Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №7**

на тему:

“Множинне спадкування. Поліморфізм”

**Виконав:**

Ст. гр. КІ-15

Височанський Сергій

**Перевірив:**

Викладач

Козак Н.Б.

Львів – 2020

**Мета роботи:** познайомитися із множинним спадкуванням класів та поліморфізмом.

***Завдання:***

Спроектувати і реалізувати ієрархію класів, що описують предметну область згідно варіанту, яка реалізується класом 1. Клас 1 в свою чергу утворюється шляхом множинного спадкування класів 2 і 3 кожен з яких в свою чергу успадковує клас 4. Додаткові вимоги:

1. Базовий клас містить мінімум один віртуальний метод, один невіртуальний метод і одну динамічно створювану властивість.

2. Забезпечити механізми коректної роботи конструкторів і деструкторів.

3. Перевантажити оператор присвоєня з метою його коректної роботи.

4. Кожен з класів має містити мінімум одину властивість і 4 методи.

5. Написати main() функцію де створити об‘єкт класу 1 і продемонструвати різницю між статичним і динамічним поліморфізмом.



*Код програми:*

#pragma once

#include"Patient.h"

#include"Doctor.h"

class CPatientDoctor:public CPatient, public CDoctor

{

public:

CPatientDoctor();

CPatientDoctor(string n, int a, string sym, string sp, int e);

CPatientDoctor(const CPatientDoctor& other);

virtual ~CPatientDoctor();

void ShowDoctorInfo()override;

void ShowPatientInfo()override;

void ShowPatientDoctorInfo();

void ShowFullInfo()override;

CPatientDoctor& operator=(const CPatientDoctor& other);

protected:

};

CPatientDoctor::CPatientDoctor():CDoctor(), CPatient()

{

}

CPatientDoctor::CPatientDoctor(string n, int a, string sym, string sp, int e) : CHuman(n,a), CDoctor(n, a, sp, e), CPatient(n, a, sym)

{

}

CPatientDoctor::CPatientDoctor(const CPatientDoctor& other)

{

CPatient::operator=(other);

CDoctor::operator=(other);

}

CPatientDoctor::~CPatientDoctor()

{

}

void CPatientDoctor::ShowDoctorInfo()

{

cout << "I am a doctor(" << \*specialty << ")" << endl;

cout << "My work experience is " << experience << " years" << endl;

cout << "I am on a sick leave!" << endl;

}

void CPatientDoctor::ShowPatientInfo()

{

cout << "I am very ill" << endl;

}

void CPatientDoctor::ShowPatientDoctorInfo()

{

cout << "I am a doctor but I am ill, so I can not treat patients." << endl << "That mean now I am the patient" << endl;

}

void CPatientDoctor::ShowFullInfo()

{

CDoctor::ShowFullInfo();

ShowPatientDoctorInfo();

GetSymptoms();

}

CPatientDoctor& CPatientDoctor::operator=(const CPatientDoctor& other)

{

CPatient::operator=(other);

CDoctor::operator=(other);

return \*this;

}

#pragma once

#include"Human.h"

class CDoctor:virtual public CHuman

{

public:

CDoctor();

CDoctor(string n, int a, string s, int e);

CDoctor(const CDoctor& other);

virtual ~CDoctor();

void GetSpecialty();

void SetSpecialty(string s);

void GetExperience();

void SetExperience(int e);

virtual void ShowDoctorInfo();

void ShowFullInfo()override;

CDoctor& operator=(const CDoctor& other);

protected:

string\* specialty = new string;

int experience;

};

CDoctor::CDoctor():CHuman()

{

\*specialty = "No specialty";

experience = 0;

}

CDoctor::CDoctor(string n, int a, string s, int e) :CHuman(n,a)

{

\*specialty = s;

experience = e;

}

CDoctor::CDoctor(const CDoctor& other)

{

this->specialty = other.specialty;

this->experience = other.experience;

CHuman::operator=(other);

}

CDoctor::~CDoctor()

{

delete specialty;

specialty = nullptr;

}

void CDoctor::GetSpecialty()

{

cout << "Specialty: " << \*specialty << endl;

}

void CDoctor::SetSpecialty(string s)

{

\*specialty = s;

}

void CDoctor::GetExperience()

