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: Добавления игрока и элементов для поля

Студент гр. 9382 _____ Демин В.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Создать класс игрока и элементов поля. Реализация на языке «чистом» C++, полностью в ООП стиле.

Задание.

Создан класс игрока, которым управляет пользователь. Объект класса игрока может перемещаться по полю, а также взаимодействовать с элементами поля. Для элементов поля должен быть создан общий интерфейс и должны быть реализованы 3 разных класса элементов, которые по разному взаимодействуют с игроком. Для взаимодействия игрока с элементом должен использоваться перегруженный оператор (Например, оператор +). Элементы поля могут добавлять очки игроку/замедлять передвижения/и.т.д.

Обязательные требования:

- Реализован класс игрока
- Реализованы три класса элементов поля
- Объект класса игрока появляется на клетке со входом
- Уровень считается пройденным, когда объект класса игрока оказывается на клетке с выходом (и при определенных условиях: например, набрано необходимое кол-во очков)
- Взаимодействие с элементами происходит через общий интерфейс
- Взаимодействие игрока с элементами происходит через перегруженный оператор

Дополнительные требования:

- Для создания элементов используется паттерн Фабричный метод/Абстрактная фабрика
- Реализовано динамическое изменение взаимодействия игрока с элементами через паттерн Стратегия. Например, при взаимодействии с определенным количеством элементов, игрок не может больше с ними взаимодействовать

Выполнение работы.

В ходе выполнение работы были разработаны основные класс элементов поля и класс игрока.

Class Player

Хранит в себе координаты x и y на игровом поле, которые изменяются в ходе передвижения с помощью методов set_cord_x(int x) и set_cord_y(int y), которые вызываются в классе Game, при управлении игрока стрелочками клавиатуры. Также класс хранит в себе поля int coins и int health, которые говорят нам сколько монеток собрал игрок и сколько у него жизнец. Определили оператор + для взаимодействия с элементами поля.

Interface Element

Представляет собой интерфейсом для следующих элементов : class Coin, class Mushroom, class Chest. Хранит в себе методы get_coin() и get_health(), которые сообщают игроку как изменятся его характеристики. Get_status() говорит игре как изменится статус игры, так как в элементе сундук, рандомным образом может выпасть моментальный выигрыш.

В движке игры были добавлены следующие условия для победы: необходимо собрать все coin на поле. После хода игрока происходит обновление экрана, с выводом статистики игрока.

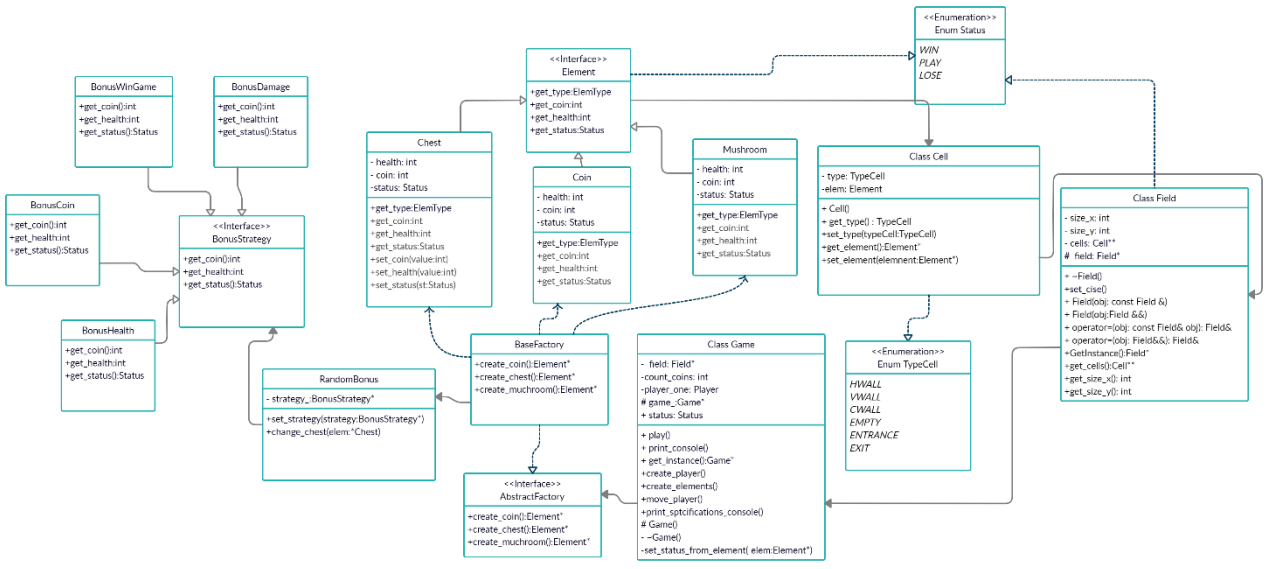

Рисунок 1. Uml диаграмма

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1	|_____ _____ _____ _____ _____| | H H M H | | H O M | | M O O | | M O | + O O M $ | O H M | | O O | | H O | |_____ _____ _____ _____ _____| Health:1 Coins:0 You need coins to win:10	Выбирается режим отрисовки клеток(1- в консоле).

