Множествено наследяване и виртуални класове

Множествено наследяване

Досега разглеждахме случаи, когато един клас наследява само един друг клас. ООП ни позволя даден клас да наследява повече от един клас. Това се случва по следният начин:

```
class A
{
    /*...*/
}
class B
{
    /*...*/
}
class C: protected A, public B
{
    /*...*/
}
```

Такова наследяване се нарича **множествено наследяване**.

Правилата за голяма четворка са същите както при единичното наследяване.

Задача 1

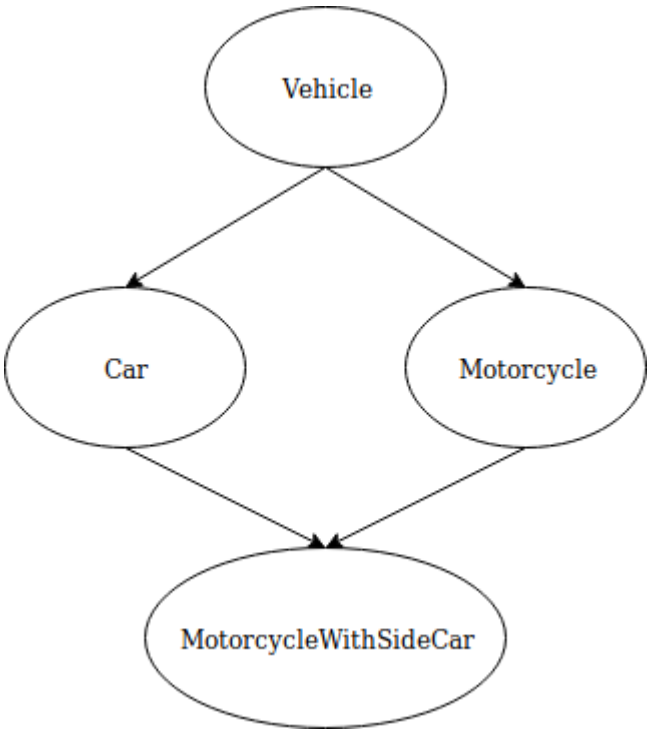
Да се напише клас `Property`, което ще описва "собственост". Класът притежава следните член-данни:

- Име на собственика
- Стойност на собствеността в лева
- Данък на собствеността в проценти

Нека класът `Car` да наследява и `Property`, като данъка да е 1% от стойността на колата, ако конските сили са < 110, и с 1% повече за всеки 100 к.н над 110 (пр. до 210hp е 2%, до 310hp е 3%, etc.)

Виртуални класове

Нека си представим следната ситуация - искаме да направим клас `MotorcycleWithSidecar` (мотор с кош) - той е и кола, и мотор (за целите на задачата) - тоест, `MotorcycleWithSidecar` трябва да наследява и `Car` и `Motorcycle`. Получава се следната схема.



Ако искаме да използваме някоя член-данна или метод на `Vehicle`, примерно, `horsepower` и `get_horsepower()`, наследената от кой клас ще бъде използвана ? (и `Car` и `Motorcycle` имат `horsepower`).

Решение за това е, класа Vehicle да бъде наследяван виртуално. Така класът `Vehicle` ще съществува точно веднъж, независимо колко пъти се наследява.

Особенности на виртуалните класове

- Необходимо е извикването на конструкторите на всички наследени класове, при всеки клас, който наследява класове - това в нашия случай означава, че ако имаме конструктор с параметри във `Vehicle`, той трябва да се извиква от `Car`, `Motorcycle` и `MotorcycleWithSidecar`
- Също така, виртуалните класове се инициализират преди всички други

Имплементация на virtual класове

Пред модификатора на видимост се слага `virtual` (може и след)

```
class Car: virtual public Vehicle, public Property
{
    private:
        int seats;
        char* fuel_type;

    public:
        Car();
        Car(const Car&);
        Car&operator=(const Car&);
        void set_seat(const int);
        void set_fuel_type(const char*);
        void print() const;
```

```
        const int get_seats() const;
};
```

```
class Motorcycle: virtual public Vehicle
{
    private:
        int luggage_capacity;
    public:
        Motorcycle();
        void set_luggage_capacity(const int new_luggage_capacity);
        const int get_luggage_capacity() const;
        void print() const;
};
```

```
class MotorcycleWithSidecar: public Car, public Motorcycle {
private:
    int wheels;
public:
    MotorcycleWithSidecar();

    void set_wheels(const int&);
    const int get_wheels() const;
};
```

Проблем възниква обаче, когато искаме да извикаме функцията `print()`. Решението на този проблем ще разгледаме в следващата тема.

Задача 2

Имплементирайте класа `MotorcylceWithSidecar`

Задача 3

Напишете следните класове със следните член-данни и подходящи методи:

Клас `Beverage` - code (цяло число), amount (дробно число)

Клас `Coffee` - origin (enum със следните стойност - Arabic, Brazilian, Indian, Colombian) - `Coffee` наследява `Beverage`

Клас `Milk` - type (enum със стойности - cow, almond, bull) - `Milk` наследява `Beverage`

Клас `CoffeeWithMilk` - price (дробно число) - наследява `Coffee` и `Milk`