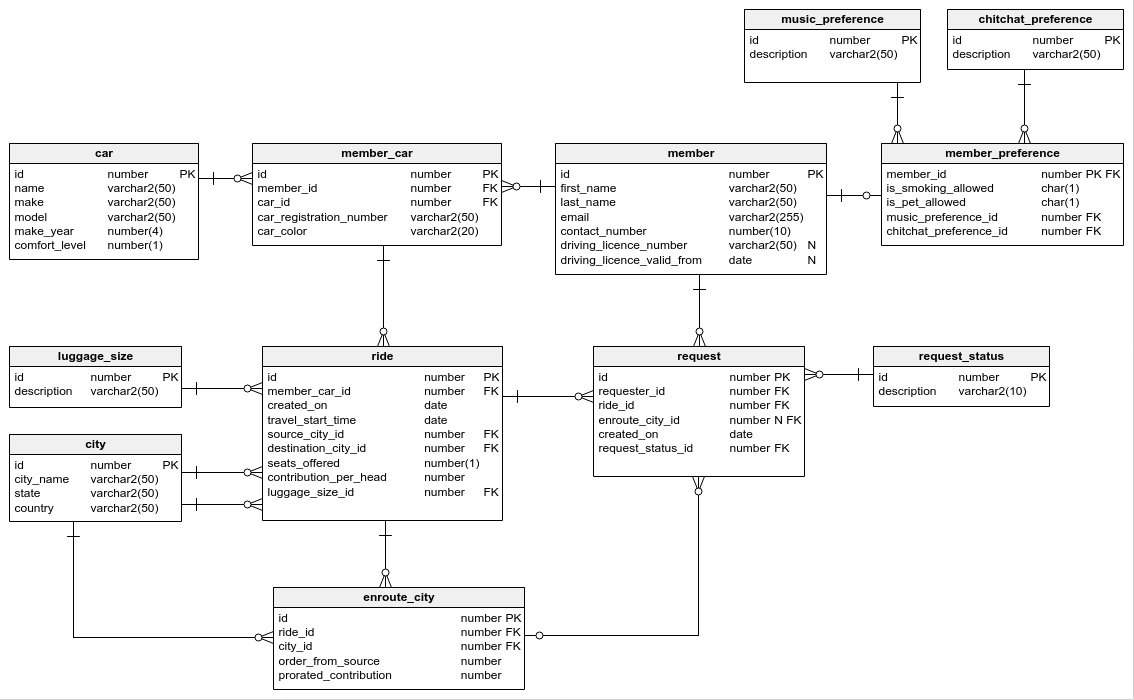
Page of

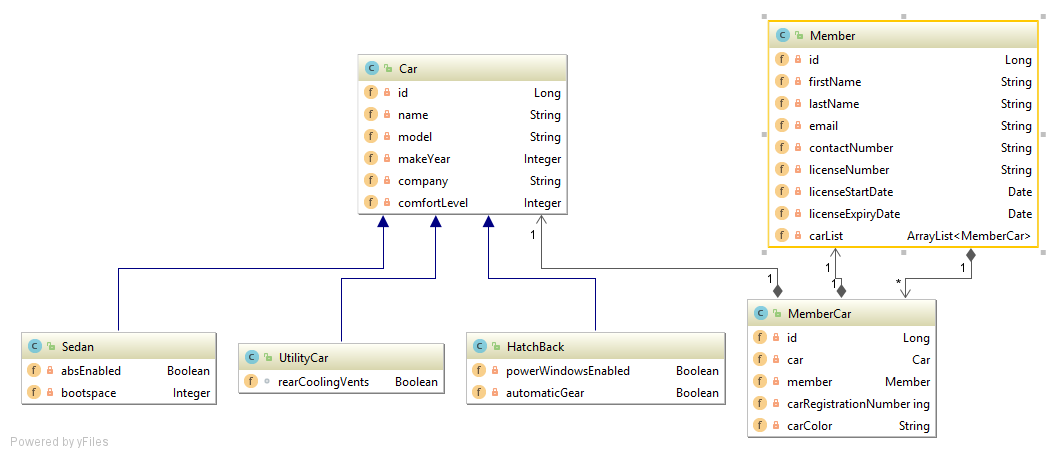
p { margin-bottom: 0.25cm; direction: ltr; line-height: 120%; text-align: left; }