{

cout << "Experience: " << experience << endl;

}

void CDoctor::SetExperience(int e)

{

experience = e;

}

void CDoctor::ShowDoctorInfo()

{

cout << "I am a doctor(" << \*specialty << ")" << endl;

cout << "My work experience is " << experience << " years" << endl << endl;

}

void CDoctor::ShowFullInfo()

{

GetName();

GetAge();

ShowDoctorInfo();

}

CDoctor& CDoctor:: operator=(const CDoctor& other)

{

this->specialty = new string;

this->specialty = other.specialty;

this->experience = other.experience;

CHuman::operator=(other);

return \*this;

}

#pragma once

#include "Human.h"

class CPatient:virtual public CHuman

{

public:

CPatient();

CPatient(string n, int a, string s);

CPatient(const CPatient& other);

virtual ~CPatient();

void GetSymptoms();

void SetSymptoms(string s);

void AddSymptoms(string s);

virtual void ShowPatientInfo();

void ShowFullInfo()override;

CPatient& operator=(const CPatient& other);

protected:

string\* symptoms = new string;

};

CPatient::CPatient():CHuman()

{

\*symptoms = "No symptoms";

}

CPatient::CPatient(string n, int a, string s) : CHuman(n, a)

{

\*symptoms = s;

}

CPatient::CPatient(const CPatient& other)

{

this->symptoms = other.symptoms;

CHuman::operator=(other);

}

CPatient::~CPatient()

{

delete symptoms;

symptoms = nullptr;

}

void CPatient::GetSymptoms()

{

cout << "Symptoms: " << \*symptoms << endl << endl;

}

void CPatient::SetSymptoms(string s)

{

\*symptoms = s;

}

void CPatient::AddSymptoms(string s)

{

\*symptoms = \*symptoms +", "+ s;

}

void CPatient::ShowPatientInfo()

{

cout << "I am a patient and I am very ill"<< endl;

}

void CPatient::ShowFullInfo()

{

GetName();

GetAge();

ShowPatientInfo();

GetSymptoms();

}

CPatient& CPatient::operator=(const CPatient& other)

{

this->symptoms = new string;

this->symptoms = other.symptoms;

CHuman::operator=(other);

return \*this;

}

#pragma once

#include<iostream>

#include<string>

using namespace std;

class CHuman

{

public:

CHuman();

CHuman(string n, int a);

CHuman(const CHuman& other);

virtual ~CHuman();

void SetName(string n);

void GetName();

void SetAge(int a);

void GetAge();

virtual void ShowFullInfo() = 0;

CHuman& operator=(const CHuman& other);

protected:

string\* name = new string;

int age;

};

CHuman::CHuman()

{

\*name = "No name";

age = 0;

}

CHuman::CHuman(string n, int a)

{

\*name = n;

age = a;

}

CHuman::CHuman(const CHuman& other)

{

this->name = other.name;

this->age = other.age;

}

CHuman::~CHuman()

{

delete name;

name = nullptr;

}

void CHuman::SetName(string n)

{

\*name = n;

}

void CHuman::GetName()

{

cout << "Name: " << \*name << endl;

}

void CHuman::SetAge(int a)

{

age = a;

}

void CHuman::GetAge()

{

cout << "Age: " << age << endl;

}

CHuman& CHuman::operator=(const CHuman& other)

{

this->name = new string;

this->name = other.name;

this->age = other.age;

return \*this;

}

#include"PatientDoctor.h"

int main()

{

cout << "----- Creating objects -----" << endl;

CPatient h1("Ivan", 34, "headache");

h1.AddSymptoms("sick");

h1.ShowFullInfo();

CDoctor h2("Ivan", 44, "eye specialist", 0);

h2.SetExperience(6);

h2.ShowFullInfo();

CPatientDoctor h3("Ivan", 47, "earache", "dentist", 2);

h3.SetExperience(5);

h3.ShowFullInfo();

CPatientDoctor h4 = h3;

h4.ShowFullInfo();

CPatientDoctor h5(h3);

h5.ShowFullInfo();

h1 = h5;

h1.ShowFullInfo();

cout << "----- Creating reference -----" << endl;

cout << "----- CHuman &ref1 = h1; -----" << endl;

