# **Звіт про виконання лабораторної роботи № 3.3D**

## «Успадкування замість

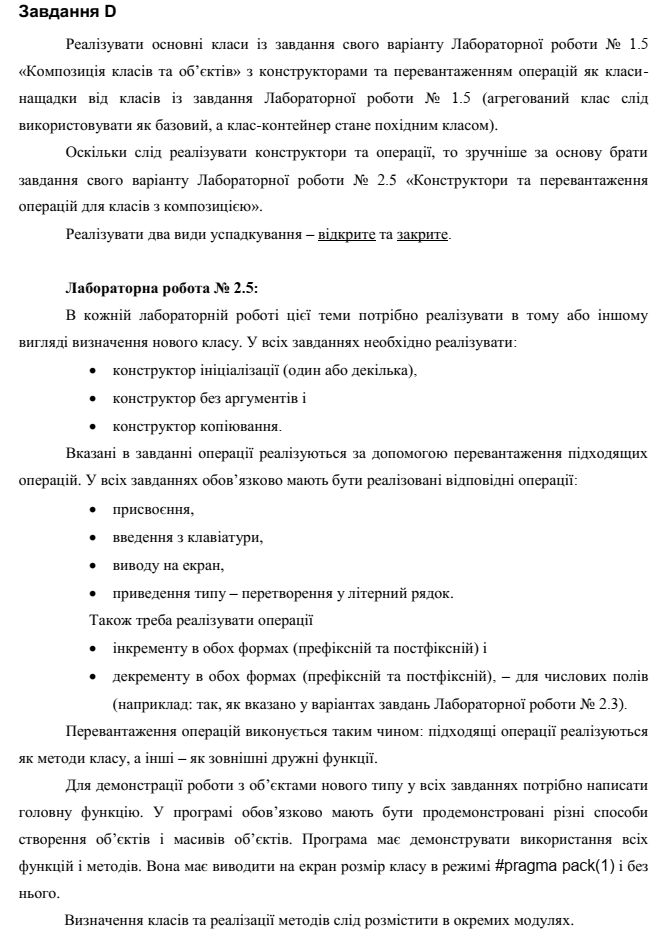
## композиції» з дисципліни «Об’єктно-орієнтоване програмування»

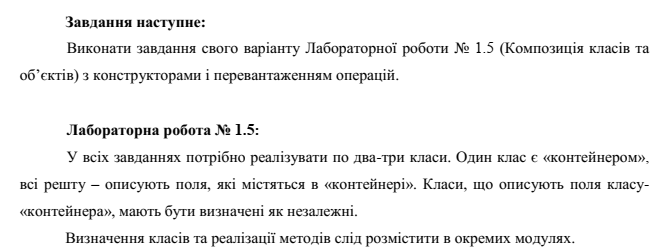
## Студента групи «**ІТ-12**» - **Степанчука Сергія**

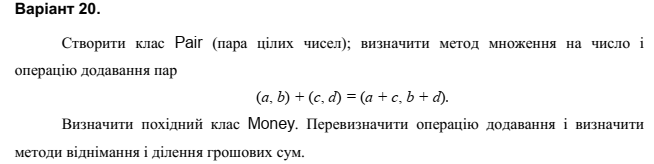
## Мета роботи

Освоїти використання успадкування.

