Лабораторна робота №3. Успадкування і віртуальні функції

Мета: Одержати практичні навички створення ієрархії класів і використання статичних компонентів класу.

<u>Основний зміст роботи.</u>

Написати програму, в якій створюється ієрархія класів. Включити поліморфні об'єкти в зв'язаний список, використовуючи статичні компоненти класу. Показати використання віртуальних функцій.

Задача

1. Створити абстрактний клас Trans з методами дозволяючими вивести на екран інформацію про транспортний засіб, а також визначити вантажопідйомність транспортного засобу.
2. Створити похідні класи: Легковая_машина (марка, номер, швидкість, вантажопідйомність), Мотоцикл (марка, номер, швидкість, вантажопідйомність, наявність коляски, при цьому якщо коляска відсутня, то вантажопідйомність рівна 0), Грузовик (марка, номер, швидкість, вантажопідйомність, наявність причепа, при цьому якщо є причіп, то вантажопідйомність збільшується в два рази) з своїми методами виведення інформації на екран, і визначення вантажопідйомності.
3. Створити базу (масив) з n машин, вивести повну інформацію з бази на екран, а також організувати пошук машин, що задовольняють вимогам вантажопідйомності.

Ієрархія класів

Код програми

Vehicle.hpp

```
#pragma once
#ifndef VEHICLE_HPP
#define VEHICLE_HPP


class Vehicle
{
protected:
        std::string _marks;
        unsigned int _number;
        unsigned int _speed;
        unsigned int _capacity;

public:

        // --------------------- Constructor and Destructor ----------------------------
        Vehicle(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity);
        Vehicle(const Vehicle& other);
        Vehicle(Vehicle&& other)noexcept;
        virtual ~Vehicle();

        // ------------------------- Getters and Setters ----------------------------
        std::string getMarks() const;
        unsigned int getNumber() const;
```

					ЛР.ОК.19.ПІ231.20.03
Змін.	Аркуш	№ докум.	Підпис	Дата	
Розробив		Дар’єв Д.О.			**Успадкування і віртуальні функції**
Перевірив		Жереб Д.В.			
Н.контр.					
Затвер.					

Літ	Аркуш	Аркушів
	1	1
ХПК		

```
		unsigned int getSpeed() const;
		unsigned int getCapacity() const;

		void setMarks(const std::string& marks);
		void setNumber(unsigned int number);
		virtual void setSpeed(unsigned int speed);
		virtual void setCapacity(unsigned int capacity);

		// ------------------------- Other Methods ----------------------------

		virtual void ShowInfo() const = 0;
		virtual void ShowInfoInTable() const = 0;

};

#endif // VEHICLE_HPP
```

Vehicle.cpp

```
#include <string>
#include <iostream>
#include "Vehicle.hpp"

// ---------------------- Constructor and Destructor ----------------------------
Vehicle::Vehicle(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity)
{
		setMarks(marks);
		setNumber(number);
		setSpeed(speed);
		setCapacity(capacity);

		std::cout << "Vehicle::Vehicle(string,uint,uint,uint) called" << std::endl;
}

Vehicle::Vehicle(const Vehicle& other)
{
		setMarks(other._marks);
		setNumber(other._number);
		setSpeed(other._speed);
		setCapacity(other._capacity);
		std::cout << "Vehicle::Vehicle(const Vehicle&) called" << std::endl;
}

Vehicle::Vehicle(Vehicle&& other)noexcept
{
		setMarks(other._marks);
		setNumber(other._number);
		setSpeed(other._speed);
		setCapacity(other._capacity);
		other._marks = "NoName";
		other._number = 0;
		other._speed = 0;
		other._capacity = 0;
		std::cout << "Vehicle::Vehicle(Vehicle&&) called" << std::endl;
}

Vehicle::~Vehicle()
{
		std::cout << "Vehicle::~Vehicle() called" << std::endl;
}

// ------------------------- Getters and Setters ----------------------------
std::string Vehicle::getMarks() const
{
		return _marks;
}

unsigned int Vehicle::getNumber() const
{
	return _number;
}

unsigned int Vehicle::getSpeed() const
{
	return _speed;
}

unsigned int Vehicle::getCapacity() const
{
	return _capacity;
}

void Vehicle::setMarks(const std::string& marks)
{

	if (marks != " " && !marks.empty()) {
		_marks = marks;
	}
	else _marks = "NoName";
}

void Vehicle::setNumber(unsigned int number)
{
	if (number > 0) {
		_number = number;
	}
	else _number = 0;
}

void Vehicle::setSpeed(unsigned int speed)
{
	if (speed > 0) {
		_speed = speed;
	}
	else _speed = 0;
}

void Vehicle::setCapacity(unsigned int capacity)
{
	if (capacity > 0) {
		_capacity = capacity;
	}
	else _capacity = 0;
}
```

