**Міністерство освіти і науки України**

**Львівський національний університет імені Івана Франка**

**Факультет електроніки та комп’ютерних технологій**

Звіт

про виконання лабораторної роботи №5

« Множинне успадкування »

Виконав :

Студент групи Феп-13

Шилін Артур

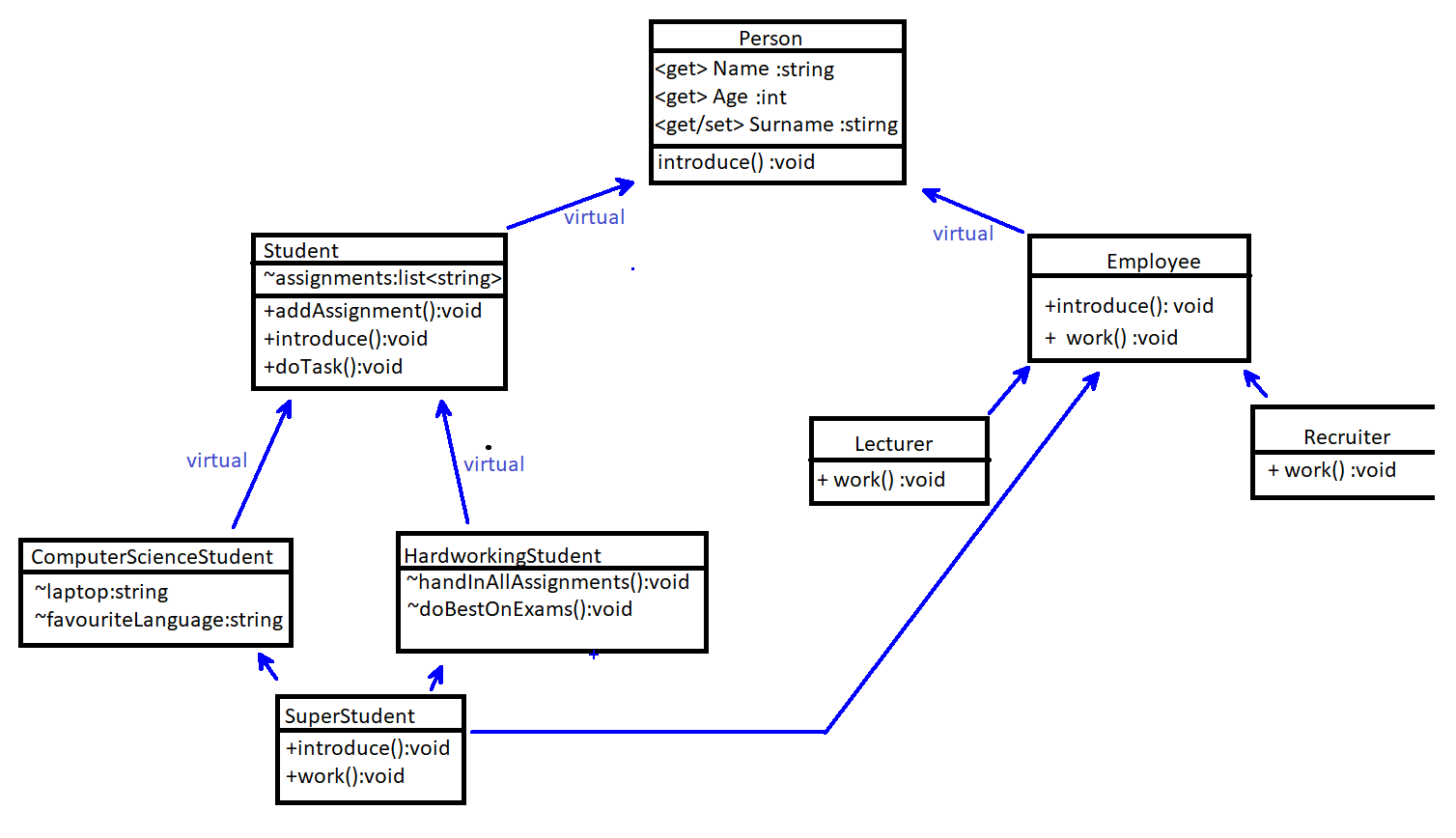
Львів-2021

**Мета роботи:** Проробити на практиці можливі варіанти множинного успадкування та методи уникнення проблем із ним.

**Завдання:** : До створеної ієрархії класів у лаб. 4. добавити класи таким чином, щоб отримати множинне успадкування. Переробити класи так, щоб отримати ромбовидне успадкування. Класи мають мати окрім конструкторів і звичайних методів ще й деструктори. Навчитись керувати порядком виклику конструкторів та деструкторів класів.

**Виконання роботи**

Обновив структуру класів із попередньої лабораторної роботи: додав класи із множинним спадкуванням (від трьох суперкласів) та із ромбовидним спадкуванням:



Щоб уникнути проблем, характерних для ромбовидного наслідування

(а саме дублювання суперкласів), додав модифікатор virtual при наслідуванні від «проблемних» класів – Student і Person.

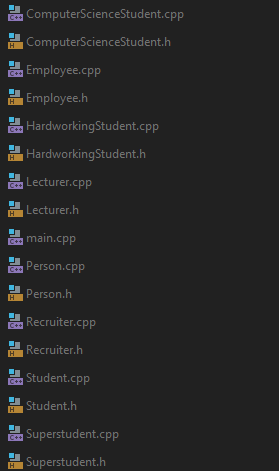
Додав конструктори да деструктори за замовчуванням, при виклику яких, у консоль виводиться повідомлення про створення/знищення об’єктів.

**Клас SuperStudent успадковується від трьох батьківських класів, а саме: ComputerScienceStudent, HardworkingStudent, Employee . Це можна помітити, глянувши:**

1. **На діаграму класів**
2. **На код програми**

**Результати роботи**

Список файлів:



*//  
// Created by artur on 07-June-21.  
//*#include "ComputerScienceStudent.h"  
  
ComputerScienceStudent::ComputerScienceStudent(): laptop("Lenovo"),favouriteLanguage("of course ukrainian!") {  
  
 std::cout<<"Computer science student created"<<std::endl;  
}  
  
ComputerScienceStudent::~ComputerScienceStudent() {  
 cout << "ComputerScienceStudent was deleted"<<std::endl;  
  
}  
  
ComputerScienceStudent::ComputerScienceStudent(Person \*person, *const* list<string> &assignments, *const* string &laptop,  
 *const* string &favouriteLanguage) :  
 Student(person, assignments),  
 laptop(laptop), favouriteLanguage(  
 favouriteLanguage) {}

*//  
// Created by artur on 07-June-21.  
//*#ifndef **OOP\_LAB\_4\_COMPUTERSCIENCESTUDENT\_H**#define **OOP\_LAB\_4\_COMPUTERSCIENCESTUDENT\_H**#include "Student.h"  
  
*class* ComputerScienceStudent: *public virtual* Student {  
*protected*:  
 string laptop;  
 string favouriteLanguage;  
*public*:  
 ComputerScienceStudent(Person \*person, *const* list<string> &assignments, *const* string &laptop,  
 *const* string &favouriteLanguage);  
  
 ComputerScienceStudent();  
  
 *virtual* ~ComputerScienceStudent();  
};  
  
  
#endif *//OOP\_LAB\_4\_COMPUTERSCIENCESTUDENT\_H*

*//  
// 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() {  
 std::cout<<"Employee was created"<<std::endl;  
}  
  
Employee::~Employee() {  
std::cout<<"Employee was deleted"<<std::endl;  
}  
  
Employee::Employee(*const* std::string &name, *const* std::string &surname, *int* age) : Person(name, surname, age) {}  
  
