game_Manager.h"

int main()
{
	Game_Manager play;
	play.start_Game();
	play.draw_and_move();
	return 0;
}
```

Файл Cell.h

```
#pragma once
#include "1_Player.h"
#include "Element.h"
#include "Opponents.h"
#include "Behavior_people.h"
#include "Behavior_steal_alco.h"
#include "Behavior_to_start.h"
#include "Behavior.h"

class Cell
{
private:
   bool pass;
   bool out;
   bool in;
   bool player_1;
```

```
    bool people;
    bool alcogol;
    bool enemy;
    bool medicine;
    bool armor;
    bool b1;
    bool b2;
    bool b3;

    Element* element;
    Player_1* player_11;
    Opponents<Behavior>* opponents;
    Behavior_people* behavior_people;
    Behavior_steal_alco* behavior_steal_alco;
    Behavior_to_start* behavior_to_start;

public:
    Cell();
    ~Cell();

    void set_unpass(bool value);
    void set_out(bool value);
    void set_in(bool value);
    void set_player_1(bool value);
    void set_people(bool value);
    void set_alcogol(bool value);
    void set_enemy(bool value);
    void set_medicine(bool value);
    void set_armor(bool value);
```

```
    void set_b1(bool);
    void set_b2(bool);
    void set_b3(bool);

    bool get_pass();
    bool get_out();
    bool get_in();
    bool get_player_1();
    bool get_people();
    bool get_alcogol();
    bool get_enemy();
    bool get_medicine();
    bool get_armor();
    bool get_b1();
    bool get_b2();
    bool get_b3();
    void set_Element(Element* elem);
    void set_Player_1(Player_1* player);
    void set_Opponents(Opponents<Behavior>* opponents);
    void set_Behavior_people(Behavior_people*);
    void set_Behavior_steal_alco(Behavior_steal_alco*);
    void set_Behavior_to_start(Behavior_to_start*);
    Element* get_Element();
    Behavior_people* get_Behavior_people();
    Behavior_steal_alco* get_Behavior_steal_alco();
    Behavior_to_start* get_Behavior_to_start();
    Opponents<Behavior>* get_Opponents();
};
```

Файл Cell.cpp

```
#include "Cell.h"

Cell::Cell()
{
   this->pass = true;
   this->in = false;
   this->out = false;
   this->player_1 = false;
   this->element = nullptr;
   this->people = false;
   this->alcogol = false;
   this->armor = false;
   this->enemy = false;
   this->medicine = false;
   this->player_11 = nullptr;
   this->opponents = nullptr;
   this->behavior_people = nullptr;
   this->behavior_steal_alco = nullptr;
   this->behavior_to_start = nullptr;
   this->b1 = false;
   this->b2 = false;
   this->b3 = false;
}

Cell::~Cell(){};

void Cell::set_unpass(bool val)
{
   this->pass = val;
```

```
    this->in = false;
    this->out = false;
}

void Cell::set_in(bool val)
{
    this->pass = true;
    this->in = val;
    this->out = false;
}

void Cell::set_out(bool val)
{
    this->pass = true;
    this->in = false;
    this->out = val;
}

void Cell::set_player_1(bool val)
{
    this->player_1 = val;
}

void Cell::set_people(bool val)
{
    this->people = val;
}

void Cell::set_alcogol(bool val)
```

```
{
    this->alcogol = val;
}

void Cell::set_armor(bool val)
{
    this->armor = val;
}

void Cell::set_enemy(bool val)
{
    this->enemy = val;
}

void Cell::set_medicine(bool val)
{
    this->medicine = val;
}

bool Cell::get_pass()
{
    return this->pass;
}

bool Cell::get_in()
{
    return this->in;
}
```

```
bool Cell::get_out()
{
   return this->out;
}

bool Cell::get_player_1()
{
   return this->player_1;
}

void Cell::set_Element(Element* elem)
{
   this->element = elem;
}

void Cell::set_Player_1(Player_1* player)
{
   this->player_11 = player;
}

void Cell::set_Opponents(Opponents<Behavior>* opponents)
{
   this->opponents = opponents;
}

void Cell::set_Behavior_people(Behavior_people* behavior_people)
{
   this->behavior_people = behavior_people;
}
```

