Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: Об’єктно-орієнтоване програмування

**Лабораторна робота №3**

**Тема:** **«**Успадкування, агрегація та композиція**»**

|  |
| --- |
| Виконав: ст. гр. КБ-23 |
| Захарченко А.О. |
| Перевірив: викладач  Козірова Наталія Леонідівна |
|  |

Кропивницький

2024

**Тема:** **Успадкування, агрегація та композиція**

**Мета: Ознайомитись з основними поняттями успадкування, агрегація та**

**композиція в ООП та навчитись їх програмно реалізовувати мовою С++.**

**Завдання: Варіант 3**

**1. Створіть клас "Компанія", який містить такі властивості:**

**• Назва компанії**

**• Адреса компанії**

**• Список підрозділів, які входять до складу компанії (використовуйте**

**агрегацію)**

**• Головний директор компанії (використовуйте композицію)**

**2. Створіть клас "Підрозділ", який містить такі властивості:**

**• Назва підрозділу**

**• Кількість працівників**

**• Керівник підрозділу**

**3. Створіть клас "Працівник", який містить такі властивості:**

**• Прізвище та ім'я працівника**

**• Посада працівника**

**4. Реалізуйте методи для додавання та видалення підрозділів зі списку**

**підрозділів компанії.**

**5. Напишіть демонстраційну функцію main(), в якій створюються об'єкти**

**класів "Компанія", "Підрозділ" та "Працівник". Додайте підрозділи до списку**

**підрозділів компанії та виведіть інформацію про них.**

**6. Продемонструйте успадкування, створивши похідний клас від**

**"Працівник" (наприклад, "Менеджер") з додатковими властивостями та**

**методами.**

**7. Розширте функціональність, додавши додаткові методи та властивості**

**до класів за власним бажанням.**

**Не забудьте додати коментарі та пояснення до коду, щоб роз'яснити його**

**функціональність та зрозумілість.**

**Лістинг:**

**Source.cpp**

#include "Company.h"

#include <iostream>

#include <string>

int main()

{

Company company("Atlas Corp.", "Bubalvar 13", "Daks Hendrik");

company.addUnit("LTR");

company.addUnit("SNF");

company.addUnit("PATRN");

SupUnit LTR("LTR", 25, "John Fardt");

company.addSupUnit(LTR);

SupUnit SNF("SNF", 30, "Ionna Hord");

company.addSupUnit(SNF);

company.addWorker("Kart", "Sovan", "Tester", 60000);

company.addWorker("Bark", "Josha", "Tech-Support", 50000);

company.addManager("Linda", "Dorba", "Head-officer", 90000, 10);

company.showInfo();

return 0;

}

**Company.cpp**

#include "Company.h"

#include <iostream>

#include <string>

using namespace std;

//Отримання імені директора

Director::Director(const string& name, int age) : directorName(name) {}

string Director::getName() const

{

return directorName;

}

//Отримання іменя юніта

Unit::Unit(const string& Name) : unitName(Name) {}

string Unit::getUnitName() const

{

return unitName;

}

//Отримання імені копанії

Company::Company(const string& Name, const string& Address, const string& directorName) : companyName(Name), companyAddress(Address)

{

director = new Director(directorName, 0);

}

//Деструктор

Company::~Company()

{

delete director;

}

//Додавання юніта

void Company::addUnit(const string& unitName)

{

units.push\_back(Unit(unitName));

}

//Додавання сапюніта

void Company::addSupUnit(const SupUnit supunit)

{

supUnits.push\_back(supunit);

}

//Доадавання працівника та менеджера

void Company::addWorker(const string& name, const string& surname, const string& position, int salary)

{

Worker newWorker(name, surname, position, salary);

workers.push\_back(newWorker);

}

void Company::addManager(const string& name, const string& surname, const string& position, int salary, int NewHuman)

{

Manager newManager(name, surname, position, salary, NewHuman);

managers.push\_back(newManager);

}

//Task-2

//отримання інфи про все

SupUnit::SupUnit(const string SupUnitName, int Employees, const string HeadUnit) : supUnitName(SupUnitName), Employees(Employees), HeadUnit(HeadUnit) {}

string SupUnit::getSupUnitName() const

{

return supUnitName;

}

int SupUnit::getEmployees() const

{

return Employees;

