Assoziationen

Beziehungen zwischen Objekten

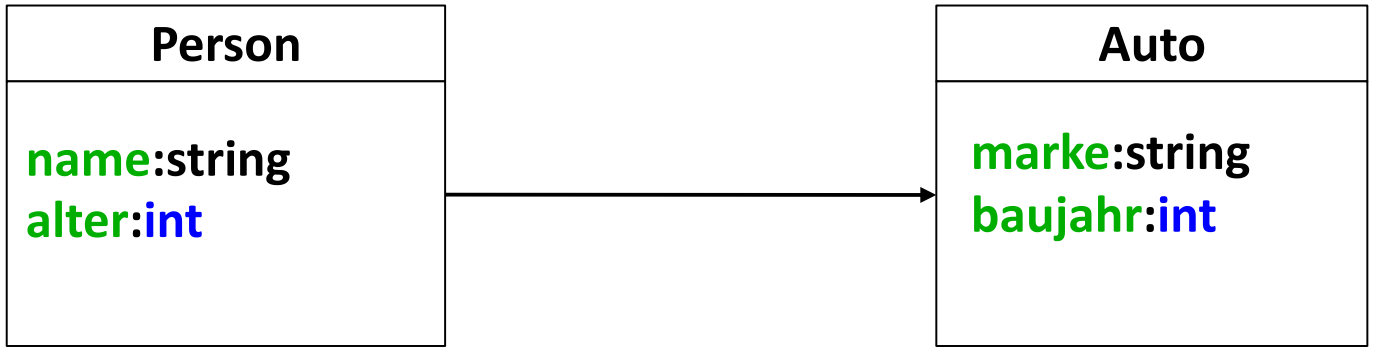
Identifizieren Sie die Objekte im Bild

- Ein Auto-Objekt
- Ein Person-Objekt
- Vier Reifen-Objekte?
- Ein Lenkrad-Objekt
- Zwei Scheinwerfer-Objekte
- Zwei Tür-Objekte?
- ...

Offensichtlich stehen die verschiedenen Objekte zu einander in **Beziehung**!

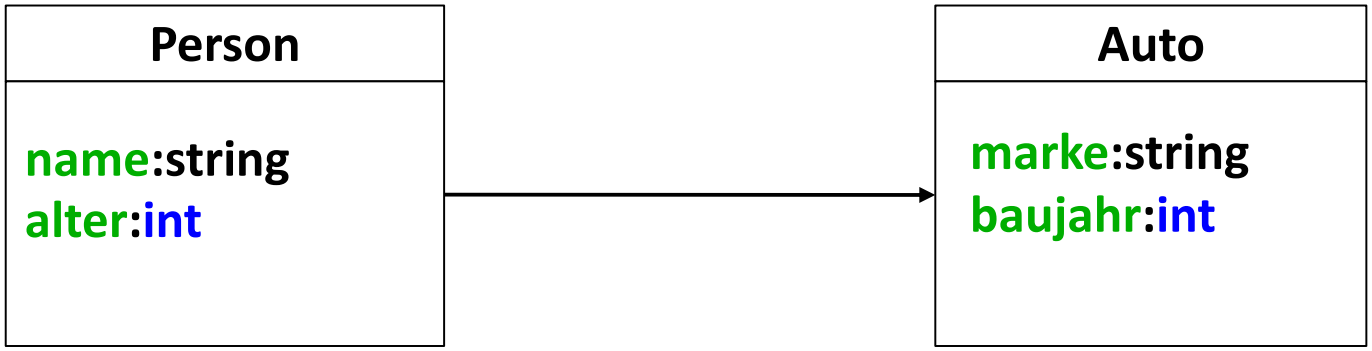
Eine Person hat ein Auto

UML:



Eine Person hat ein Auto

UML:



C#:

```
class Person{
string name;
int alter;
Auto auto;
}
```

```
class Auto{
string marke;
int baujahr;
}
```

In C# werden hat-Beziehungen durch **Referenzvariablen** realisiert. Im obigen Beispiel kann von Auto nicht zu Person navigiert werden, da nur die Klasse **Person** eine Instanzvariable von **Auto** besitzt aber nicht umgekehrt!

Objekte miteinander verbinden

```
class Person{
string name;
int alter;
Auto auto;
}
```

```
class Auto{
string marke;
int baujahr;
}
```

Hier wird die **Beziehung** zwischen Person und Auto gesetzt.

