8. Rozhraní/Interface

- u dědičnost jsme závislí na třídách
  - příklad Zvíře, Vozidlo, které mají/sdílejí funkce
- my ale nechceme být závislí na třídě, chceme být závislí na implementaci (funkci)

př1.

Třídy mohou reprezentovat lidé a my v naší restauraci nechceme být závislí na Pepovi jako kuchaři. My chceme být závislí na někom, kdo umí vařit svíčkovou. A je jedno jestli to je Pepa, Lucka, Libor nebo někdo jiný.

př2.

Obchod nechce být závislý na Monetě, KB, Paypalu on chce být závislý na transakci respektive platební metodě. Zadám částku a implementace od někud odečte peníze a zašle na účet obchodu.

- jmenná konvence
  - interface začíná vždy I
  - př. IMovable
    IFlyable
- vše co interface předepisuje, je vždy public
  - nepředepisuje privátní proměnný
- nemá implementaci
- nemá tělo
- zápis

```
public interface IVehicle {
    string Brand {get;}   - pokud je pouze get hodnota je pouze READ ONLY
    String Color {get; set;}   - hodnotu lze i měnit mimo třídu
    void Drive(int kilometers);
}
```

Použití

- stejně jako u dědičnosti jiných servisbouje
- auto implementuje IVehicle

```
internal class Car : IVehicle
{
}
```

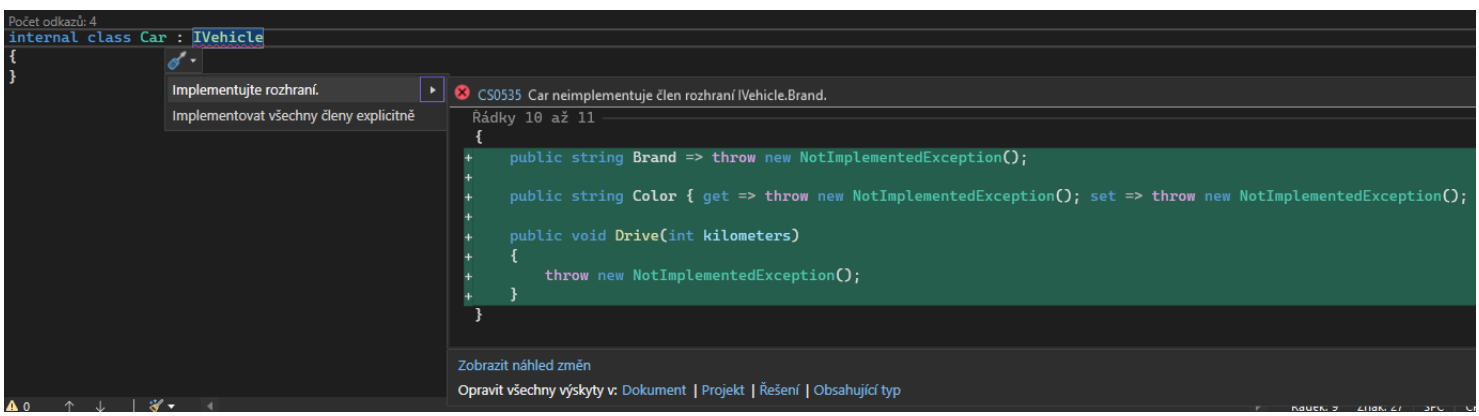
interface UKOL_Sprava_vozoveho_parku.IVehicle

CS0535: Car neimplementuje člen rozhraní IVehicle.Brand.

CS0535: Car neimplementuje člen rozhraní IVehicle.Color.

CS0535: Car neimplementuje člen rozhraní IVehicle.Drive(int).

- IDE nám samo řekne co nám chybí doimplementovat



- pomocí šroubováku nebo žárovky "Implementovat rozhraní"

```
internal class Car : IVehicle
{
    Počet odkazů: 1
    public string Brand => throw new NotImplementedException();

    Počet odkazů: 1
    public string Color { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

    Počet odkazů: 2
    public void Drive(int kilometers)
    {
        throw new NotImplementedException();
    }
}
```

– následně upravíme a máme hotovo

```
internal class Car : IVehicle
{
    Počet odkazů: 2
    public string Brand { get; }

    Počet odkazů: 2
    public string Color { get; set; }

    Počet odkazů: 2
    public Car(string brand, string color)
    {
        this.Brand = brand;
        this.Color = color;
    }

    Počet odkazů: 2
    public void Drive(int kilometers)
    {
        Console.WriteLine($"Drive with steering wheel {kilometers}km.");
    }
}
```

```
internal class Motorcycle : IVehicle
{
    Počet odkazů: 2
    public string Brand { get; }

    Počet odkazů: 2
    public string Color { get; set; }

    Počet odkazů: 2
    public Motorcycle(string brand, string color)
    {
        this.Brand = brand;
        this.Color = color;
    }

    Počet odkazů: 2
    public void Drive(int kilometers)
    {
        Console.WriteLine($"Drive with handlebars {kilometers}km.");
    }
}
```

– my nechceme být závislí na autě nebo motorce, ale na něčem, co umí ujet počet kilometrů v tomto případě na vozidle, respektive IVehicle. Je jedno jaké, ale odveze nás.