Выводы.

В ходе выполнения лабораторной работы были созданы класс игрока и элементов поля, изучен ООП стиль.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include "Game/Game.h"


int main() {
   Game* game=Game::GetInstance();
    return 0;
}
```

Название файла: Cell.h

```
//
// Created by vikto on 13.09.2020.
//

#ifndef MYGAME_CELL_H
#define MYGAME_CELL_H
enum TypeCell{
    HWALL,
    VWALL,
    CWALL,
    EMPTY,
    ENTRANCE,
    EXIT
};


class Cell {

    int x;
    int y;
    TypeCell type;
public:
    Cell();

    void SetCord(int x, int y);
    void SetType(TypeCell typeCell);
    TypeCell GetType();
    int GetCordX() const;
    int GetCordY() const;
```

```
    Cell *next;
    Cell *prev;


    //список
};


#endif //MYGAME_CELL_H
```
Название файла: Cell.cpp
```
//
// Created by vikto on 13.09.2020.
//

#include "Cell.h"

Cell::Cell() {
    type = EMPTY;
    next = nullptr;
    prev = nullptr;
    x = 0;
    y = 0;
}

void Cell::SetCord(int x, int y) {
    this->x = x;
    this->y = y;
}

void Cell::SetType(TypeCell typeCell) {
    this->type = typeCell;
}

TypeCell Cell::GetType() {
    return type;
}

int Cell::GetCordY() const {
    return y;
}

int Cell::GetCordX() const {
    return x;
}
```

Название файла: Feild.h

```
//
// Created by vikto on 13.09.2020.
//

#ifndef MYGAME_FIELD_H
#define MYGAME_FIELD_H


#include "Cell/Cell.h"

class Field {
    Cell **cells;
    int size_x;
    int size_y;


protected:

    Field();

    static Field *field_;


public:
    ~Field();

    Field(const Field &obj);

    Field(Field &&obj);

    //Field(Field &other) = delete;

    Field &operator=(const Field &obj);

    Field &operator=(Field &&obj);

    static Field *GetInstance();

    Cell **GetCells() const;

    int GetSizeX() const;

    int GetSizeY() const;
```

```
private:
    void MakeListCells(Cell **cells_for_list) const;

};


#endif //MYGAME_FIELD_H
```

Название файла: Feild.cpp

```
//
// Created by vikto on 13.09.2020.
//

#include "Field.h"

Field *Field::field_ = nullptr;


Field *Field::GetInstance() {
    if (field_ == nullptr) {
        field_ = new Field();
    }
    return field_;
}

Field::Field() {
    size_x = 100;
    size_y = 10;
    //сделать границы поля, вход и выход
    cells = new Cell *[size_y];
    for (int i = 0; i < size_y; i++) {
        cells[i] = new Cell[size_x];
    }

    for (int i = 0; i < size_y; ++i) {
        for (int j = 0; j < size_x; ++j) {
            cells[i][j].SetCord(j, i);
            if (i == 0 || i == size_y - 1) {
                cells[i][j].SetType(HWALL);
            }
            if (j == 0 || j == size_x - 1) {
                cells[i][j].SetType(VWALL);
            }
            if (i == 0 && j == 0 || i == 0 && j == size_x - 1 || j == 0 && i ==
size_y - 1 ||
```

```
            j == size_x - 1 && i == size_y - 1) {
            cells[i][j].SetType(CWALL);
        }
        if (j == 0 && i == size_y / 2) {
            cells[i][j].SetType(ENTRANCE);
        }

        if (j == size_x - 1 && i == size_y / 2) {
            cells[i][j].SetType(EXIT);
        }
      }
   }

   MakeListCells(cells);
}

Field::~Field() {
   for (int i = 0; i < size_y; ++i) {
      delete cells[i];
   }
   delete[] cells;
   delete field_;

}

Cell **Field::GetCells() const {
   return cells;
}

int Field::GetSizeX() const {
   return size_x;
};

int Field::GetSizeY() const {
   return size_y;
}


Field::Field(const Field &obj) {
   this->size_x=obj.size_x;
   this->size_y=obj.size_y;
   this->cells = new Cell *[size_y];
   for (int i = 0; i < this->size_y; i++) {
      this->cells[i] = new Cell[size_x];
   }
```