```
void Cell::set_Behavior_steal_alco(Behavior_steal_alco* behavior_steal_alco)
{
   this->behavior_steal_alco = behavior_steal_alco;
}

void Cell::set_Behavior_to_start(Behavior_to_start* behavior_to_start)
{
   this->behavior_to_start = behavior_to_start;
}

bool Cell::get_alcogol()
{
   return this->alcogol;
}

bool Cell::get_armor()
{
   return this->armor;
}

bool Cell::get_enemy()
{
   return this->enemy;
}

bool Cell::get_medicine()
{
```

```
    return this->medicine;
}

bool Cell::get_people()
{
    return this->people;
}

Element* Cell::get_Element()
{
    return this->element;
}

void Cell::set_b1(bool val)
{
    b1 = val;
}

void Cell::set_b2(bool val)
{
    b2 = val;
}

void Cell::set_b3(bool val)
{
    b3 = val;
}

bool Cell::get_b1()
```

```
{
   return b1;
}

bool Cell::get_b2()
{
   return b2;
}

bool Cell::get_b3()
{
   return b3;
}

Opponents<Behavior>* Cell::get_Opponents()
{
   return opponents;
}

Behavior_people* Cell::get_Behavior_people()
{
   return behavior_people;
}

Behavior_steal_alco* Cell::get_Behavior_steal_alco()
{
   return behavior_steal_alco;
}
```

```
Behavior_to_start* Cell::get_Behavior_to_start()
{
   return behavior_to_start;
}
```

Файл Field.h

```
#pragma once
#include "Cell.h"
#include "1_Player.h"

class Field
{
private:
   Cell** ptr = nullptr;
   int width, height;
   static Field* object;

   Field(int width, int height);
   ~Field();
   Field(const Field& ref_Field);
   Field& operator=(const Field& ref_Field);
   Field(Field&& ref_Field);
   Field& operator=(Field&& ref_Field);

public:
   static Field* get_Field(int x, int y);
   void In(int x, int y, bool val);
   void Out(int x, int y, bool val);
   void Unpass(int x, int y, bool val);
   void Player_1(int x, int y, bool val);
```

```
    void del_Player_1(int x, int y, bool val);

friend class Game_Manager;
friend class Iterator;
};
```

Файл Field.cpp

```
#include "Field.h"

Field* Field::object = nullptr;

Field::Field(int x, int y) : width(x), height(y)
{
    this->ptr = new Cell* [this->width];
    for (int i = 0; i < this->width; i++)
    {
        this->ptr[i] = new Cell [this->height];
    }
}

Field::~Field()
{
    for (int i = 0; i < this->width; i++)
    {
        delete[] this->ptr[i];
    }
    delete[] this->ptr;
}

Field* Field::get_Field(int x, int y)
```

```
{
   object = new Field(x, y);
   return object;
}

void Field::In(int x, int y, bool val)
{
   if (x >= 0 && x < this->width && y >= 0 && y < this->height)
   {
      this->ptr[x][y].set_in(val);
   }
}

void Field::Out(int x, int y, bool val)
{
   if (x >= 0 && x < this->width && y >= 0 && y < this->height)
   {
      this->ptr[x][y].set_out(val);
   }
}

void Field::Unpass(int x, int y, bool val)
{
    if (x >= 0 && x < this->width && y >= 0 && y < this->height)
    {
       this->ptr[x][y].set_unpass(0);
    }
}
```

```
void Field::Player_1(int x, int y, bool val)
{
   if (x >= 0 && x < this->width && y >= 0 && y < this->height)
   {
      this->ptr[x][y].set_player_1(1);
   }
}

void Field::del_Player_1(int x, int y, bool val)
{
   if (x >= 0 && x < this->width && y >= 0 && y < this->height)
   {
      this->ptr[x][y].set_player_1(0);
   }
}

Field::Field(const Field& ref_Field)
{
   this->width = ref_Field.width;
   this->height = ref_Field.height;
   this->ptr = new Cell* [ref_Field.width];
   for (int i = 0; i < ref_Field.width; i++)
   {
      this->ptr[i] = new Cell[ref_Field.height];
      for (int j = 0; j < ref_Field.height; j++)
      {
         this->ptr[i][j] = ref_Field.ptr[i][j];
      }
   }
```

```
}

Field& Field::operator=(const Field& ref_Field)
{
   if (&ref_Field == this)
   {
      return *this;
   }

   if (this != &ref_Field){
      for (int i = 0; i < this->width; i++)
      {
         delete[] this->ptr[i];
      }
      delete[] this->ptr;
   }

   this->width = ref_Field.width;
   this->height = ref_Field.height;
   this->ptr = new Cell* [ref_Field.width];
   for (int i = 0; i < ref_Field.width; i++)
   {
      this->ptr[i] = new Cell[ref_Field.height];
      for (int j = 0; j < ref_Field.height; j++)
      {
         this->ptr[i][j] = ref_Field.ptr[i][j];
      }
   }
   return *this;
```