Employee::Employee(*const* std::string &name, *const* std::string &surname, *int* age, *int* sallary) : Person(name, surname,  
 age),  
 sallary(sallary) {}

*//  
// Created by artur on 07-June-21.  
//*#ifndef **OOP\_LAB\_4\_EMPLOYEE\_H**#define **OOP\_LAB\_4\_EMPLOYEE\_H**#include "Person.h"  
  
*const int* MIN\_SALLARY = 6000;  
  
*class* Employee: *public virtual* Person {  
*public*:  
 *int* sallary = MIN\_SALLARY;  
 Employee();  
  
 Employee(*const* std::string &name, *const* std::string &surname, *int* age, *int* sallary);  
  
 Employee(*const* std::string &name, *const* std::string &surname, *int* age);  
 *virtual void* work();  
  
 *virtual* ~Employee();  
  
*private*:  
 *void* introduce() *override*;  
  
};  
  
  
#endif *//OOP\_LAB\_4\_EMPLOYEE\_H*

*//  
// Created by artur on 07-June-21.  
//*#include "HardworkingStudent.h"  
  
*void* HardworkingStudent::handInAllAssignments()  
{  
 *while* (!assignments.empty()) {  
 doTask();  
 }  
 cout << "All assignments done!";  
}  
  
HardworkingStudent::HardworkingStudent() {  
 std::cout<<"Hardworking student created"<<std::endl;  
}  
  
HardworkingStudent::~HardworkingStudent() {  
  
 cout << "HardWorkingStudent was deleted"<<std::endl;  
}  
  
HardworkingStudent::HardworkingStudent(Student \*student) : Student(student) {}  
  
*void* HardworkingStudent::doBestOnExams() {  
 cout << "Exam result is 100/100. Easy!";  
}

*//  
// Created by artur on 07-June-21.  
//*#ifndef **OOP\_LAB\_4\_HARDWORKINGSTUDENT\_H**#define **OOP\_LAB\_4\_HARDWORKINGSTUDENT\_H**#include "Student.h"  
  
*class* HardworkingStudent : *public virtual* Student{  
*public*:  
 HardworkingStudent();  
  
 HardworkingStudent(Student \*student);  
  
 *virtual* ~HardworkingStudent();  
  
*protected*:  
 *void* handInAllAssignments();  
 *void* doBestOnExams();  
};  
  
  
#endif *//OOP\_LAB\_4\_HARDWORKINGSTUDENT\_H*

*//  
// Created by artur on 07-June-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) {}

*//  
// Created by artur on 07-June-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*

#include <iostream>  
#include "Person.h"  
#include "Student.h"  
#include "Recruiter.h"  
#include "Lecturer.h"  
#include "Superstudent.h"  
  
*using namespace* std;  
*int* main() {  
 Superstudent \*student = *new* Superstudent();  
 cout<<endl;  
 student->introduce();  
 student->work();  
 cout<<endl;  
 *delete*(student);  
}

*//  
// Created by artur on 07-June-21.  
//*#include <iostream>  
#include "Person.h"  
  
Person::Person(*const* std::string &name, *const* std::string &surname, *int* age) : name(name), surname(surname), age(age) {}  
  
std::string Person::getName() {  
 *return* name;  
}  
  
std::string Person::getSurname() {  
 *return* surname;  
}  
  
*int* Person::getAge() {  
 *return* age;  
}  
  
*void* Person::setSurname(std::string newSurname) {  
 std::cout << "Surname was changed to " << newSurname << " !" << std::endl;  
 surname = newSurname;  
}  
  
*void* Person::introduce() {  
 std::cout << "Hello, my name is " << name << " " << surname << " and I am " << age << " years old.";  
}  
  
Person::Person() : name("Artur"), surname("Shylin"), age(18) {  
 std::cout<<"Person created"<<std::endl;  
}  
  
Person::~Person() {  
 std::cout << "Person was deleted"<<std::endl;  
  