## Умова завдання



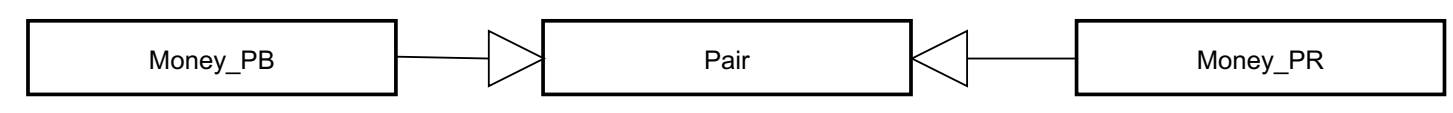




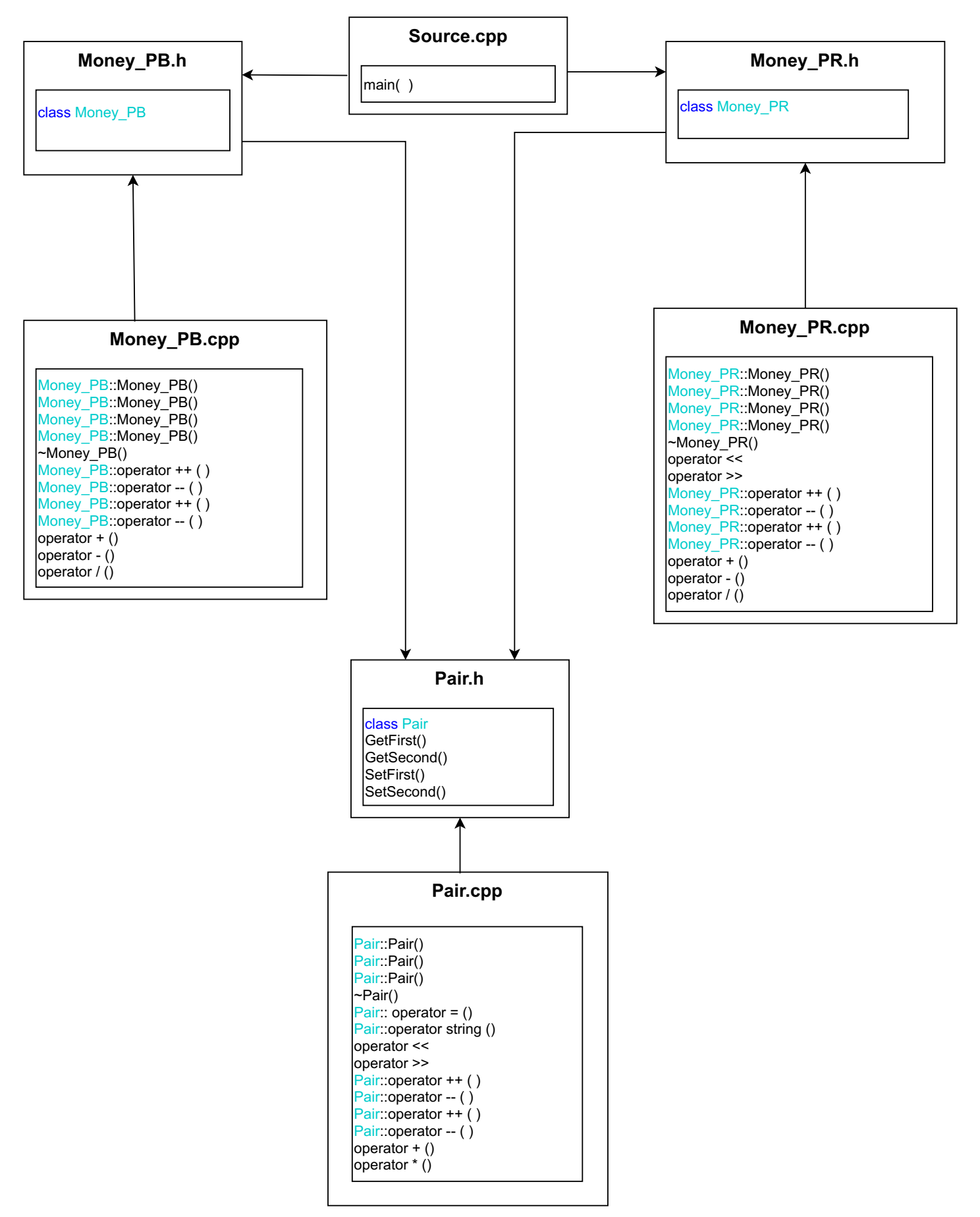
## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/OOP_Lab_3.3D>

## UML-діаграма класів



## Структурна схема



## Текст програми

### // Pair.h

#pragma once

#pragma pack(1)

#include <iostream>

#include <sstream>

#include <string>

#include <math.h>

using namespace std;

class Pair

{

private:

int first, second;

public:

int GetFirst() const { return first; }

int GetSecond() const { return second; }

void SetFirst(int value) { first = value; }

void SetSecond(int value) { second = value; }

Pair();

Pair(const int first, const int second);

Pair(const Pair& S);

~Pair(void);

Pair& operator = (const Pair& s);

operator string () const;

friend ostream& operator << (ostream& out, const Pair& s);

friend istream& operator >> (istream& in, Pair& s);