}

string SupUnit::getHeadUnit() const

{

return HeadUnit;

}

//Task-3

//інфа про працівників

Worker::Worker(const string& WorkerName, const string& WorkerSurname, const string& WorkerPosition, int WorkerSalary)

: WorkerName(WorkerName), WorkerSurname(WorkerSurname), WorkerPosition(WorkerPosition), WorkerSalary(WorkerSalary) {}

string Worker::getWorkerName() const

{

return WorkerName;

}

string Worker::getWorkerSurname() const

{

return WorkerSurname;

}

string Worker::getWorkerPosition() const

{

return WorkerPosition;

}

int Worker::getWorkerSalary() const

{

return WorkerSalary;

}

//вивід інфи у консоль

void Company::showInfo() const

{

cout << "Company name: " << companyName << endl;

cout << "Company address: " << companyAddress << endl;

cout << "Director: " << director->getName() << endl;

cout << "Units: " << endl;

for (const auto& unit : units)

{

cout << "- " << unit.getUnitName() << endl;

}

cout << "SupUnits: " << endl;

for (const auto& supUnit : supUnits)

{

cout << "- " << supUnit.getSupUnitName()<< "\nEmployees: " << supUnit.getEmployees()<< "\nHeadUnit : " << supUnit.getHeadUnit() << endl;

}

cout << "Workers: " << endl;

for (const auto& worker : workers)

{

cout << "- " << worker.getWorkerName() << " " << worker.getWorkerSurname() << " Position: " << worker.getWorkerPosition() <<" Salary:" << worker.getWorkerSalary() << endl;

}

cout << "Managers: " << endl; // Додаємо інформацію про менеджерів

for (const auto& manager : managers) {

cout << "- " << manager.getWorkerName() << " " << manager.getWorkerSurname() << " Position: " << manager.getWorkerPosition() << " Salary: " << manager.getWorkerSalary() << " New Humans: " << manager.getNewHuman() << endl;

}

}

**Company.h**

#pragma once

#ifndef COMPANY\_H

#define COMPANY\_H

#include <string>

#include <vector>

using namespace std;

class SupUnit {

private:

string supUnitName;

int Employees;

string HeadUnit;

public:

SupUnit(const string supUnitName, int Employees, const string HeadUnit);

string getSupUnitName() const;

int getEmployees() const;

string getHeadUnit() const;

};

class Director {

private:

string directorName;

int directorAge;

public:

Director(const string& name, int age);

string getName() const;

};

class Unit {

private:

string unitName;

public:

Unit(const string& unitName);

string getUnitName() const;

};

class Worker {

private:

string WorkerName;

string WorkerSurname;

string WorkerPosition;

int WorkerSalary;

public:

Worker(const string& WorkerName, const string& WorkerSurname, const string& WorkerPosition, int WorkerSalary);

string getWorkerName() const;

string getWorkerSurname() const;

string getWorkerPosition() const;

int getWorkerSalary() const;

};

//клас менеджер наслідується від класу працівник

class Manager : public Worker

{

private:

int NewHuman;

public:

Manager(const string& WorkerName, const string& WorkerSurname, const string& WorkerPosition, int WorkerSalary, int NewHuman) : Worker(WorkerName, WorkerSurname, WorkerPosition, WorkerSalary), NewHuman(NewHuman) {}

int getNewHuman() const

{

return NewHuman;

}

};

class Company {

private:

string companyName;

string companyAddress;

vector<Unit> units;

Director\* director;

vector<SupUnit> supUnits;

vector<Worker> workers;

vector<Manager> managers;

public:

Company(const string& Name, const string& Address, const string& directorName);

~Company();

void addUnit(const string& unitName);

void addSupUnit(const SupUnit supunit);

void addWorker(const string& name, const string& surname, const string& position, int salary);

