Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №7**

на тему:

“Множинне спадкування. Поліморфізм”

**Виконав:**

Ст. гр. КІ-15

Залевський К.С.

**Перевірив:**

Викладач

Козак Н.Б.

Львів – 2020

**Мета роботи:** познайомитися із множинним спадкуванням класів та поліморфізмом.

***Завдання:***

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. Додаткові вимоги:

1. Базовий клас містить мінімум один віртуальний метод, один невіртуальний метод і одну динамічно створювану властивість.

2. Забезпечити механізми коректної роботи конструкторів і деструкторів.

3. Перевантажити оператор присвоєня з метою його коректної роботи.

4. Кожен з класів має містити мінімум одину властивість і 4 методи.

5. Написати main() функцію де створити об‘єкт класу 1 і продемонструвати різницю між статичним і динамічним поліморфізмом.





*Код програми:*

#include <iostream>

using namespace std;

//Газоелектричний бойлер

class CBoiler //Класс 4

{

public:

CBoiler()

{

cout << "Constructor CBoiler" << endl;

capacity = new double;

\*capacity = 0;

state = false;

efficiency = 0;

}

virtual ~CBoiler()

{

cout << "Destructor CBoiler" << endl;

delete this->capacity;

this->capacity = nullptr;

}

virtual void SmartTurnOn(double w\_capacity) = 0;

virtual void TurnOff() = 0;

virtual void TimeNeed(double w\_capacity) = 0;

void SetCapacity(double Capacity)

{

\*capacity = Capacity;

}

void SetEfficiency(int efficiency)

{

this->efficiency = efficiency;

}

double TimeToFinishBoiling(double water\_capacity, int efficiency)

{

if (water\_capacity > \* this->capacity)

{

cout << "Error"<<endl;

return 0;

}

return water\_capacity/efficiency;

}

protected:

double\* capacity;

bool state;

int efficiency;

};

class CElectricBoiler:public CBoiler //Класс 2

{

public:

CElectricBoiler()

{

cout << "Constructor CElectricBoiler" << endl;

EnergyUsage = 0;

efficiency = 0;

}

CElectricBoiler(double energyusage) : CElectricBoiler()

{

cout << "#2 Constructor CElectricBoiler" << endl;

EnergyUsage = energyusage;

}

CElectricBoiler(double energyusage, int efficiency): CElectricBoiler(energyusage)

{

cout << "#3 Constructor CElectricBoiler" << endl;

this->efficiency = efficiency;

}

~CElectricBoiler() override

{

cout << "Destructor CBoiler" << endl;

}

//Method's

void TimeNeed(double w\_capacity)override

{

if (this->efficiency == 0)

{

cout << "Error" << endl;

return;

}

cout << "For ElectricBoiler need time: " << efficiency / w\_capacity << ";" << endl;

}

void SmartTurnOn(double w\_capacity) override

{

if (this->state)

{

cout << "Error: Electric boiler already on!" << endl;

return;

}

if (this->capacity=0)

{

cout << "Error, add water!" << endl;

return;

}

if (\*this->capacity < w\_capacity)

{

cout << "Error: too much water!" << endl;

return;

}

cout<< "Turn on Electric Boiler!" <<endl;

state = true;

}

void TurnOff() override

{

cout << "TurnOff Electric Bolier";

state = false;

}

void SetEnergyUsage(double energyusage)

{

this->EnergyUsage = energyusage;

}

double GetEnergyUsage()

{

return this->EnergyUsage;

}

protected:

double EnergyUsage;

};

class CGasBoiler:public CBoiler //Класс 3

{

public:

CGasBoiler()

{

cout << "Constructor CGasBoiler" << endl;

GasCapacity = 0;

efficiency = 0;

}

CGasBoiler(double gascapacity) :CGasBoiler()

{

cout << "#2 Constructor CGasBoiler" << endl;

GasCapacity = gascapacity;

}

CGasBoiler(double gascapacity, int efficiency) :CGasBoiler(gascapacity)

{

cout << "#3 Constructor CGasBoiler" << endl;

this->efficiency = efficiency;

}

~CGasBoiler() override

{

cout << "Destructor CGasBoiler" << endl;

}

//Method's

void TimeNeed(double w\_capacity)override

{

if (this->efficiency == 0)

{

cout << "Error" << endl;

return;

}

cout << "For GasBoiler need time: " << efficiency / w\_capacity << ";" << endl;

}

void SmartTurnOn(double w\_capacity) override

{

if (this->state)

{

cout << "Error: Gas boiler already on!" << endl;

return;

}

if (this->capacity = 0)

{

cout << "Error, add water!" << endl;

return;

}

if (\*this->capacity < w\_capacity)

{

cout << "Error: too much water!" << endl;

return;

}

cout << "Turn on Gas Boiler!" << endl;

state = true;

}

void TurnOff() override

{

cout << "TurnOff Electric Bolier";

state = false;

}

void SetGasCapacity(double gascapacity)

{

this->GasCapacity = gascapacity;

}

protected:

double GasCapacity;

};

class CMixedBoiler:public CElectricBoiler, public CGasBoiler //Класс 1

{

public:

CMixedBoiler()

{

cout << "Constructor CMixedBoiler" << endl;

power = 0;

}

CMixedBoiler(double power):CMixedBoiler()

{

cout << "Constructor CMixedBoiler" << endl;

this->power = power;

}

~CMixedBoiler() override

{

cout << "Destructor CMixedBoiler" << endl;

}

CMixedBoiler operator=(const CMixedBoiler& other)

{

this->power = other.power;

this->GasCapacity = other.GasCapacity;

this->EnergyUsage = other.EnergyUsage;

this->::CBoiler::efficiency = other.::CBoiler::efficiency;

this->::CBoiler::capacity = other.::CBoiler::capacity;

}

void TimeNeed(double w\_capacity)override

{

if (this->::CBoiler::efficiency == 0)

{

cout << "Error" << endl;

return;

}

cout << "For MixedBoiler need time: " << ::CBoiler::efficiency / w\_capacity << ";" << endl;

}

void SetPower(double power)

{

this->power = power;

}

double GetPower()

{

return this->power;

}

void SmartTurnOn(double w\_capacity)override

{

if (this->::CBoiler::state)

{

cout << "Error: CMixed boiler already on!" << endl;

return;

}

if (this->::CBoiler::capacity = 0)

{

cout << "Error, add water!" << endl;

return;

}

if (\*this->::CBoiler::capacity < w\_capacity)

{

cout << "Error: too much water!" << endl;

return;

}

if (this->GasCapacity == 0)

{

cout << "Error: no Gas!" << endl;

return;

}

cout << "MixedBoiler start!" << endl;

::CBoiler::state = true;

}

void TurnOff() override

{

cout << "MixedBoiler turn off!" << endl;

::CBoiler::state = false;

}

private:

double power;

};

int main()

{

CMixedBoiler mb;

mb.SetEnergyUsage(2.7);

mb.SetGasCapacity(5.7);

mb.SetPower(8.0);

//Поліморфізм

CGasBoiler gb;

CElectricBoiler eb;

//Статичний поліморфізм

gb.SetEfficiency(3);

gb.TimeNeed(5.6);

eb.SetEfficiency(5);

eb.TimeNeed(5.6);

//Динамічний Поліморфізм

CBoiler\* m;

m = &gb;

m->SetEfficiency(7);

m->TimeNeed(5.6);

m = &eb;

m->SetEfficiency(8);

m->TimeNeed(5.6);

}

*Вікно результату:*



*Висновок:* я познайомився з множинним спадкуванням класів та з поліморфізмом.