```
}

Field::Field(Field&& ref_Field)
{
   this->ptr = ref_Field.ptr;
   this->width = ref_Field.width;
   this->height = ref_Field.height;
   ref_Field.ptr = nullptr;
   ref_Field.width = 0;
   ref_Field.height = 0;
}

Field& Field::operator=(Field&& ref_Field)
{
   if (&ref_Field == this)
   {
      return *this;
   }

   if (this != &ref_Field){
      for (int i = 0; i < this->width; i++)
      {
         delete[] this->ptr[i];
      }
      delete[] this->ptr;
   }

   this->ptr = ref_Field.ptr;
   this->width = ref_Field.width;
```

```
    this->height = ref_Field.height;
    ref_Field.ptr = nullptr;
    ref_Field.width = 0;
    ref_Field.height = 0;
    return *this;
}
```

Файл FieldIterator.h:

```
#pragma once
#include "Field.h"

class Iterator
{
private:
    int cell_x, cell_y;
    int width, height;
    const Field* field;

public:
    Iterator(const Field* f);
    Iterator(int i = 0, int j = 0);
    Iterator begin();
    Iterator end();

    void operator++();
    void operator--();
    bool operator==(const Iterator& field_2);
    bool operator!=(const Iterator& field_2);
    Cell& operator*();
```

```
    Cell& get_Cell();

    void next();
    void back();
    void up();
    void down();
    void left();
    void right();
};
```

Файл FieldIterator.cpp:

```
#include "FieldIterator.h"

Iterator::Iterator(const Field* f)
{
    this->field = f;
    for (int i = 0; i < f->width; i++)
    {
        for (int j = 0; j < f->height; j++)
        {
            this->cell_x = i;
            this->cell_y = j;
        }
    }
}

Iterator::Iterator(int i, int j)
{
```

```
    this->cell_x = i;
    this->cell_y = j;
}

Iterator Iterator::begin()
{
    for (int i = 0; i < this->width; i++)
    {
        for (int j = 0; j < this->height; j++)
        {
            if (this->field->ptr[i][j].get_in())
            {
                return Iterator(i,j);
            }
        }
    }
}

Iterator Iterator::end()
{
    for (int i = 0; i < this->width; i++)
    {
        for (int j = 0; j < this->height; j++)
        {
            if (this->field->ptr[i][j].get_out())
            {
                return Iterator(i,j);
            }
        }
```

```
    }
}

void Iterator::next()
{
    if ((this->cell_y + 1) == this->height && (this->cell_x + 1) == this->width)
    {
        return;
    }
    if ((this->cell_y + 1) < this->height)
    {
        this->cell_y++;
    }
    else
    {
        this->cell_x++;
    }
}

void Iterator::back()
{
    if ((this->cell_y + 1) == 1 && (this->cell_x + 1) == 1)
    {
        return;
    }
    if ((this->cell_y + 1) > 1)
    {
        this->cell_y--;
```

```
        }
        else
        {
            this->cell_x--;
        }
    }

    void Iterator::up()
    {
        if (this->cell_y > 0 && this->field->ptr[this->cell_x][this->cell_y - 1].get_pass())
        {
            this->cell_y--;
        }
    }

    void Iterator::down()
    {
        if (this->cell_y < this->height && this->field->ptr[this->cell_x][this->cell_y + 1].get_pass())
        {
            this->cell_y++;
        }
    }

    void Iterator::left()
    {
        if (this->cell_x > 0 && this->field->ptr[this->cell_x - 1][this->cell_y].get_pass())
```

```
        {
            this->cell_x--;
        }
    }

    void Iterator::right()
    {
        if (this->cell_x < this->width && this->field->ptr[this->cell_x + 1][this->cell_y].get_pass())
        {
            this->cell_x++;
        }
    }

    void Iterator::operator++()
    {
        this->next();
    }

    void Iterator::operator--()
    {
        this->back();
    }

    bool Iterator::operator==(const Iterator& field_2)
    {
        return this->cell_x == field_2.cell_x && this->cell_y == field_2.cell_y && this->field == field_2.field;
    }
```

```
bool Iterator::operator!=(const Iterator& field_2)
{
   return this->cell_x != field_2.cell_x || this->cell_y != field_2.cell_y || this->field != field_2.field;
}

Cell& Iterator::get_Cell()
{
   return Field::object->ptr[this->cell_x][this->cell_y];
}

Cell& Iterator::operator*()
{
   return this->get_Cell();
}
```

Файл 1_Player.h:

```
#pragma once

class Player_1
{
private:
   int hp;
   int pos_x, pos_y;
   int armor;
   int zomb;
   int alco;
   int saved_people;
```