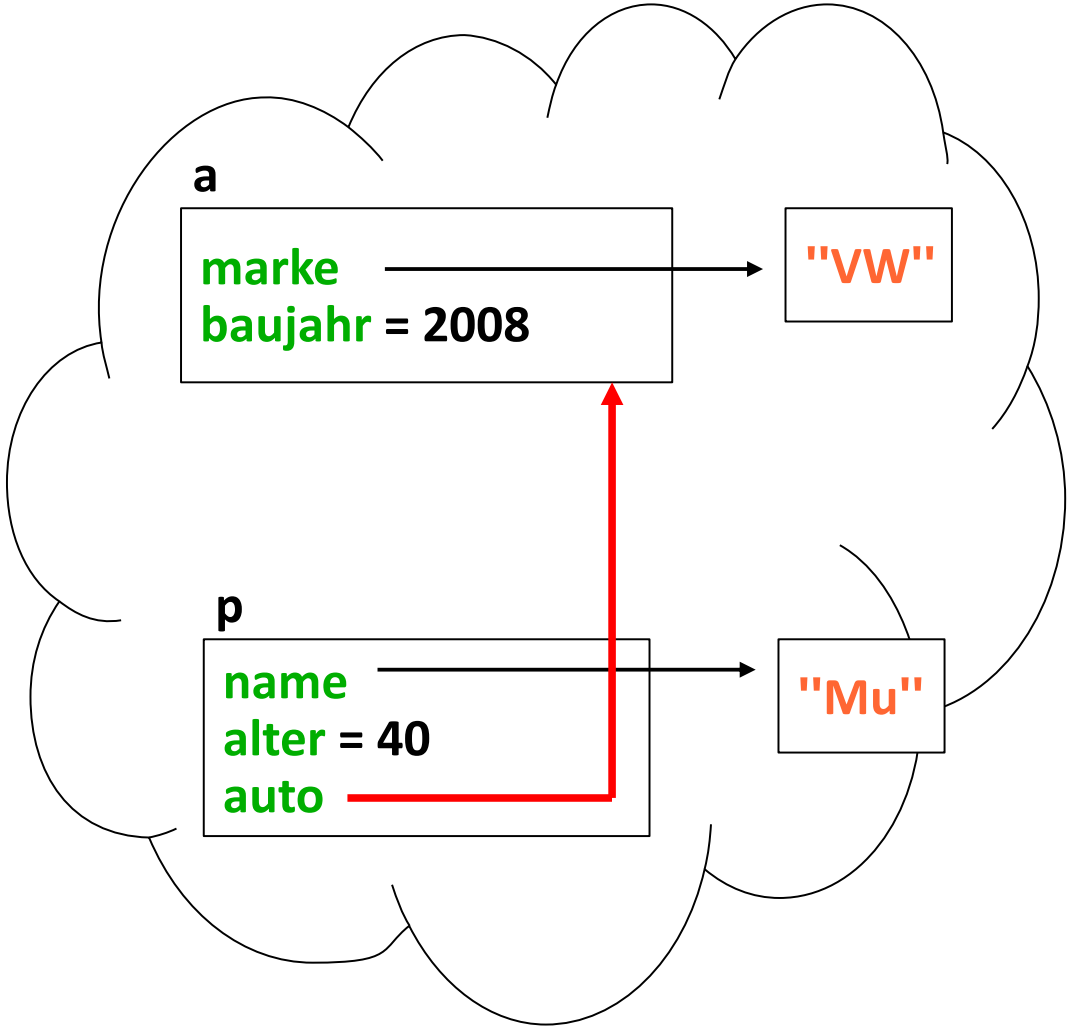
```
Auto a = new Auto();
  a.marke = "VW";
  a.baujahr = 2014;

Person p = new Person();
      p.name = "Monika";
      p.alter = 21;
      p.auto = a;
```

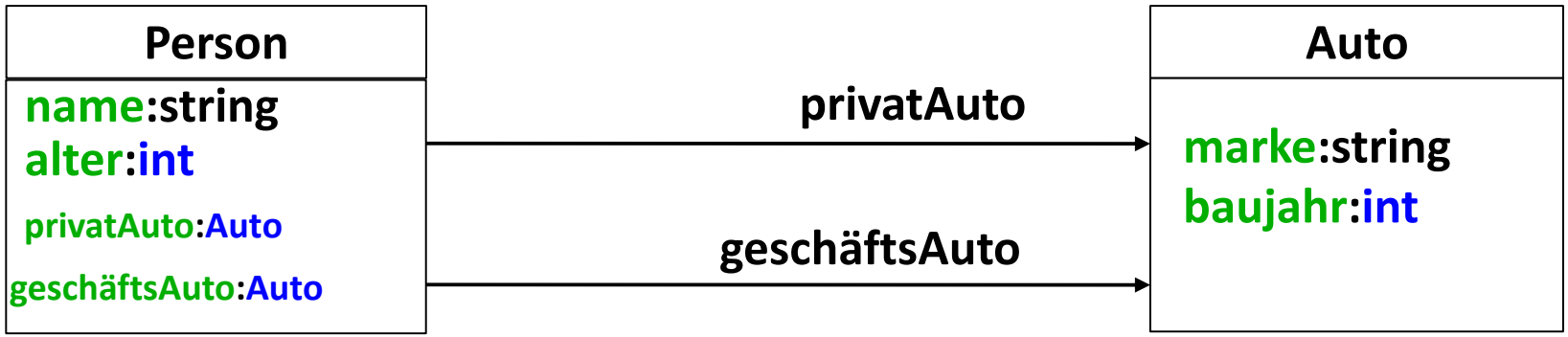
Beziehungen sind Referenzen

```
Auto a = new Auto();
a.marke = "VW";
a.baujahr = 2008;
```

