**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: **Интерфейсы, полиморфизм**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Калмак Д.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2021

## **Цель работы.**

Реализовать классы игрока, врагов и предметов, соблюдая принцип полиморфизма.

## **Задание.**

Могут быть три типа элементов располагающихся на клетках:

Игрок - объект, которым непосредственно происходит управление. На поле может быть только один игрок. Игрок может взаимодействовать с врагом (сражение) и вещами (подобрать).

Враг - объект, который самостоятельно перемещается по полю. На поле врагов может быть больше одного. Враг может взаимодействовать с игроком (сражение).

Вещь - объект, который просто располагается на поле и не перемещается. Вещей на поле может быть больше одной.

Требования:

Реализовать класс игрока. Игрок должен обладать собственными характеристиками, которые могут изменяться в ходе игры. У игрока должна быть прописана логика сражения и подбора вещей. Должно быть реализовано взаимодействие с клеткой выхода.

Реализовать три разных типа врагов. Враги должны обладать собственными характеристиками (например, количество жизней, значение атаки и защиты, и.т.д. Желательно, чтобы у врагов были разные наборы характеристик). Реализовать логику перемещения для каждого типа врага. В случае смерти врага он должен исчезнуть с поля. Все враги должны быть объединены своим собственным интерфейсом.

Реализовать три разных типа вещей. Каждая вещь должна обладать собственным взаимодействием на ход игры при подборе. (например, лечение игрока). При подборе, вещь должна исчезнуть с поля. Все вещи должны быть объединены своим собственным интерфейсом.

Должен соблюдаться принцип полиморфизма.

## **Выполнение работы.**

Написан заголовочный файл Classes\_game.h, в котором написаны все игровые классы: class Object\_of\_game; class Creature; class Player; class Nazgul; class Bolg; class Smaug; class Items; class Inventory; class Armor; class Weapon; class Medicine;

Для взаимодействия объектов реализован интерфейс Object\_of\_game с методом interact().

Класс Creature использует интерфейс Object\_of\_game. Этот класс является основой для классов игрока и врагов. У класса реализованы поля actual.hp, max\_hp, actual\_armor, actual\_weapon, что соответствует текущему здоровью, максимальному здоровью, текущей броне и текущему оружию. Переопределен метод взаимодействия interact. Определены методы установки оружия set\_weapon(), брони set\_armor(), также выдачи информации о броне get\_armor(), оружии get\_weapon(), здоровье get\_hp(), а также метод изменяющий здоровье increase\_hp().

Реализован класс игрока class Player, который наследуется от Creature и конструктором заполняет информацию о герое. Классы врагов class Bolg, class Nazgul, class Smaug наследуются от Creature и конструктором заполняют информацию о себе — здоровье, количество брони, оружие.

Класс Items использует интерфейс Object\_of\_game. Этот класс содержит метод взаимодействия interact.

Реализован класс брони Armor, который наследуется от класса Items. Поле value содержит прочность брони. Метод get\_value() возвращает информацию о броне. Метод interact() взаимодействие с существом. Реализован класс оружия Weapon, который наследуется от класса Items. Поле dmg содержит урон оружия. Метод get\_dmg() возвращает информацию о оружии. Метод interact() взаимодействие с существом. Реализован класс аптечки Medicine, который наследуется от класса Items. Поле heal\_size содержит количество здоровья в аптечке. Метод get\_healsize() возвращает информацию о количестве здоровья в аптечке. Метод interact() взаимодействие с существом.

В класс Interface\_cell добавлены методы, которые определены в классе Cell, set\_objectofgame() и get\_objectofgame(), которые устанавливают объект на клетку и возвращают информацию об объекте с клетки соответственно.

Реализован класс инвентаря Inventory. Поле хранит размер и массив предметов. Метод selection заполняет инвентарь предметами.

Разработанный код см. в Приложении А.

Разработанную диаграмму классов UML см. в Приложении Б.

## **Тестирование.**

Результаты тестирования представлены в Таблице 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Результаты тестирования | | | |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1 | 15 15 | Hp player: 200 Armor player: 20 Damage player: 30  Armor Bolg: 15 Hp Bolg: 100  Did he survive?  interact creature  Yes!  Hp Bolg: 85  Bolg need medicine!  Hp in medicine: 20  interact item  Bolg got medicine and his Hp: 100  pb0000000000000  001000101110000  01100011100111e  000011001110100  000001011011110  010001111001110  000100101111110  010011110001110  011000101111100  001110000111010  010001010011110  000010001001000  011011101011110  001010010010000  i01101001011000 | Использование методов игрока и врага. Взаимодействие игрока с врагом. Взаимодействие врага с предметом. |

## **Выводы.**

Таким образом, были реализованы классы игрока, врагов, а также предметов. У классов реализован интерфейс для взаимодействия.

# **ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

**Файл Classes\_game:**

#ifndef CLASSES\_GAME\_H

#define CLASSES\_GAME\_H

using namespace std;

class *Object\_of\_game*;

class *Creature*;

class *Player*;

class *Nazgul*;

class *Bolg*;

class *Smaug*;

class *Items*;

class *Inventory*;

class *Armor*;

class *Weapon*;

class *Medicine*;

#endif

**Файл Object\_of\_game.h:**

#ifndef OBJECT\_OF\_GAME\_H

#define OBJECT\_OF\_GAME\_H

#include <iostream>

#include "Classes\_game.h"

class *Object\_of\_game*{

public:

virtual bool interact(*Creature*\* *obj*) = 0;

};

#endif

**Файл Creature.h:**

#ifndef CREATURE\_H

#define CREATURE\_H

#include "Object\_of\_game.h"

class *Creature*: public *Object\_of\_game*{

private:

int actual\_hp;

int max\_hp;

