НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



**З В І Т**

до лабораторної роботи **№7**

# з курсу: «Програмування, частина 2

# (Об'єктно-орієнтоване програмування)»

# на тему: «Множинне спадкування. Поліморфізм»

Варіант **№8**

***Виконав:*** *студент групи КІ-103 Кобзєв Роман*

***Прийняв:*** *Науличний В.В*

**Львів – 2024**

***Мета роботи:*** познайомитися із множинним спадкуванням класів та поліморфізмом.

**Завдання**

Спроектувати і реалізувати ієрархію класів, що описують предметну область згідно варіанту, яка реалізується класом 1. Клас 1 в свою чергу утворюється шляхом множинного спадкування класів 2 і 3 кожен з яких в свою чергу успадковує клас 4. Додаткові вимоги:

1. Базовий клас містить мінімум один віртуальний метод, один невіртуальний метод і одну динамічно створювану властивість.

2. Забезпечити механізми коректної роботи конструкторів і деструкторів.

3. Перевантажити оператор присвоєня з метою його коректної роботи.

4. Кожен з класів має містити мінімум одину властивість і 4 методи.

5. Написати main() функцію де створити об‘єкт класу 1 і продемонструвати різницю між статичним і динамічним поліморфізмом.

| *№* | *Завдання* | | | | |
| --- | --- | --- | --- | --- | --- |
| 16 | Предметна область | Клас 1 | Клас 2 | Клас 3 | Клас 4 |
| Аудіо-відео плеєр | CAudioVideoPlayer | CVideoPlayer | CAudioPlayer | CDevice |

**Текст програми**

#include <iostream>

#include <string>

using namespace std;

class CDevice {

protected:

    string manufacturer;

    int year;

public:

    CDevice() : manufacturer("Unknown"), year(0) {}

    CDevice(const string& man, int y) : manufacturer(man), year(y) {}

    virtual ~CDevice() {}

    virtual void play() const {

        cout << "Playing from device" << endl;

    }

    void setManufacturer(const string& man) { manufacturer = man; }

    void setYear(int y) { year = y; }

    string getManufacturer() const { return manufacturer; }

    int getYear() const { return year; }

    CDevice& operator=(const CDevice& other) {

        if (this != &other) {

            manufacturer = other.manufacturer;

            year = other.year;

        }

        return \*this;

    }

};

class CAudioPlayer : virtual public CDevice {

private:

    int bitRate;

public:

    CAudioPlayer() : CDevice(), bitRate(0) {}

    CAudioPlayer(const string& man, int y, int br) : CDevice(man, y), bitRate(br) {}

    void setBitRate(int br) { bitRate = br; }

    int getBitRate() const { return bitRate; }

    void play() const override {

        cout << "Playing audio from audio player" << endl;

    }

    void stop() const {

        cout << "Stopping audio from audio player" << endl;

    }

    void pause() const {

        cout << "Pausing audio from audio player" << endl;

    }

    void rewind() const {

        cout << "Rewinding audio from audio player" << endl;

    }

};

class CVideoPlayer : virtual public CDevice {

private:

    int resolution;

public:

    CVideoPlayer() : CDevice(), resolution(0) {}

    CVideoPlayer(const string& man, int y, int res) : CDevice(man, y), resolution(res) {}

    void setResolution(int res) { resolution = res; }

    int getResolution() const { return resolution; }

    void play() const override {

        cout << "Playing video from video player" << endl;

    }

    void stop() const {

        cout << "Stopping video from video player" << endl;

    }

    void pause() const {

        cout << "Pausing video from video player" << endl;

    }

    void forward() const {

        cout << "Forwarding video from video player" << endl;

    }

};

class CAudioVideoPlayer : public CAudioPlayer, public CVideoPlayer {

public:

    CAudioVideoPlayer() : CDevice(), CAudioPlayer(), CVideoPlayer() {}

    CAudioVideoPlayer(const string& man, int y, int br, int res) : CDevice(man, y), CAudioPlayer(man, y, br), CVideoPlayer(man, y, res) {}

    void play() const override {

        cout << "Playing audio and video from audio-video player" << endl;

    }

    void stop() const {

        cout << "Stopping audio and video from audio-video player" << endl;

    }

    void pause() const {

        cout << "Pausing audio and video from audio-video player" << endl;

    }

    void forward() const {

        cout << "Forwarding audio and video from audio-video player" << endl;

    }

};

int main() {

    CAudioVideoPlayer avPlayer("Sony", 2022, 320, 1080);

    avPlayer.play();  // Calls overridden method in CAudioVideoPlayer

    return 0;

}

**Висновок**

Розробка ієрархії класів згідно з вимогами завдання дозволила створити гнучку та розширювану модель предметної області. Реалізація віртуальних і невіртуальних методів, динамічних властивостей, перевантаження операторів та коректної роботи конструкторів і деструкторів забезпечила ефективне управління ресурсами та поліморфізм. Програма-драйвер продемонструвала функціональність розроблених класів, підкресливши можливості як статичного, так і динамічного поліморфізму.