}  
  
Person::Person(Person\* person) : name(person->getName()), surname(person->getSurname()), age(person->age) {  
  
}

*//  
// Created by artur on 07-June-21.  
//*#ifndef **OOP\_LAB\_4\_PERSON\_H**#define **OOP\_LAB\_4\_PERSON\_H**#include "iostream"  
  
*class* Person {  
*public*:  
 Person();  
  
 Person(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();  
  
 *virtual* ~Person();  
  
*private*:  
 std::string name;  
 std::string surname;  
 *int* age;  
};  
  
  
#endif *//OOP\_LAB\_4\_PERSON\_H*

*//  
// Created by artur on 07-June-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) {}

*//  
// Created by artur on 07-June-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*

*//  
// Created by artur on 07-June-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*

*//  
// Created by artur on 07-June-21.  
//*#ifndef **OOP\_LAB\_4\_STUDENT\_H**#define **OOP\_LAB\_4\_STUDENT\_H**#include "Person.h"  
#include "list"  
  
*using namespace* std;  
  
*class* Student : *public virtual* Person {  
  
*public*:  
 Student();  
  
 Student(Student \*);  
  
  
 Student(Person \*person, *const* list<string> &assignments);  
  
 *void* addAssignment(string assignment);  
  
 *void* doTask();  
  
 *virtual* ~Student();  
  
*protected*:  
  
 *void* introduce() *override*;  
  
 list<string> assignments;  
  
  
};  
  
  
#endif *//OOP\_LAB\_4\_STUDENT\_H*

*//  
// Created by artur on 07-June-21.  
//*#include "Superstudent.h"  
  
Superstudent::Superstudent() {  
 cout << "Superstudent was created"<<std::endl;  
}  
  
Superstudent::~Superstudent() {  
 cout << "Superstudent was deleted"<<std::endl;  
}  
  
*void* Superstudent::introduce() {  
 cout << "Hello, my name is" << getName()<<" "<< getSurname() << ", I`m" << getAge() <<  
 " y. o.." << "My favourite language is: " << favouriteLanguage << endl;  
}  
  
*void* Superstudent::work() {  
 cout<<"Working as a cashier in McDonalds..."<<endl;  
}