```
        MakeListCells(this->cells);

        for (int i = 0; i < size_y; ++i) {
            for (int j = 0; j < this->size_x;++j) {
                this->cells[i][j].SetType(obj.cells[i][j].GetType());
                this->cells[i][j].SetCord(obj.cells[i][j].GetCordX(),obj.cells[i][j].GetCordY());
            }
        }

    }


    Field::Field(Field &&obj) {

        this->size_x=obj.size_x;
        this->size_y=obj.size_y;

        this->cells=obj.cells;
        obj.cells= nullptr;

        obj.cells= nullptr;
        obj.size_y=0;
        obj.size_x=0;
    }
    Field &Field::operator=(const Field & obj) {
        if(&obj == this)
            return *this;

        this->size_x=obj.size_y;
        this->size_x=obj.size_x;

        for (int i = 0; i < this->size_x; i++){
            delete [] this->cells[i];
        }
        delete [] this->cells;

        this->cells = new Cell *[size_y];
        for (int i = 0; i < this->size_y; i++) {
            this->cells[i] = new Cell[size_x];
        }

        MakeListCells(this->cells);
```

```
    for (int i = 0; i < this->size_y; ++i) {
        for (int j = 0; j < this->size_x;++j) {
            this->cells[i][j].SetType(obj.cells[i][j].GetType());
            this->cells[i][j].SetCord(obj.cells[i][j].GetCordX(),obj.cells[i][j].GetCordY());
        }
    }


    return *this;
}

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

    this->size_x=obj.size_y;
    this->size_x=obj.size_x;

    for (int i = 0; i < this->size_x; i++){
        delete [] this->cells[i];
    }
    delete [] this->cells;



    this->cells=obj.cells;
    obj.cells= nullptr;
    obj.size_x=0;
    obj.size_y=0;

    return *this;
}

void Field::MakeListCells(Cell **cells_for_list) const {
    Cell *p_temp = nullptr;

    for (int i = 0; i < size_y; ++i) {
        for (int j = 0; j < size_x; ++j) {
            cells_for_list[i][j].prev = p_temp;
            if (j != size_x - 1) {
                cells_for_list[i][j].next = &cells_for_list[i][j + 1];
            }
            if (j == size_x - 1 && i != size_y - 1) {
                cells_for_list[i][j].next = &cells_for_list[i + 1][0];
```

```
            }
            p_temp = &cells_for_list[i][j];
        }
    }
};
```

Название файла: Iterator.h

```
//
// Created by vikto on 13.09.2020.
//

#ifndef MYGAME_ITERATOR_H
#define MYGAME_ITERATOR_H
#include "../../Game/Field/Cell/Cell.h"


template <class U>
class Iterator {
public:
    virtual U *Next()=0;
    virtual U *begin()=0;
    virtual U *getElem()=0;
    virtual U *Prev()=0;
    virtual bool hasMore()=0;
};


class CellIterator: public Iterator<Cell> {
//методы для iteratinga
    Cell* temp;
    Cell* head;
    public:
    CellIterator(Cell &temp);
    Cell * getElem() override;
    Cell* begin() override;
    Cell* Next() override;
    Cell* Prev() override;
    bool hasMore() override;
};


#endif //MYGAME_ITERATOR_H
```

Название файла: Iterator.cpp

```
//
```

```
// Created by vikto on 14.09.2020.
//
#include "Iterator.h"


CellIterator::CellIterator(Cell &temp):temp(&temp) {
    head=this->temp;
}
//ничего не сделано
Cell* CellIterator::Next() {
    temp=temp->next;
    return temp;
}

bool CellIterator::hasMore() {
    if(temp== nullptr){
        return false;
    }else{
        return true;
    }
}

Cell* CellIterator::Prev() {
    temp=temp->prev;
    return temp;
}

Cell *CellIterator::begin() {
    temp=head;
    return head;
}

Cell *CellIterator::getElem() {
    return temp;
}
```

Название файла: Game.h

```
//
// Created by vikto on 13.09.2020.
//
#include <iostream>
#include "Field/Field.h"
#ifndef MYGAME_GAME_H
#define MYGAME_GAME_H
```

```
class Game {
    ~Game();
protected:
    Game();
    static Game* game_;

public:
    void play();
    void PrintConsole();
    Game(Game &other) = delete;
    void operator=(const Game &) = delete;
    static Game *GetInstance();

private:
    Field* field;
};


#endif //MYGAME_GAME_H

#endif //OOP_DISPLAY_H
```

Название файла: Game.cpp

```
//
// Created by vikto on 13.09.2020.
//

#include "Game.h"
#include "../Tools/MyCollections/collections.h"

Game *Game::game_ = nullptr;