```
public:
    Player_1();

    int get_hp();
    int get_pos_x();
    int get_pos_y();
    int get_armor();
    int get_zomb();
    int get_alco();
    int get_saved_people();

    void change_place(int, int);
    void steal_hp();
    void add_hp();
    void steal_armor();
    void add_armor();
    void steal_zomb();
    void add_zomb();
    void steal_alco();
    void add_alco();
    void add_saved_people();

        void restart();
};
```

Файл 1_Player.cpp:

```
#include "1_Player.h"
```

```
Player_1::Player_1()
{
   this->hp = 3;
   this->pos_x = 0;
   this->pos_y = 0;
   this->armor = 0;
   this->zomb = 0;
   this->alco = 0;
   this->saved_people = 0;
}

void Player_1::restart()
{
   this->hp = 3;
   this->pos_x = 0;
   this->pos_y = 0;
   this->armor = 0;
   this->zomb = 0;
   this->alco = 0;
   this->saved_people = 0;
}

int Player_1::get_hp()
{
   return this->hp;
}

int Player_1::get_pos_x()
```

```
{
   return this->pos_x;
}

int Player_1::get_pos_y()
{
   return this->pos_y;
}

int Player_1::get_armor()
{
   return this->armor;
}

int Player_1::get_zomb()
{
   return this->zomb;
}

int Player_1::get_alco()
{
   return this->alco;
}

int Player_1::get_saved_people()
{
   return this->saved_people;
}
```

```
void Player_1::change_place(int x, int y)
{
   this->pos_x = x;
   this->pos_y = y;
}

void Player_1::steal_hp()
{
   if (this->hp > 0)
   {
      this->hp--;
   }
}

void Player_1::add_hp()
{
   if (this->hp < 3)
   {
      this->hp++;
   }
}

void Player_1::steal_armor()
{
   if (this->armor > 0)
   {
      this->armor--;
   }
}
```

```
void Player_1::add_armor()
{
   if (this->armor < 3)
   {
      this->armor++;
   }
}

void Player_1::steal_zomb()
{
   if (this->zomb > 0)
   {
      this->zomb--;
   }
}

void Player_1::add_zomb()
{
   if (this->zomb < 5)
   {
      this->zomb++;
   }
}

void Player_1::steal_alco()
{
   if (this->alco > 0)
   {
```

```
        this->alco--;
    }
}

void Player_1::add_alco()
{
    if (this->alco < 3)
    {
        this->alco++;
    }
}

void Player_1::add_saved_people()
{
    if (this->saved_people < 3)
    {
        this->saved_people++;
    }
}
```

Файл Element.h:

```
#pragma once
#include "1_Player.h"

class Element
{
public:
    virtual void operator+(Player_1&) = 0;
};
```

Файл Element.cpp:

```
#include "Element.h"
```

Файл Medicine.h:

```
#pragma once
#include "Element.h"

class Medicine:public Element
{
public:
   void operator+(Player_1&);
};
```

Файл Medicine.cpp:

```
#include "Medicine.h"

void Medicine::operator+(Player_1& player_1)
{
   player_1.add_hp();
}
```

Файл Armor.h:

```
#pragma once
#include "Element.h"

class Armor:public Element
{
public:
```

```
    void operator+(Player_1&);
};
```

Файл Armor.cpp:

```
#include "Armor.h"

void Armor::operator+(Player_1& player_1)
{
    player_1.add_armor();
}
```

Файл People.h:

```
#pragma once
#include "Element.h"

class People:public Element
{
public:
    void operator+(Player_1&);
};
```

Файл People.cpp:

```
#include "People.h"

void People::operator+(Player_1& player_1)
{
    player_1.add_saved_people();
}
```

Файл Alcogol.h:

```
#pragma once
#include "Factory_Elements.h"
#include "Alcogol.h"

class Factory_Alcogol:public Factory_Element
{
public:
   Element* createElement();
};
```

Файл Alcogol.cpp:

```
#include "Factory_Alcogol.h"

Element* Factory_Alcogol::createElement()
{
   return new Alcogol;
}
```

Файд Factory_ Elements.h:

```
#pragma once
#include "Element.h"

class Factory_Element
{
public:
   virtual Element* createElement() = 0;
};
```

Файл Factory_Elements.cpp:

```
#include "Factory_Elements.h"
```

Файл Factory_Medicine.h:

```
#pragma once
#include "Factory_Elements.h"
#include "Medicine.h"

class Factory_Medicine:public Factory_Element
{
public:
   Element* createElement();
};
```

Файл Factory_Medicine.cpp:

```
#include "Factory_Medicine.h"

Element* Factory_Medicine::createElement()
{
   return new Medicine;
}
```

Файл Factory_People.h:

```
#pragma once
#include "Factory_Elements.h"
#include "People.h"
```

```
class Factory_People:public Factory_Element
{
public:
   Element* createElement();
};
```

Файл Factory_People.cpp:

```
#include "Factory_People.h"

Element* Factory_People::createElement()
{
   return new People;
}
```

Файл Factory_Enemy.h:

```
#pragma once
#include "Factory_Elements.h"
#include "Enemy.h"

class Factory_Enemy:public Factory_Element
{
public:
   Element* createElement();
};
```

Файл Factory_Enemy.cpp:

```
#include "Enemy.h"

void Enemy::operator+(Player_1& player_1)
```

```
{
   if (player_1.get_armor() == 0)
   {
      player_1.steal_hp();
   }
   else
   {
      player_1.steal_armor();
   }
}
```

Файл Factory_Armor.h:

```
#pragma once
#include "Factory_Elements.h"
#include "Armor.h"

class Factory_Armor:public Factory_Element
{
public:
   Element* createElement();
};
```

Файл Factory_Armor.cpp:

```
#include "Factory_Armoor.h"

Element* Factory_Armor::createElement()
{
   return new Armor;
}
```

Файл Factory_Alcogol.h:

```
#pragma once
#include "Factory_Elements.h"
#include "Alcogol.h"

class Factory_Alcogol:public Factory_Element
{
public:
   Element* createElement();
};
```

Файл Factory_Alcogol.cpp:

```
#include "Factory_Alcogol.h"

Element* Factory_Alcogol::createElement()
{
   return new Alcogol;
}
```

Файл Game_Manager.h:

```
#include "Field.h"
#include <SFML/Graphics.hpp>
#include "FieldIterator.h"
#include "1_Player.h"
#include "Factory_Alcogol.h"
#include "Factory_Armoor.h"
#include "Factory_Enemy.h"
```

```
#include "Factory_Medicine.h"
#include "Factory_People.h"
#include "Log_player.h"
#include "Log_print_file.h"
#include "Opponents.h"
#include "Behavior.h"
#include "Behavior_people.h"
#include "Behavior_steal_alco.h"
#include "Behavior_to_start.h"

class Game_Manager
{
public:
   void start_Game();
   void draw_and_move();

private:
   Field* field = Field::get_Field(30, 10);

   Behavior_people behavior_people;
   Behavior_steal_alco behavior_steal_alco;
   Behavior_to_start behavior_to_start;

   Player_1 man;
    Factory_People factory_people; //0 - условные номера для вызовов в функциях
   Factory_Alcogol factory_alcogol; //1
   Factory_Armor factory_armor; //2
   Factory_Enemy factory_enemy; //3
```

```
    Factory_Medicine factory_medicine; //4

    Log_print_file logs;
    Log_player logs_change;

    void set_Field();
    void set_Player_and_Elements_and_Logs();
    void set_Player_and_logs(int, int);
    void set_Elements_and_logs(int, int, int);
    void change_move_player(int, int);
    void change_move_player_help(int);
    void helper_draw(sf::RenderWindow, sf::Sprite);
    void helper_draw(sf::RenderWindow, sf::Sprite, int, int, int, int);
    void check_behavior();

    int w = 32;
    int x = 0;
    int y = 0;
};
```

Файл Game_Manager.cpp:

```
#include "Game_Manager.h"

using namespace sf;

void Game_Manager::start_Game()
{
    set_Field();
    set_Player_and_Elements_and_Logs();
```

```
}

void Game_Manager::draw_and_move()
{
        RenderWindow app(VideoMode(32 * field->width, 32 * (field->height+2)), "Cool game :)");
    Texture t;
    Sprite s(t);
    Font font;
    Text Player_1_info;
      t.loadFromFile("C:/Users/Eldorado/Documents/qwe/oop/govno/fantasy-tileset.png");
    font.loadFromFile("19849.ttf");
    Player_1_info.setFont(font);
    Player_1_info.setCharacterSize(20);
    Player_1_info.setFillColor(Color::Red);
    Player_1_info.setStyle(Text::Bold);
    Player_1_info.setPosition(Vector2f(0,(field->height)*32));

    while (app.isOpen())
    {
       app.clear();
       for (int i = 0; i < field->width; i++)
       {
          for (int j = 0; j < field->height; j++)
             {
                if (!field->ptr[i][j].get_in() && !field->ptr[i][j].get_out() && field->ptr[i][j].get_pass())
                   {
```

