Министерство образования Республики Беларусь  
Учреждение образования  
«Брестский государственный технический университет»  
Кафедра ИИТ

Лабораторная работа №3

«Иерархии наследования, динамический полиморфизм в приложениях на языках С++, С#  
По дисциплине: «Объектное моделирование интеллектуальных систем»

Выполнил:  
Студент 3 курса  
Группы ИИ-23  
Макаревич Н.Р.

Проверила:  
Демидович А.Г.

Брест 2024

**Цель работы:**

1. Изучить особенности организации и реализации множественного наследования классов.

2. Изучить варианты организации иерархии классов с использованием отношений агрегации, композиции, использования.

3. Изучить проектирование иерархий классов с использованием динамического полиморфизма.

Множественное наследование

Анализировать предметную область и разработать классы для демонстрации всех видов наследования. Для этого:

- описать исходный класс. Состав минимальный, включая конструкторы, деструктор, перегруженный оператор. Включить в классы члены-данные, требующие динамического распределения памяти;

- для заданной предметной области разработать дополнительные классы для демонстрации всех видов наследования;

- описать классы на UML (язык - русский);

- описать диаграмму классов

(уровень ассоциаций) на UML (язык - русский) без раскрытия классов;

- уточнить диаграмму классов (типы отношений) на UML (язык - русский) без раскрытия классов;

- описать уточненную диаграмму классов UML с раскрытием классов;

- специфицировать, прокомментировать при необходимости данные и методы класса.

Организация иерархий классов

Для указанной предметной области:

– реализовать иерархии классов, используя наследование и используя агрегацию (состав минимальный, включая конструкторы, деструктор, перегруженный оператор):

- для каждого описать диаграмму UML (язык - русский);

- описать диаграмму UML (язык - английский);

- специфицировать данные и методы класса;

- разработать диаграмму классов (уровень ассоциаций);

- уточнить диаграмму классов (типы отношений). Программировать.

Реализовать приложение для обработки объектов предметной области MIX (целое и вещественное), использовав агрегацию и наследование для организации иерархии классов.

Динамический полиморфизм

1. Продемонстрировать для своей предметной области (без применения виртуальных функций) использование собственных указателей классов и указателей базовых классов для доступа к членам классов иерархии со стороны объектов.
2. Повторить предыдущий пункт с применением виртуальных функций.
3. Реализовать приложение с использованием динамического полиморфизма. Вариант реализации – консольный с управлением через простейшее меню.

**Предметная область: "Фестиваль"**

**1 (Множественное наследование).**

Описание исходного класса "Музыкант":

class Musician{

public:

std::string name;

Musician(std::string);

~Musician();

virtual void play() const;

Musician& operator = (const Musician& other);

};

Класс "Музыкант" содержит данные об имени музыканта.

Дополнительные классы для демонстрации всех видов наследования:

a) Класс "Drummer" и “Guitarist”:

class Drummer: public Musician{

public:

Drummer() : Musician("Unknown drummer") {}

void play() const ;

};

class Guitarist: public Musician{

public:

Guitarist() : Musician("Unknown guitarist") {}

void play() const;

};

б) Класс "Band":

class Band{

public:

std::string name;

std::vector<Musician\*> members;

Band(std::string);

void addMember(Musician\*);

void makeStandartBand();

void play();

};

в) Класс "Fest":

class Fest{

public:

std::string name;

Fest(const std::string& name);

void addBand(const std::string& name);

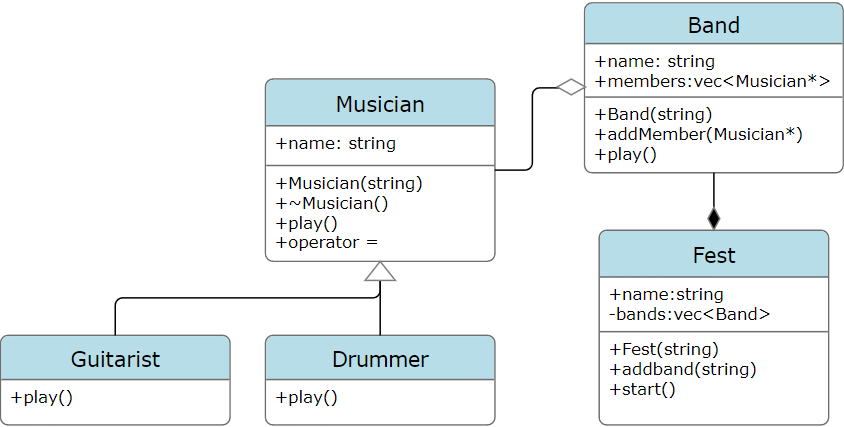
void start();

private:

std::vector<Band> bands;

};

**Описание классов:**



1. Демонстрация использования собственных указателей классов и указателей базовых классов для доступа к членам классов иерархии со стороны объектов :

**Код программы:**

void Band::makeStandartBand() {

Musician\* guitarist = new Guitarist();

Musician\* drummer = new Drummer();

Musician\* singer = new Musician("Unknown singer");

addMember(guitarist);

addMember(drummer);

addMember(singer);}

void Band::play() {

std::cout << name << " band is perfoming\n";

for (int i = 0; i < members.size(); i++) {

std::cout << members[i]->name << ": ";

members[i]->play();

}}

3. Реализовать приложение с использованием динамического полиморфизма через консольное меню:

**Код программы:**

#include <iostream>

#include "Fest.h"

int main() {

int choise;

std::string name = "";

Fest festival(name);

while (true) {

system("cls");

std::cout << "1. Set up festival\n";

std::cout << "2. Add band\n";

std::cout << "3. Start festival\n";

std::cout << "4.Exit\n";

std::cin >> choise;

switch (choise) {

case 1:

system("cls");

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

std::cin >> name;

festival = Fest(name);

break;

case 2:

system("cls");

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

std::cin >> name;

festival.addBand(name);

break;

case 3:

system("cls");

festival.start();

system("pause");

break;

case 4:

return 0;

default:

std::cout << "Wrong option\n";

break;

}}}

**Вывод:** Изучил особенности организации и реализации множественного наследования классов. Изучил варианты организации иерархии классов с использованием отношений агрегации, композиции, использования. Изучил проектирования иерархий классов с использованием динамического полиморфизма.