**Міністерство освіти і науки України**

**Львівський національний університет імені Івана Франка**

**Факультет електроніки та комп’ютерних технологій**

Звіт

про виконання лабораторної роботи №4

« Успадкування та поліморфізм»

Виконав :

Студент групи феп-12

Галяк Олег

Львів-2021

**Мета роботи:** Навчитись використовувати функціонал мови С++ для створення складних гіллястих ієрархій класів

**Завдання:** За аналогією до поданого прикладу розробити свою ієрархію класів,

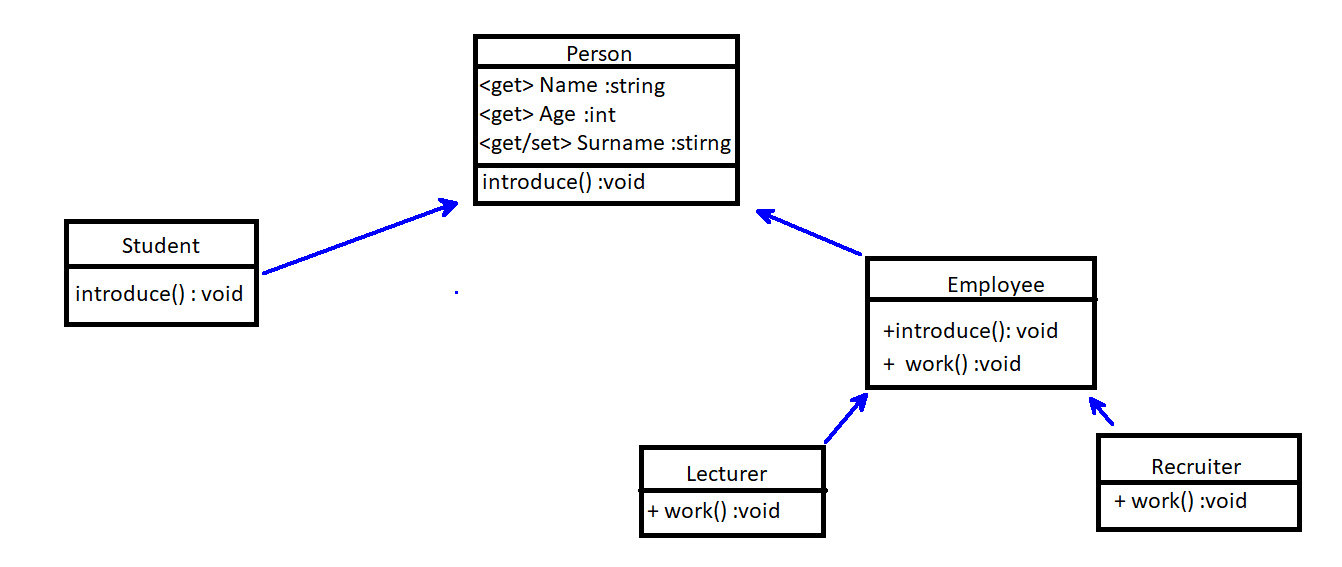
яка реалізує принципи успадкування та поліморфізму (мінімум 5 класів).

Використати перевизначення та перевантаження методів. Тематику обрати

самостійно.

**Виконання роботи**

Створив ієрархію класів із наступною структурою:



В класі Main написав функціонал для демонстрування поліморфізму для субкласів Person та субкласів Employee:

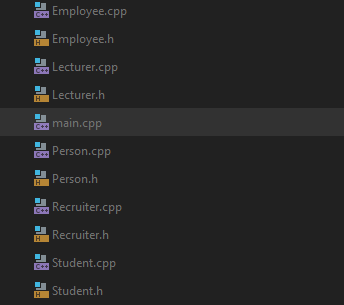


**Результати роботи**

**\*Примітка:** код цієї лабораторної роботи також доступний у моєму Git-репози-

торії за посиланням: <https://github.com/olehhaliak/oop/tree/master/OOP_lab_2/Lab_3>

