

Lab: CarRentalCompany

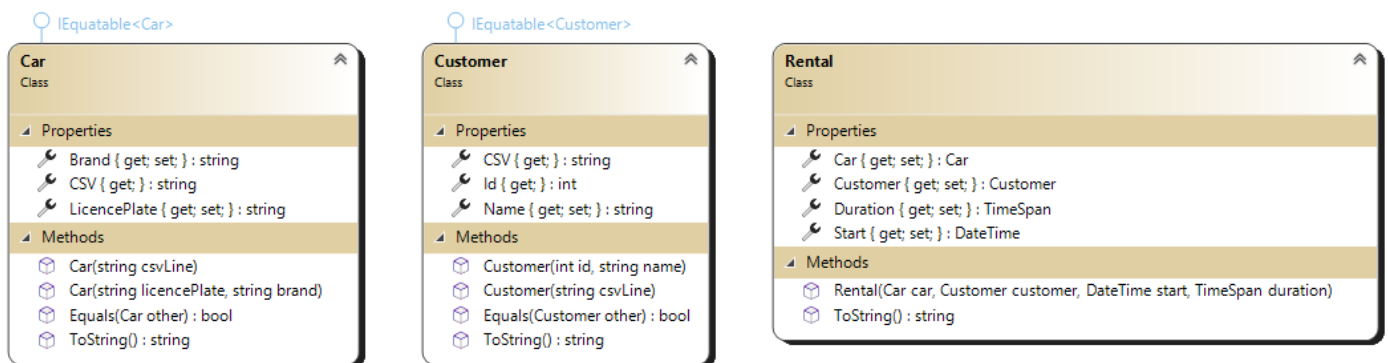
In dem Beispiel geht es um einen Autovermieter, bei dem die Kunden (**Customer**) Autos (**Car**) ausborgen können. Welches Auto, von welchem Kunden, wann und für wie lange ausgeborgt wird, implementiert die Klasse **Rental**.

Verwende eine **Klassenbibliothek** für die Implementierung der Logikklassen, die in den Aufgaben 1 – 4 beschreiben sind.

Aufgabe 1: Basisklassen

Implementiere die Klassen **Car**, **Customer** und **Rental** dem Klassendiagramm entsprechend. Beachte dabei, dass **Car** und **Customer** das Interface **IEquatable** implementieren sollen. Dabei gilt folgendes:

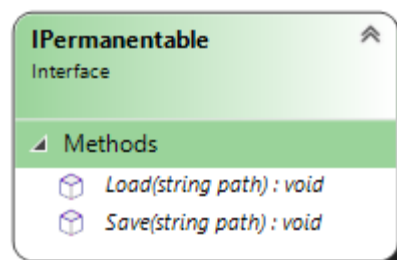
- **Car**: zwei Autos sind dann gleich, wenn sie das gleiche Kennzeichen haben.
- **Customer**: zwei Kunden sind dann gleich, wenn sie die gleichen IDs und den gleichen Namen haben.



Die Objekte der Klasse **Car** und **Customer** sollen in eine **.csv**-Datei gespeichert und daraus gelesen werden können. Dazu dient einerseits die Property **CSV** (liefert die csv-Darstellung des Objekts als string) und der Konstruktor, der eine csv-Zeile als Parameter entgegennimmt.

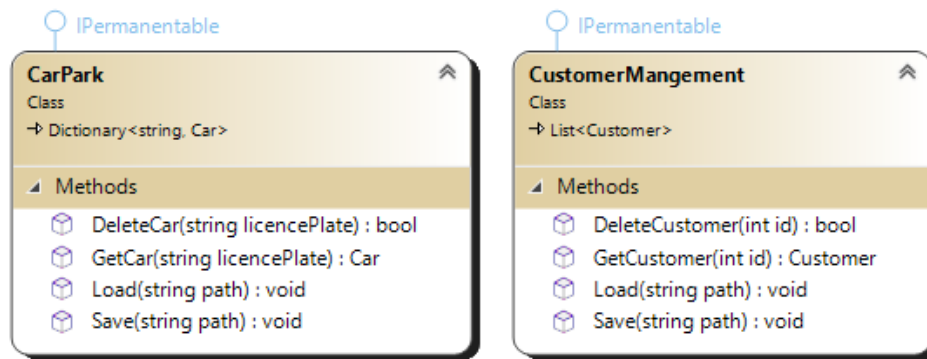
Aufgabe 2: IPermanentable

Definiere das Interface **IPermanentable** dem folgenden Diagramm entsprechend. Klassen, die das Interface implementieren, sollen dafür geeignet sein, ihre Daten permanent, d.h. auf der Festplatte speichern zu können.



