**Лабораторна робота № 4**

**Побудова множинного наслідування класів(С++ )**

**Інтерфейси- С#**

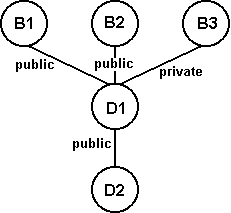
Мета:

навчитися організовувати класи та похідні класи з своїми даними та властивостями: визначати структуру класу, типи даних та методів, організовувати необхідні конструктори та деструктори, використовувати специфікатори доступу; будувати ієрархію класів, використовуючи множинне наслідування; визначати та оперувати об’єктами цих класів; отримати практичні вміння та навички проектування та побудови ієрархії класів. Отримати практичні навички використання інтерфейсів.

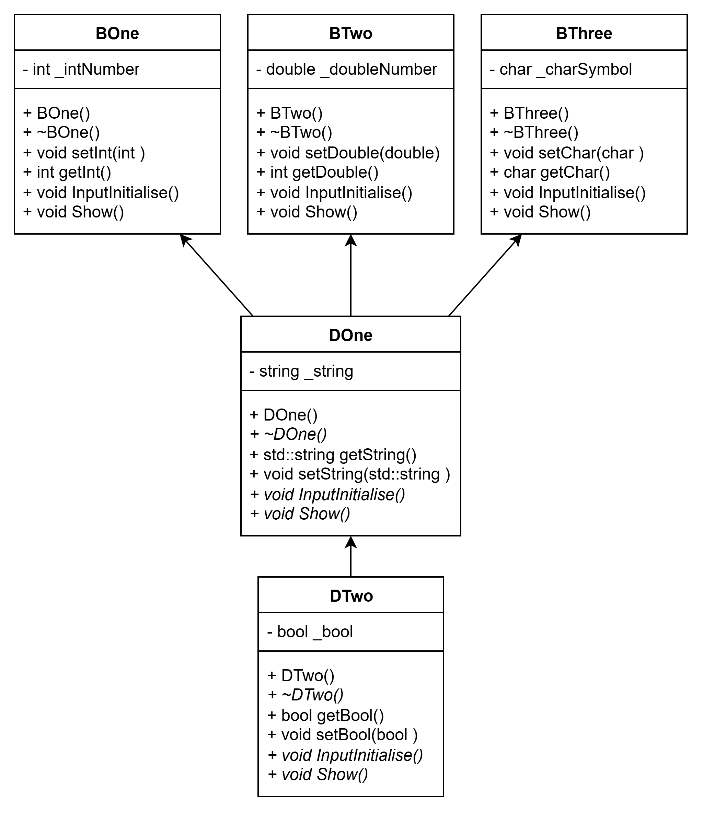
**Хід роботи**

**Завдання 1**

Необхідно побудувати ієрархію класів що відповідає схемі наслідування, наведеній у варіанті завдання. Кожен клас повинен містити конструктор-ініціалізатор, і функцію *show()* для виведення значень.



**Class diagram**

****

**Code**

**BOne.hpp**

#pragma once

#ifndef BONE\_HPP

#define BONE\_HPP

class BOne

{

private:

int \_intNumber;

public:

BOne();

virtual ~BOne();

void setInt(int number);

int getInt();

virtual void InputInitialise();

virtual void Show();

};

#endif // BONE\_HPP

**BOne.cpp**

#include <iostream>

#include "BOne.hpp"

BOne::BOne()

{

std::cout << "BOne::Bone() called" << std::endl;

}

BOne::~BOne()

{

std::cout << "BOne::~BOne() called" << std::endl;

}

void BOne::setInt(int number)

{

\_intNumber = number;

}

int BOne::getInt()

{

return \_intNumber;

}

void BOne::InputInitialise()

{

std::cout << "Enter BOne (int) - ";

std::cin >> \_intNumber;

}

void BOne::Show()

{

std::cout << "BOne: " << getInt() << std::endl;

}

**BTwo.hpp**

#pragma once

#ifndef BTWO\_HPP

#define BTWO\_HPP

class BTwo

{

private:

double \_doubleNumber;

public:

BTwo();

virtual ~BTwo();

void setDouble(int number);

int getDouble();

virtual void InputInitialise();

virtual void Show();

};

#endif // BTWO\_HPP

**BTwo.cpp**

#include <iostream>

#include "BTwo.hpp"

BTwo::BTwo() {

std::cout << "BTwo::BTwo() called" << std::endl;

}

BTwo::~BTwo()

{

std::cout << "BTwo::~BTwo() called" << std::endl;

}

void BTwo::setDouble(int number)

{

\_doubleNumber = number;

}

int BTwo::getDouble()

{

return \_doubleNumber;

}

void BTwo::InputInitialise() {

std::cout << "Enter BTwo (double) - ";

std::cin >> \_doubleNumber;

}

void BTwo::Show() {

std::cout << "BTwo: " << getDouble() << std::endl;

}

**BThree.hpp**

#pragma once

#ifndef BTHREE\_HPP

#define BTHREE\_HPP

class BThree

{

private:

int \_charSymbol;

public:

BThree();

virtual ~BThree();

void setChar(int number);

int getChar();

virtual void InputInitialise();

virtual void Show();

};

#endif // BTHREE\_HPP

**BThree.cpp**

#include <iostream>

#include "BThree.hpp"

BThree::BThree() {

std::cout << "BThree::BThree() called" << std::endl;

}

BThree::~BThree()

{

std::cout << "BThree::~BThree() called" << std::endl;

}

void BThree::setChar(int number)

{

\_charSymbol = number;

}

int BThree::getChar()

{

return \_charSymbol;

}

void BThree::InputInitialise() {

std::cout << "Enter BThree (char) - ";

std::cin.ignore();

\_charSymbol = std::cin.get();

}

void BThree::Show() {

std::cout << "BThree: " << getChar() << std::endl;

}

**DOne.hpp**

#pragma once

#ifndef DONE\_HPP

#define DONE\_HPP

#include <string>

#include "BOne.hpp"

#include "BTwo.hpp"

#include "BThree.hpp"

class DOne : public BOne, public BTwo, private BThree

{

std::string \_string;

public:

DOne();

~DOne()override;

std::string getString();

void setString(std::string other);

void InputInitialise() override;

void Show()override;

};

#endif // DONE\_HPP

**DOne.cpp**

#include <iostream>

#include "DOne.hpp"

DOne::DOne()

:BOne(), BTwo(), BThree()

{

std::cout << "DOne::DOne() called" << std::endl;

}

DOne::~DOne()

{

std::cout << "DOne::~DOne() called" << std::endl;

}

std::string DOne::getString()

{

return std::string();

}

void DOne::setString(std::string other)

{

\_string = other;

}

void DOne::InputInitialise()

{

BOne::InputInitialise();

BTwo::InputInitialise();

BThree::InputInitialise();

std::cout << "Enter DOne (string) - ";

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

std::getline(std::cin, \_string);

}

void DOne::Show() {

BOne::Show();

BTwo::Show();

BThree::Show();

std::cout << "DOne: " << \_string << std::endl;

}

**DTwo.hpp**

#pragma once

#ifndef DTWO\_HPP

#define DTWO\_HPP

#include "DOne.hpp"

class DTwo : public DOne

{

bool \_bool;

public:

DTwo();

~DTwo()override;

bool getBool();

void setBool(bool other);

void InputInitialise()override;

void Show()override;

};

#endif //DTWO\_HPP

**DTwo.cpp**

#include <iostream>

#include "DTwo.hpp"

DTwo::DTwo()

:DOne()

{

std::cout << "DTwo::DTwo() called" << std::endl;

}

DTwo::~DTwo() {

std::cout << "DTwo::~DTwo() called" << std::endl;

}

bool DTwo::getBool()

{

return \_bool;

}

void DTwo::setBool(bool other)

{

\_bool = other;

}

void DTwo::InputInitialise()