Структура файлів:



Person.h:

*//  
// Created by olehh on 08-Mar-21.  
//*#ifndef **OOP\_LAB\_4\_PERSON\_H**#define **OOP\_LAB\_4\_PERSON\_H**#include "iostream"  
  
*class* Person {  
*public*:  
 Person();  
  
 Person(*const* std::string &name, *const* std::string &surname, *int* age);  
 std::string getName();  
 std::string getSurname();  
 *int* getAge();  
 *void* setSurname(std::string);  
 *virtual void* introduce();  
*private*:  
 std::string name;  
 std::string surname;  
 *int* age;  
};  
  
  
#endif *//OOP\_LAB\_4\_PERSON\_H*

Student.h :

*//  
// Created by olehh on 08-Mar-21.  
//*#ifndef **OOP\_LAB\_4\_STUDENT\_H**#define **OOP\_LAB\_4\_STUDENT\_H**#include "Person.h"  
  
*class* Student: *public* Person {  
*void* introduce() *override*;  
  
*public*:  
 Student(*const* std::string &name, *const* std::string &surname, *int* age, *int* yearOfEducation,  
 *const* std::string &university, *const* std::string &faculty, *const* std::string &speciality);  
  
 Student();  
  
  
*private*:  
 *int* yearOfEducation;  
 std::string university;  
 std::string faculty;  
 std::string speciality;  
};  
  
  
#endif *//OOP\_LAB\_4\_STUDENT\_H*

Employee:

*//  
// Created by olehh on 08-Mar-21.  
//*#ifndef **OOP\_LAB\_4\_STUDENT\_H**#define **OOP\_LAB\_4\_STUDENT\_H**#include "Person.h"  
  
*class* Student: *public* Person {  
*void* introduce() *override*;  
  
*public*:  
 Student(*const* std::string &name, *const* std::string &surname, *int* age, *int* yearOfEducation,  
 *const* std::string &university, *const* std::string &faculty, *const* std::string &speciality);  
  
 Student();  
  
  
*private*:  
 *int* yearOfEducation;  
 std::string university;  
 std::string faculty;  
 std::string speciality;  
};  
  
  
#endif *//OOP\_LAB\_4\_STUDENT\_H*

Lecturer.h:

*//  
// Created by olehh on 08-Mar-21.  
//*#ifndef **OOP\_LAB\_4\_LECTURER\_H**#define **OOP\_LAB\_4\_LECTURER\_H**#include "Employee.h"  
  
*class* Lecturer: *public* Employee {  
*public*:  
 Lecturer(*const* std::string &name, *const* std::string &surname, *int* age);  
  
 *void* work() *override*;  
};  
  
  
#endif *//OOP\_LAB\_4\_LECTURER\_H*

Recruiter.h:

*//  
// Created by olehh on 08-Mar-21.  
//*#ifndef **OOP\_LAB\_4\_RECRUITER\_H**#define **OOP\_LAB\_4\_RECRUITER\_H**#include "Employee.h"  
  
*class* Recruiter: *public* Employee {  
*public*:  
 Recruiter();  
  
 Recruiter(*const* std::string &name, *const* std::string &surname, *int* age);  
  
 *void* introduce() *override*;  
  
 *void* work() *override*;  
 *void* work(std::string ocupation);  
};  
  
  
#endif *//OOP\_LAB\_4\_RECRUITER\_H*

Person.cpp:

*//  
// Created by olehh on 08-Mar-21.  
//*#include "Employee.h"  
  
*void* Employee::work() {  
std::cout<<getName()<< "is working";  
}  
  
*void* Employee::introduce() {  
 std::cout<<"Good afternoon, my name is "<<getName()<<" "<<getSurname()<<" and I am Employee"<<std::endl;  
  
}  
  
Employee::Employee() {}  
  
Employee::Employee(*const* std::string &name, *const* std::string &surname, *int* age) : Person(name, surname, age) {}

Student.cpp:

*//  
// Created by olehh on 08-Mar-21.  
//*#include "Student.h"  
  
*void* Student::introduce() {  
 std::cout<<"Sup! I`m "<<getName()<<" "<<getSurname()<<", I`m "<<getAge()<<" y.o. "  
 <<"I`m studying in "<<university<<" at "<<faculty<<" faculty at "<<speciality<<std::endl;  
}  
  
Student::Student(*const* std::string &name, *const* std::string &surname, *int* age, *int* yearOfEducation,  
 *const* std::string &university, *const* std::string &faculty, *const* std::string &speciality) : Person(  
 name, surname, age), yearOfEducation(yearOfEducation), university(university), faculty(faculty), speciality(  
 speciality) {}  
  
Student::Student() : Person() ,  
 yearOfEducation(1),university("LNU"),faculty("electronics"),speciality("software engineering"){}

Employee.cpp:

*//  
// Created by olehh on 08-Mar-21.  
//*#include "Employee.h"  
  
*void* Employee::work() {  
std::cout<<getName()<< "is working";  
}  
  
*void* Employee::introduce() {  
 std::cout<<"Good afternoon, my name is "<<getName()<<" "<<getSurname()<<" and I am Employee"<<std::endl;  
  
}  
  
Employee::Employee() {}  
  
Employee::Employee(*const* std::string &name, *const* std::string &surname, *int* age) : Person(name, surname, age) {}

Lecturer.cpp:

*//  
// Created by olehh on 08-Mar-21.  
//*#include "Lecturer.h"  
  
*void* Lecturer::work() {  
 std::cout<<getName()<<" is educating students";  
}  
  
Lecturer::Lecturer(*const* std::string &name, *const* std::string &surname, *int* age) : Employee(name, surname, age) {}

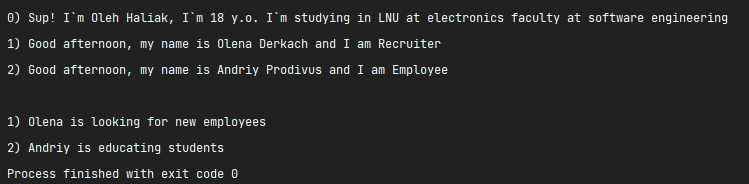
Recruiter.cpp:

*//  
// Created by olehh on 08-Mar-21.  
//*#include "Recruiter.h"  
  
*void* Recruiter::introduce() {  
 std::cout<<"Good afternoon, my name is "<<getName()<<" "<<getSurname()<<" and I am Recruiter"<<std::endl;  
  
}  
  
*void* Recruiter::work() {  
 std::cout<< getName()<<" is looking for new employees"<<std::endl;  
}  
  
*void* Recruiter::work(std::string ocupation) {  
 std::cout<< getName()<<" is looking for new "<<ocupation<<std::endl;  
}  
  
Recruiter::Recruiter() {}  
  
Recruiter::Recruiter(*const* std::string &name, *const* std::string &surname, *int* age) : Employee(name, surname, age) {}

Main.cpp:

#include <iostream>  
#include "Person.h"  
#include "Student.h"  
#include "Recruiter.h"  
#include "Lecturer.h"  
  
  
*int* main() {  
  
 Person \* person[3] = {  
 *new* Student(),  
 *new* Recruiter("Olena","Derkach",31),  
 *new* Lecturer("Andriy","Prodivus",49)  
 };  
  
 *for* (*int* i = 0; i < 3; ++i) {  
 std::cout<<i<<") ";  
 person[i]->introduce();  
 }  
 std::cout<<std::endl;  
 *for* (*int* i = 1; i < 3; ++i) {  
 std::cout<<i<<") ";  
 ( (Employee \*) person[i])->work();  
 }  
 *return* 0;  
}

Вивід (output) програми у консолі:



**Висновки:**

На цій лабораторній роботі я навчився використовувати функціонал мови С++ для створення складних гіллястих ієрархій класів.