Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи № 7, варіант 9

З дисципліни « ПРОГРАМУВАННЯ, ЧАСТИНА 2 (ОБ’ЄКТНООРІЄНТОВАНЕ ПРОГРАМУВАННЯ) »

На тему: «Множинне спадкування. Поліморфізм»

Виконав: ст. гр. КІ-15

Горбачук В.Р.

Перевірив: викладач

Козак Н.Б.

Львів – 2020

**Завдання(9):** 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.

**Предметна область** – колонка з флешплеєром;

**Клас 1 –** CFatherWorker;

**Клас 2 –** CWorker;

**Клас 3 –** CFather;

**Клас 4 –** CHuman.

**Код програми:**

#include <iostream>

#include <string>

using namespace std;

class Human

{

public:

virtual void prints() = 0;

void Print()

{

cout << endl << "Overload function in base class" << endl << endl;

}

};

class Father : public Human

{

private:

string name;

string status;

int age;

int weight;

public:

string GetName()

{

return name;

}

string GetStatus()

{

return status;

}

int GetAge()

{

return age;

}

int GetWeight()

{

return weight;

}

void SetName(string name1)

{

name = name1;

}

void SetStatus(string status1)

{

status = status1;

}

void SetAge(int age1)

{

age = age1;

}

void SetWeight(int weight1)

{

weight = weight1;

}

void prints() override

{

cout << endl << "Virtual Method" << endl;

cout << endl << GetName() << endl << "Status of human: " << GetStatus() << endl;

cout << " Age of human" << GetAge() << endl << "Weight of : " << GetWeight() << endl;

}

void Print()

{

cout << endl << "Overloaded Methods" << endl;

cout << endl << "Father: " << GetName() << endl << "Status: " << GetStatus() << endl;

cout << " Age " << GetAge() << endl << "Weight: " << GetWeight() << endl;

}

Father(string newstatus, int newage, int newweight)

{

name = "Steav";

status = newstatus;

age = newage;

weight = newweight;

}

Father() {}

};

class Worker : public Human

{

private:

string name;

string work;

int hour;

int cash;

public:

string GetName()

{

return name;

}

string GetWork()

{

return work;

}

int GetHour()

{

return hour;

}

int GetCash()

{

return cash;

}

void SetName(string name1)

{

name = name1;

}

void SetWork(string work1)

{

work = work1;

}

void SetHour(int hour1)

{

hour = hour1;

}

void SetCash(int cash1)

{

cash = cash1;

}

void prints() override

{

cout << endl << "Virtual Method" << endl;

cout << "Worker: " << GetName() << endl << "Work: " << GetWork() << endl;

cout << "Working day: " << GetHour() << "h " << endl << "Salary: " << GetCash() << "hrn" << endl;

}

void Print()

{

cout << endl << "Overloaded Methods" << endl;

cout << "Worker: " << GetName() << endl << "Work: " << GetWork() << endl;

cout << "Working day: " << GetHour() << "h " << endl << "Salary: " << GetCash() << "hrn" << endl;

}

Worker(string newwork, int newhour, int newcash)

{

name = "Steav";

work = newwork;

hour = newhour;

cash = newcash;

}

Worker() {}

};

class FatherWorker : public Father, public Worker

{

public:

void prints(Human\* human)

{

human->prints();

}

void Print(Human\* human)

{

human->Print();

cout << "Overloaded function in FatherWorker" << endl;

}

};

int main()

{

int a = 0, b = 0, c = 0, d = 0;

Father m("married", 30, 85);

Worker k("Builder", 8, 1500);

FatherWorker mix;

while (a != 3)

{

cout << "Which way you`d like to try?" << endl;

cout << "1.Virtual" << endl << "2.Overload" << endl << "3.Exit" << endl;

cin >> a;

if (a == 1)

{

cout << "How do You Want to use your FatherWorker?" << endl << "1.Father 2.Worker" << endl;

cin >> c;

if (c == 1)

{

mix.prints(&m);

}

else

{

mix.prints(&k);

}

}

else

{

cout << "How do You Want to use your FatherWorker?" << endl << "1.Father 2.Worker" << endl;

cin >> c;

if (c == 1)

{

mix.Print(&k);

}

else

{

mix.Print(&m);

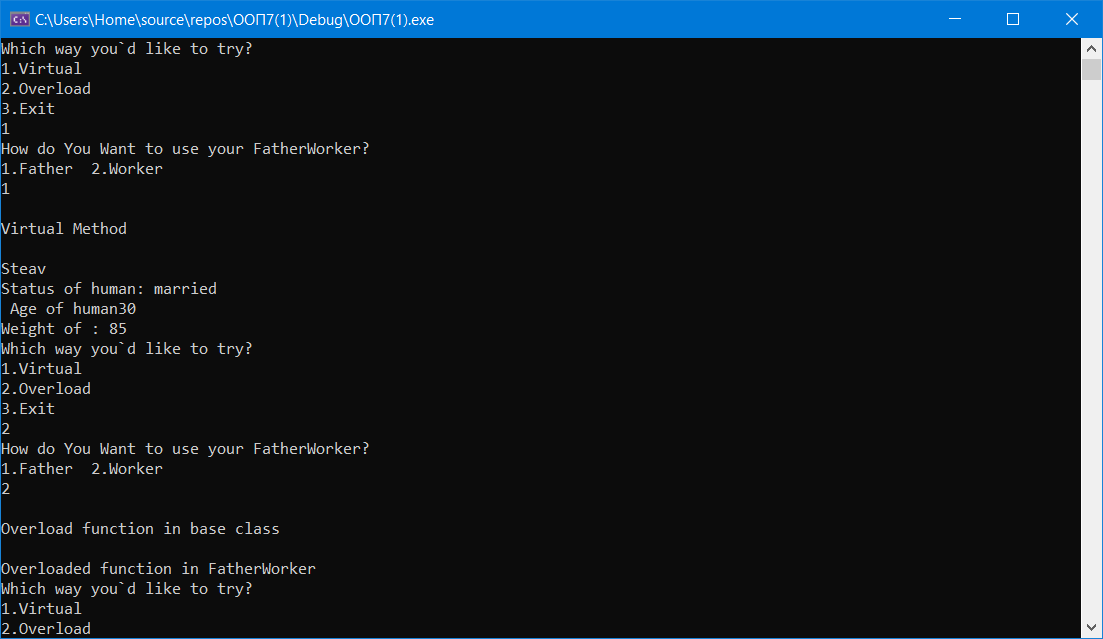
}

}

}

}

**Результат роботи програми:**



**Висновок:** на даній лабораторній роботі я ознайомився із множинним спадкуванням класів та поліформізмом.