Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Кафедра ЕОМ



**ЗВІТ**

З лабораторної роботи № 7

З дисципліни: “Програмування, частина 2 (Об'єктно-орієнтоване програмування)”

На тему: “МНОЖИННЕ СПАДКУВАННЯ. ПОЛІМОРФІЗМ”

Варіант 18

Виконав: ст. гр. КІ-15

Марков Л.Ю.

Прийняв:

Козак Н.Б.

Львів – 2020

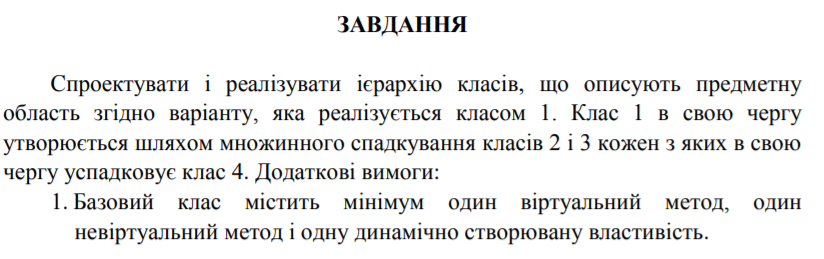
Мета: познайомитися із множинним спадкуванням класів та поліморфізмом.

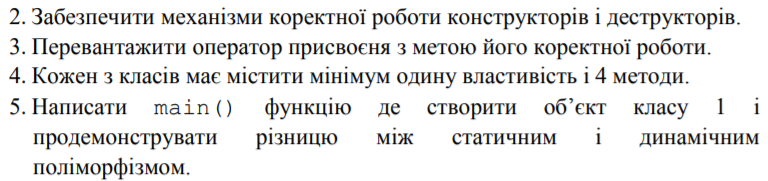
ТЕОРЕТИЧНІ ВІДОМОСТІ

Множинне спадкуування

Якщо спадкування здійснюється від декількох батьківських класів одночасно, тоді воно називається множинним спадкуванням. Визначальним для похідного класу породженого множинним спадкуванням є те, що він явно чи неявно повинен успадковувати характеристики декількох базових класів. Основні принципи одинарного спадкування, зокрема спадкування членів, модифікаторів доступу до членів базових класів, розширення та обмеження характеристик, без жодних доповнень можуть бути перенесені на множинне спадкування. Неявним множинним спадкуванням можуть бути випадки змішаного спадкування. Результатом цих спадкуваннь є ієрархія, в якій похідний клас неявно (через один проміжний) успадкував характеристики двох базових класів.

Варіант 18:







Виконання:

#include <iostream>

#include <string>

using namespace std;

class CEngine {

protected:

int\* fuelConsumption = new int,

fuelLeft;

string model;

public:

CEngine() {

\*fuelConsumption = 0;

this->fuelLeft = 0;

this->model = "Unknown";

}

CEngine(int FuelConsumption, int fuelLeft, string model) {

\*fuelConsumption = FuelConsumption;

this->fuelLeft = fuelLeft;

this->model = model;

}

int getFuelConsumption() {

return \*fuelConsumption;

}

void setFuelConsumption() {

cout << "Enter new fuel consumption: " << endl;

cin >> \*fuelConsumption;

}

void setFuelConsumption(int FuelConsumption) {

\*fuelConsumption = FuelConsumption;

}

int getFuelLeft() {

return fuelLeft;

}

void setFuelLeft() {

cout << "Enter new fuel level: " << endl;

cin >> fuelLeft;

}

void setFuelLeft(int fuelLevel) {

this->fuelLeft = fuelLeft;

}

string getModel() {

return model;

}

void setModel() {

cout << "Enter new model: " << endl;

cin >> model;

}

void setModel(string model) {

this->model = model;

}

virtual void getAll() {};

~CEngine() {

fuelConsumption = NULL;

delete fuelConsumption;

//cout << "Object engine destroyed" << endl;

}

};

class CDieselEngine : virtual public CEngine {

protected:

int cylinders,

compression;

public:

CDieselEngine() {

this->cylinders = 0;

this->compression = 0;

}

CDieselEngine(int cylinders, int compression) {

this->cylinders = cylinders;

this->compression = compression;

}

int getCylinders() {

return cylinders;

}

void setCylinders() {

cout << "Enter new cylinders: " << endl;

cin >> cylinders;

}

void setCylinders(int cylinders) {

this->cylinders = cylinders;

}

int getCompression() {

return compression;

}

void setCompression() {

cout << "Enter new compression: " << endl;

cin >> compression;

}

void setCompressiomm(int compression) {

this->compression = compression;

}

~CDieselEngine() {};

};

class CGasEngine : virtual public CEngine {

protected:

int valves;

public:

CGasEngine() {

this->valves = 0;

}

CGasEngine(int valves) {

this->valves = valves;

}

int getValves() {

return valves;

}

void setValves(int valves) {

this->valves = valves;

}

void setValves() {

cout << "Enter new valves: " << endl;

cin >> valves;

}

~CGasEngine() {};

};

class CMixedEngine : virtual public CGasEngine, CDieselEngine {

protected:

int price;

public:

CMixedEngine() {

this->price = 0;

}

CMixedEngine(int price) {

this->price = price;

}

int getPrice() {

return price;

}

void setPrice() {

cout << "Enter new price: " << endl;

cin >> price;

}

void setPrice(int price) {

this->price = price;