Car.hpp

```
#pragma once
#ifndef CAR_HPP
#define CAR_HPP


class Car : public Vehicle
{
public:
	// ---------------------- Constructor and Destructor -----------------------------
	Car(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity);
	Car(const Car& other);
	Car(Car&& other)noexcept;
	~Car()override;
```

```
		// ------------------------- Getters and Setters ----------------------------
		void setSpeed(unsigned int speed) override;
		// ------------------------- Other Methods ----------------------------
		void ShowInfo() const override;
		void ShowInfoInTable() const override;

	};

	#endif // CAR_HPP
```

Car.cpp

```
#include <string>
#include <iostream>
#include <iomanip>
#include "Vehicle.hpp"
#include "Car.hpp"

Car::Car(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity)
	: Vehicle(marks, number, speed, capacity)
{
	std::cout << "Car::Car(string,uint,uint,uint) called" << std::endl;
}

Car::Car(const Car& other)
	: Vehicle(other)
{
	std::cout << "Car::Car(const Car&) called" << std::endl;
}

Car::Car(Car&& other)noexcept
	: Vehicle(std::move(other))
{
	std::cout << "Car::Car(Car&&) called" << std::endl;
}

Car::~Car()
{
	std::cout << "Car::~Car() called" << std::endl;
}

void Car::setSpeed(unsigned int speed)
{
	if (speed > 0) {
		_speed = speed;
	}
	_speed = 100;
}

void Car::ShowInfo() const
{
	std::cout << "Car Info: " << std::endl;
	std::cout << "Marks: " << _marks << std::endl;
	std::cout << "Number: " << _number << std::endl;
	std::cout << "Speed: " << _speed << std::endl;
	std::cout << "Capacity: " << _capacity << std::endl;
}

void Car::ShowInfoInTable() const
{
	std::cout << std::left;
	std::cout << std::setw(15) << _marks
		<< std::setw(15) << _number
		<< std::setw(15) << _speed
		<< std::setw(15) << _capacity << std::endl;
```

```
        std::cout << std::right;
}
```
Motorcycle.hpp
```
#pragma once
#ifndef MOTORCYCLE_HPP
#define MOTORCYCLE_HPP

class Motorcycle : public Vehicle
{
        bool _hasSidecar;
public:
        // ---------------------- Constructor and Destructor ----------------------------
        Motorcycle(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity, bool hasSidecar);
        Motorcycle(const Motorcycle& other);
        Motorcycle(Motorcycle&& other)noexcept;
        ~Motorcycle()override;
        // ------------------------- Getters and Setters ----------------------------
        bool getHasSidecar() const;
        void setHasSidecar(bool hasSidecar);
        void setSpeed(unsigned int speed) override;
        void setCapacity(unsigned int capacity) override;
        // ------------------------- Other Methods ----------------------------
        void ShowInfo() const override;
        void ShowInfoInTable() const override;
};

#endif // MOTORCYCLE_HPP
```
Motorcycle.cpp
```
#include <string>
#include <iostream>
#include <iomanip>
#include "Vehicle.hpp"
#include "Motorcycle.hpp"

// ---------------------- Constructor and Destructor ----------------------------
Motorcycle::Motorcycle(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity, bool
hasSidecar)
        : Vehicle(marks, number, speed, capacity), _hasSidecar(hasSidecar)
{
        std::cout << "Motorcycle::Motorcycle(string,uint,uint,uint,bool) called" << std::endl;
}

Motorcycle::Motorcycle(const Motorcycle& other)
        : Vehicle(other), _hasSidecar(other._hasSidecar)
{
        std::cout << "Motorcycle::Motorcycle(const Motorcycle&) called" << std::endl;
}

Motorcycle::Motorcycle(Motorcycle&& other) noexcept
        : Vehicle(std::move(other)), _hasSidecar(other._hasSidecar)
{
        std::cout << "Motorcycle::Motorcycle(Motorcycle&&) called" << std::endl;
}

Motorcycle::~Motorcycle()
{
        std::cout << "Motorcycle::~Motorcycle() called" << std::endl;
}
// ------------------------- Getters and Setters ----------------------------
bool Motorcycle::getHasSidecar() const
{
        return _hasSidecar;
}

void Motorcycle::setHasSidecar(bool hasSidecar)
{
	_hasSidecar = hasSidecar;
}

void Motorcycle::setSpeed(unsigned int speed)
{
	if (speed > 0) {
		_speed = speed;
	}
	_speed = 150;
}

void Motorcycle::setCapacity(unsigned int capacity)
{
	if (capacity < 0 && getHasSidecar()) {
		_capacity = 0;
	}
	else {
		_capacity = capacity;
	}
}
// -------------------------- Other Methods ----------------------------
void Motorcycle::ShowInfo() const
{
	std::cout << "Motorcycle Info: " << std::endl;
	std::cout << "Marks: " << _marks << std::endl;
	std::cout << "Number: " << _number << std::endl;
	std::cout << "Speed: " << _speed << std::endl;
	std::cout << "Capacity: " << _capacity << std::endl;
	std::cout << "Has Sidecar: " << (_hasSidecar ? "Yes" : "No") << std::endl;
}

void Motorcycle::ShowInfoInTable() const
{
	std::cout << std::left;
	std::cout << std::setw(15) << _marks
		<< std::setw(15) << _number
		<< std::setw(15) << _speed
		<< std::setw(15) << _capacity
		<< std::setw(15) << (_hasSidecar ? "Sidecar" : "NoSidecar") << std::endl;
	std::cout << std::right;
}
```