```
    s.setTextureRect(IntRect(0, 1 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
    //проходимая
}
if (field->ptr[i][j].get_in())
{
    s.setTextureRect(IntRect(5 * w, 1 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
    //вход
}
if (field->ptr[i][j].get_out())
{
    s.setTextureRect(IntRect(1 * w, 3 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
    //выход
}
if (!field->ptr[i][j].get_pass())
{
    s.setTextureRect(IntRect(0 * w, 3 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
    //непроходимая
}
if (field->ptr[i][j].get_player_1())
{
    s.setTextureRect(IntRect(5 * w, 18 * w, w, w));
```

```
    s.setPosition(i*w, j*w);
    app.draw(s);
  }
  if (field->ptr[i][j].get_people())
  {
    s.setTextureRect(IntRect(1 * w, 20 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
  }
  if (field->ptr[i][j].get_enemy())
  {
    s.setTextureRect(IntRect(0 * w, 18 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
  }
  if (field->ptr[i][j].get_medicine())
  {
    s.setTextureRect(IntRect(0 * w, 20 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
  }
  if (field->ptr[i][j].get_armor())
  {
    s.setTextureRect(IntRect(7 * w, 13 * w, w, w));
    s.setPosition(i*w, j*w);
    app.draw(s);
  }
  if (field->ptr[i][j].get_alcogol())
  {
```

```
                    s.setTextureRect(IntRect(6 * w, 5 * w, w, w));
                    s.setPosition(i*w, j*w);
                    app.draw(s);
                }
                  if (field->ptr[i][j].get_b1() || field->ptr[i][j].get_b2() || field->ptr[i][j].get_b3())
                {
                    s.setTextureRect(IntRect(7 * w, 18 * w, w, w));
                    s.setPosition(i*w, j*w);
                    app.draw(s);

                }
            }
        }

        if (man.get_hp() == 0)
        {
                    Player_1_info.setString("Game over\nYou saved " + std::to_string(man.get_saved_people()) + " rabbits");
            app.draw(Player_1_info);
        }
          else if (man.get_saved_people() == 3 && x == field->width-1 && y == field->height-1)
        {
            Player_1_info.setString("Victory\nYou saved everyone");
            app.draw(Player_1_info);
        }
        else
        {
```

```
                Player_1_info.setString("Hp: " + std::to_string(man.get_hp()) +
"\tarmor: " + std::to_string(man.get_armor()) + "\tsaved poeple: " +
std::to_string(man.get_saved_people()) + "\talco: " +
std::to_string(man.get_alco()));
            app.draw(Player_1_info);
        }
        Event e;

        while(app.pollEvent(e))
        {
            if (e.type == Event::Closed)
                app.close();
            if (e.type == Event::KeyPressed)
            {
                if (e.key.code == Keyboard::Escape) app.close();
                if (e.key.code == Keyboard::Left)
                {
                    if ((x-1) >= 0)
                    {
                        if (field->ptr[x-1][y].get_pass())
                        {
                            change_move_player(-1, 0);
                        }
                    }
                }
                if (e.key.code == Keyboard::Right)
                {
                    if ((x+1) < field->width)
                    {
```

```
        if (field->ptr[x+1][y].get_pass())
        {
            change_move_player(1, 0);
        }
    }
}
if (e.key.code == Keyboard::Up)
{
    if ((y-1) >= 0)
    {
        if (field->ptr[x][y-1].get_pass())
        {
            change_move_player(0, -1);
        }
    }
}
if (e.key.code == Keyboard::Down)
{
    if ((y+1) < field->height)
    {
        if (field->ptr[x][y+1].get_pass())
        {
            change_move_player(0, 1);
        }
    }
}
if (e.key.code == Keyboard::R)
{
    field->del_Player_1(x,y,0);
```

```
                field->ptr[x][y].set_player_1(0);
                x = 0;
                y = 0;
                man.restart();
                start_Game();
            }
        }
    }
    app.display();
  }
}

void Game_Manager::change_move_player_help(int way)
{
  *(field->ptr[x][y].get_Element()) + man;
  switch(way)
  {
  case 0:
    field->ptr[x][y].set_people(0);
    logs_change.print_parametrs(&logs, 4);
    break;
  case 1:
    field->ptr[x][y].set_enemy(0);
    logs_change.print_parametrs(&logs, 3);
    break;
  case 2:
    field->ptr[x][y].set_medicine(0);
    logs_change.print_parametrs(&logs, 0);
    break;
```

