

Інститут спеціального зв'язку та захисту інформації
Національного технічного університету України
"Київський політехнічний інститут ім. Ігоря Сікорського"

Спеціальна кафедра №5

ЛАБОРАТОРНА РОБОТА

з навчальної дисципліни
«Об'єктно-орієнтоване програмування»

Тема: Дослідження об'єктно-орієнтованих технологій
оброблення виняткових ситуацій в програмах

Виконав: курсант Жванський Роман

Перевірив: доцент Спеціальної кафедри №5
Куліков В.М.

Київ 2023

I. Дослідження засобів створення ієрархій класів в мові C++.

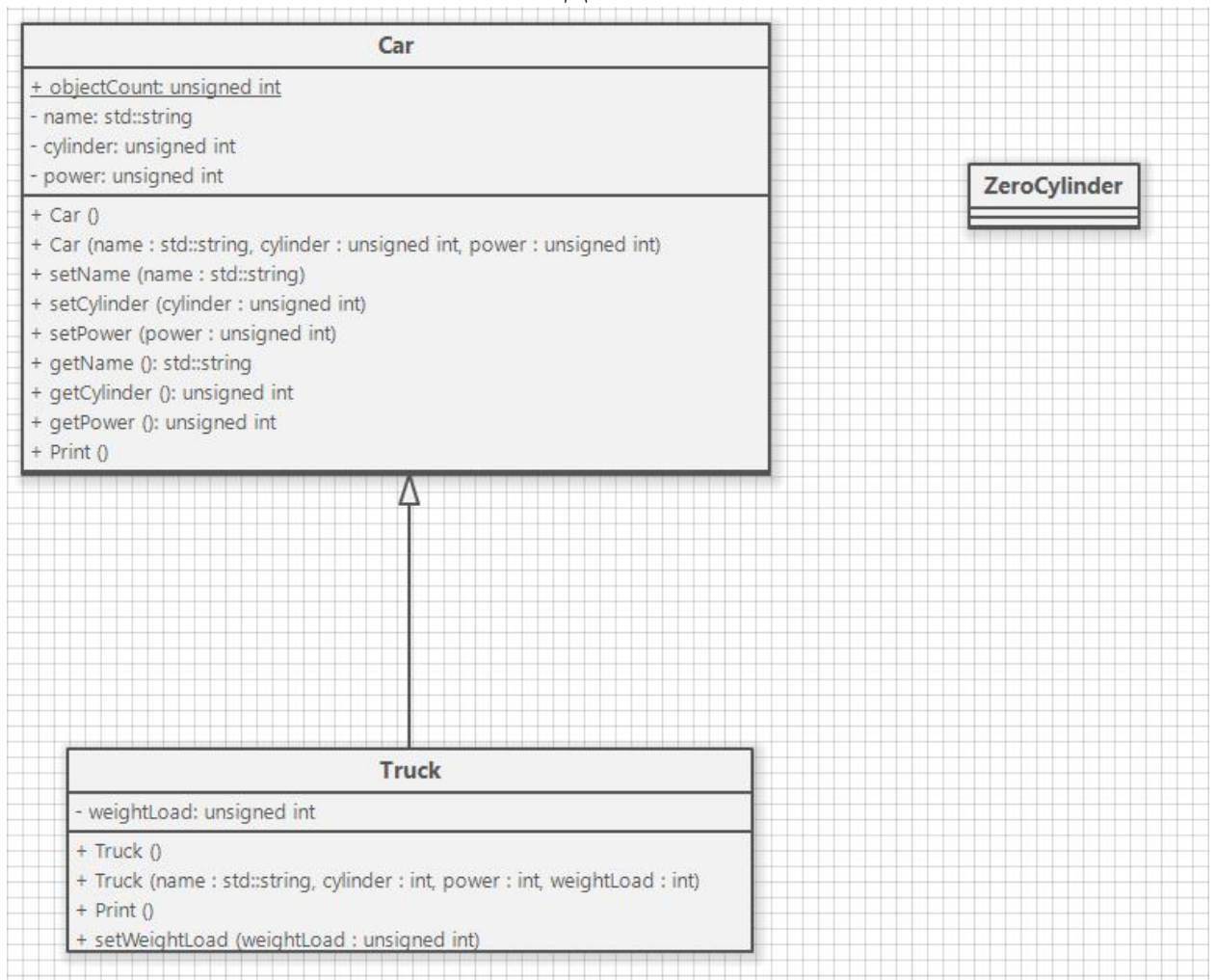
1.1 Завдання

Завдання 1 (ЛР № 2, 1 розділ)

Створити клас машина, який має марку (показчик на рядок), кількість циліндрів, потужність. Визначити конструктори, деструктор і функцію друку. Створити public-похідний клас - вантажівки, який має вантажопідйомність кузова. Визначити конструктори за замовчуванням і з різною кількістю параметрів, деструктори, функцію друку. Визначити функції перепризначення марки і вантажопідйомності.

