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: ст. гр. КН-23 |
| Гребенюк Д.О. |
| Перевірив: асистент  Козірова Н.Л. |
|  |

Кропивницький

2024

**Мета:** Ознайомитись з основними поняттями успадкування, агрегація та композиція в ООП та навчитись їх програмно реалізовувати мовою С++.

**Варіант 4**

**Завдання 1**

1. Створіть клас "Музичний гурт", який містить такі властивості:

• Назва гурту

• Стиль музики

• Список музикантів, які входять до гурту (використовуйте агрегацію)

• Керівник гурту (використовуйте композицію)

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

• Ім'я музиканта

• Інструмент, на якому він грає

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

• Назва інструменту

• Рік виготовлення

4. Реалізуйте методи для додавання та видалення музикантів зі списку музикантів гурту.

5. Напишіть демонстраційну функцію main(), в якій створюються об'єкти класів "Музичний гурт", "Музикант" та "Інструмент". Додайте музикантів до

списку музикантів гурту та виведіть інформацію про них.

6. Продемонструйте успадкування, створивши похідний клас від

"Музикант" (наприклад, "Вокаліст") 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 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.

**Завдання 1**

*Лістинг instrument.h*

#ifndef INSTRUMENT\_H

#define INSTRUMENT\_H

#include <string>

class Instrument

{

public:

Instrument(std::string name, int manufactureYear);

std::string getName() const;

void setName(const std::string &newName);

int getManufactureYear() const;

void setManufactureYear(int newManufactureYear);

private:

std::string name;

int manufactureYear;

};

#endif // INSTRUMENT\_H

*Лістинг instrument.cpp*

#include "instrument.h"

Instrument::Instrument(std::string name, int manufactureYear)

{

this->name = name;

this->manufactureYear = manufactureYear;

}

std::string Instrument::getName() const

{

return name;

}

void Instrument::setName(const std::string &newName)

{

name = newName;

}

int Instrument::getManufactureYear() const

{

return manufactureYear;

}

void Instrument::setManufactureYear(int newManufactureYear)

{

manufactureYear = newManufactureYear;

}

*Лістинг musician.h*

#ifndef MUSICIAN\_H

#define MUSICIAN\_H

#include <string>

#include "instrument.h"

class Musician

{

public:

Musician(std::string name, Instrument instrument);

std::string getName() const;

void setName(const std::string &newName);

Instrument getInstrument() const;

void setInstrument(const Instrument &newInstrument);

private:

std::string name;

Instrument instrument;

};

#endif // MUSICIAN\_H

*Лістинг musician.cpp*

#include "musician.h"

Musician::Musician(std::string name, Instrument instrument) : instrument(instrument)

{

this->name = name;

}

std::string Musician::getName() const

{

return name;

}

void Musician::setName(const std::string &newName)

{

name = newName;

}

Instrument Musician::getInstrument() const

{

return instrument;

}

void Musician::setInstrument(const Instrument &newInstrument)

{

instrument = newInstrument;

}

*Лістинг musicband.h*

#ifndef MUSICBAND\_H

#define MUSICBAND\_H

#include <string>

#include <vector>

#include "musician.h"

#include "manager.h"

class MusicBand

{

public:

MusicBand(std::string name, std::string musicStyle, Manager manager);

std::string getName() const;

void setName(const std::string &newName);

std::string getMusicStyle() const;

void setMusicStyle(const std::string &newMusicStyle);

std::vector<Musician \*> getMusicians() const;

void setMusicians(const std::vector<Musician \*> &newMusicians);

Manager getManager() const;

void setManager(const Manager &newManager);

void addMusician(Musician\* musician);

void removeMusician(Musician\* musician);

private:

std::string name;

std::string musicStyle;

std::vector<Musician\*> musicians;

Manager manager;

};

#endif // MUSICBAND\_H

*Лістинг musicband.cpp*

#include "musicband.h"

#include <algorithm>

#include <iostream>

MusicBand::MusicBand(std::string name, std::string musicStyle, Manager manager) : manager(manager)

{

this->name = name;

this->musicStyle = musicStyle;

}

Manager MusicBand::getManager() const

{

return manager;

}

void MusicBand::setManager(const Manager &newManager)

{

manager = newManager;

}

std::vector<Musician \*> MusicBand::getMusicians() const

{

return musicians;

}

void MusicBand::setMusicians(const std::vector<Musician \*> &newMusicians)