Pair& operator ++ ();

Pair& operator -- ();

Pair operator ++ (int);

Pair operator -- (int);

friend Pair operator + (Pair&, Pair&);

friend Pair operator \* (Pair&, int);

};

### // Pair.cpp

#include "Pair.h"

Pair::Pair()

: first(0), second(0)

{}

Pair::Pair(const int First, const int Second)

: first(First), second(Second)

{}

Pair::Pair(const Pair& v)

: first(v.first), second(v.second)

{}

Pair::~Pair(void)

{}

Pair& Pair::operator = (const Pair& n)

{

first = n.first;

second = n.second;

return \*this;

}

Pair::operator string () const

{

stringstream ss;

ss << endl;

ss << "a = " << first << endl;

ss << "b = " << second << endl;

ss << "(" << first << ", " << second << ")" << endl;

return ss.str();

}

ostream& operator <<(ostream& out, const Pair& s)

{

return out << string(s);

}

istream& operator >>(istream& in, Pair& s)

{

int first, second;

cout << "a = "; in >> first;

cout << "b = "; in >> second;

s.SetFirst(first);

s.SetSecond(second);

return in;

}

Pair& Pair::operator --()

{

--first;

return \*this;

}

Pair& Pair::operator ++()

{

++first;

return \*this;

}

Pair Pair::operator --(int)

{

Pair first(\*this);

second--;

return first;

}

Pair Pair::operator ++(int)

{

Pair first(\*this);

second++;

return first;

}

Pair operator + (Pair& p1, Pair& p2)

{

Pair tmp;

tmp.first = p1.first + p2.first;

tmp.second = p1.second + p2.second;

return tmp;

}

Pair operator \* (Pair& p, int n)

{

Pair tmp;

tmp.first = p.first \* n;

tmp.second = p.second \* n;

return tmp;

}

### // Money\_PB.h

#pragma once

#pragma pack(1)

#include "Pair.h"

class Money\_PB : public Pair

{

public:

Money\_PB();

Money\_PB(const int first, const int second);

Money\_PB(const Money\_PB& s);

Money\_PB(double p);

~Money\_PB(void);

Money\_PB& operator ++ ();

Money\_PB& operator -- ();

Money\_PB operator ++ (int);

Money\_PB operator -- (int);

friend Money\_PB operator + (Money\_PB&, Money\_PB&);

friend Money\_PB operator - (Money\_PB&, Money\_PB&);

friend double operator / (Money\_PB&, Money\_PB&);

};

### // Money\_PB.cpp

#include "Money\_PB.h"

using namespace std;

Money\_PB::Money\_PB()

: Pair()

{}

Money\_PB::Money\_PB(const int F, const int S)

{

int a = F;

int b = S;

while (b > 99)

{

b -= 100;

a++;

}

while (b < -99)

{

b += 100;

a--;

}

SetFirst(F);

SetSecond(S);

}

Money\_PB::Money\_PB(double p)

{

int a = (int)p;

p -= a;

p \*= 100;

int b = (int)p;

while (b > 99)

{

b -= 100;

a++;

}

while (b < -99)

{

b += 100;

a--;

}

SetFirst(a);

SetSecond(b);

}

Money\_PB::Money\_PB(const Money\_PB& v)

{

int a = v.GetFirst();

int b = v.GetSecond();

while (b > 99)

{

b -= 100;

a++;

}

while (b < -99)

{

b += 100;

a--;

}

SetFirst(a);

SetSecond(b);

}

Money\_PB::~Money\_PB(void)

{}

///////////////

Money\_PB operator + (Money\_PB& p1, Money\_PB& p2)

{

Money\_PB tmp;

int P\_F = p1.GetFirst() + p2.GetFirst();

int P\_S = p1.GetSecond() + p2.GetSecond();

if (P\_S > 99)

{

P\_S = P\_S - 100;

P\_F = P\_F + 1;

}

tmp.SetFirst(P\_F);

tmp.SetSecond(P\_S);

return tmp;

}

Money\_PB operator - (Money\_PB& p1, Money\_PB& p2)

{

Money\_PB tmp;

int P\_F = p1.GetFirst() - p2.GetFirst();

int P\_S = p1.GetSecond() - p2.GetSecond();

while (P\_S > 99)