Game::Game() {
    field = Field::GetInstance();
    play();
}

Game *Game::GetInstance() {
    if (game_ == nullptr) {
        game_ = new Game();
    }
    return game_;
```

```
}

void Game::play() {
   int mode = 0;
   std::cout << "Select mode:1-ConsoleMode\n";
   std::cin >> mode;
   if (mode == 1) PrintConsole();
}


void Game::PrintConsole() {
   //использоване итератора
   int size_x = field->GetSizeX();
   int size_y = field->GetSizeY();

   Cell **cells = field->GetCells();
   //without Iterator
   /*
   for (int i = 0; i < size_y; ++i) {
      for (int j = 0; j < size_x; ++j) {
         if (cells[i][j].GetType() == EMPTY) {
            std::cout << " ";
         }
         if (cells[i][j].GetType() == HWALL) {
            std::cout << "_";
         }
         if (cells[i][j].GetType() == VWALL || cells[i][j].GetType() == CWALL)
{
            std::cout << "|";
         }

         if (cells[i][j].GetType() == EXIT) {
            std::cout << "$";
         }
         if (cells[i][j].GetType() == ENTRANCE) {
            std::cout << "#";
         }
      }
      std::cout << "\n";
   }*/
   //with Iterator
   CellCollection cont(cells);

   Iterator<Cell> *it = cont.CreateIterator();
   for (it->begin(); it->hasMore(); it->Next()) {
```

```
            if (it->getElem()->GetType() == EMPTY) {
                std::cout << " ";
            }
            if (it->getElem()->GetType() == HWALL) {
                std::cout << "_";
            }
            if (it->getElem()->GetType() == VWALL || it->getElem()->GetType() ==
CWALL) {
                std::cout << "|";
            }

            if (it->getElem()->GetType() == EXIT) {
                std::cout << "$";
            }
            if (it->getElem()->GetType() == ENTRANCE) {
                std::cout << "#";
            }
            if (it->getElem()->GetCordX() == size_x - 1) {
                std::cout << "\n";
            }
        }

    }

    Game::~Game() {
        delete field;
        delete game_;

    }
```

Название файла: collections.cpp

```
    //
    // Created by vikto on 13.09.2020.
    //

    #include "collections.h"

    CellCollection::CellCollection(Cell** cells) {
        this->cells=cells;
    }

    Iterator<Cell> *CellCollection::CreateIterator() {
        CellIterator* a = new CellIterator(cells[0][0]);
        return a;
    }
```

Название файла: collections.h

```
//
// Created by vikto on 13.09.2020.
//

#ifndef MYGAME_MYCOLLECTIONS_H
#define MYGAME_MYCOLLECTIONS_H
#include "../../Game/Field/Cell/Cell.h"
#include "../Iterator/Iterator.h"

template <class U>
class Collection {
   virtual Iterator<U> *CreateIterator()=0;
};

class CellCollection: public Collection<class Cell>{
   //хранит коллекцию
   Cell** cells;
public:
   CellCollection(Cell** cells);
   Iterator<Cell> *CreateIterator() override;

};


#endif //MYGAME_MYCOLLECTIONS_H
```

Название файла: player.h

```
//
// Created by vikto on 18.10.2020.
//

#ifndef MYGAME_PLAYER_H
#define MYGAME_PLAYER_H

#include "../Elements/Element.h"

class Player {
   int cord_x;
   int cord_y;
public:
   Player();
   int get_cord_x() const;
```

```
    int get_cord_y() const;

    void set_cord_x(int x);

    void set_cord_y(int y);

    Player operator+(const Element &d2) ;
    int coins;
    int health;


};


#endif //MYGAME_PLAYER_H
```

Название файла: player.cpp

```
//
// Created by vikto on 18.10.2020.
//

#include "Player.h"

int Player::get_cord_x() const {
    return cord_x;
}
int Player::get_cord_y() const {
    return cord_y;  }

void Player::set_cord_x(int x) {
    cord_x=x;
}
void Player::set_cord_y(int y) {
    cord_y=y;
}

Player Player::operator+(const Element &d2) {
    this->coins=coins+d2.get_coin();
    this->health=health+d2.get_health();
    return *this;
}

Player::Player() {
    health=1;
    coins=0;
```

```
    cord_x=0;
    cord_y=0;
}
```
Название файла: Element.h

```
//
// Created by vikto on 19.10.2020.
//

#ifndef MYGAME_ELEMENT_H
#define MYGAME_ELEMENT_H
#include "../StatusEnum.h"
enum ElemType{
    MUSHROOM,COIN,CHEST
};

class Element {
public:
    virtual ElemType get_type() const =0;
    virtual int get_coin()const=0;
    virtual int get_health()const=0;
    virtual Status get_status()const=0;
};

#endif //MYGAME_ELEMENT_H
```