Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Лабораторная работа №5

По дисциплине ООТПиСП за 5 семестр

Выполнил:

Студент группы ПО-6(1)
3-го курса

Мартынович Даниил

Проверил:

Хацкевич М. В.

Брест 2022

Проект лабирит

Door.h

```
#ifndef DOOR_H
#define DOOR_H

#include <MapSite.h>
#include <Room.h>

class Door : public MapSite
{
public:
    Door(Room * room1 = 0, Room * room2 = 0, bool isOpen = true);
    void enter() override;
    Room * otherSideFrom(Room * room);
    virtual Door * clone();
    virtual void initialize(Room * room1 = 0, Room * room2 = 0, bool isOpen = true);

protected:
    Room * _room1;
    Room * _room2;
    bool _isOpen;
};

#endif // DOOR_H
```

Door.cpp

```
#include "Door.h"
#include <iostream>

Door::Door(Room * room1, Room * room2, bool isOpen)
{
    this->_room1 = room1;
    this->_room2 = room2;
    this->_isOpen = isOpen;
    std::cout << "create door" << std::endl;
}

void Door::enter()
{
    if(this->_isOpen)
    {
        std::cout << "go throw open door" << std::endl;
    }
    else
    {
        std::cout << "bam! closed door" << std::endl;
    }
}

Room * Door::otherSideFrom(Room * room)
{
    if (this->_room1 == room)
    {
        return this->_room1;
    }
    return this->_room2;
}

Door * Door::clone()
{
    return new Door(*this);
}

void Door::initialize(Room * room1, Room * room2, bool isOpen)
{
    this->_room1 = room1;
    this->_room2 = room2;
    this->_isOpen = isOpen;
}
```

MagicDoor.h

```
#ifndef MAGICDOOR_H
#define MAGICDOOR_H

#include <Door.h>

class MagicDoor : public Door
{
public:
    MagicDoor(Room * room1 = 0, Room * room2 = 0, bool isOpen = true);
    Door * clone() override;
    void initialize(Room * room1 = 0, Room * room2 = 0, bool isOpen = true) override;

};

#endif // MAGICDOOR_H
```

MagicDoor.cpp

```
#include "MagicDoor.h"
#include <iostream>

MagicDoor::MagicDoor(Room * room1, Room * room2, bool isOpen) :
    Door(room1, room2, isOpen)
{
    std::cout << "magic" << std::endl;
}

Door * MagicDoor::clone()
{
    return new MagicDoor(*this);
}

void MagicDoor::initialize(Room * room1, Room * room2, bool isOpen)
{
    this->_room1 = room1;
    this->_room2 = room2;
    this->_isOpen = isOpen;
}
```

Maze.h

```
#ifndef MAZE_H
#define MAZE_H

#include <Room.h>

class Maze
{
public:
    Maze();
    void addRoom(Room *room);
    virtual Maze * clone();
};


#endif // MAZE_H
```

Maze.cpp

```
#include "Maze.h"
#include <iostream>

Maze::Maze()
{

}

void Maze::addRoom(Room *room)
{
    std::cout << "add room to maze" << std::endl;
}

Maze * Maze::clone()
{
    return new Maze(*this);
}
```

Room.h

```
#ifndef ROOM_H
#define ROOM_H

#include <MapSite.h>

enum Direction {North, South, East, West};

class Room : public MapSite {
public:
    Room(int roomNo);
    MapSite * getSide(Direction directiom) const;
    void setSide(Direction direction, MapSite * mapSite);
    void enter() override;
    virtual Room * clone();

private:
    MapSite* _sides[4];
    int _rbomNumber;
};


#endif // ROOM_H
```

Room.cpp

```
#include "Room.h"
#include <iostream>

Room::Room(int roomNo)
{
    this->_rbomNumber = roomNo;
    std::cout << "create room" << std::endl;
}

MapSite * Room::getSide(Direction direction) const
{
    return this->_sides[direction];
}

void Room::setSide(Direction direction, MapSite * mapSite)
{
    this->_sides[direction] = mapSite;
}

void Room::enter()
{
    std::cout << "step in room" << std::endl;
}

Room * Room::clone()
{
    return new Room(*this);
}
```

Вывод: научился использовать паттерны.