{

P\_S -= 100;

P\_F++;

}

while (P\_S < -99)

{

P\_S += 100;

P\_F--;

}

tmp.SetFirst(P\_F);

tmp.SetSecond(P\_S);

return tmp;

}

double operator / (Money\_PB& p1, Money\_PB& p2)

{

double P1 = p1.GetFirst() + 0.01 \* p1.GetSecond();

double P2 = p2.GetFirst() + 0.01 \* p2.GetSecond();

return P1 / P2;

}

Money\_PB& Money\_PB::operator ++()

{

this->SetFirst(this->GetFirst() + 1);

return \*this;

}

Money\_PB& Money\_PB::operator --()

{

this->SetFirst(this->GetFirst() - 1);

return \*this;

}

Money\_PB Money\_PB::operator ++(int)

{

Money\_PB a(\*this);

this->SetFirst(this->GetFirst() + 1);

return a;

}

Money\_PB Money\_PB::operator --(int)

{

Money\_PB a(\*this);

this->SetFirst(this->GetFirst() - 1);

return a;

}

### // Money\_PR.h

#pragma once

#pragma pack(1)

#include "Pair.h"

class Money\_PR : private Pair

{

public:

using Pair::GetFirst;

using Pair::GetSecond;

using Pair::SetFirst;

using Pair::SetSecond;

using Pair::Pair;

Money\_PR();

Money\_PR(const int first, const int second);

Money\_PR(const Money\_PR& s);

Money\_PR(const double p);

~Money\_PR(void);

friend ostream& operator << (ostream& out, const Money\_PR& s);

friend istream& operator >> (istream& in, Money\_PR& s);

Money\_PR& operator ++ ();

Money\_PR& operator -- ();

Money\_PR operator ++ (int);

Money\_PR operator -- (int);

friend Money\_PR operator + (Money\_PR&, Money\_PR&);

friend Money\_PR operator - (Money\_PR&, Money\_PR&);

friend double operator / (Money\_PR&, Money\_PR&);

};

### // Money\_PR.cpp

#include "Money\_PR.h"

using namespace std;

Money\_PR::Money\_PR()

: Pair()

{}

Money\_PR::Money\_PR(const int F, const int S)

{

int a = F;

int b = S;

while (b > 99)

{

b -= 100;

a++;

}

while (b < -99)

{

b += 100;

a--;

}

SetFirst(F);

SetSecond(S);

}

Money\_PR::Money\_PR(double p)

{

int a = (int)p;

p -= a;

p \*= 100;

int b = (int)p;

while (b > 99)

{

b -= 100;

a++;

}

while (b < -99)

{

b += 100;

a--;

}

SetFirst(a);

SetSecond(b);

}

Money\_PR::Money\_PR(const Money\_PR& v)

{

int a = v.GetFirst();

int b = v.GetSecond();

while (b > 99)

{

b -= 100;

a++;

}

while (b < -99)

{

b += 100;

a--;

}

SetFirst(a);

SetSecond(b);

}

Money\_PR::~Money\_PR(void)

{}

ostream& operator <<(ostream& out, const Money\_PR& s)

{

out << string(s);

return out;

}

istream& operator >>(istream& in, Money\_PR& s)

{

int first, second;

cout << "a = "; in >> first;

cout << "b = "; in >> second;

s.SetFirst(first);

s.SetSecond(second);

return in;

}

///////////////

Money\_PR operator + ( Money\_PR& p1, Money\_PR& p2)

{

Money\_PR tmp;

int P\_F = p1.GetFirst() + p2.GetFirst();

int P\_S = p1.GetSecond() + p2.GetSecond();

if (P\_S > 99)

{

P\_S = P\_S - 100;

P\_F = P\_F + 1;

}

tmp.SetFirst(P\_F);

tmp.SetSecond(P\_S);

return tmp;

}

Money\_PR operator - ( Money\_PR& p1, Money\_PR& p2)

{

Money\_PR tmp;

int P\_F = p1.GetFirst() - p2.GetFirst();

int P\_S = p1.GetSecond() - p2.GetSecond();

while (P\_S > 99)

{

P\_S -= 100;

P\_F++;

}

while (P\_S < -99)

{

P\_S += 100;

P\_F--;