{

musicians = newMusicians;

}

std::string MusicBand::getMusicStyle() const

{

return musicStyle;

}

void MusicBand::setMusicStyle(const std::string &newMusicStyle)

{

musicStyle = newMusicStyle;

}

std::string MusicBand::getName() const

{

return name;

}

void MusicBand::setName(const std::string &newName)

{

name = newName;

}

void MusicBand::addMusician(Musician \*musician)

{

musicians.push\_back(musician);

}

void MusicBand::removeMusician(Musician \*musician)

{

auto elementPosition = std::find(musicians.begin(), musicians.end(), musician);

if (elementPosition == musicians.end()) {

std::cout << "No musician was found in the list" << std::endl;

return;

}

musicians.erase(elementPosition);

}

*Лістинг manager.h*

#ifndef MANAGER\_H

#define MANAGER\_H

#include <string>

class Manager

{

public:

Manager(std::string name);

std::string getName() const;

void setName(const std::string &newName);

private:

std::string name;

};

#endif // MANAGER\_H

*Лістинг manager.cpp*

#include "manager.h"

Manager::Manager(std::string name) {

this->name = name;

}

std::string Manager::getName() const

{

return name;

}

void Manager::setName(const std::string &newName)

{

name = newName;

}

*Лістинг vocalist.h*

#ifndef VOCALIST\_H

#define VOCALIST\_H

#include "musician.h"

class Vocalist : public Musician

{

public:

Vocalist(std::string name, Instrument instrument, std::string voiceType);

std::string getVoiceType() const;

void setVoiceType(const std::string &newVoiceType);

private:

std::string voiceType; // soprano/tenor

};

#endif // VOCALIST\_H

*Лістинг vocalist.cpp*

#include "vocalist.h"

Vocalist::Vocalist(std::string name, Instrument instrument, std::string voiceType)

: Musician(name, instrument)

{

this->voiceType = voiceType;

}

std::string Vocalist::getVoiceType() const

{

return voiceType;

}

void Vocalist::setVoiceType(const std::string &newVoiceType)

{

voiceType = newVoiceType;

}

*Лістинг main.cpp*

#include <iostream>

#include <vector>

#include "musicband.h"

#include "musician.h"

#include "vocalist.h"

using namespace std;

int main()

{

Instrument guitar("Guitar", 1999);

Instrument drum("Drum", 2004);

Instrument piano("Piano", 2017);

Instrument microphone("Microphone", 2024);

std::string musicBandName;

std::cout << "Enter the name of the music band: ";

std::getline(std::cin, musicBandName);

std::string musicBandStyle;

std::cout << "Enter the style of the music band: ";

std::getline(std::cin, musicBandStyle);

std::string managerName;

std::cout << "Enter the name of the music band leader: ";

std::getline(std::cin, managerName);

int musiciansCount = 0;

std::cout << "Enter the number of musicians in the band: ";

std::cin >> musiciansCount;

std::cin.ignore();

MusicBand musicBand(musicBandName, musicBandStyle, Manager(managerName));

std::vector<Musician \*> musicians;

for (int index = 0; index < musiciansCount; index++) {

std::string musicianName;

std::cout << "Enter the name of musician #" << index + 1 << ": ";

std::getline(std::cin, musicianName);

std::string musicianInstrumentName;

std::cout << "Enter the name of the musician's instrument (guitar, drum or piano): ";

std::getline(std::cin, musicianInstrumentName);

Instrument \*musicianInstrument = nullptr;

if (musicianInstrumentName == "guitar")

musicianInstrument = &guitar;

else if (musicianInstrumentName == "drum")

musicianInstrument = &drum;

else if (musicianInstrumentName == "piano")

musicianInstrument = &piano;

else {

std::cout << "You didn't specify an instrument for the musician." << std::endl;

return 1;

}

musicians.push\_back(new Musician(musicianName, \*musicianInstrument));

musicBand.addMusician(musicians.back());

}

std::cout << std::endl;

std::cout << "========== About band ==========" << std::endl;

std::cout << "Band name: " << musicBand.getName() << std::endl;

std::cout << "Band style: " << musicBand.getMusicStyle() << std::endl;

std::cout << "Band leader: " << musicBand.getManager().getName() << std::endl;

std::cout << "Musicians:" << std::endl;

for (const Musician\* musician : musicBand.getMusicians()) {

std::cout << "- " << musician->getName() << " (" << musician->getInstrument().getName() << ")" << std::endl;