Car Pooling System - Requirement 1

You are a very active member of a Nature Club in your organization. In one of the meetings, it was discussed to build a car pooling system to help cut down the pollution. Being very active and tech savvy, you wish to contribute towards the development of system. One of the members being an architect has understood the requirement and would be sharing you with smaller requirements.  
  
The complete data model designed by the architect is given below.



Since you are an object oriented specialist and you want build a part of the system initially, you choose to create the basic classes like cars, members their relationships and few small functionalities. The overall model that you will be implementing today is given below.



**Requirement 1:**

The users of the system are going to be general public who own cars. They are classified as members in our system.

1. Create a Member Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field name** | **Type** |
| id | Long |
| firstName | String |
| lastName | String |
| email | String |
| contactNumber | String |
| license Number | String |
| licenseStartDate | Date (java.util) |
| licenseExpiryDate | Date (java.util) |

1. Mark all the attributes as private
2. Create / Generate appropriate Getters & Setters
3. Add a default constructor and a parameterized constructor to take in all attributes.
4. When the “member” object is printed, it should display the following details:

Member: firstname, lastname

Member contact details: contactNumber, email

1. Two members are considered same if they have same email and contactNumber. Implement the logic in the appropriate function. (Case – Insensitive)

The Input to your program would be details of two members, You need to display their details as given in "e" and use the function to compare the two members and display if the members are same or unique. Refer to Sample IO.

**Hint:**

The template code hints at which methods are to be implemented to complete the requirement.

**Sample Input and Output 1:**  
**[All text in bold corresponds to input and the rest corresponds to output.]**

Member1 :  
id:  
**1**  
first name:  
**Arun**  
last name:  
**Kumar**  
email:  
**arun123@gmail.com**  
contact number:  
**9878767655**  
license number:  
**TN38QW1232343**  
license start date:  
**12-12-2010**  
license expiry date:  
**13-12-2020**  
Member2 :  
id:  
**2**  
first name:  
**Mohamed**  
last name:  
**Safiq**  
email:  
**safiq1243@gmail.com**  
contact number:  
**9667826601**  
license number:  
**TN33VA1238743**  
license start date:  
**01-05-2013**  
license expiry date:  
**01-04-2125**  
  
Member 1  
Name: Arun , Kumar  
Member contact details: 9878767655 , arun123@gmail.com  
  
Member 2  
Name: Mohamed , Safiq  
Member contact details: 9667826601 , safiq1243@gmail.com  
Member 1 and Member 2 are different  
  
**Sample Input and Output 2:**  
**[All text in bold corresponds to input and the rest corresponds to output.]**  
  
Member1 :  
id:  
**1**  
first name:  
**Sam**  
last name:  
**Nath**  
email:  
**Sam123@gmail.com**  
contact number:  
**9456738498**  
license number:  
**TN45AS123456**  
license start date:  
**12-12-2010**  
license expiry date:  
**13-12-2021**  
Member2 :  
id:  
**2**  
first name:  
**Swamy**  
last name:  
**Nathan**  
email:  
**Sam123@gmail.com**  
contact number:  
**9456738498**  
license number:  
**TN54DF321456**  
license start date:  
**01-05-2012**  
license expiry date:  
**01-05-2123**  
  
Member 1  
Name: Sam , Nath  
Member contact details: 9456738498 , Sam123@gmail.com  
  
Member 2  
Name: Swamy , Nathan  
Member contact details: 9456738498 , Sam123@gmail.com  
Member 1 is same as Member 2