Truck.hpp

```
#pragma once
#ifndef TRUCK_HPP
#define TRUCK_HPP

class Truck : public Vehicle
{
private:
	bool _hasTrailer;
public:
	// --------------------- Constructor and Destructor ----------------------------
	Truck(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity, bool hasTrailer);
	Truck(const Truck& other);
	Truck(Truck&& other)noexcept;
	~Truck()override;
	// -------------------------- Getters and Setters ----------------------------
	bool getHasTrailer() const;
	void setHasTrailer(bool hasTrailer);
```

```
        void setSpeed(unsigned int speed) override;
        void setCapacity(unsigned int capacity) override;
        // -------------------------- Other Methods -----------------------------
        void ShowInfo() const override;
        void ShowInfoInTable() const override;
};

#endif // TRUCK_HPP
```

Truck.cpp

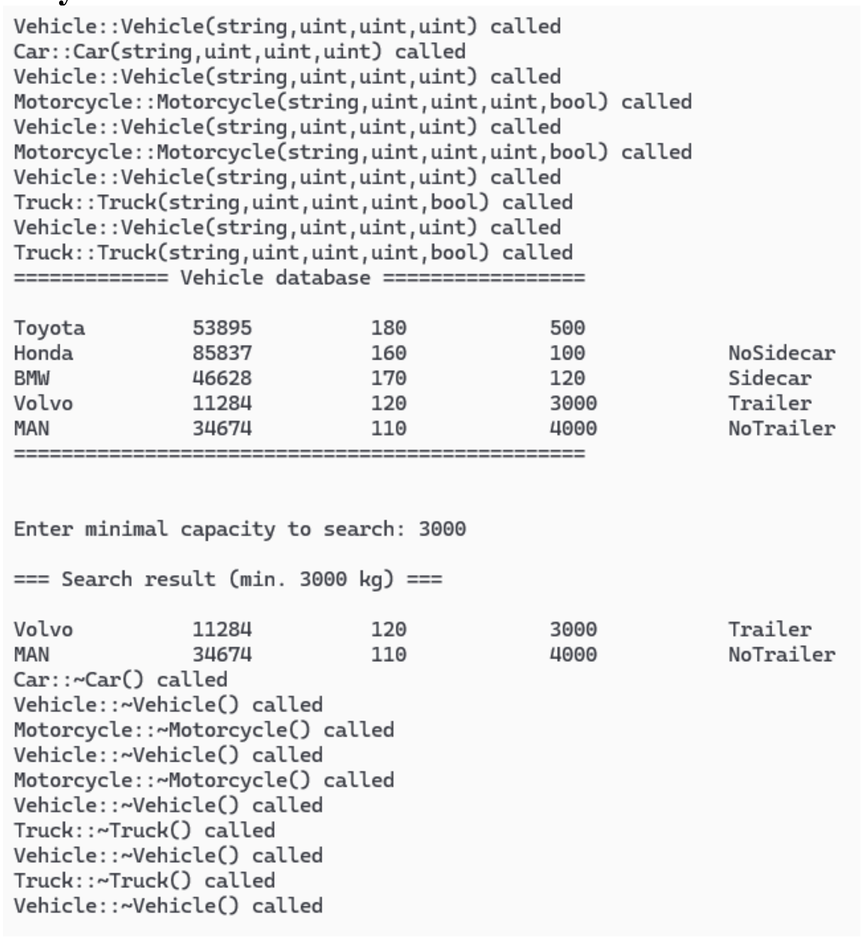
```
#include <string>
#include <iostream>
#include <iomanip>
#include "Vehicle.hpp"
#include "Truck.hpp"

// ---------------------- Constructor and Destructor -----------------------------
Truck::Truck(const std::string& marks, unsigned int number, unsigned int speed, unsigned int capacity, bool hasTrailer)
        : Vehicle(marks, number, speed, capacity), _hasTrailer(hasTrailer)
{
        std::cout << "Truck::Truck(string,uint,uint,uint,bool) called" << std::endl;
}

Truck::Truck(const Truck& other)
        : Vehicle(other), _hasTrailer(other._hasTrailer)
{
        std::cout << "Truck::Truck(const Truck&) called" << std::endl;
}

Truck::Truck(Truck&& other) noexcept
        : Vehicle(std::move(other)), _hasTrailer(other._hasTrailer)
{
        std::cout << "Truck::Truck(Truck&&) called" << std::endl;
}

Truck::~Truck()
{
        std::cout << "Truck::~Truck() called" << std::endl;
}
// -------------------------- Getters and Setters ----------------------------
bool Truck::getHasTrailer() const
{
        return _hasTrailer;
}

void Truck::setHasTrailer(bool hasTrailer)
{
        _hasTrailer = hasTrailer;
}

void Truck::setSpeed(unsigned int speed)
{
        if (speed > 0) {
                _speed = speed;
        }
        _speed = 80;
}

void Truck::setCapacity(unsigned int capacity)
{
        if (capacity < 0) { _capacity = 0; }
        else if (getHasTrailer()) { _capacity = capacity * 2; }
        else _capacity = capacity;
}
//		-------------------------- Other Methods -----------------------------
void Truck::ShowInfo() const
{
		std::cout << "Truck Info: " << std::endl;
		std::cout << "Marks: " << _marks << std::endl;
		std::cout << "Number: " << _number << std::endl;
		std::cout << "Speed: " << _speed << std::endl;
		std::cout << "Capacity: " << _capacity << std::endl;
		std::cout << "Has Trailer: " << (_hasTrailer ? "Yes" : "No") << std::endl;
}

void Truck::ShowInfoInTable() const
{
		std::cout << std::left;
		std::cout << std::setw(15) << _marks
				<< std::setw(15) << _number
				<< std::setw(15) << _speed
				<< std::setw(15) << _capacity
				<< std::setw(15) << (_hasTrailer ? "Trailer" : "NoTrailer") << std::endl;
		std::cout << std::right;
}
```

