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. КI-23 |
| Чепіль В.О. |
| Перевірив викладач:  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++.

**Варіант:** 4

**Завдання:**

**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.

**Лістинг завдання 1(Task\_1):**

**MusicBand.h:**

#ifndef MUSICBAND\_H

#define MUSICBAND\_H

#include <string>

#include <vector>

#include "musician.h"

#include "vocalist.h"

using namespace std;

class MusicBand

{

public:

MusicBand(string name, string style, Musician leader);

void setBandName(string name);

string getBandName();

void setStyleMusic(string style);

string getStyleMusic();

void setBandLeader(Musician leader);

string getBandLeader();

void addMusicianBand(Musician musician);

void addVocalistBand(Vocalist vocalist);

void deleteBandMember(Musician musician);

void printMusicBand();

private:

string bandName;

string styleMusic;

vector<Musician> listMusicians; // агрегація

vector<Vocalist> listVocalists; // агрегація

Musician bandLeader; // композиція

};

#endif // MUSICBAND\_H

**MusicBand.cpp:**

#include "musicBand.h"

#include <iostream>

MusicBand::MusicBand(string name, string style, Musician leader)

: bandName(name), styleMusic(style), bandLeader(leader) {

listMusicians.push\_back(leader);

}

void MusicBand::setBandName(string name)

{

bandName = name;

}

string MusicBand::getBandName()

{

return bandName;

}

void MusicBand::setStyleMusic(string style)

{

styleMusic = style;

}

string MusicBand::getStyleMusic()

{

return styleMusic;

}

void MusicBand::setBandLeader(Musician leader)

{

bandLeader = leader;

}

string MusicBand::getBandLeader()

{

return bandLeader.getMusicianName();

}

void MusicBand::addMusicianBand(Musician musician)

{

listMusicians.push\_back(musician);

}

void MusicBand::addVocalistBand(Vocalist vocalist)

{

listVocalists.push\_back(vocalist);

}

void MusicBand::deleteBandMember(Musician musician)

{

int index = -1;

for (int i = 0; i < static\_cast<int>(listMusicians.size()); ++i)

{

if (listMusicians[i].getMusicianName() == musician.getMusicianName())

{

index = i;

if (listMusicians[i].getMusicianName() == getBandLeader())

{

musician.setMusicianName("Not leader");

setBandLeader(musician);

}

break;

}

}

if (index != -1)

{

listMusicians.erase(listMusicians.begin() + index);

return;

}

index = -1;

for (int i = 0; i < static\_cast<int>(listVocalists.size()); ++i)

{

if (listVocalists[i].getMusicianName() == musician.getMusicianName())

{

index = i;

break;

}

}

if (index != -1)

{

listVocalists.erase(listVocalists.begin() + index);

}

}

void MusicBand::printMusicBand()

{

cout << "Band name: " << bandName << endl;

cout << "Style music: " << styleMusic << endl;

cout << "Band leader: " << MusicBand::getBandLeader() << endl;

for (Musician musician : listMusicians)

{

cout << "Musician: " << musician.getMusicianName()

<< ", Tool: " << musician.getTool().getToolName()

<< " (" << musician.getTool().getYearManufacture() << ")" << endl;

}

for (Vocalist vocalist : listVocalists)

{

cout << "Vocalist: " << vocalist.getMusicianName()

<< ", Tool: " << vocalist.getTool().getToolName()

<< " (" << vocalist.getTool().getYearManufacture() << "), "

<< "Type: " << vocalist.getVocalistType() << endl;

}

cout << endl;

}

**Musician.h:**

#ifndef MUSICIAN\_H

#define MUSICIAN\_H

#include <string>

#include "tool.h"

using namespace std;

class Musician

{

public:

Musician(string name, Tool tool);

void setMusicianName(string name);

string getMusicianName();

Tool getTool();

private:

string musicianName;

Tool tool; // агрегація

};

#endif // MUSICIAN\_H

**Musician.cpp:**

#include "musician.h"

Musician::Musician(string name, Tool tool) : musicianName(name), tool(tool) {}

void Musician::setMusicianName(string name)

{

musicianName = name;

}

string Musician::getMusicianName()

{

return musicianName;

}

Tool Musician::getTool()

{

return tool;

}

**Tool.h:**

#ifndef TOOL\_H

#define TOOL\_H

#include <string>

using namespace std;

class Tool

{

public:

Tool(string name, int year);

void setToolName(string name);

string getToolName();

void setYearManufacture(int year);

int getYearManufacture();

private:

string toolName;

int yearManufacture;

};

#endif // TOOL\_H

**Tool.cpp:**

#include "tool.h"

Tool::Tool(string name, int year) : toolName(name), yearManufacture(year) {}

void Tool::setToolName(string name)

{

toolName = name;

}

string Tool::getToolName()

{

return toolName;

}

void Tool::setYearManufacture(int year)

{

yearManufacture = year;

}

int Tool::getYearManufacture()

{

return yearManufacture;

}

**Vocalist.h:**

#ifndef VOCALIST\_H

#define VOCALIST\_H

#include "musician.h"

class Vocalist : public Musician

{

public:

Vocalist(string name, Tool tool, string vocalistType);

string getVocalistType();

private:

string vocalistType;

};

#endif // VOCALIST\_H

**Vocalist.cpp:**

#include "vocalist.h"

Vocalist::Vocalist(string name, Tool tool, string vocalistType)

: Musician(name, tool), vocalistType(vocalistType) {}

string Vocalist::getVocalistType()

{

return vocalistType;

}

**main.cpp:**

#include <iostream>

#include "musicband.h"

using namespace std;

int main()