}

CMixedEngine(int FuelConsumption, int fuelLeft, string model, int cylinders, int compression, int valves, int price) {

\*fuelConsumption = FuelConsumption;

this->fuelLeft = fuelLeft;

this->model = model;

this->cylinders = cylinders;

this->compression = compression;

this->valves = valves;

this->price = price;

}

void setAll() {

int choice;

while (true) {

cout << "What do you want to do ?" << endl <<

"1 - get fuel consumption " << endl <<

"2 - get fuel left" << endl <<

"3 - get model" << endl <<

"4 - get cylinders" << endl <<

"5 - get compression" << endl <<

"6 - get valves" << endl <<

"7 - get price" << endl <<

"8 - set fuel consumption " << endl <<

"9 - set fuel left" << endl <<

"10 - set model" << endl <<

"11 - set cylinders" << endl <<

"12 - set compression" << endl <<

"13 - set valves" << endl <<

"14 - set price" << endl <<

"15 - get all values" << endl <<

"0 - close the app" << endl;

cin >> choice;

switch (choice) {

case 1: cout << getFuelConsumption() << endl; break;

case 2: cout << getFuelLeft() << endl; break;

case 3: cout << getModel() << endl; break;

case 4: cout << getCylinders() << endl; break;

case 5: cout << getCompression() << endl; break;

case 6: cout << getValves() << endl; break;

case 7: cout << getPrice() << endl; break;

case 8: setFuelConsumption(); break;

case 9: setFuelLeft(); break;

case 10: setModel(); break;

case 11: setCylinders(); break;

case 12: setCompression(); break;

case 13: setValves(); break;

case 14: setPrice(); break;

case 15: getAll(); break;

case 0: break;

default: cout << "Wrong input" << endl; break;

}

if (choice == 0)

break;

}

}

void getAll() override {

cout << "Fuel consumption: " << \*fuelConsumption << endl <<

"Fuel left: " << fuelLeft << endl <<

"Model: " << model << endl <<

"Cylinders: " << cylinders << endl <<

"Compression: " << compression << endl <<

"Valves: " << valves << endl <<

"Price: " << price << endl;

}

void operator = (const CMixedEngine& other) {

this->fuelConsumption = other.fuelConsumption;

this->fuelLeft = other.fuelLeft;

this->model = other.model;

this->cylinders = other.cylinders;

this->compression = other.compression;

this->valves = other.valves;

this->price = other.price;

}

CMixedEngine(const CMixedEngine& other) {

this->fuelConsumption = other.fuelConsumption;

this->fuelLeft = other.fuelLeft;

this->model = other.model;

this->cylinders = other.cylinders;

this->compression = other.compression;

this->valves = other.valves;

this->price = other.price;

}

~CMixedEngine() {};

};

int main() {

CMixedEngine obj(10, 2, "TDI 50", 4, 14, 16, 10000);

CMixedEngine obj1(obj);

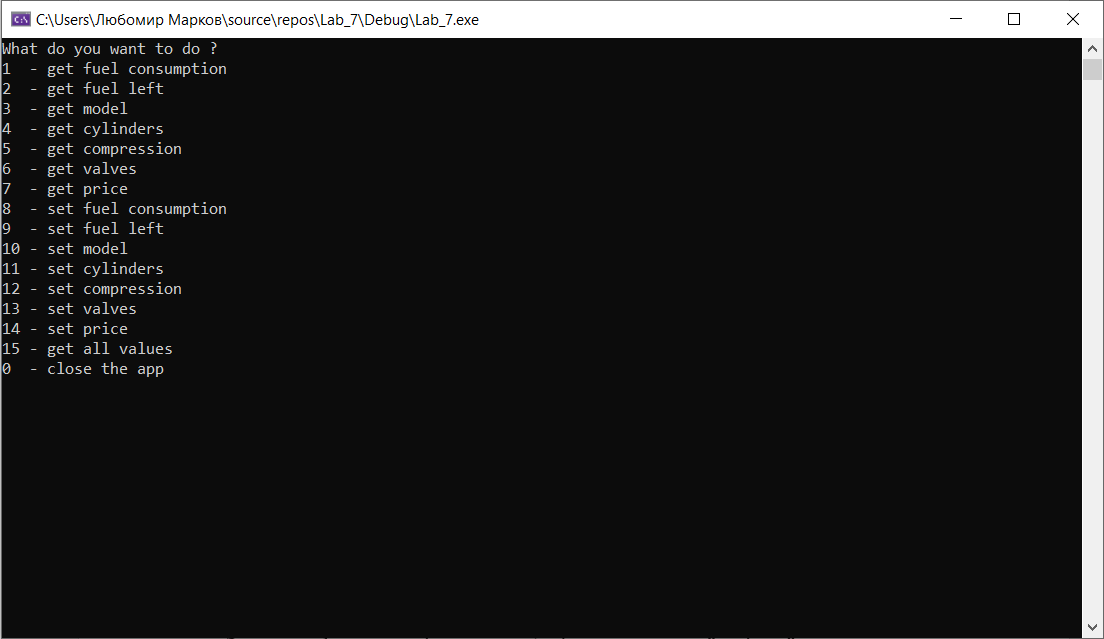
obj1.setModel("TDI 45");

obj1.setAll();

return 0;

}

**Результат:**



Висновок: на цій лабораторній роботі я познайомився із множинним спадкуванням класів та поліформізмом.