Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №3

з дисципліни «Об’єктно-орієнтоване програмування»

**Тема:** Успадкування, агрегація та композиція

Виконав: студент гр. КН-23

**Чубенко.І.К**

Перевірив:Асистент  
**Козірова Н.Л.**

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 1**

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.

**Завдання 2**

**Варіант 6**

1. Створіть клас "Університет", який містить такі властивості:

• Назва університету

• Адреса університету

• Список факультетів, які належать до університету (використовуйте

агрегацію)

2. Створіть клас "Факультет", який містить такі властивості:

• Назва факультету

• Декан факультету (використовуйте композицію)

• Список студентів, які навчаються на факультеті (використовуйте

агрегацію)

3. Створіть клас "Декан", який містить такі властивості:

• Ім'я декана

• Рік призначення на посаду декана

4. Реалізуйте методи для додавання та видалення студентів зі списку

студентів факультету.

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

класів "Університет", "Факультет" та "Декан". Додайте факультети до списку

факультетів університету, додайте студентів до списку студентів факультету

та виведіть інформацію про них.

6. Продемонструйте успадкування, створивши похідний клас від

"Студент" (наприклад, "Магістр") 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

Завдання 1

#include <iostream>

#include <Windows.h>

#include "Animal.h"

using namespace std;

class Zoo {

private:

Animal animal; // Композиція - об'єкт класу Animal є частиною Zoo

public:

Zoo(string name, string species, string age) {

animal.setName(name);

animal.setSpecies(species);

animal.setAge(age);

}

void showAnimalInfo() {

cout << "Тварина в зоопарку: " << animal.getName() << ", "

<< animal.getSpecies() << ", " << animal.getAge() << endl;

}

};

class PetOwner {

private:

Animal& pet; // Посилання на об'єкт Animal

public:

PetOwner(Animal& animal) : pet(animal) {}

void showPetInfo() {

cout << "Тварина власника: " << pet.getName() << ", "

<< pet.getSpecies() << ", " << pet.getAge() << endl;

}

};

// Похідний клас, що використовує наслідування

class Dog : public Animal {

private:

string breed;

public:

Dog(string name, string age, string breed) {

setName(name);

setSpecies("Собака");

setAge(age);

this->breed = breed;

}

void showDogInfo() {

cout << "Собака: " << getName() << ", Порода: " << breed << ", Вік: " << getAge() << endl;

}

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Zoo zooAnimal("Слон", "Ссавець", "25");

zooAnimal.showAnimalInfo();

Animal myPet;

myPet.setName("Buddy");

myPet.setSpecies("Собака");

myPet.setAge("5");

PetOwner owner(myPet);

owner.showPetInfo();

Dog myDog("Rex", "3", "Німецька вівчарка");

myDog.showDogInfo();

return 0;

}

Animal.h

#ifndef ANIMAL\_H

#define ANIMAL\_H

#include <string>

using namespace std;

class Animal {

private:

string name;

string species;

string age;

public:

void setName(string n) { name = n; }

string getName() const { return name; }

void setSpecies(string s) { species = s; }

string getSpecies() { return species; }

void setAge(string a) { age = a; }

string getAge() { return age; }

};

#endif

Завдання 2

#include <iostream>

#include <Windows.h>

#include <vector>

#include <string>

using namespace std;

class Dean {

private:

string name;

int yearAppointed;

public:

Dean(string name, int yearAppointed) : name(name), yearAppointed(yearAppointed) {}

string getName() const { return name; }

int getYearAppointed() const { return yearAppointed; }

};

class Student {

protected:

string name;

int id;

public:

Student(string name, int id) : name(name), id(id) {}

string getName() const { return name; }

int getId() const { return id; }

};

// Похідний клас "Магістр" від "Студент"

class MasterStudent : public Student {

private:

string thesisTopic;

public:

MasterStudent(string name, int id, string thesisTopic) : Student(name, id), thesisTopic(thesisTopic) {}

// Метод для отримання теми дисертації

string getThesisTopic() const { return thesisTopic; }

};

"

class Faculty {

private:

string name;

Dean dean;

vector<Student\*> students;

public:

Faculty(string name, Dean dean) : name(name), dean(dean) {}

// Методи для додавання та видалення студентів

void addStudent(Student\* student) {

students.push\_back(student);

}

void removeStudent(int id) {

for (auto it = students.begin(); it != students.end(); ++it) {

if ((\*it)->getId() == id) {

students.erase(it);

break;

}

}

}

// Методи для отримання інформації про факультет

string getName() const { return name; }

Dean getDean() const { return dean; }

// Метод для виведення інформації про студентів

void showStudents() const {

cout << "Студенти факультету " << name << ":" << endl;

for (const auto& student : students) {

cout << "Ім'я: " << student->getName() << ", ID: " << student->getId() << endl;

}

}

};

class University {

private:

string name;

string address;

vector<Faculty\*> faculties;

public:

University(string name, string address) : name(name), address(address) {}

// Методи для додавання та видалення факультетів

void addFaculty(Faculty\* faculty) {

faculties.push\_back(faculty);

}

void removeFaculty(string name) {

for (auto it = faculties.begin(); it != faculties.end(); ++it) {

if ((\*it)->getName() == name) {

faculties.erase(it);

break;

}

}

}

// Методи для отримання інформації про університет

string getName() const { return name; }

string getAddress() const { return address; }

// Метод для виведення інформації про факультети

void showFaculties() const {

cout << "Факультети університету " << name << ":" << endl;

for (const auto& faculty : faculties) {

cout << "Факультет: " << faculty->getName() << ", Декан: " << faculty->getDean().getName() << endl;

}

}

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Dean dean("Іван Іванов", 2015);

Faculty faculty("Інформатика", dean);

Student student1("Петро Петров", 1);

Student student2("Марія Іванова", 2);

//Успадкування, створивши похідний клас

MasterStudent masterStudent("Олексій Сидоров", 3, "Розробка штучного інтелекту");

// Додавання студентів до факультету

faculty.addStudent(&student1);

faculty.addStudent(&student2);

faculty.addStudent(&masterStudent);

University university("Центральний національний університет", "вул. Головна, 1");

university.addFaculty(&faculty);

cout << "Інформація про університет:" << endl;

cout << "Назва: " << university.getName() << ", Адреса: " << university.getAddress() << endl;

university.showFaculties();

faculty.showStudents();

faculty.removeStudent(2);

cout << "\nПісля видалення студента з ID 2:" << endl;

faculty.showStudents();

return 0;

}

Висновок

У даній лабораторній роботі успішно реалізовані три основні концепції об'єктно-орієнтованого програмування (ООП): **композиція**, **агрегація** та **успадкування**. Кожен із підходів був продемонстрований у відповідних завданнях.

1. **Композиція**:
   * Створено класи, де один об'єкт включається в інший. Наприклад, клас **"Факультет"** включає об'єкт класу **"Декан"** як невід'ємну частину.
2. **Агрегація**:
   * Клас **"Університет"** використовує агрегацію для зберігання списку факультетів, а клас **"Факультет"** — для зберігання списку студентів. Це дозволяє об'єктам існувати незалежно від класів-контейнерів.
3. **Успадкування**:
   * Реалізовано похідний клас **"Магістр"** від базового класу **"Студент"** з додатковими властивостями (наприклад, тема дисертації). Це демонструє можливості наслідування для розширення функціональності базових класів.