*Armor*\* actual\_armor;

*Weapon*\* actual\_weapon;

public:

Creature(int *max\_hp*, *Armor*\* *actual\_armor*, *Weapon*\* *actual\_weapon*);

void set\_armor(*Armor*\* *armor*);

void set\_weapon(*Weapon*\* *weapon*);

int get\_hp();

int get\_armor();

int get\_weapon();

bool interact(*Creature*\* *other*);

void increase\_hp(int *hp*);

};

#endif

**Файл Creature.cpp:**

#include "Weapon.h"

#include "Armor.h"

*Creature*::Creature(int *max\_hp*, *Armor*\* *actual\_armor*, *Weapon*\* *actual\_weapon*): actual\_hp(*max\_hp*), max\_hp(*max\_hp*),

actual\_armor(*actual\_armor*), actual\_weapon(*actual\_weapon*) {}

void *Creature*::set\_armor(*Armor*\* *armor*){

actual\_armor = *armor*;

}

void *Creature*::set\_weapon(*Weapon*\* *weapon*){

actual\_weapon = *weapon*;

}

int *Creature*::get\_hp(){

return actual\_hp;

}

int *Creature*::get\_armor(){

return actual\_armor->get\_value();

}

int *Creature*::get\_weapon(){

return actual\_weapon->get\_dmg();

}

bool *Creature*::interact(*Creature*\* *other*){

cout << "interact creature" << endl;

// first stage - damage armor

int armor = *other*->get\_armor();

int damage = get\_weapon();

int dif = armor - damage;

// second stage - damage hp

if (dif >=0){}

else *other*->increase\_hp(dif);

return *other*->actual\_hp > 0;

}

void *Creature*::increase\_hp(int *hp*){

actual\_hp = min(max\_hp, actual\_hp + *hp*);

}

**Файл Player.h:**

#ifndef PLAYER\_H

#define PLAYER\_H

#include "Creature.h"

#include "Armor.h"

#include "Weapon.h"

class *Player*: public *Creature*{

public:

Player(): *Creature*(200, **new** *Armor*(20), **new** *Weapon*(30)) {}

};

#endif

**Файл Bolg.h:**

#ifndef ENEMY\_BOLG\_H

#define ENEMY\_BOLG\_H

#include "Creature.h"

#include "Armor.h"

#include "Weapon.h"

class *Bolg*: public *Creature*{

public:

Bolg(): *Creature*(100, **new** *Armor*(15), **new** *Weapon*(15)) {}

};

#endif

**Файл Nazgul.h:**

#ifndef ENEMY\_NAZGUL\_H

#define ENEMY\_NAZGUL\_H

#include "Creature.h"

#include "Armor.h"

#include "Weapon.h"

class *Nazgul*: public *Creature*{

public:

Nazgul(): *Creature*(200, **new** *Armor*(20), **new** *Weapon*(20)) {}

};

#endif

**Файл Smaug.h:**

#ifndef ENEMY\_SMAUG\_H

#define ENEMY\_SMAUG\_H

#include "Creature.h"

#include "Armor.h"

#include "Weapon.h"

class *Smaug*: public *Creature*{

public:

Smaug(): *Creature*(500, **new** *Armor*(50), **new** *Weapon*(50)) {}

};

#endif

**Файл Items.h:**

#ifndef ITEMS\_H

#define ITEMS\_H

#include "Object\_of\_game.h"

#include "Classes\_game.h"

class *Items*: public *Object\_of\_game*{

public:

bool interact(*Creature*\* *obj*){

//interaction with creatures

return true;

}

};

#endif

**Файл Armor.h:**

#ifndef ARMOR\_H

#define ARMOR\_H

#include "Creature.h"

#include "Items.h"

class *Armor*: public *Items*{

private:

int value;

public:

Armor(int *value*): value(*value*){}

int get\_value(){

return value;

}

// interaction with the creature

bool interact(*Creature*\* *obj*){

cout << "interact item" << endl;

*obj*->set\_armor(*this*);

return true;

}

};

#endif

**Файл Weapon.h:**

#ifndef WEAPON\_H

#define WEAPON\_H

#include "Creature.h"

#include "Items.h"

class *Weapon*: public *Items*{

private:

int dmg;

public:

Weapon(int *dmg*): dmg(*dmg*){}

int get\_dmg(){

return dmg;

}

// interaction with the creature

bool interact(*Creature* \**obj*){

cout << "interact item\n";

*obj*->set\_weapon(*this*);

return true;

}

};

#endif

**Файл Inventory.h:**

#ifndef INVENTORY\_H

#define INVENTORY\_H

#include "Items.h"

class *Inventory*{

private:

int size;

*Items*\*\* inventory\_items;

public:

Inventory(int *size*);

void selection(*Items*\* *item*);

};

#endif

**Файл Inventory.cpp:**

#include "Inventory.h"

*Inventory*::Inventory(int *size*): size(*size*){

// fill empty inventory

inventory\_items = **new** *Items*\*[*size*];

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

inventory\_items[i] = nullptr;

}

}

void *Inventory*::selection(*Items*\* *item*) {

// Got item

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

inventory\_items[i] = *item*;

}

}

**Файл Medicine.h:**

#ifndef MEDICINE\_H

#define MEDICINE\_H

#include "Creature.h"

#include "Items.h"

class *Medicine*: public *Items*{

private:

int heal\_size;

public:

Medicine(int *heal\_size*): heal\_size(*heal\_size*) {}

int get\_healsize(){

return heal\_size;

}

// interaction with the creature

bool interact(*Creature*\* *obj*){

cout << "interact item" << endl;

*obj*->increase\_hp(heal\_size);

return true;

}

};

#endif

# **ПРИЛОЖЕНИЕ Б UML ДИАГРАММА КЛАССОВ**