CHuman& ref1 = h1;

ref1.ShowFullInfo();

cout << "----- ref1 = h2; -----" << endl;

ref1 = h2;

ref1.ShowFullInfo();

cout << "----- CHuman &ref2 = h2; -----" << endl;

CHuman& ref2 = h2;

ref2.ShowFullInfo();

cout << "----- CHuman &ref3 = h2; -----" << endl;

CHuman& ref3 = h2;

ref3.ShowFullInfo();

cout << "----- Creating dynamic objects -----" << endl;

cout << "----- CHuman\* ptr1 = new CPatient(\"Taras\", 25, \"cough\"); -----" << endl;

CHuman\* ptr1 = new CPatient("Taras", 22, "cough");

ptr1->SetAge(25);

ptr1->GetName();

ptr1->GetAge();

ptr1->ShowFullInfo();

cout << "----- CHuman\* ptr2 = new CDoctor(\"Oleg\", 23, \"endocrinologist\", 5); -----" << endl;

CHuman\* ptr2 = new CDoctor("Oleg", 0, "endocrinologist", 5);

ptr2->SetAge(23);

ptr2->GetName();

ptr2->GetAge();

ptr2->ShowFullInfo();

cout << "----- CHuman\* ptr3 = new CPatientdoctor(\"Yura\", 45, \"cold\",\"terapist\", 4); -----" << endl;

CHuman\* ptr3 = new CPatientDoctor("Yura", 0, "cold","terapist", 4);

ptr3->SetAge(45);

ptr3->GetName();

ptr3->GetAge();

ptr3->ShowFullInfo();

cout << "----- CPatient \*ptr4 = new CPatient(\"Ostap\", 19, \"sore throat\"); -----" << endl;

CPatient\* ptr4 = new CPatient("Ostap", 0, "sore throat");

ptr4->SetAge(19);

ptr4->GetName();

ptr4->GetAge();

ptr4->ShowFullInfo();

cout << "----- CDoctor \*ptr5 = new CDoctor(\"Igor\", 54, \"Surgeon\", 15); -----" << endl;

CDoctor\* ptr5 = new CDoctor("Igor", 52, "Surgeon", 15);

ptr5->SetAge(54);

ptr5->GetName();

ptr5->GetAge();

ptr5->ShowFullInfo();

cout << "----- CPatientDoctor \*ptr6 = new CPatientDoctor(\"Mark\", 33, \"fever\", \"Dermatologists\", 10); -----" << endl;

CPatientDoctor\* ptr6 = new CPatientDoctor("Mark", 33, "fever", "Dermatologists", 10);

ptr6->SetAge(33);

ptr6->GetName();

ptr6->GetAge();;

ptr6->ShowFullInfo();

delete ptr1;

delete ptr2;

delete ptr3;

delete ptr4;

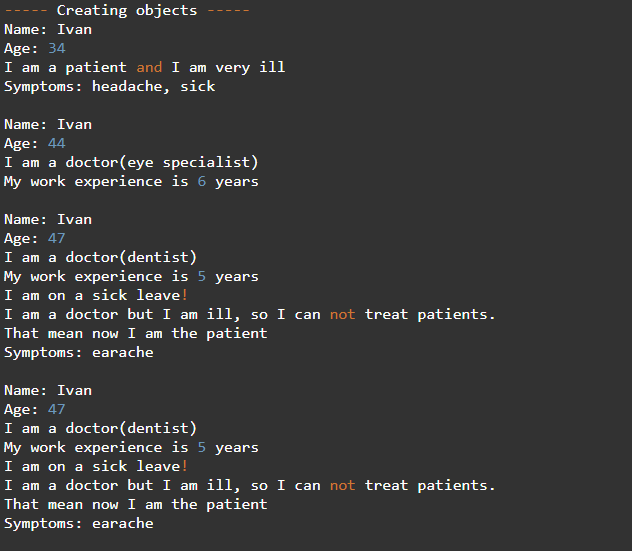
delete ptr5;

