## Cars

Build a hierarchy of interfaces and classes:



## Define an Interface IPerson

Define an interface **IPerson** with properties for **Name** and **Age**. Define a class **Citizen** which implements **IPerson** and has a constructor which takes a **string** name and an **int** age.

## Multiple Implementation

Using the code from the previous task, define an interface **IIdentifiable** with a **string** property **Id** and an interface **IBirthable** with a **string** property **Birthdate** and implement them in the **Citizen** class. Rewrite the Citizen constructor to accept the new parameters.

## Ferrari

Model an application which contains a **class Ferrari** and an **interface**. Your task is simple, you have a **car - Ferrari**, its model is **"****488-Spider"** and it has a **driver**. Your Ferrari should have functionality to **use brakes** and **push the gas pedal**. When the **brakes** are pushed down **print "****Brakes!"**, and when the **gas pedal** is pushed down - **"****Go!"**. As you may have guessed this functionality is typical for all cars, so you should **implement an interface** to describe it.

Your task is to **create a Ferrari** and **set the driver's name** to the passed one in the input. After that, print the info.

## Telephony

You have a business - **manufacturing cell phones**. But you have no software developers, so you call some friends of yours and ask them to help you create a cell phone software. They have already agreed and you started working on the project. The project consists of one main **model – a Smartphone**. Each of your smartphones should have functionalities of **calling other phones** and **browsing in the world wide web.**

These friends of yours though are very busy, so you decide to write the code on your own. Here is the mandatory assignment:

You should have a **model** - **Smartphone** and two separate functionalities which your smartphone has - to **call other phones** and to **browse in the world wide web**. You should end up with **one class** and **two interfaces**.

## Explicit Interfaces

Create 2 interfaces **IResident** and **IPerson**. **IResident** should have a **name**, **country** and a method **GetName()**. **IPerson** should have a **name**, an **age** and a method **GetName()**. Create a class Citizen which implements both **IResident** and **IPerson**, explicitly declare that IResident’s **GetName()** method should return “Mr/Ms/Mrs ” before the name while IPerson’s **GetName()** method should return just the name. You will receive lines of citizen information from the console until the command “**End**” is received. Each will be in the format **“<name> <country> <age>**” for each line create the corresponding citizen and print his **IPerson’s GetName()** and his **IResitent’s GetName().**

## Collection Hierarchy

Create 3 different string collections – **AddCollection**, **AddRemoveCollection** and **MyList**.

The **AddCollection** should have:

* Only a single method **Add** which adds an item to the **end** of the collection.

The **AddRemoveCollection** should have:

* An **Add** method– which adds an item to the **start** of the collection.
* A **Remove** method which removes the **last** item in the collection.

The **MyList** collection should have:

* An **Add** method which adds an item to the **start** of the collection.
* A **Remove** method which removes the **first** element in the collection.
* A **Used** property which displays the number of elements currently in the collection.

Create interfaces which define the collections functionality, think how to model the relations between interfaces to reuse code. Add an extra bit of functionality to the methods in the custom collections, **add** methods should return the index in which the item was added, **remove** methods should return the item that was removed.

## Sudoku

Create console application that solves this Sudoku