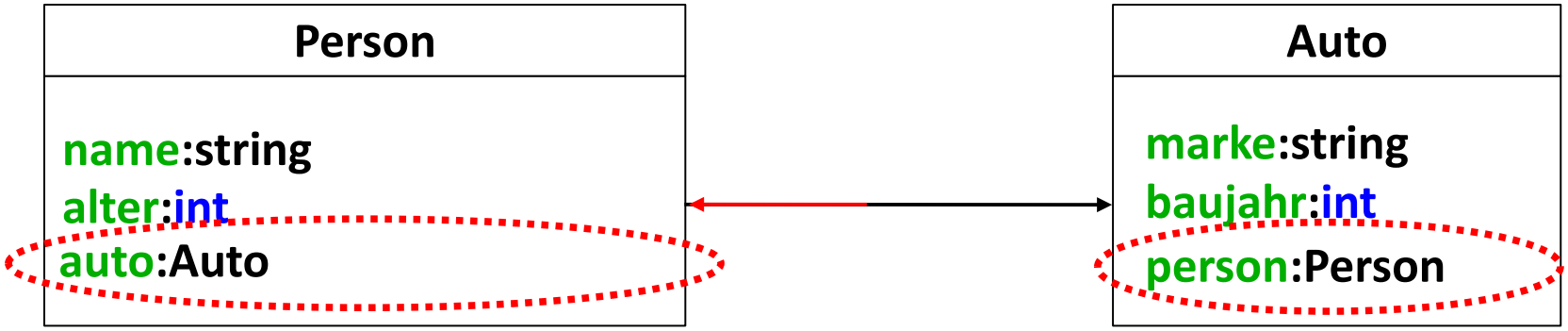
```
Person p = new Person();
p.name = "Mu";
p.alter = 40;
p.auto = a;
```



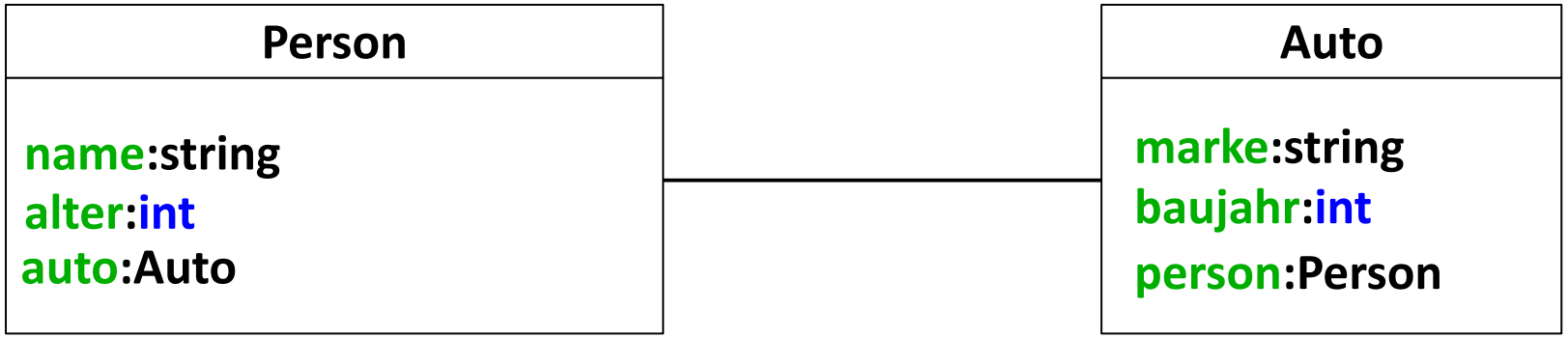
Mehrere Beziehungen zu einer Klasse modellieren



Bidirektionale Beziehung

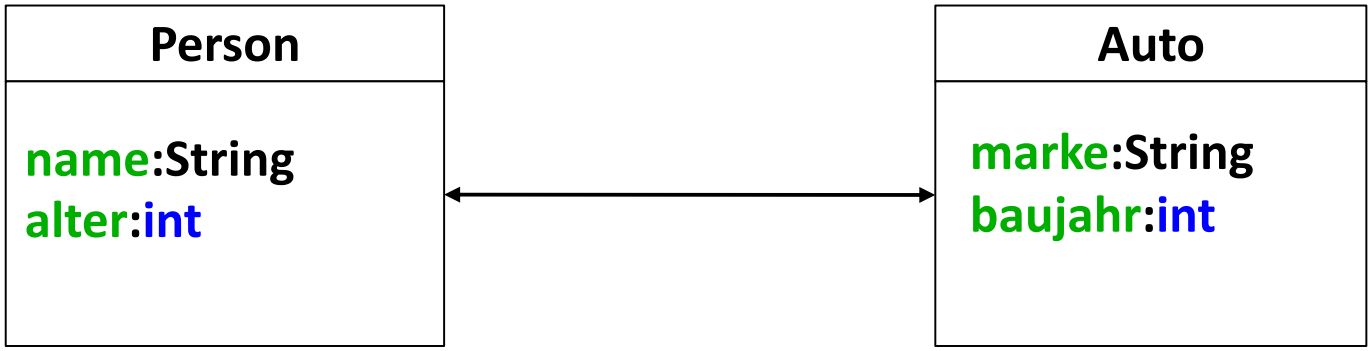


… oder man lässt die Pfeile ganz weg ...