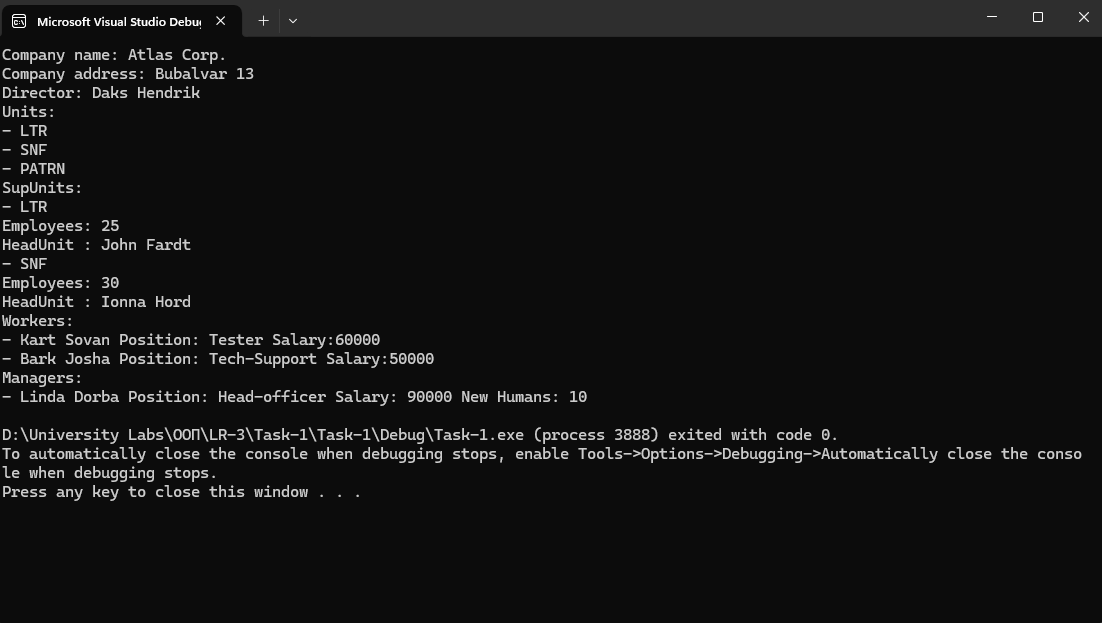
void addManager(const string& name, const string& surname, const string& position, int salary, int NewHuman);

void showInfo() const;

};

#endif

**Результат:**

****

**Завдання 2:**

**Створіть використовуючи композицію два класи, які матимуть свої**

**властивості, перший реалізуйте за допомогою включення (composition), а**

**другий за допомогою посилання (reference), контейнером для цих класів**

**використайте клас з лабораторної роботи №1, створіть також третій –**

**похідний клас, використовуючи наслідування, базовим класом може бути або**

**клас з лр№1, або класи які ви створили за допомогою композиції. За**

**бажанням можете зобразити схематично як ваші класи залежать один від**

**одного.**

**Person.cpp:**

#include "Person.h"

void Person::setname(const string& name) {

this->name = name;

}

void Person::setage(int age) {

this->age = age;

}

void Person::setaddress(const string& address) {

this->address = address;

}

string Person::getname() const {

return name;

}

int Person::getage() const {

return age;

}

string Person::getaddress() const {

return address;

}

void Passport::setPassportNumber(const string& passportNumber) {

this->passportNumber = passportNumber;

}

void Passport::setNationality(const string& nationality) {

this->nationality = nationality;

}

string Passport::getPassportNumber() const {

return passportNumber;

}

string Passport::getNationality() const {

return nationality;

}

Job::Job(const string& jobTitle, double salary) : jobTitle(jobTitle), salary(salary) {}

string Job::getJobTitle() const {

return jobTitle;

}

double Job::getSalary() const {

return salary;

}

void Job::setJobTitle(const string& jobTitle) {

this->jobTitle = jobTitle;

}

void Job::setSalary(double salary) {

this->salary = salary;

}

Employee::Employee(const Person& person, const Passport& passport, Job& job)

: person(person), passport(passport), job(job) {}

Person Employee::getPerson() const {

return person;

}

Passport Employee::getPassport() const {

return passport;

}

Job& Employee::getJob() const {

return job;

}

Manager::Manager(const Person& person, const Passport& passport, Job& job, int teamSize)

: Employee(person, passport, job), teamSize(teamSize) {}

int Manager::getTeamSize() const {

return teamSize;

}

void Manager::setTeamSize(int teamSize) {

this->teamSize = teamSize;

}

**Person.h:** #pragma once

#ifndef PERSON\_H

#define PERSON\_H

#include <string>

using namespace std;

class Person {

private:

string name;

int age;

string address;

public:

void setname(const string& name);

void setage(int age);

void setaddress(const string& address);

string getname() const;

int getage() const;

string getaddress() const;