Main.cpp

```
#include <iostream>
#include <vector>
#include "Vehicle.hpp"
#include "Car.hpp"
#include "Motorcycle.hpp"
#include "Truck.hpp"



int main() {

		std::vector<Vehicle*> base;

		base.push_back(new Car("Toyota", 53895, 180, 500));
		base.push_back(new Motorcycle("Honda", 85837, 160, 100, false));
		base.push_back(new Motorcycle("BMW", 46628, 170, 120, true));
		base.push_back(new Truck("Volvo", 11284, 120, 3000, true));
		base.push_back(new Truck("MAN", 34674, 110, 4000, false));

		std::cout << "============= Vehicle database =================\n\n";
		for (auto t : base) {
				t->ShowInfoInTable();
		}
		std::cout << "=============================================\n\n";
		double required;
		std::cout << "\nEnter minimal capacity to search: ";
		std::cin >> required;

		std::cout << "\n=== Search result (min. " << required << " kg) ===\n\n";
		bool found = false;
		for (auto t : base) {
				if (t->getCapacity() >= required) {
						t->ShowInfoInTable();
						found = true;
				}
		}
		if (!found) {
				std::cout << "There are no vehicles with this load capacity..\n";
		}
```

```
	for (auto t : base) {
		delete t;
	}

	return 0;
}
```

Результат



```
Vehicle::Vehicle(string,uint,uint,uint) called
Car::Car(string,uint,uint,uint) called
Vehicle::Vehicle(string,uint,uint,uint) called
Motorcycle::Motorcycle(string,uint,uint,uint,bool) called
Vehicle::Vehicle(string,uint,uint,uint) called
Motorcycle::Motorcycle(string,uint,uint,uint,bool) called
Vehicle::Vehicle(string,uint,uint,uint) called
Truck::Truck(string,uint,uint,uint,bool) called
Vehicle::Vehicle(string,uint,uint,uint) called
Truck::Truck(string,uint,uint,uint,bool) called
============= Vehicle database =================

Toyota         53895          180            500
Honda          85837          160            100            NoSidecar
BMW            46628          170            120            Sidecar
Volvo          11284          120            3000           Trailer
MAN            34674          110            4000           NoTrailer
================================================


Enter minimal capacity to search: 3000

=== Search result (min. 3000 kg) ===

Volvo          11284          120            3000           Trailer
MAN            34674          110            4000           NoTrailer
Car::~Car() called
Vehicle::~Vehicle() called
Motorcycle::~Motorcycle() called
Vehicle::~Vehicle() called
Motorcycle::~Motorcycle() called
Vehicle::~Vehicle() called
Truck::~Truck() called
Vehicle::~Vehicle() called
Truck::~Truck() called
Vehicle::~Vehicle() called
```

Висновок

На лабораторній роботі було одержано практичні навички створення ієрархії класів і використання статичних компонентів класу.