Implementierung: Birektionale Beziehung
Eine Person hat ein Auto, ein Auto hat eine Person

UML:



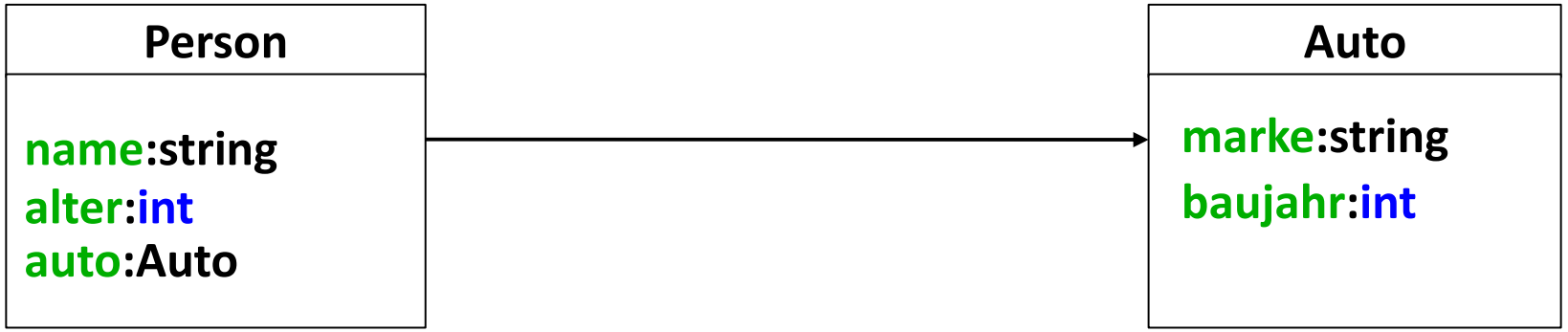
C#:

```
class Person{
String name;
int alter;
Auto auto;
}
```

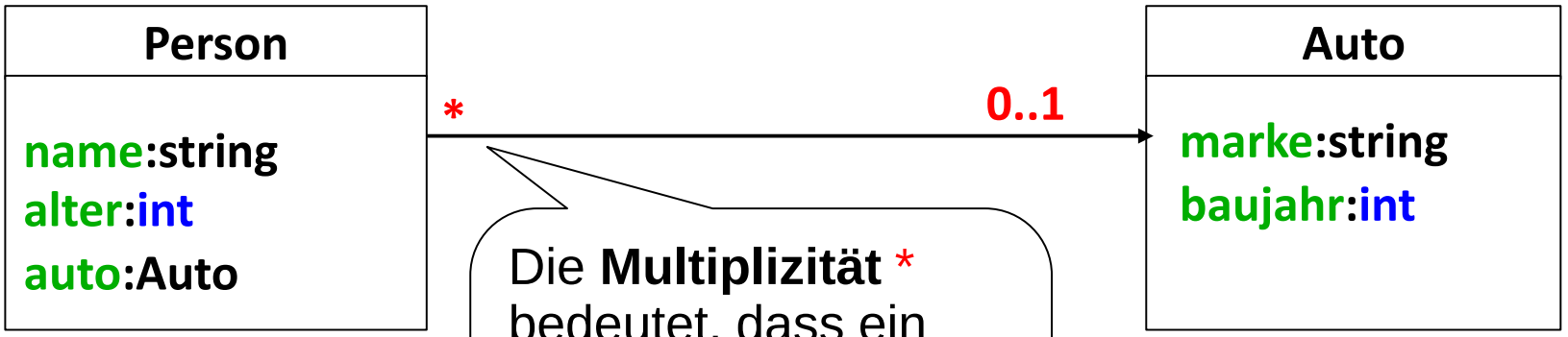
```
class Auto{
String marke;
int baujahr;
Person person;
}
```

Multiplizität

Multiplizität



Das obige Diagramm ohne Angabe von Multiplizitäten ist gleichbedeutend mit dem folgenden Diagramm:



Die **Multiplizität** * bedeutet, dass ein Auto **beliebig vielen** (0 bis unendlich) Personen gehören kann.