{

DOne::InputInitialise();

std::cout << "Enter DTwo (bool(0,1 or True,False)) - ";

std::cin >> \_bool;

}

void DTwo::Show() {

DOne::Show();

std::cout << "DTwo: " << (getBool() ? "True" : "False") << std::endl;

}

**Main.cpp**

#include <iostream>

#include <string>

#include "DTwo.hpp"

int main()

{

{

DTwo obj;

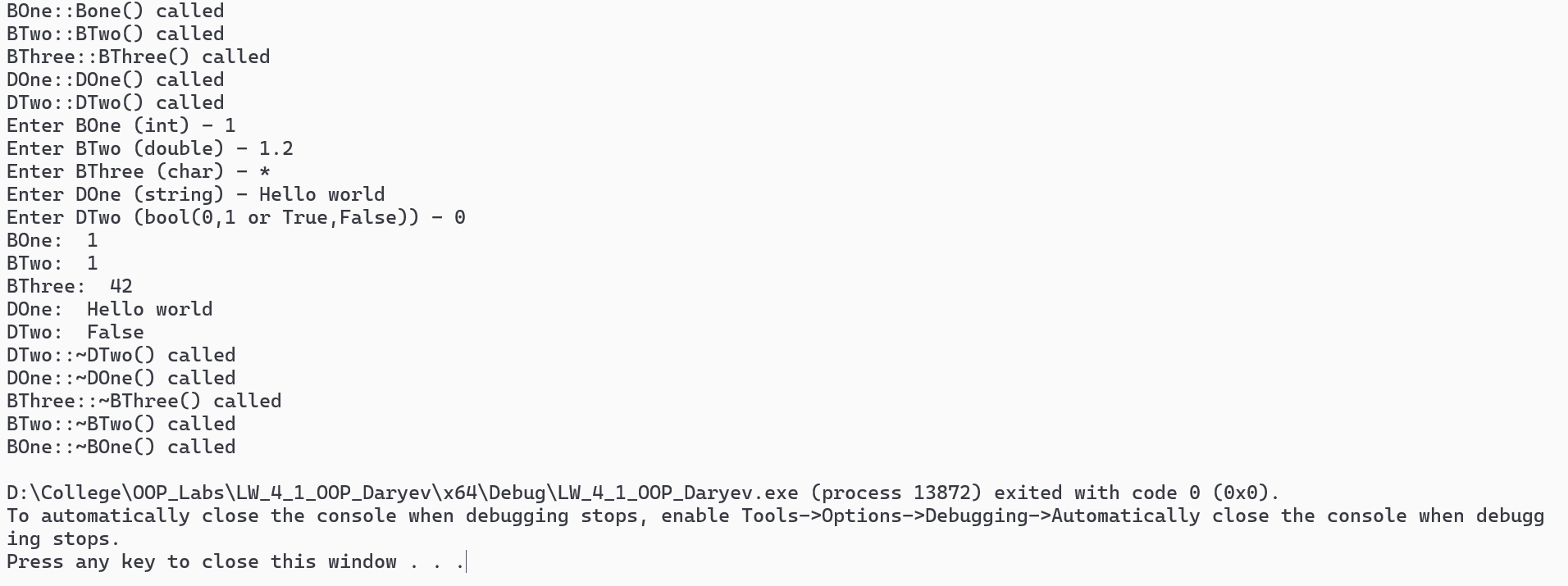
obj.InputInitialise();

obj.Show();

}

}

**Result**

****

**Завдання 2(WinForm)**

Розробити ієрархію класів разом з інтерфейсами в C# Windows Form

**Висновок:** на лабораторній роботі було засвоєно знання як організовувати класи та похідні класи з своїми даними та властивостями: визначати структуру класу, типи даних та методів, організовувати необхідні конструктори та деструктори, використовувати специфікатори доступу , будувати ієрархію класів, використовуючи множинне наслідування, визначати та оперувати об’єктами цих класів, отримати практичні вміння та навички проектування та побудови ієрархії класів та отримано практичні навички використання інтерфейсів в C#.