}

tmp.SetFirst(P\_F);

tmp.SetSecond(P\_S);

return tmp;

}

double operator / ( Money\_PR& p1, Money\_PR& p2)

{

double P1 = p1.GetFirst() + 0.01 \* p1.GetSecond();

double P2 = p2.GetFirst() + 0.01 \* p2.GetSecond();

return P1 / P2;

}

Money\_PR& Money\_PR::operator ++()

{

this->SetFirst(this->GetFirst() + 1);

return \*this;

}

Money\_PR& Money\_PR::operator --()

{

this->SetFirst(this->GetFirst() - 1);

return \*this;

}

Money\_PR Money\_PR::operator ++(int)

{

Money\_PR a(\*this);

this->SetFirst(this->GetFirst() + 1);

return a;

}

Money\_PR Money\_PR::operator --(int)

{

Money\_PR a(\*this);

this->SetFirst(this->GetFirst() - 1);

return a;

}

### // Source.cpp

#include <iostream>

#include "Money\_PB.h"

#include "Money\_PR.h"

using namespace std;

int main()

{

//////////////////////////пара//////////////////////////

Pair p1, p2;

cout << "Input first pair" << endl;

cin >> p1;

cout << "Input second pair" << endl;

cin >> p2;

cout << endl;

cout << "First pair a and b";

cout << p1 << endl;

cout << "Second pair a is c, b is d";

cout << p2 << endl;

//додавання пар

cout << "Sum of pairs:" << p1 + p2 << endl << endl;

//множення на число

int number;

cout << "(number, you want to multiply on) n = ";

cin >> number; cout << endl;

cout << "First pair multyplied on n : " << p1 \* number << endl;

cout << "Second pair multiplyed on n: " << p2 \* number << endl;

cout << endl;

cout << "p1: " << p1 << endl;

cout << "++p1: " << ++p1 << endl;

cout << "--p1: " << --p1 << endl;

cout << "p1++: " << p1++ << endl;

cout << "p1--: " << p1-- << endl << endl;

cout << "--------------------------------------------" << endl << endl;

//////////////////////////гроші//////////////////////////

cout << "PUBLIC" << endl << endl;

{

Money\_PB m1, m2;

cout << endl;

cout << "Input money 1: ";

cin >> m1;

cout << endl;

cout << "Input money 2: ";

cin >> m2;

cout << endl;

cout << "Money 1: ";

cout << m1 << endl;

cout << "Money 2: ";

cout << m2 << endl;

//додавання пар

cout << "Sum of money: " << m1 + m2 << endl;

//віднімання пар

cout << "Difference of money: " << m1 - m2 << endl;

//ділення пар

cout << "Division of money: " << m1 / m2 << endl;

cout << "m1: " << m1 << endl;

cout << "++m1: " << ++m1 << endl;

cout << "--m1: " << --m1 << endl;

cout << "m1++: " << m1++ << endl;

cout << "m1--: " << m1-- << endl << endl;

}

cout << "--------------------------------------------" << endl << endl;

cout << "PRIVATE" << endl << endl;

{

Money\_PR m1, m2;

cout << endl;

cout << "Input money 1: ";

cin >> m1;

cout << endl;

cout << "Input money 2: ";

cin >> m2;

cout << endl;

cout << "Money 1: ";

cout << m1 << endl;

cout << "Money 2: ";

cout << m2 << endl;

//додавання пар

cout << "Sum of money: " << m1 + m2 << endl;

//віднімання пар

cout << "Difference of money: " << m1 - m2 << endl;

//ділення пар

cout << "Division of money: " << m1 / m2 << endl;

cout << "--------------------------------------------" << endl << endl;

cout << "m1: " << m1 << endl;

cout << "++m1: " << ++m1 << endl;

cout << "--m1: " << --m1 << endl;

cout << "m1++: " << m1++ << endl;

cout << "m1--: " << m1-- << endl << endl;

}

}

## UnitTest

### Код

#include "pch.h"

#include "CppUnitTest.h"

#include "../Lab\_3.3-D/Pair.h"

#include "../Lab\_3.3-D/Pair.cpp"

#include "../Lab\_3.3-D/Money\_PB.h"

#include "../Lab\_3.3-D/Money\_PB.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest33D