1.2 Об'єктні моделі мовою UML (діаграми)

Завдання 1



1.3 Тексти програм(з коментарями, що пояснюють відповідність розроблених програм варіанту завдання)

Main.cpp

```
#include <iostream>
#include "Car.h"
#include <signal.h>

void signal_handler(int signal_num)
{
    std::cout << "Signal! WeigtLoad > 0\n"; // обробка виклику сигналу
}

int main(){
    signal(SIGINT, signal_handler); // створення сигналу
    CAR::Car c1;
    try {
        c1.setPower(0); // задаєм 0, щоб викликати стандартку помилку
    }
    catch (int a) {
        std::cerr << "catch(int). Error code: " << a << std::endl; //
        стандартна помилка код котрої зберігається в а
    }
    try {
        c1.setCylinder(0); // задаємо 0 для виклика користувацької помилки
    }
    catch (CAR::ZeroCylinder) {
        std::cerr << "catch(CARR::ZeroCylinder)" << std::endl; // користувацька
        помилка
    }
    CAR::Car c2, c3;
    CAR::Truck w;
    w.setWeightLoad(0);
    std::cout << "Num car: " << CAR::Car::objectCount << "\n"; // вивід кількості
    об'єктів
}
```

```

        c1.setName(nullptr); // виклик помилки assert
    return 0;
}

```

Car.h

```

#ifndef Car_h
#define Car_h

#include <iostream>
#include <string>

namespace CAR {
    class ZeroCylinder {}; // виключення користувача
    class Car {
    public:
        static unsigned int objectCount;

        Car();
        Car(std::string* name, unsigned int cylinder, unsigned int power);

        void setName(std::string* name);
        void setCylinder(unsigned int cylinder);
        void setPower(unsigned int power);
        std::string getName();
        unsigned int getCylinder();
        unsigned int getPower();
        virtual void Print();
    private:
        std::string* name;
        unsigned int cylinder;
        unsigned int power;
    };

    class Truck : public Car
    {
    public:

```

```
    Truck() : Car(new std::string("default"), 0, 0), weightLoad(0){};

    Truck(std::string* name, int cylinder, int power, int weightLoad) : Car(name,
power, cylinder), weightLoad(weightLoad) {};

    virtual void Print();

    void setWeightLoad(unsigned int weightLoad);

private:
    unsigned int weightLoad;
};

}

#endif /* Car_h */
```