}

std::cout << std::endl;

char addVocalist;

std::cout << "Do you want to add a vocalist? (y/n): ";

std::cin >> addVocalist;

std::cin.ignore();

if (addVocalist == 'y' || addVocalist == 'Y') {

std::string vocalistName;

std::cout << "Enter vocalist's name: ";

std::getline(std::cin, vocalistName);

std::string vocalistVoiceType;

std::cout << "Enter vocalist's voice type (e.g., tenor, soprano): ";

std::getline(std::cin, vocalistVoiceType);

Vocalist newVocalist(vocalistName, microphone, vocalistVoiceType);

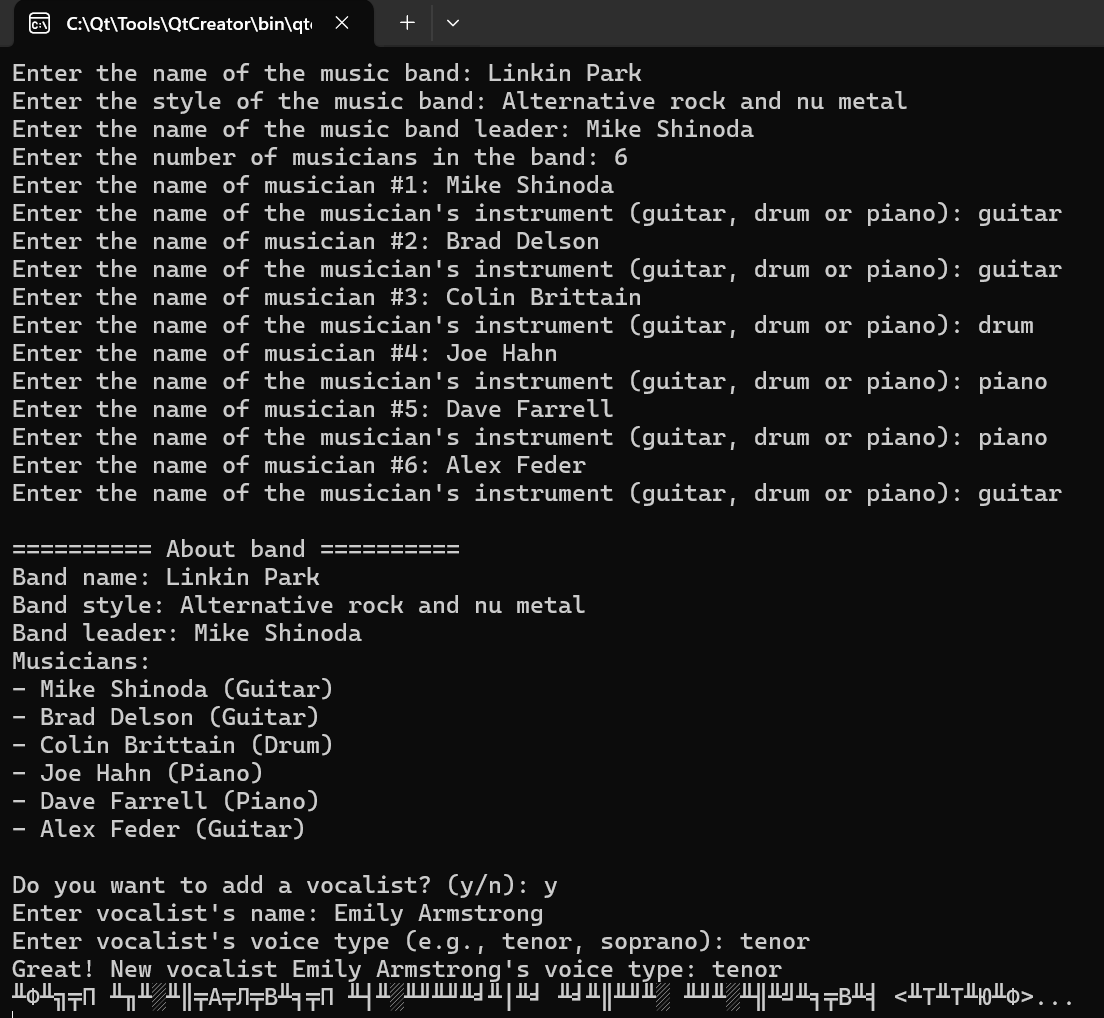
std::cout << "Great! New vocalist " << newVocalist.getName() << "'s voice type: " << newVocalist.getVoiceType() << std::endl;