Aufgabe 3: Die Klassen mit den Collections (List/Dictionary)

Die Autovermietung hat naturgemäß einen Fuhrpark mit mehreren Autos (**CarPark**) und viele Kunden, die in einer eigenen Klasse (**CustomerManagement**) verwaltet werden.

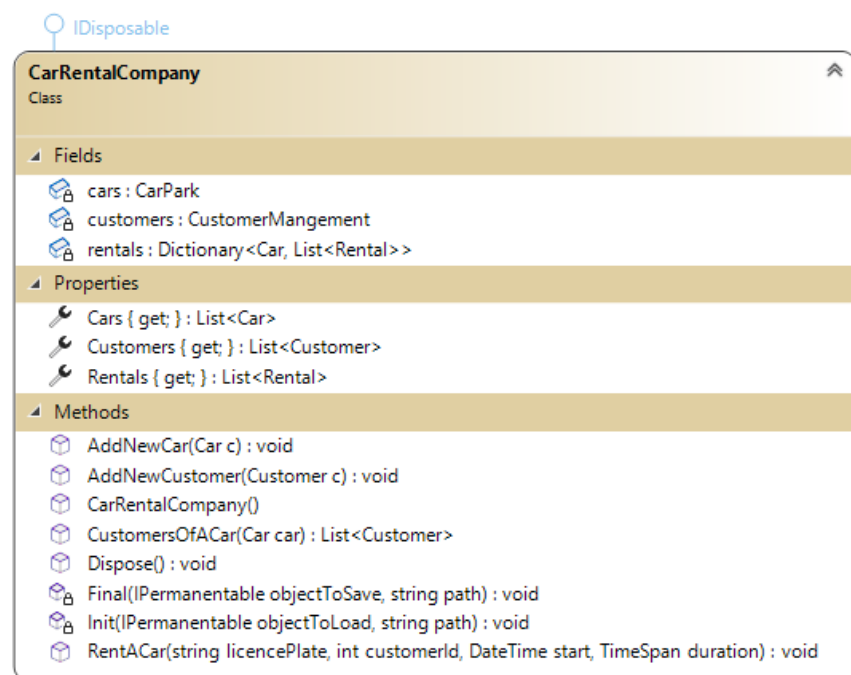


Die Klasse **CarPark** erbt von **Dictionary<string, Car>** (siehe Diagramm), wobei der Key die LicencePlate ist. Die Klasse **CustomerMangement** erbt von **List<Customer>**. Die Klassen implementieren zusätzlich auch das Interface **IPermanentable**.

- **Save** speichert die Objekte aus dem Dictionary bzw. Liste in eine (siehe path) .csv Datei. Dabei wird die CSV-Property der Klasse verwendet.
- **Load** lädt die Daten aus der dem Parameter entsprechenden csv Datei in das Dictionary bzw. List. Dabei wird jener Konstruktor der Klasse verwendet, der eine csv-Zeile als Parameter entgegennimmt.
- **Delete** löscht ein Objekt, **Get...** liefert ein gesuchtes Objekt. Verwende für die Suche in **CustomerMangement** LINQ.

Aufgabe 4: Die Klassen für die Anwendung

Die **Kunden**, die **Autos** und ihre **Vermietungen** werden zusammen in einer eigenen Klasse verwaltet:



Felder:

Definiere je ein (private!) Feld für den Fuhrpark (**CarPark**) und für das Kundenmanagement (**CustomerMangement**). Die **Vermietungen (rentals)** werden ebenfalls in einem Feld vom Typ

Dictionary<Car, List<Rental>> gespeichert. Da man sich öfters dafür interessiert, an welche Kunden und wann ein Auto vermietet wurde, ist der Key das Car-Objekt und Value ist eine Liste von Rental-Objekten. So kann man leicht und schnell zu einem Auto alle gebuchten Vermietungen finden.