};

class Passport {

private:

string passportNumber;

string nationality;

public:

void setPassportNumber(const string& passportNumber);

void setNationality(const string& nationality);

string getPassportNumber() const;

string getNationality() const;

};

class Job {

private:

string jobTitle;

double salary;

public:

Job(const string& jobTitle, double salary);

string getJobTitle() const;

double getSalary() const;

void setJobTitle(const string& jobTitle);

void setSalary(double salary);

};

// Клас Employee (контейнер для Passport і Job)

class Employee {

private:

Person person;

Passport passport;

Job& job;

public:

Employee(const Person& person, const Passport& passport, Job& job);

Person getPerson() const;

Passport getPassport() const;

Job& getJob() const;

};

class Manager : public Employee {

private:

int teamSize;

public:

Manager(const Person& person, const Passport& passport, Job& job, int teamSize);

int getTeamSize() const;

void setTeamSize(int teamSize);

};

#endif

**Task-1.cpp:**

#include <iostream>

#include "Person.h"

using namespace std;

int main() {

Person person;

person.setname("John Doe");

person.setage(30);

person.setaddress("123 Elm Street");

Passport passport;

passport.setPassportNumber("A1234567");

passport.setNationality("American");

Job job("Software Engineer", 85000);

Employee employee(person, passport, job);

Manager manager(person, passport, job, 10);

cout << "Employee Name: " << employee.getPerson().getname() << endl;

cout << "Passport Number: " << employee.getPassport().getPassportNumber() << endl;

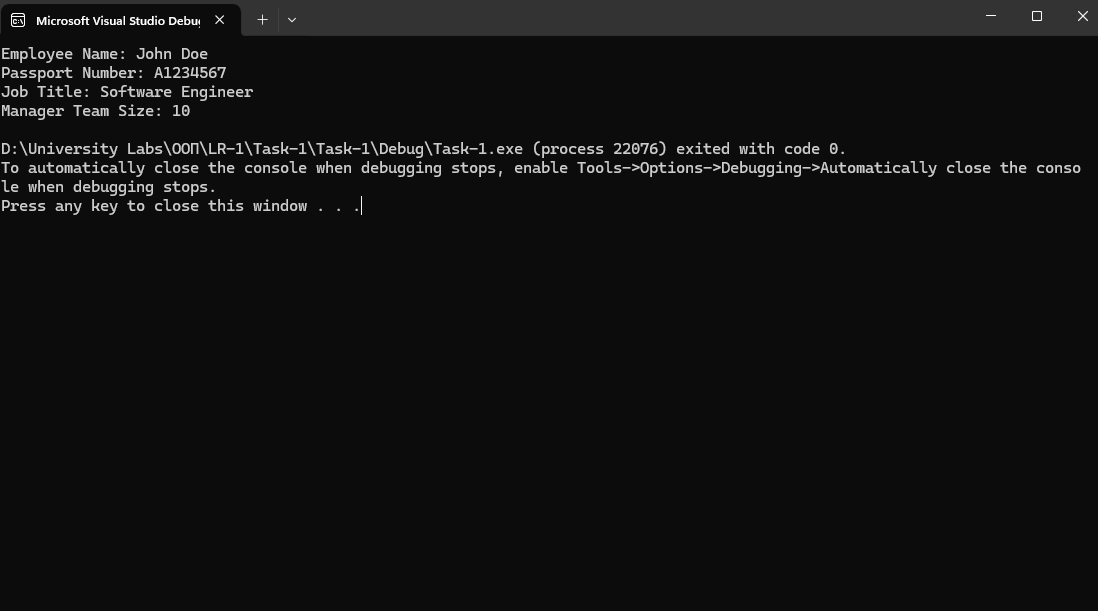
cout << "Job Title: " << employee.getJob().getJobTitle() << endl;

cout << "Manager Team Size: " << manager.getTeamSize() << endl;

return 0;

}

**Резульат:**

****

**Висновок:**

В ході виконання ЛР-3 було опановано поняття термінів:агрегація:композиція та наслідування. Перераховані терміни були використані в ході виконання роботи.

Також було покращено навички написання коду на мові програмування С++. Було використано додаткові джерела інформації для виконання поставлених задач.

Також хочу зауважити, щ були застосовані параметризовані констуктори.