}

return 0;

}

*Результат виконання*



**Завдання 2**

*Лістинг employee.h*

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

#include "jobposition.h"

#include "department.h"

class Employee

{

public:

Employee();

std::string getName();

void setName(std::string newName);

int getId();

void setId(int newId);

double getSalary();

void setSalary(double newSalary);

JobPosition getJobPosition();

void setJobPosition(JobPosition newJobPosition);

Department\* getDepartment();

void setDepartment(Department\* newDepartment);

private:

std::string name;

int id;

double salary;

JobPosition jobPosition;

Department\* department;

};

#endif // EMPLOYEE\_H

*Лістинг employee.cpp*

#include "employee.h"

Employee::Employee() {

name = "";

id = -1;

salary = 0.0;

department = nullptr;

}

std::string Employee::getName() {

return name;

}

void Employee::setName(std::string newName) {

name = newName;

}

int Employee::getId() {

return id;

}

void Employee::setId(int newId) {

id = newId;

}

double Employee::getSalary() {

return salary;

}

void Employee::setSalary(double newSalary) {

salary = newSalary;

}

JobPosition Employee::getJobPosition() {

return jobPosition;

}

void Employee::setJobPosition(JobPosition newJobPosition) {

jobPosition = newJobPosition;

}

Department\* Employee::getDepartment() {

return department;

}

void Employee::setDepartment(Department\* newDepartment) {

department = newDepartment;

}

*Лістинг jobposition.h*

#ifndef JOBPOSITION\_H

#define JOBPOSITION\_H

#include <string>

class JobPosition

{

public:

JobPosition();

std::string getTitle();

void setTitle(std::string newTitle);

double getSalary();

void setSalary(double newSalary);

private:

std::string title;

double salary;

};

#endif // JOBPOSITION\_H

*Лістинг jobposition.cpp*

#include "jobposition.h"

JobPosition::JobPosition() {

title = "";

salary = 0.0;

}

std::string JobPosition::getTitle() {

return title;

}

void JobPosition::setTitle(std::string newTitle) {

title = newTitle;

}

double JobPosition::getSalary() {

return salary;

}

void JobPosition::setSalary(double newSalary) {

salary = newSalary;

}

*Лістинг department.h*

#ifndef DEPARTMENT\_H

#define DEPARTMENT\_H

#include <string>

class Department

{

public:

Department();

std::string getName();

void setName(std::string newName);

std::string getManager();

void setManager(std::string newManager);

private:

std::string name;

std::string manager;

};

#endif // DEPARTMENT\_H

*Лістинг department.cpp*

#include "department.h"

Department::Department() {

name = "";

manager = "";

}

std::string Department::getName() {

return name;

}

void Department::setName(std::string newName) {

name = newName;

}

std::string Department::getManager() {

return manager;

}

void Department::setManager(std::string newManager) {

manager = newManager;

}

*Лістинг developer.h*

#ifndef DEVELOPER\_H

#define DEVELOPER\_H

#include <string>

#include "employee.h"

class Developer : public Employee

{

public:

Developer();

std::string getProgrammingLanguage();

void setProgrammingLanguage(std::string newLanguage);

private:

std::string programmingLanguage;

};

#endif // DEVELOPER\_H

*Лістинг developer.cpp*

#include "developer.h"

Developer::Developer() {

programmingLanguage = "";

}

std::string Developer::getProgrammingLanguage() {

return programmingLanguage;

}

void Developer::setProgrammingLanguage(std::string newLanguage) {

programmingLanguage = newLanguage;

}

*Лістинг main.cpp*

#include <iostream>

#include "developer.h"

#include "jobposition.h"

#include "department.h"

int main() {

std::string developerName;

int developerId;

double developerSalary;

std::string programmingLanguage;

std::cout << "Enter developer name: ";

std::getline(std::cin, developerName);

std::cout << "Enter developer ID: ";

std::cin >> developerId;

std::cout << "Enter developer salary: ";

std::cin >> developerSalary;

std::cout << "Enter programming language: ";

std::cin.ignore();

std::getline(std::cin, programmingLanguage);

std::string jobTitle;

double jobSalary;

std::cout << "Enter job position title: ";

std::getline(std::cin, jobTitle);

std::cout << "Enter job position salary: ";

std::cin >> jobSalary;