Properties:

- **Cars** liefert die Liste der Autos aus dem Fuhrpark (Feld cars).
- **Customers** liefert die Liste der Kunden (aus dem Feld customers).
- **Rentals** ermittelt alle Vermietungen aus dem Dictionary rentals als Liste. (verwende dazu LINQ SelectMany!)

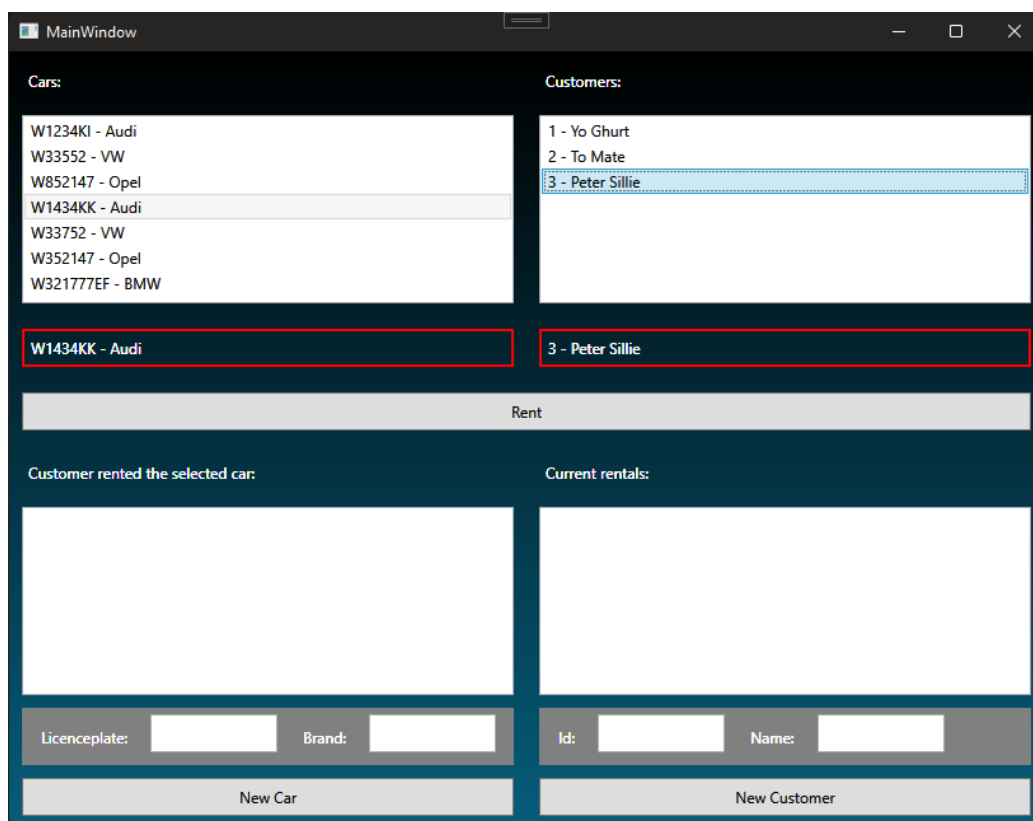
Methoden:

Beginne mit den beiden Methoden **Init** und **Final**. Diese nehmen ein Objekt vom Typ **IPermanentable** und einen **Pfad** als Parameter entgegen. Init ruft die **Load**, Final die **Save** Methode für das Objekt (Parameter) auf.

Um zu Beginn alle Daten (Kunden und Autos) aus den Files zu laden, braucht man nur noch aus dem Konstruktor die **Init** für diese beiden Felder (cars und customers) aufrufen.

Aufgabe 5: Testen der bisher implementierten Logikklassen

Füge zu deiner Solution ein neues WPF-Projekt hinzu, um die grafische Oberfläche zu implementieren.