```
    case 3:
        field->ptr[x][y].set_armor(0);
        logs_change.print_parametrs(&logs, 1);
        break;
    case 4:
        field->ptr[x][y].set_alcogol(0);
        logs_change.print_parametrs(&logs, 2);
        break;
    }
}

void Game_Manager::change_move_player(int x_change, int y_change)
{
    x = x + x_change;
    y = y + y_change;
    man.change_place(x, y);
    field->ptr[x][y].set_Player_1(&man);
    field->Player_1(x,y,1);
    field->del_Player_1(x-x_change,y-y_change,0);
    logs_change.print_parametrs(&logs, 5);
    if (field->ptr[x][y].get_people())
    {
        change_move_player_help(0);
    }
    if (field->ptr[x][y].get_enemy())
    {
        change_move_player_help(1);
    }
    if (field->ptr[x][y].get_medicine())
```

```
            {
                change_move_player_help(2);
            }
            if (field->ptr[x][y].get_armor())
            {
                change_move_player_help(3);
            }
            if (field->ptr[x][y].get_alcogol())
            {
                change_move_player_help(4);
            }
                if (field->ptr[x][y].get_b1() || field->ptr[x][y].get_b2() || field->ptr[x]
[y].get_b3())
            {
                check_behavior();
            }
        }

        void Game_Manager::check_behavior()
        {
            if (field->ptr[x][y].get_b1())
            {
                field->ptr[x][y].set_b1(0);
                *(field->ptr[x][y].get_Behavior_people()) - man;
                logs_change.print_parametrs(&logs, 6);
            }
            if (field->ptr[x][y].get_b3())
            {
                field->ptr[x][y].set_b3(0);
```

```
        *(field->ptr[x][y].get_Behavior_to_start()) - man;
        field->del_Player_1(x,y,0);
        field->ptr[x][y].set_player_1(0);
        x = man.get_pos_x();
        y = man.get_pos_y();
        field->ptr[x][y].set_player_1(1);
        logs_change.print_parametrs(&logs, 8);
    }
    if (field->ptr[x][y].get_b2())
    {
        field->ptr[x][y].set_b2(0);
        *(field->ptr[x][y].get_Behavior_steal_alco()) - man;
        logs_change.print_parametrs(&logs, 7);
    }
}

void Game_Manager::set_Field()
{
    for (int i = 2; i < 30; i++)
    {
        field->Unpass(i, 0, 0);
    }
    for (int i = 0; i < 8; i++)
    {
        field->Unpass(29, i, 0);
    }
    for (int i = 1; i < 10; i++)
    {
        field->Unpass(0, i, 0);
```

```
    }
    for (int i = 1; i < 29; i++)
    {
        field->Unpass(i, 9, 0);
    }
field->In(0,0,1);
field->Player_1(0,0,1);
field->Out(29,9,1);
field->Unpass(7,2,0);
field->Unpass(7,3,0);
field->Unpass(7,4,0);
field->Unpass(4,5,0);
field->Unpass(4,6,0);
field->Unpass(4,7,0);
field->Unpass(4,8,0);
field->Unpass(2,6,0);
field->Unpass(2,7,0);
field->Unpass(6,6,0);
field->Unpass(6,7,0);
field->Unpass(1,4,0);
field->Unpass(4,7,0);
for (int i = 9; i < 29; i++)
    {
        field->Unpass(i, 2, 0);
    }
    field->Unpass(10,6,0);
    for (int i = 11; i < 28; i++)
    {
        field->Unpass(i, 6, 0);
```

```
        }
        field->Unpass(10,3,0);
        field->Unpass(10,4,0);
        field->Unpass(9,4,0);
        field->Unpass(9,5,0);
        field->Unpass(9,6,0);
    field->Unpass(4,8,0);
    field->Unpass(2,4,0);
    field->Unpass(4,4,0);
    field->Unpass(6,5,0);
    field->Unpass(2,0,0);
    field->Unpass(2,1,0);
    field->Unpass(2,2,0);
    field->Unpass(3,2,0);
    field->Unpass(4,2,0);
    field->Unpass(5,2,0);
    field->Unpass(6,2,0);
    field->Unpass(6,4,0);
    }

    void Game_Manager::set_Player_and_logs(int x, int y)
    {
        field->ptr[x][y].set_Player_1(&man);
        logs_change.set_player(&man);
            logs.add_logs("Игрок установлен в: " + std::to_string(x) + ", " +
std::to_string(y) + "\n");
    }