JobPosition jobPosition;

jobPosition.setTitle(jobTitle);

jobPosition.setSalary(jobSalary);

std::string departmentName;

std::string departmentManager;

std::cout << "Enter department name: ";

std::cin.ignore();

std::getline(std::cin, departmentName);

std::cout << "Enter department manager name: ";

std::getline(std::cin, departmentManager);

Department department;

department.setName(departmentName);

department.setManager(departmentManager);

Developer developer;

developer.setName(developerName);

developer.setId(developerId);

developer.setSalary(developerSalary);

developer.setProgrammingLanguage(programmingLanguage);

developer.setJobPosition(jobPosition);

developer.setDepartment(&department);

std::cout << std::endl;

std::cout << "Developer created successfully!" << std::endl;

std::cout << "Developer Name: " << developer.getName() << std::endl;

std::cout << "Developer ID: " << developer.getId() << std::endl;

std::cout << "Developer Salary: " << developer.getSalary() << std::endl;

std::cout << "Programming Language: " << developer.getProgrammingLanguage() << std::endl;

std::cout << "Job Position: " << developer.getJobPosition().getTitle() << std::endl;

std::cout << "Job Salary: " << developer.getJobPosition().getSalary() << std::endl;

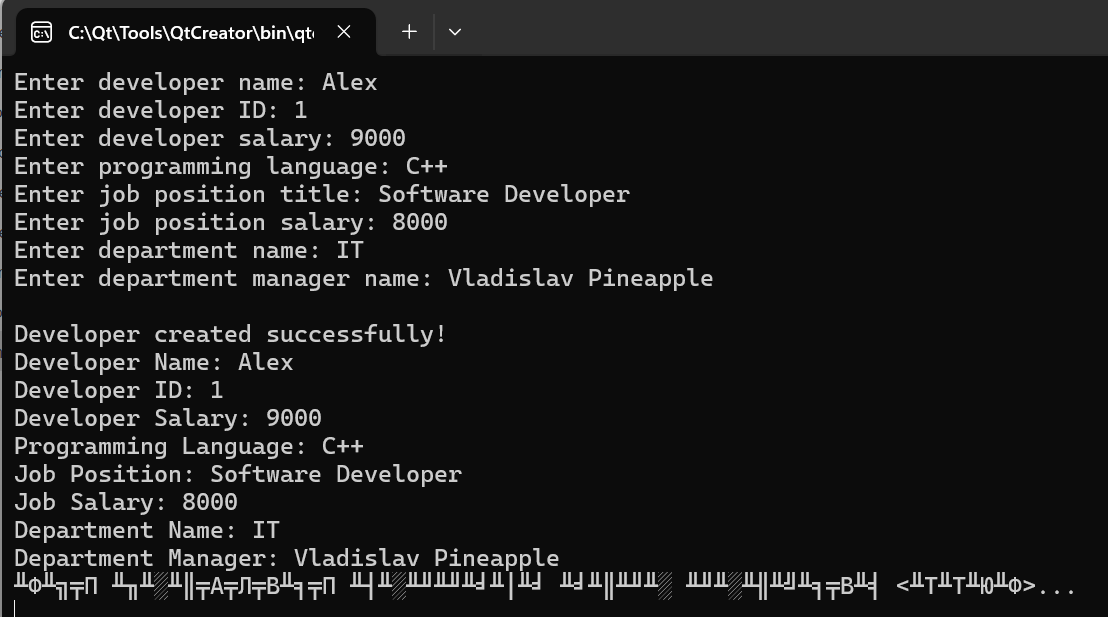
std::cout << "Department Name: " << developer.getDepartment()->getName() << std::endl;

std::cout << "Department Manager: " << developer.getDepartment()->getManager() << std::endl;

return 0;

}

*Результат виконання*



**Висновок**

В ході виконання лабораторної роботи я ознайомився з основними принципами об'єктно-орієнтованого програмування, такими як успадкування, агрегація та композиція. Я навчився реалізовувати ці концепції мовою C++, створюючи класи, що взаємодіють між собою через ці механізми. Зокрема, було створено класи для музичного гурту, музикантів, інструментів, а також керівника гурту, що продемонструвало використання агрегації та композиції. Крім того, я реалізував успадкування, створивши похідний клас “Вокаліст” від класу “Музикант”. Це дало змогу поглибити розуміння принципів ООП та їх практичної реалізації у програмуванні.