- Definiere ein **CarRentalCompany** Feld im **MainWindow**, um die dort implementierte Anwendungslogik verwenden zu können.

- Wenn du die **CarRentalCompany** Klasse wie oben beschreiben implementiert hast, musst du in **MainWindow** nur noch die **ItemsSources** für die beiden oberen **ListBoxes** richtig setzen, um die Daten aus der Datei zu sehen.
- Implementiere für beiden **ListBoxes** das **SelectionChanged**-Event, um das jeweils ausgewählte Objekt im Label darunter anzeigen zu können.

Aufgabe 6: Neue Objekte speichern

- Implementiere in **CarRentalCompany** die beiden Methoden **AddNewCar** und **AddNewCustomer**. Diese fügen das neue Objekt (Parameter!) in das entsprechende Feld ein.
- Die Klasse **CarRentalCompany** soll auch das Interface **IDisposable** (siehe Folien!) implementieren. In der Methode **Dispose** wird die Methode **Final** für die beiden Felder **cars** und **customers** aufgerufen. Diese speichert die Daten dieser Collections.
- Implementiere in **MainWindow** die beiden Buttons **New Car** und **New Customer**, indem du die soeben implementierte **AddNewCar** bzw. **AddNewCustomer** Methoden aufrufst.
- Damit die Daten beim Beenden des Programms auf keinem Fall verloren gehen können, definieren wir zu **Closing**-Event des **MainWindows** eine Methode, wo wir **Dispose** von unserem **CarRentalCompany** Objekt aufrufen. Dies speichert die Daten (**cars** und **customers**) in die **.csv**-Dateien.

Aufgabe 7: Autos vermieten

Wenn das alles funktioniert, können wir damit beginnen, unsere Autos an die Kunden zu vermieten.

CarRentalCompany: Folgende Methoden müssen noch implementiert werden:

- **RentACar** – bekommt die **licencePlate**, die **customerId**, **Startdatum** und **Dauer** als Parameter. (Achtung: halte dich an diese Signatur!!)

Verwende **GetCar** (aus **CarPark**) und **GetCustomer** (aus **CustomerManagement**) um die gesuchten **Car**- und **Customer**-Objekte zu finden. Erstelle damit ein **Rental**-Objekt und führe es in dein **Dictionary** ein. Beachte, dass es hier zwei Möglichkeiten gibt:

1. Es gibt **noch keinen Key** mit diesem **Car**-Objekt: – es solle ein neuer Eintrag in das **Dictionary** mit diesem **Car**-Objekt als **Key** gemacht werden.
2. Es gibt **schon ein Key** mit diesem **Car**-Objekt: – es soll in die Liste der **Rental** Objekte, die bei diesem **Car**-Objekt als **Value** gespeichert wurde, das neue **Rental**-Objekt hinzugefügt werden.

