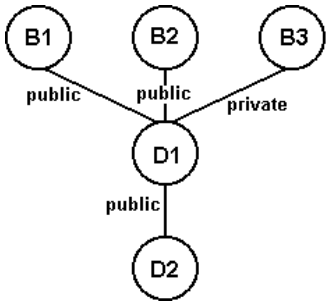
Лабораторна робота № 4
Побудова множинного наслідування класів(C++)
Інтерфейси- C#

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.

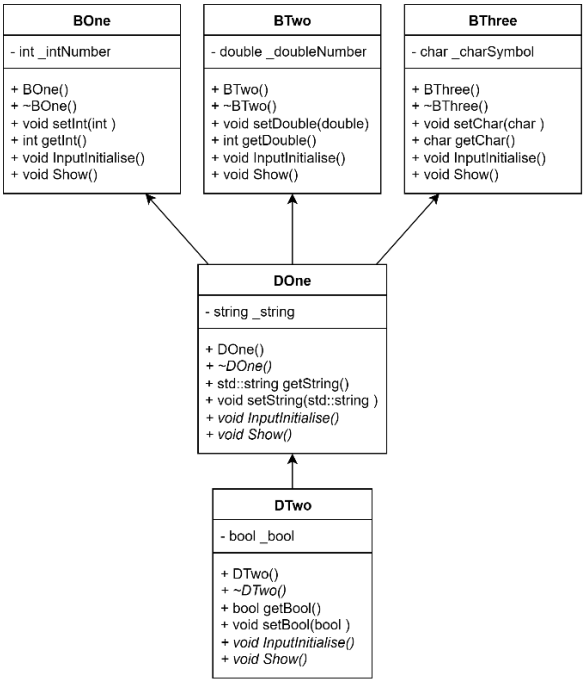
Хід роботи

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



					ЛР.ОК.15.ПІ233.02.09	
Змін.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*		
Розробив	Дар’єв Д.О.				*Літ*	*Аркуш* 1 / *Аркушів* 1
Перевірив						
Н.контр.					ХПК	
Затвер.						

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();
}
```

```cpp
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

```cpp
#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

```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

```
BOne::Bone() called
BTwo::BTwo() called
BThree::BThree() called
DOne::DOne() called
DTwo::DTwo() called
Enter BOne (int) - 1
Enter BTwo (double) - 1.2
Enter BThree (char) - *
Enter DOne (string) - Hello world
Enter DTwo (bool(0,1 or True,False)) - 0
BOne:  1
BTwo:  1
BThree:  42
DOne:  Hello world
DTwo:  False
DTwo::~DTwo() called
DOne::~DOne() called
BThree::~BThree() called
BTwo::~BTwo() called
BOne::~BOne() called

D:\College\OOP_Labs\LW_4_1_OOP_Daryev\x64\Debug\LW_4_1_OOP_Daryev.exe (process 13872) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugg
ing stops.
Press any key to close this window . . .
```

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#.