{

TEST\_CLASS(UnitTest33D)

{

public:

TEST\_METHOD(TestMethod1)

{

Money\_PB m1(6, 0), m2(2, 0);

int D = m1.GetFirst() + m2.GetFirst();

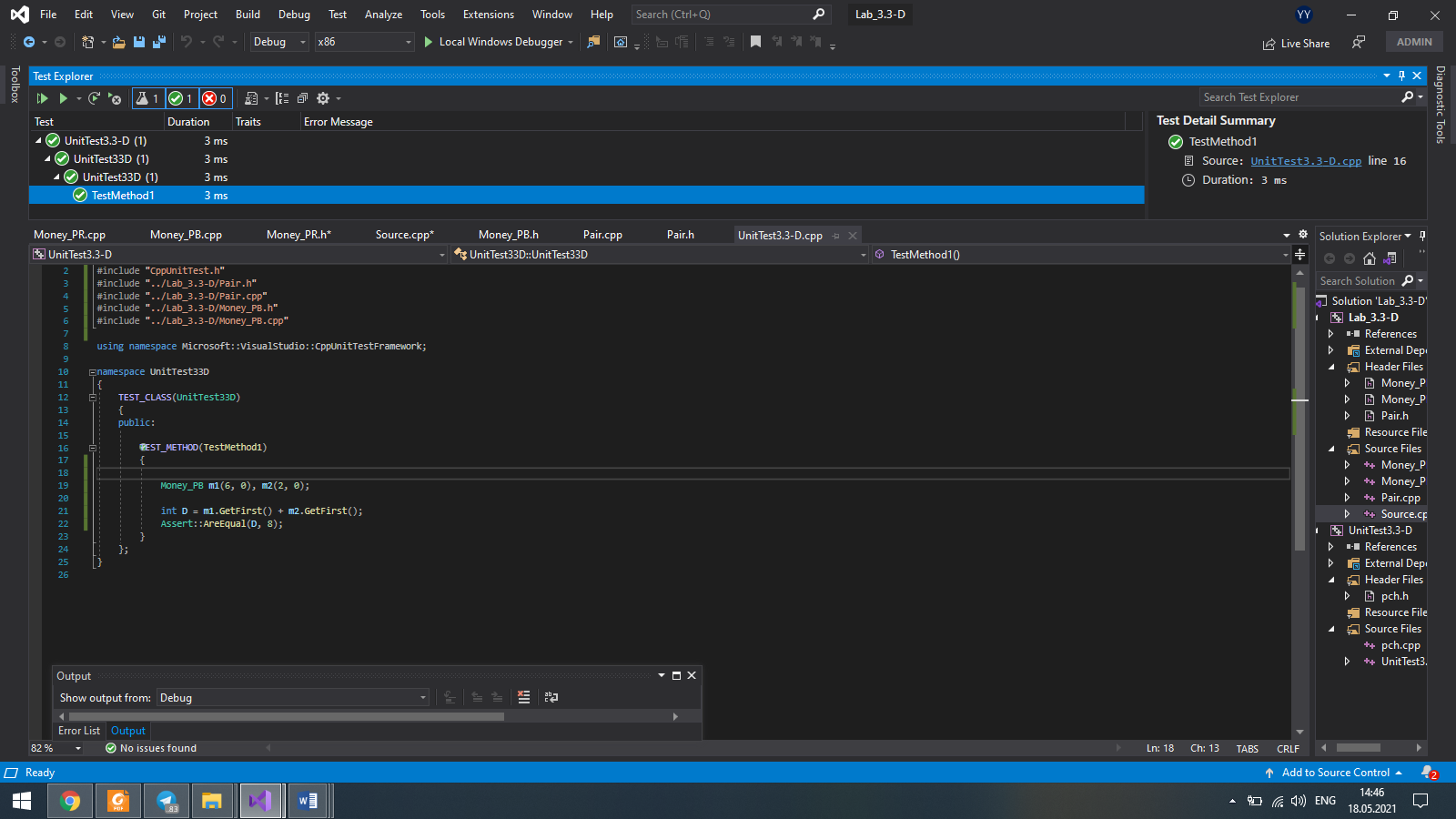
Assert::AreEqual(D, 8);

}

};

}

### Результат



## Висновок

Освоїв використання успадкування.