Aufgabe 1

Entwerfen Sie ein UML-Diagramm für das folgende Szenario:
Ein Kraftfahrzeugfahrer hat eine Personalnummer, einen Vor- und Nachnamen, eine Adresse sowie einen Führerschein. Der Führerschein hat ein Ausstellungsdatum sowie eine Klasse (z. B. Klasse A, Klasse B usw..). Jede Adresse besteht aus einer Straße, einer Hausnummer, einem Postleitzahl und einem Ort.
Beachten Sie die Datenkapselung!

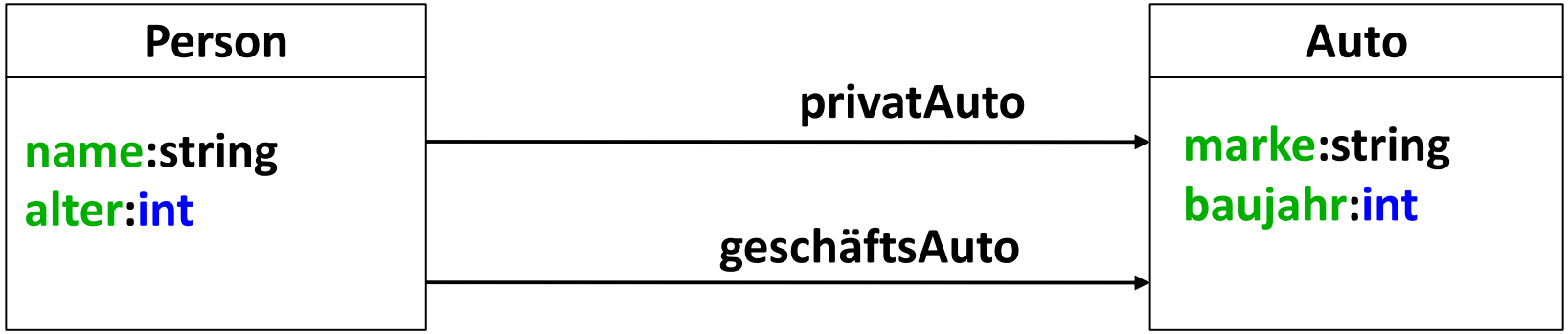
Überführen Sie das Diagramm in C#.

Erstellen Sie die folgenden Objekte und geben Sie deren Daten aus:

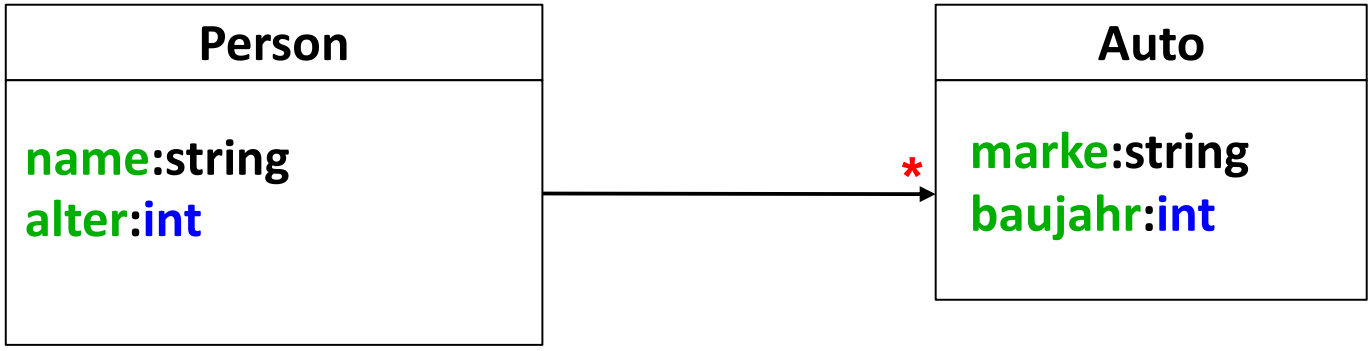
- Fahrer Michael Anton hat die Personalnummer 1. Sein Führerschein der Klasse A ist am 13. April 2013 ausgestellt. Michael Anton wohnt am Leopoldring 3, in 79199 Kirchzarten.
- Fahrerin Michaela Antonella hat die Personalnummer 2. Ihr Führerschein der Klasse B ist am 13.01.2009 ausgestellt. Sie wohnt in der Bahnhofstr. 7, in 79108 Kenzingen.

Aufgabe 2

Überführen Sie das folgende Diagramm in C# und erstellen Sie eine Person mit zwei Autos.

