***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №5**

**По дисциплине ППвИС за III семестр**

**Тема: «Иерархии классов. Наследование»**

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Михняев А.Л.

Брест 2018

ЦЕЛЬ РАБОТЫ

1. Изучение правил наследования классов.

2. Реализация одиночного наследования классов.

3. Изучение управления методами и свойствами производных классов через объекты производных классов и через указатели на объекты производных классов.

4. Изучение правил описания наследования и диаграмм классов в языке UML.

СПИСОК ЗАДАЧ

Индивидуальные задания на разработку классов. Реализовать одиночное прямое наследование.

*Выбранная предметная область:* Онлайн игра.

*Описание тестового примера.*

В качестве тестовых входных данных будем использовать следующие:

1. Name of game: “Dota 2”

Release year: 2013

Price: 0;

Owner: “valve”;

Engine: “Source 2”

Name of user: “Ilya”

Nickname of user: “Ilya2000”

*Описание классов:*

1. Имя: App

Назначение: хранит информацию о приложении

1. Имя: Game (потомок App)

Назначение: хранит информацию об игре.

1. Имя: OnlineGame (потомок Game)

Назначение: хранит информацию об онлайн игре, содержит список игроков, находящихся онлайн.

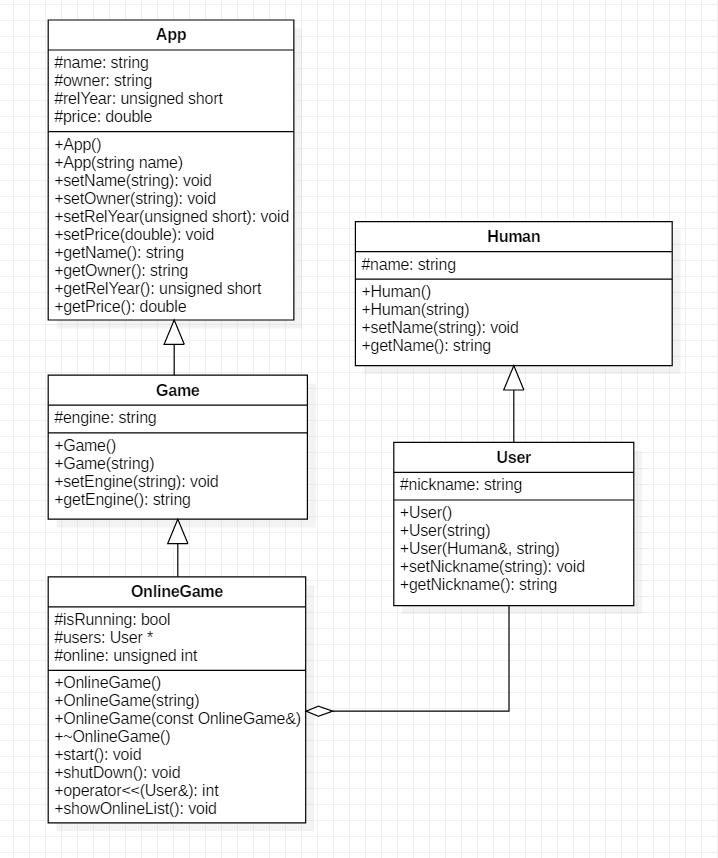
1. Имя: Human

Назначение: хранит информацию о человеке, а именно его имя.

1. Имя: User (потомок Human, тип наследования – приватный)

Назначение: хранит информацию о пользователе, а именно его никнейм.

UML:



*Листинг кода программы:*

// classes.h

#pragma once

#include <string>

void log(std::string author, std::string message);

class App

{

protected:

std::string name;

std::string owner;

unsigned short relYear;

double price;

public:

App();

App(std::string name);

void setName(std::string name);

void setOwner(std::string owner);

void setRelYear(unsigned short relYear);

void setPrice(double price);

std::string getName();

std::string getOwner();

unsigned short getRelYear();

double getPrice();

};

class Game : public App

{

protected:

std::string engine;

public:

Game();

Game(std::string name);

void setEngine(std::string engine);

std::string getEngine();

friend void showInfo(const Game& game);

};

void showInfo(const Game& game);

class Human

{

protected:

std::string name;

public:

Human();

Human(std::string name);

void setName(std::string name);

std::string getName();

};

class User : Human

{

protected:

std::string nickname;

public:

User();

User(std::string nickname);

User(Human& human, std::string nickname);

void setNickname(std::string);

std::string getNickname();

};

class OnlineGame : public Game

{

bool isRunning;

User \* users;

unsigned int online;

public:

OnlineGame();

OnlineGame(std::string name);

OnlineGame(const OnlineGame& obj);

~OnlineGame();

void start();

void shutDown();

int operator<<(User& user);

void showOnlineList();

};

//classes.cpp

#include "classes.h"

#include <iostream>

void log(std::string author, std::string message)

{

std::cout << author << ": " << message << std::endl;

}

App::App()

:name("noname"), owner("noname"), relYear(0), price(0)

{}

App::App(std::string name)