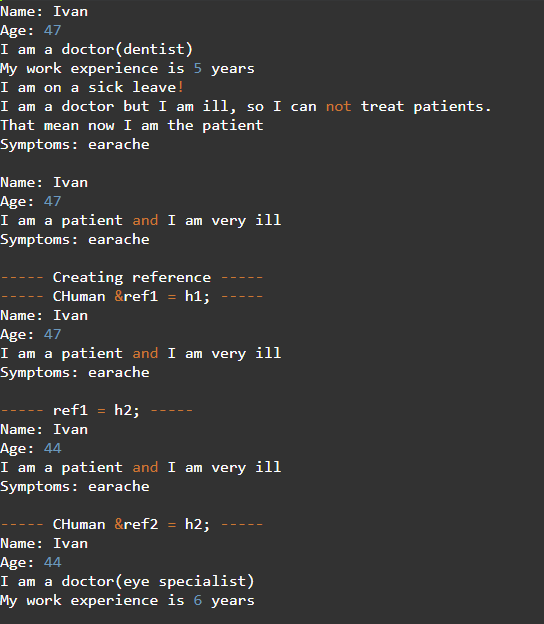
delete ptr6;

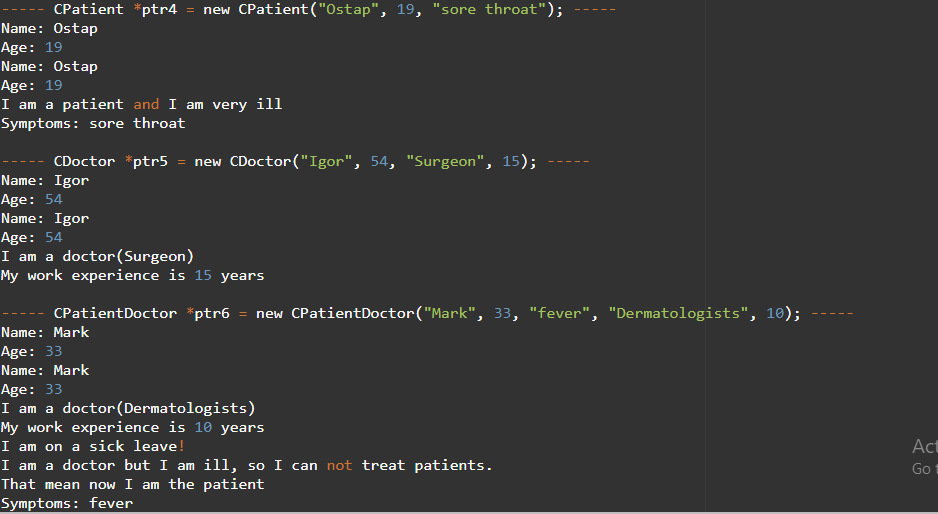
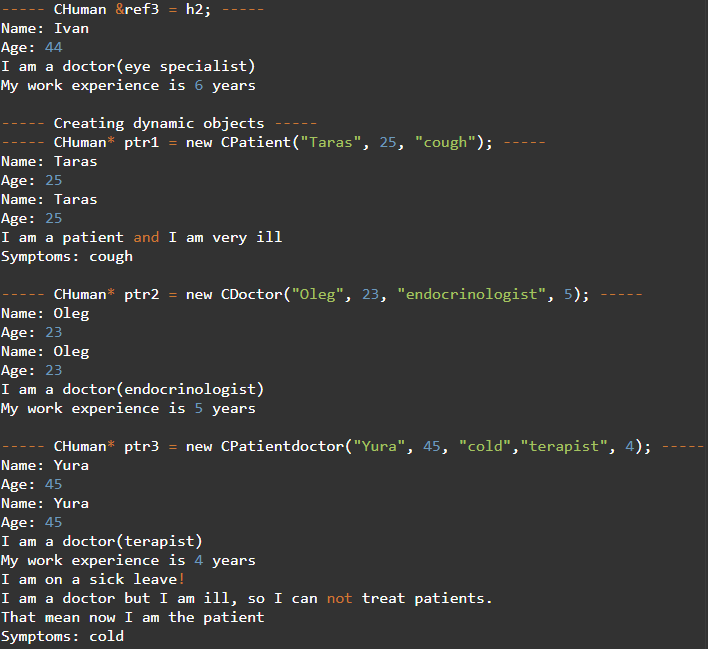
return 0;

}

*Вікно результату:*







*Висновок:* я познайомився з множинним спадкуванням класів та з поліморфізмом.