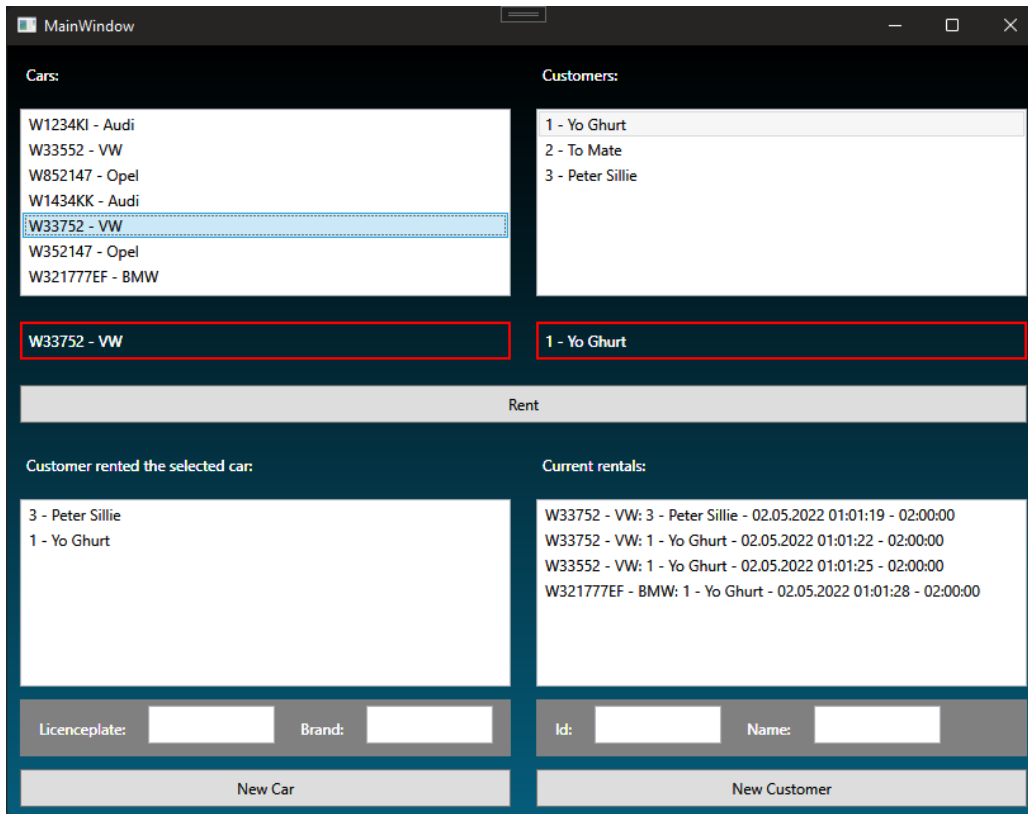
Hinweis:

Beachte, dass **TryGetValue**, bzw. **ContainsKey** für das **Dictionary** mit einem **Car**-Objekt als **Key** nur dann funktioniert, wenn die Klasse **Car** das Interface **IEquatable** implementiert.

- **CustomersOfACar** – liefert die Liste der **Rental**-Objekte, die beim **Key car** gespeichert wurde, bzw. eine leere Liste, falls der **Key** nicht gefunden wurde.

MainWindow:

Implementiere den **Button Rent**. Dabei wird das ausgewählte Auto an den ausgewählten Kunden vermietet. Die ListBox links im mittleren Bereich zeigt die Kunden, die dieses Auto bereits gemietet haben an (Methode **CustomersOfACar**). Die rechte ListBox im mittleren Bereich zeigt alle Vermietungen an (Property **Rentals**).



The screenshot shows a Windows application titled "MainWindow" with a dark-themed interface. It is divided into several sections:

- Cars:** A list box containing car details: W1234KI - Audi, W33552 - VW, W852147 - Opel, W1434KK - Audi, **W33752 - VW** (highlighted with a blue dashed border), W352147 - Opel, and W321777EF - BMW.
- Customers:** A list box containing customer names: 1 - Yo Ghurt, 2 - To Mate, and 3 - Peter Sillie.
- Selection:** Below the lists, two red-bordered boxes show the selected items: "W33752 - VW" and "1 - Yo Ghurt".
- Rent:** A large grey button labeled "Rent" is centered below the selection boxes.
- Customer rented the selected car:** A list box showing the selected customer: 3 - Peter Sillie and 1 - Yo Ghurt.
- Current rentals:** A list box showing a log of rentals: W33752 - VW: 3 - Peter Sillie - 02.05.2022 01:01:19 - 02:00:00, W33752 - VW: 1 - Yo Ghurt - 02.05.2022 01:01:22 - 02:00:00, W33552 - VW: 1 - Yo Ghurt - 02.05.2022 01:01:25 - 02:00:00, and W321777EF - BMW: 1 - Yo Ghurt - 02.05.2022 01:01:28 - 02:00:00.
- Input Fields:** At the bottom, there are two rows of input fields. The first row has "Licenceplate:" and "Brand:" labels with corresponding text boxes. The second row has "Id:" and "Name:" labels with corresponding text boxes.
- Buttons:** At the very bottom, there are two buttons: "New Car" and "New Customer".

Gutes Gelingen!