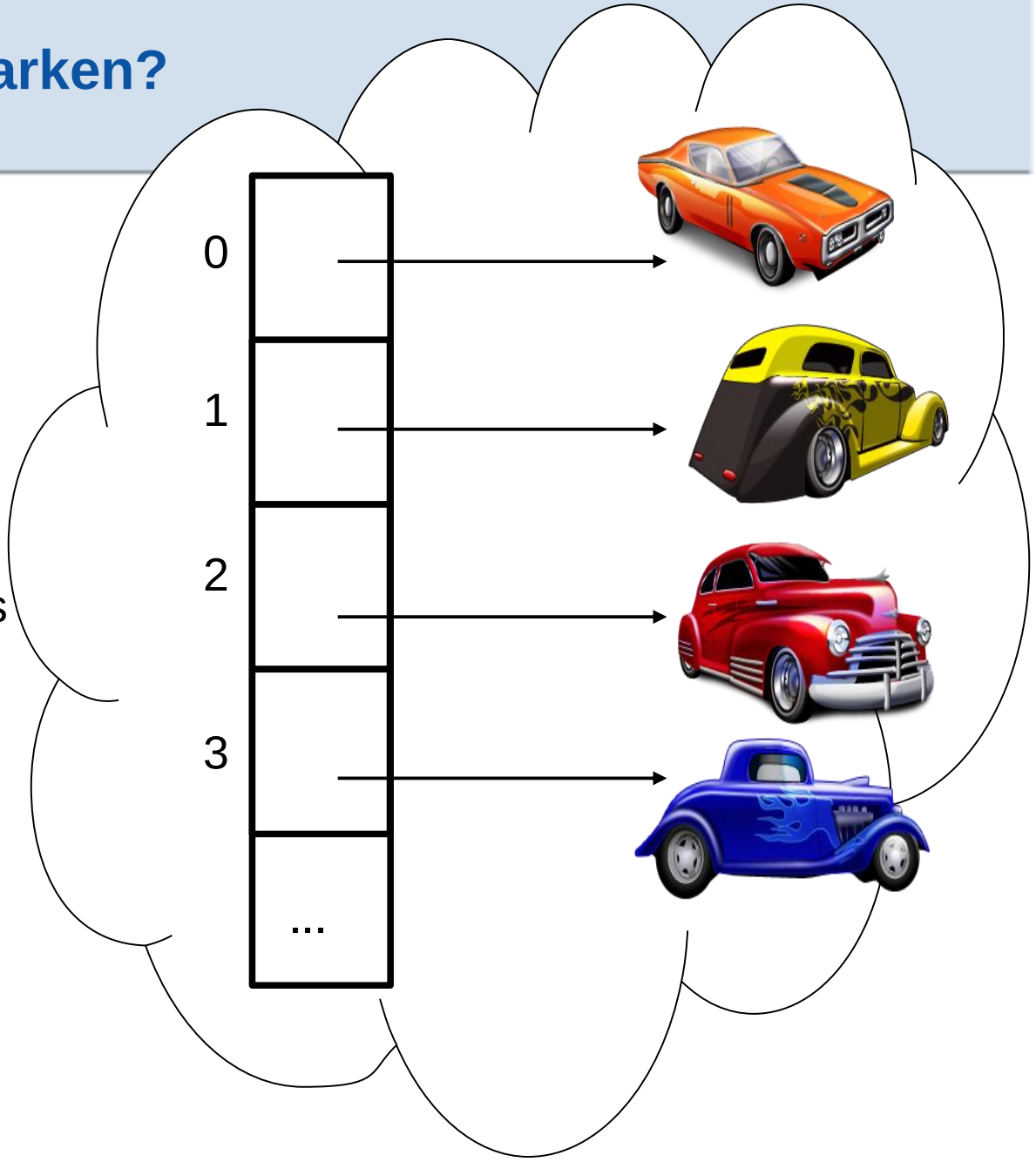


Eine Person kann auch beliebig viele Autos besitzen

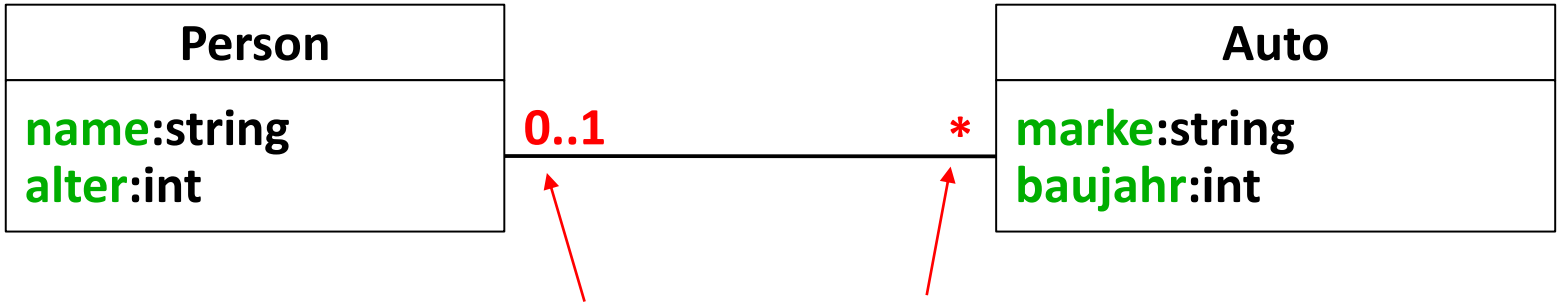


Wo alle Auto parken?