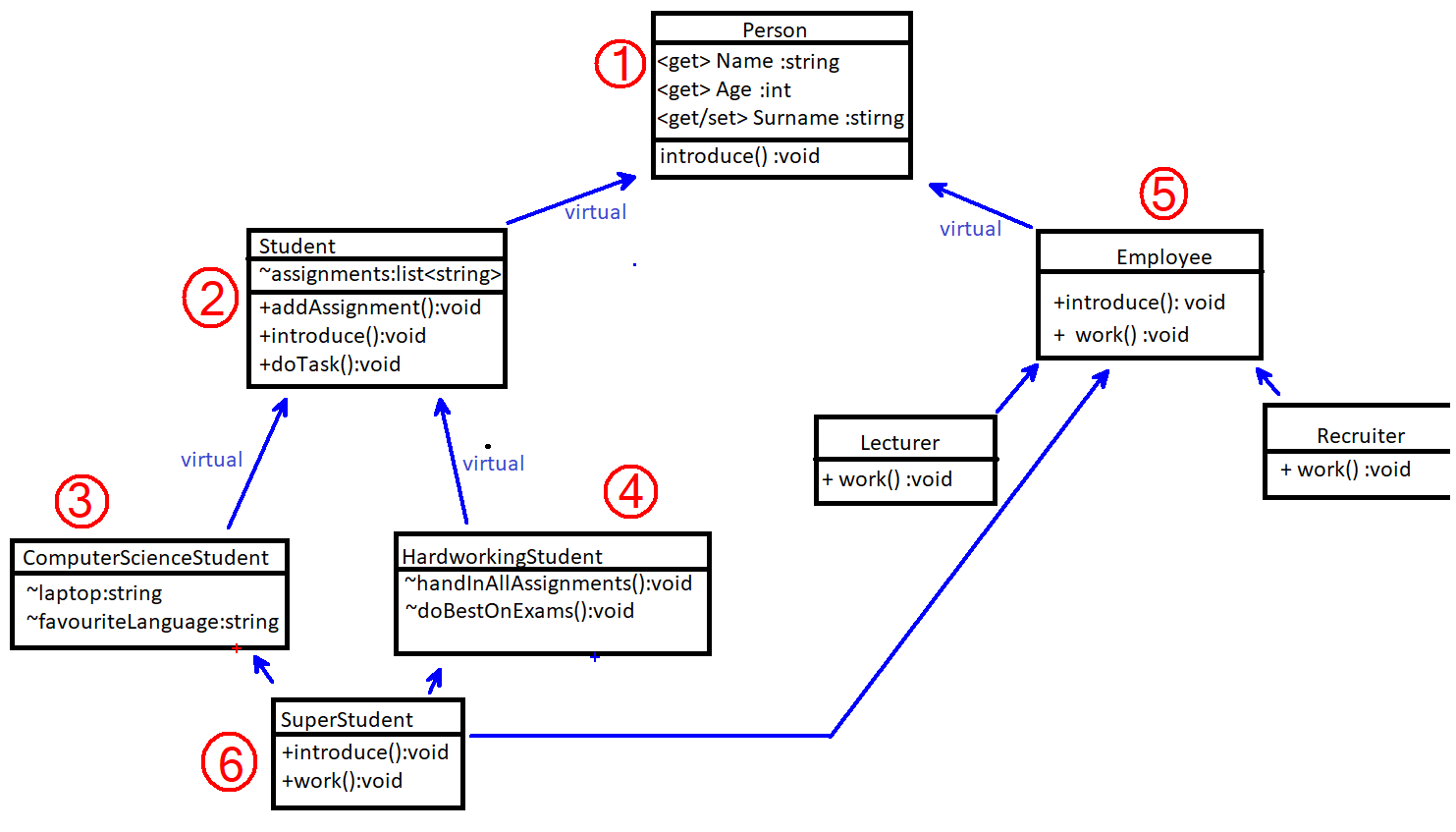
*//  
// Created by artur on 07-June-21.  
//*#ifndef **OOP\_LAB\_4\_SUPERSTUDENT\_H**#define **OOP\_LAB\_4\_SUPERSTUDENT\_H**#include "ComputerScienceStudent.h"  
#include "HardworkingStudent.h"  
#include "Employee.h"  
  
*class* Superstudent : *public* ComputerScienceStudent, *public* HardworkingStudent, *public* Employee{  
*public*:  
 Superstudent();  
  
 *void* introduce() *override*;  
  
 *virtual* ~Superstudent();  
  
 *void* work() *override*;  
};  
  
  
#endif *//OOP\_LAB\_4\_SUPERSTUDENT\_H*

Як ми можемо помітити, при створенні класу Superstudent(**SuperStudent – клас, що успадковується від трьох батьківських класів, що можна помітити по скріншоту, розміщеному на 5 рядків нижче**), спочатку створюються об’єкт суперкласу найвищого рівня – Person.

Коли ж у нас йде «розгалуження» класів із одним рівнем, першим викличеться конструктор об’єкту, котрий був зазначений першим у списку суперкласів. В моєму випадку список наслідуваних класів наступний:



Отже, конструктори будуть викликані в такому порядку:



При знищені об’єкту Superstudent, порядок виклику деструкторів буде абсолютно протилежний:

* Починаючи від субкласів
* Якщо декілька класів з одним рівнем абстракції – вибір останнього в списку.

**Висновки:**

На цій лабораторній роботі, я вивчив та закріпив на практиці поняття множинного спадкування, зрозумів та навчився вирішувати проблему «ромбоподібного наслідування».