    void Game_Manager::set_Elements_and_logs(int x, int y, int log)
```

```
        {
            switch(log)
            {
            case 0:
                field->ptr[x][y].set_Element(factory_people.createElement());
                field->ptr[x][y].set_people(1);
                 logs.add_logs("Кролики установлены в: " + std::to_string(x) + ", " +
std::to_string(y) + "\n");
                break;
            case 1:
                field->ptr[x][y].set_Element(factory_enemy.createElement());
                field->ptr[x][y].set_enemy(1);
                 logs.add_logs("Гоблины установлены в: " + std::to_string(x) + ", " +
std::to_string(y) + "\n");
                break;
            case 2:
                field->ptr[x][y].set_Element(factory_medicine.createElement());
                field->ptr[x][y].set_medicine(1);
                 logs.add_logs("Аптечки установлены в: " + std::to_string(x) + ", " +
std::to_string(y) + "\n");
                break;
            case 3:
                field->ptr[x][y].set_Element(factory_armor.createElement());
                field->ptr[x][y].set_armor(1);
                   logs.add_logs("Броня установлена в: " + std::to_string(x) + ", " +
std::to_string(y) + "\n");
                break;
            case 4:
                field->ptr[x][y].set_Element(factory_alcogol.createElement());
```

```
            field->ptr[x][y].set_alcogol(1);
              logs.add_logs("Бутыль установлена в: " + std::to_string(x) + ", " +
std::to_string(y) + "\n");
            break;
        case 5:
            field->ptr[x][y].set_Behavior_people(&behavior_people);
            field->ptr[x][y].set_b1(1);
             logs.add_logs("Врагу установлено поведение b1 (кража кролика) в
позиции: " + std::to_string(x) + ", " + std::to_string(y) + "\n");
            break;
        case 6:
            field->ptr[x][y].set_Behavior_steal_alco(&behavior_steal_alco);
            field->ptr[x][y].set_b2(1);
              logs.add_logs("Врагу установлено поведение b2 (кража бутыли) в
позиции: " + std::to_string(x) + ", " + std::to_string(y) + "\n");
            break;
        case 7:
            field->ptr[x][y].set_Behavior_to_start(&behavior_to_start);
            field->ptr[x][y].set_b3(1);
            logs.add_logs("Врагу установлено поведение b3 (отправка в начало)
в позиции: " + std::to_string(x) + ", " + std::to_string(y) + "\n");
            break;
        }
    }

    void Game_Manager::set_Player_and_Elements_and_Logs()
    {
        set_Player_and_logs(0, 0);
        set_Elements_and_logs(3, 1, 0);
```

```
    set_Elements_and_logs(28, 1, 0);
    set_Elements_and_logs(1, 8, 0);
    set_Elements_and_logs(6, 3, 1);
    set_Elements_and_logs(3, 8, 1);
    set_Elements_and_logs(7, 5, 1);
    set_Elements_and_logs(1, 5, 2);
    set_Elements_and_logs(9, 3, 2);
    set_Elements_and_logs(7, 7, 3);
    set_Elements_and_logs(10, 5, 3);
    set_Elements_and_logs(28, 3, 4);
    set_Elements_and_logs(9, 7, 5);
    set_Elements_and_logs(10, 7, 6);
    set_Elements_and_logs(11, 7, 7);
}
```

Opponents.h:

```
#pragma once

template <class T> class Opponents{};
```

Opponents.cpp:

```
#include "Opponents.h"
```

Behavior.h:

```
#pragma once
#include "1_Player.h"

class Behavior
{
public:
    virtual void operator-(Player_1&) = 0;
```

```
    virtual ~Behavior(){};
};
```

Behavior.cpp:

```
#include "Behavior.h"
```

Behavior_steal_people.h:

```
#pragma once
#include "Behavior.h"

class Behavior_people:public Behavior
{
public:
    void operator-(Player_1&);
};
```

Behavior_steal_people.cpp:

```
#include "Behavior_people.h"

void Behavior_people::operator-(Player_1& player_1)
{
    player_1.steal_people();
}
```

Behavior_steal_alco.h:

```
#pragma once
#include "Behavior.h"

class Behavior_steal_alco:public Behavior
{
    public:
    void operator-(Player_1&);
};
```

Behavior_steal_alco.cpp:

```
#include "Behavior_steal_alco.h"

void Behavior_steal_alco::operator-(Player_1& player_1)
{
   player_1.steal_alco();
}
```

Behavior_to_start.h:

```
#pragma once
#include "Behavior.h"

class Behavior_to_start:public Behavior
{
public:
   void operator-(Player_1&);
};
```

Behavior_to_start.cpp:

```
#include "Behavior_to_start.h"

void Behavior_to_start::operator-(Player_1& player_1)
{
   player_1.change_place(0, 0);
}
```