In **C#** jedenfalls kann er die Autos im **Speicher** parken wo er will. Er muss sich nur die Adressen der Autos z. B. in einer **Liste** merken...



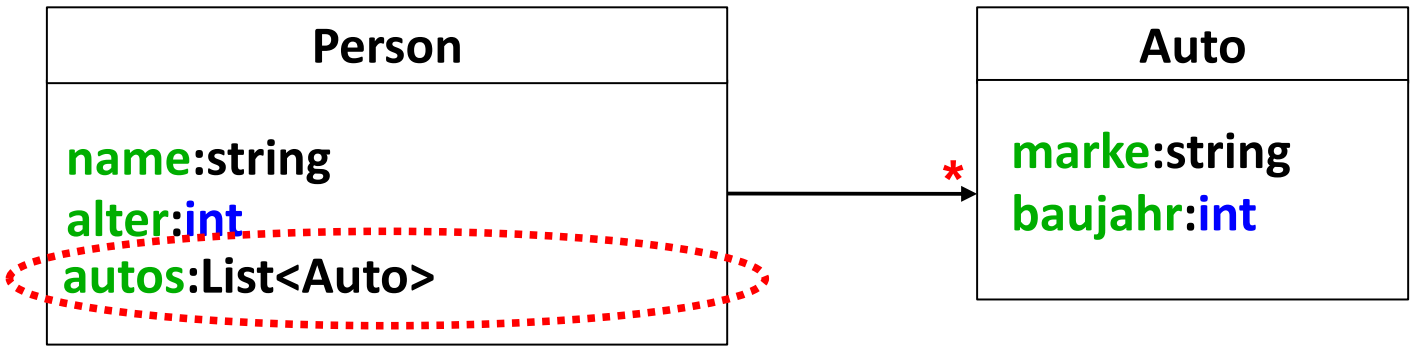
Eine Person kann mehrere Autos besitzen



Jedes Auto gehört höchstens einer Person

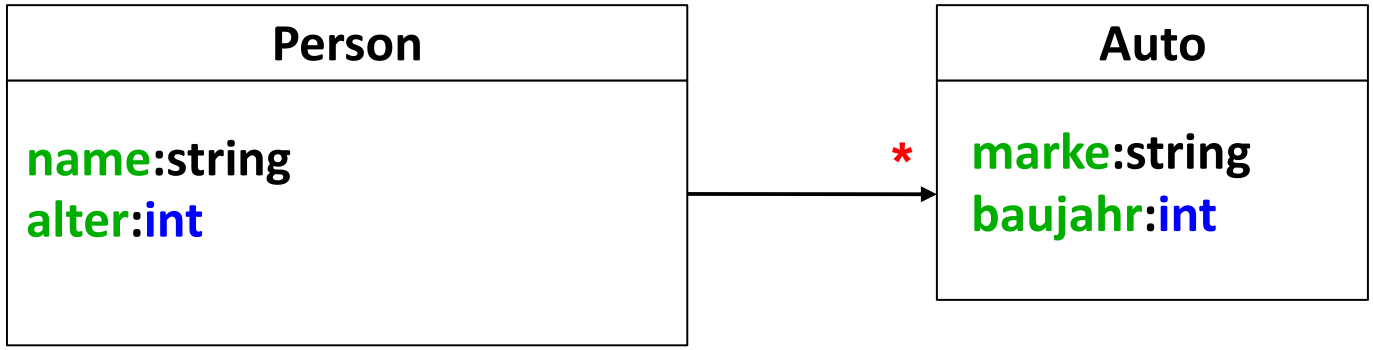
Jeder Person hat mehrere oder kein Auto.

Darstellung in UML



Implementierung in C#

UML:



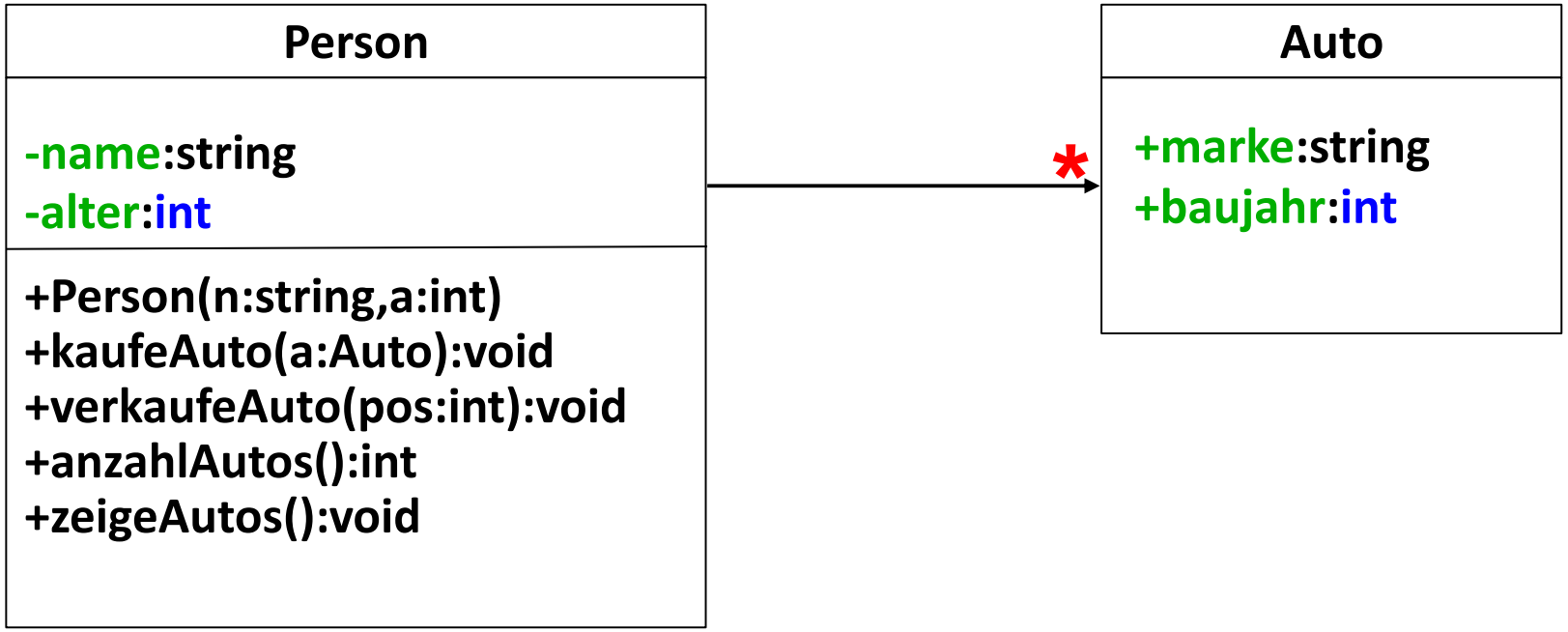
C#:

```
class Person{
string name;
int alter;
List<Auto> autos;
}
```

```
class Auto{
string marke;
int baujahr;
}
```

Aufgabe 3

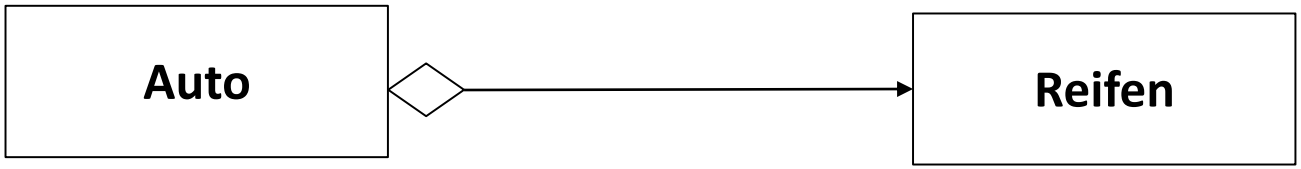
Überführen Sie das folgende Diagramm in C# und erstellen Sie eine Person mit ein paar Autos. Nutzen Sie die Methoden



Aggregation, Komposition

Aggregation: „Teil von“

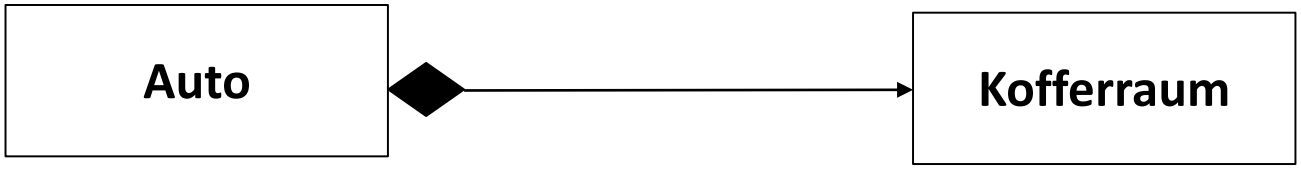
- Eine **Komposition** ist ebenfalls eine **„ist Teil von“-Beziehung**.



- Jedes Objekt der Klasse **Reifen** ist nur Komponente eines einzigen Objektes der Klasse **Auto** (**Aggregat**-Klasse).
- Die Aggregation verlangt im Gegensatz zu Komposition **nicht**, dass nach dem Entfernen einer Zeitschrift auch deren Artikel gelöscht werden.
- In der Implementierung unterscheidet sich eine Aggregation nicht von der Implementierung einer normalen Assoziation. Aggregation wird eher zum besseren Verständnis des Modell verwendet.

Komposition: „Teil-von“

- Eine **Komposition** ist ebenfalls eine **„ist Teil von“-Beziehung**.



- Jedes Objekt der Klasse **Kofferraum** ist Komponente eines einzigen Objektes der Klasse **Auto**.
- **Die Komposition verlangt**: wird ein Auto gelöscht, so muss auch der Kofferraum gelöscht werden! In der Praxis ignorieren wir dieses Detail…

Beispiele für Kompositionen