{

// Створення інструментів

Tool guitar("Guitar", 2020);

Tool dudka("Dudka", 2018);

Tool baraban("Baraban", 2010);

Tool microphone("Microphone", 2021);

// Створення музикантів

Musician musician1("Sanya", guitar);

Musician musician2("Taras", dudka);

Musician musician3("Vadim", baraban);

// Створення вокалістів

Vocalist vocalist1("Sofia", microphone,"Solo");

Vocalist vocalist2("Ivan", microphone,"Backing vocalist");

Vocalist vocalist3("Oleg", microphone, "Choir singer");

// Створення музичного гурту

MusicBand band("Rock Band", "Rock", musician1);

band.addMusicianBand(musician2);

band.addMusicianBand(musician3);

band.addVocalistBand(vocalist1);

band.addVocalistBand(vocalist2);

band.addVocalistBand(vocalist3);

// Видалення учасника гурту

//band.deleteBandMember(musician1);

// Встановлення лідера

//band.setBandLeader(vocalist1);

//виведення інформоції про гурт

band.printMusicBand();

return 0;

}

**Лістинг завдання 2(Task\_2):**

**Address.h:**

#ifndef ADDRESS\_H

#define ADDRESS\_H

#include <string>

using namespace std;

class Address

{

public:

Address(string street, string city);

string getStreet();

string getCity();

void setStreet(string newStreet);

void setCity(string newCity);

private:

string street;

string city;

};

#endif // ADDRESS\_H

**Address.cpp:**

#include "address.h"

Address::Address(string street, string city) : street(street), city(city) {}

string Address::getStreet()

{

return street;

}

string Address::getCity()

{

return city;

}

void Address::setStreet(string newStreet)

{

street = newStreet;

}

void Address::setCity(string newCity)

{

city = newCity;

}

**Department.h:**

#ifndef DEPARTMENT\_H

#define DEPARTMENT\_H

#include <string>

using namespace std;

class Department

{

public:

Department(string name);

string getName();

void setName(string newName);

private:

string name;

};

#endif // DEPARTMENT\_H

**Department.cpp:**

#include "department.h"

Department::Department(string name) : name(name) {}

string Department::getName()

{

return name;

}

void Department::setName(string newName)

{

name = newName;

}

**Employee.h:**

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

#include "address.h"

#include "department.h"

using namespace std;

class Employee

{

public:

Employee(string name, int id, int salary, Address addr, Department& dept);

string getName();

int getId();

int getSalary();

Address getAddress();

Department& getDepartment();

private:

string name;

int id;

int salary;

Address address; // композиція

Department& department; // посилання

};

#endif // EMPLOYEE\_H

**Employee.cpp:**

#include "employee.h"

Employee::Employee(string name, int id, int salary, Address addr, Department &dept)

: name(name), id(id), salary(salary), address(addr), department(dept) {}

string Employee::getName()

{

return name;

}

int Employee::getId()

{

return id;

}

int Employee::getSalary()

{

return salary;

}

Address Employee::getAddress()

{

return address;

}

Department &Employee::getDepartment()

{

return department;

}

**Manager.h:**

#ifndef MANAGER\_H

#define MANAGER\_H

#include "employee.h"

class Manager : public Employee

{

public:

Manager(string name, int id, int salary, Address addr, Department& dept, int bonus);

void setBonus(int newBonus);

int getBonus();

private:

int bonus;

};

#endif // MANAGER\_H

**Manager.cpp:**

#include "manager.h"

Manager::Manager(string name, int id, int salary, Address addr, Department& dept, int bonus)

: Employee(name, id, salary, addr, dept), bonus(bonus) {}

void Manager::setBonus(int newBonus)

{

bonus = newBonus;

}

int Manager::getBonus()

{

return bonus;

}

**main.cpp:**

#include <iostream>

#include "employee.h"

#include "address.h"

#include "department.h"

#include "manager.h"

using namespace std;

int main()

{

Address address1("Shevchenka St.", "Kyiv");

Address address2("Kovalenka St.", "Kyiv");

Department dept("IT");

Employee employee("Taras", 1, 15000, address1, dept);

Manager manager("Oleg", 2, 150000, address2, dept, 100);

cout << "Name: " << employee.getName() << endl;

cout << "ID: " << employee.getId() << endl;

cout << "Salary: " << employee.getSalary() << endl;

cout << "Address: " << employee.getAddress().getStreet() << ", " << employee.getAddress().getCity() << endl;

cout << "Department: " << employee.getDepartment().getName() << endl;

cout << endl;

cout << "Name: " << manager.getName() << endl;

cout << "ID: " << manager.getId() << endl;

cout << "Salary: " << manager.getSalary() << endl;

cout << "Address: " << manager.getAddress().getStreet() << ", " << employee.getAddress().getCity() << endl;

cout << "Department: " << manager.getDepartment().getName() << endl;

cout << "Bonus: " << manager.getBonus() << endl;

return 0;

}

**Висновок:**

У ході роботи було розглянуто основні поняття об'єктно-орієнтованого програмування (ООП) — успадкування, агрегація та композиція. Ми ознайомилися з теоретичними основами цих концепцій та навчилися їх реалізовувати на практиці за допомогою мови програмування С++.

Успадкування дозволяє створювати нові класи на основі існуючих, що сприяє повторному використанню коду та розширенню функціональності.

Агрегація — це слабкий зв’язок між об'єктами, де один клас містить посилання на інші, але не відповідає за їх життєвий цикл.

Композиція — сильніший зв’язок, де клас повністю контролює життєвий цикл об'єктів, які містяться в ньому.