: name(name), owner("noname"), relYear(0), price(0)

{}

void App::setName(std::string name)

{

this->name = name;

}

void App::setOwner(std::string owner)

{

this->owner = owner;

}

void App::setRelYear(unsigned short relYear)

{

this->relYear = relYear;

}

void App::setPrice(double price)

{

this->price = price;

}

std::string App::getName()

{

return this->name;

}

std::string App::getOwner()

{

return this->owner;

}

unsigned short App::getRelYear()

{

return this->relYear;

}

double App::getPrice()

{

return this->price;

}

Game::Game()

: engine("noname")

{}

Game::Game(std::string name)

: engine("noname")

{

this->name = name;

}

void Game::setEngine(std::string engine)

{

this->engine = engine;

}

std::string Game::getEngine()

{

return this->engine;

}

void showInfo(const Game& game)

{

std::cout << "Name: " << game.name << std::endl;

std::cout << "Owner: " << game.owner << std::endl;

std::cout << "Engine: " << game.engine << std::endl;

std::cout << "Release year: " << game.relYear << std::endl;

std::cout << "Price: " << game.price << std::endl;

}

OnlineGame::OnlineGame()

: isRunning(false), users(NULL), online(0)

{}

OnlineGame::OnlineGame(std::string name)

: isRunning(false), users(NULL), online(0)

{

this->name = name;

}

OnlineGame::OnlineGame(const OnlineGame& obj)

{

this->name = obj.name;

this->engine = obj.engine;

this->isRunning = obj.isRunning;

this->online = obj.online;

this->owner = obj.owner;

this->price = obj.price;

this->relYear = obj.relYear;

this->users = new User[this->online];

for (unsigned short i = 0; i < this->online; i++)

this->users[i] = obj.users[i];

}

OnlineGame::~OnlineGame()

{

if (this->online!=0)

delete[] this->users;

}

void OnlineGame::start()

{

if (this->isRunning)

{

log(this->name, "Cannot start the server.");

log(this->name, "Server is already started.");

return;

}

this->isRunning = true;

log(this->name, "Server has been started.");

}

void OnlineGame::shutDown()

{

if (!this->isRunning)

{

log(this->name, "Cannot shut down the server.");

log(this->name, "Server is already shutted down.");

return;

}

this->isRunning = false;

delete[] this->users;

this->online = 0;

log(this->name, "Server has been shutted down.");

}

int OnlineGame::operator<<(User& user)

{

if (!this->isRunning)

{

log(this->name, (user.getNickname() + " cannot join the server."));

log(this->name, "Server is not running.");

return 0;

}

User \* users\_copy = new User[this->online];

for (unsigned short i = 0; i < this->online; i++)

users\_copy[i] = this->users[i];

delete[] this->users;

this->online++;

this->users = new User[this->online];

for (unsigned short i = 0; i < (this->online - 1); i++)

this->users[i] = users\_copy[i];

this->users[this->online - 1] = user;

log(this->name, (user.getNickname() + " has been connected."));

return 1;

}

void OnlineGame::showOnlineList()

{

if (!this->isRunning)

{

log(this->name, "Cannot access the online list.");

log(this->name, "Server is not running.");

return;

}

std::cout << "\tOnline list (" << this->name << "): "<< std::endl;

for (int i = 0; i < this->online; i++)

std::cout << i + 1 <<".\t" << this->users[i].getNickname() << std::endl;

}

Human::Human()

: name("noname")

{}

Human::Human(std::string name)

: name(name)

{}

void Human::setName(std::string name)

{

this->name = name;

}

std::string Human::getName()

{

return this->name;

}

User::User()

{

this->name = "noname";

this->nickname = "no\_nickname";

}

User::User(Human& human, std::string nickname)

{

this->name = human.getName();

this->nickname = nickname;

}

User::User(std::string nickname)

{

this->name = "noname";

this->nickname = nickname;

}

void User::setNickname(std::string nickname)

{

this->nickname = nickname;

}

std::string User::getNickname()

{

return this->nickname;

}

// main.cpp

#include <iostream>

#include "classes.h"

int main()

{

OnlineGame dota2;

Game \* ptr\_dota2 = &dota2;

ptr\_dota2->setName("Dota 2");

ptr\_dota2->setOwner("Valve");

ptr\_dota2->setEngine("Source 2");

ptr\_dota2->setPrice(0);

ptr\_dota2->setRelYear(2013);

showInfo(\*ptr\_dota2);

dota2.setName("Dota 2 update");

dota2.setOwner("Valve update");

dota2.setEngine("Source 2 update");

dota2.setPrice(0);

dota2.setRelYear(2018);

showInfo(dota2);

Human human("Ilya");

User user(human, "Ilya2000");

dota2 << user;

dota2.start();

dota2 << user;

dota2.showOnlineList();

dota2.shutDown();

dota2.showOnlineList();

system("pause");

}

*Листинг результата:*



Вывод: изучил правила наследования классов, правила управления методами и свойствами производных классов через объекты производных классов и через указатели на объекты производных классов, изучил правила описания наследования и диаграмм классов в языке UML.