Car.cpp

```
#include "Car.h"
#include <signal.h>
#include <cassert>

using namespace CAR;

unsigned int Car::objectCount = 0;

Car::Car() :
    name(new std::string("default")),
    cylinder(0),
    power(0) {
    ++objectCount;
};

Car::Car(std::string* name, unsigned int cylinder, unsigned int power) :
    name(name),
    cylinder(cylinder),
    power(power) {
    ++objectCount;
};

void Car::setName(std::string* name) {
    assert(name != nullptr); // перевірка name на пустоту за допомогою assert
    delete name;
    this->name = name;
}

void Car::setCylinder(unsigned int cylinder) {
    if (cylinder == 0) throw ZeroCylinder(); // перевірка циліндрів на 0 та виклик
    користувачького виключення, якщо помилка
    this->cylinder = cylinder;
}

void Car::setPower(unsigned int power) {
```

```

        if (power == 0) throw - 1; // перевірка потужності на 0 та виклик стандартного
        виключення з кодом помилки -1
        this->power = power;
    }

    std::string Car::getName() {
        return *name;
    }

    unsigned int Car::getCylinder() {
        return cylinder;
    }

    unsigned int Car::getPower() {
        return power;
    }

    void CAR::Car::Print()
    {
        std::cout << "Name : " << *name << "\n";
        std::cout << "Cyliders : " << cylinder << "\n";
        std::cout << "Power : " << power << " HP\n";
    }

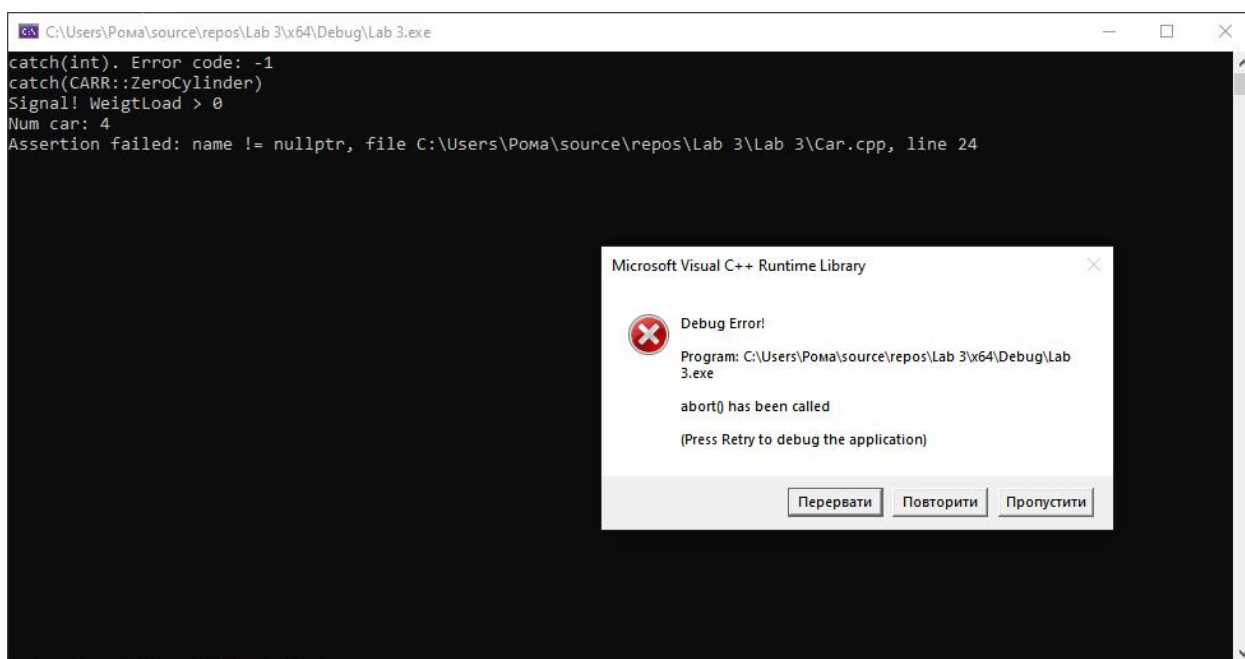
    void CAR::Truck::Print()
    {
        Car::Print();
        std::cout << "Weight load : " << weightLoad << " tons\n";
    }

    void CAR::Truck::setWeightLoad(unsigned int weightLoad)
    {
        if (weightLoad == 0)
        {
            raise(SIGINT); // викликаю сигнал, щоб повідомити про помилку
        }
        this->weightLoad = weightLoad;
    }

```


1.4 Висновки

Було створено програму, яка виконує всі поставлені задачі лабораторної роботи. Оголошення, визначення і головна програма в окремих файлах. Було використано простір імен, щоб уникнути конфліктів з іншими компонентами програми. Також, було застосовано механізми обробки помилок за допомогою засобів try, throw, catch та класів виключень. Для обробки помилок також було використано механізми підтверджень (assert) та сигналів (signal). Було передбачено лічильник кількості. Далі наведено приклад роботи програми :



The screenshot shows a Windows command prompt window titled "C:\Users\Роман\source\repos\Lab 3\x64\Debug\Lab 3.exe". The output text is as follows:

```
catch(int). Error code: -1
catch(CARR::ZeroCylinder)
Signal! WeightLoad > 0
Num car: 4
Assertion failed: name != nullptr, file C:\Users\Роман\source\repos\Lab 3\Lab 3\Car.cpp, line 24
```

Overlaid on the command prompt is a "Microsoft Visual C++ Runtime Library" error dialog box. It contains the following text:

Debug Error!
Program: C:\Users\Роман\source\repos\Lab 3\x64\Debug\Lab 3.exe
abort() has been called
(Press Retry to debug the application)

At the bottom of the dialog box are three buttons: "Перервати" (Break), "Повторити" (Retry), and "Пропустити" (Ignore).

Спочатку йде стандартна помилка, наступний рядок користувачка помилка, лічильний об'єктів і помилка assert, яка призводить до крашу програми. Програма працює вірно.