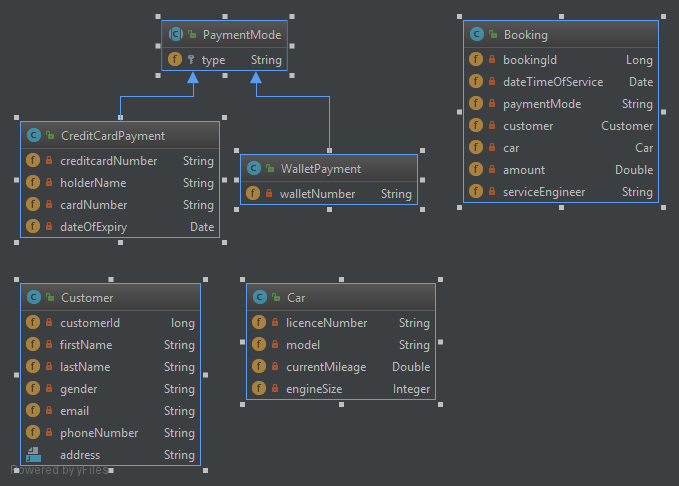
**Car-Service Management System - Requirement 1**

Your friend, a mechanical engineer is very passionate about cars and wants to be an entrepreneur. He decides to start his own car service center and with his knowledge about cars has the capability to service all different brands of cars. You are also very keen to help him out in any means possible.   
One fine day, your friend approaches you to help him setup and automate the process of tracking various customers and their service feedbacks. You decide to quickly build a small system to solve the problem.   
  
Based on the class diagram given below, you start to build a prototype of the application.



**Requirement 1:**

Let’s start off by creating a customer class based on the below mentioned specifications.

1. Create a Customer Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_customerId | long |
| \_firstName | string |
| \_lastName | string |
| \_gender | string |
| \_email | string |
| \_phoneNumber | string |
| \_address | string |

1. Mark all the attributes as private
2. Create / Generate appropriate Properties.
3. Add a default constructor and a parameterized constructor to take in all attributes in the given order: (long \_customerId, string \_firstName, string \_lastName, string \_gender, string \_email, string \_phoneNumber, string \_address)
4. When the “customer” object is printed, it should display the following details: [Override the ToString method]

**Print format:  
                            Customer: \_firstname, \_lastname**  
**Contact details: \_phoneNumber, \_email, \_address.**

1. Two customers are considered same if they have the same name  (both firstname and lastname), email and phone number. Implement the logic in the appropriate function. (Case – Insensitive) [Override the Equals method]

The Input to your program would be details of two customers, you need to display their details as given in "requirement e(refer above)" and compare the two customers and display if the customers are same or unique.

If the two customers are same then print "**Customer 1 is same as Customer 2**", else print "Customer 1 and Customer 2 are different"

**Sample INPUT & OUTPUT 1:**

Customer1 :  
customer id:  
**1**  
first name:  
**Arun**  
last name:  
**Kumar**  
gender:  
**Male**  
email:  
**arun123@gmail.com**  
phone number:  
**9897969594**  
address:  
**Coimbatore North**  
Customer2 :  
customer id:  
**123**  
first name:  
**Arun**  
last name:  
**Kumar**  
gender:  
**Male**  
email:  
**arun123@gmail.com**  
phone number:  
**9897969594**  
address:  
**Coimbatore North**  
  
Customer 1  
Customer:Arun,Kumar  
Contact details:9897969594,arun123@gmail.com,Coimbatore North  
  
Customer 2  
Customer:Arun,Kumar  
Contact details:9897969594,arun123@gmail.com,Coimbatore North  
Customer 1 is same as Customer 2

**Sample Input and Output 2:**

Customer1 :

customer id:

**1**

first name:

**Vijay**

last name:

**Kumar**

gender:

**Male**

email:

**Vijayajit@gmail.com**

phone number:

**9876541234**

address:

**North Chennai**

Customer2 :

customer id:

**23**

first name:

**Karmega**

last name:

**Kulazhi**

gender:

**Female**

email:

**Karmegam@gmail.com**

phone number:

**7785674563**

address:

**Kurangani, Theni, Tamilnadu**

Customer 1

Customer:Vijay,Kumar

Contact details:9876541234,Vijayajit@gmail.com,North Chennai

Customer 2

Customer:Karmega,Kulazhi

Contact details:7785674563,Karmegam@gmail.com,Kurangani, Theni, Tamilnadu

Customer 1 and Customer 2 are different

**Car-Service Management System - Requirement 2**

**Requirement 2:**

1. Create a Car Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_licenceNumber | string |
| \_model | string |
| \_currentMileage | double |
| \_engineSize | int |

1. Mark all the attributes as private & Create appropriate Properties.

1. Add a default constructor and a parameterized constructor to take in all attributes in the given order (string \_licenceNumber, string \_model, double \_currentMileage, int \_engineSize)

1. Add a static **FindCar** method with \_licenseNumber and carList as input and returns the Car if the car object is found or null if the car object is not found.

1. Add a static **FindCarList** method in Car class which will take a \_model and carList as input and return a list of cars for the given \_model or null if no cars found. For these conditions, please refer the content to be printed in Sample IO.  
      Override the ToString method to print the car details :  
        
   **Print format:**  
   **Licence Number: \_licenceNumber**  
   **Model: \_model**

**Sample Input & Output 1:**  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**1**  
Licence Number:  
**MH1420110062821**  
Model:  
**Verna**  
Current Mileage:  
**24.35**  
Engine Size:  
**1461**  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**1**  
Licence Number:  
**MH1420110062823**  
Model:  
**Swift**  
Current Mileage:  
**17.35**  
Engine Size:  
**1231**  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**1**  
Licence Number:  
**MH1420110045821**  
Model:  
**Verna**  
Current Mileage:  
**14.35**  
Engine Size:  
**1231**  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**2**  
Licence Number  
**MH1420110062821**  
Licence Number:MH1420110062821  
Model:Verna  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**3**  
Model  
**Verna**  
Licence Number:MH1420110062821  
Model:Verna  
Licence Number:MH1420110045821  
Model:Verna  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**2**  
Licence Number  
**MH1420110062823**  
Licence Number:MH1420110062823  
Model:Swift  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**3**  
Model  
**Figo**  
Car Figo not found  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**4**  
  
**Sample Input and output 2:**  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**1**  
Licence Number:  
**TN1220151247856**  
Model:  
**Verna**  
Current Mileage:  
**22**  
Engine Size:  
**1230**  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**2**  
Licence Number  
**TN1220151247415**  
Licence Number not present  
Menu:  
1) Add a Car  
2) Find a Car  
3) Find CarList  
4) Exit  
**4**

**Car-Service Management System** **- Requirement 3**

**Requirement 3**

There are always typical human entry errors that need to be validated so that the data being saved in the system is valid and can be used for later processing. Simple Rules that needs to be taken care:

Create a class named as **Program**, which contains following static methods.

a) Create a method **ValidateLicenceNumber**(string \_licenceNumber) which takes license number(string)  and returns boolean based on the following rules.

A licence number is valid if all the below rules are true.

* 2 chars-state name(Upper-case) - 2 characters A to Z)
* 2 numbers-branch code(integers) - valid range between 10-50
* 4 numbers-licence issued year - valid range between 2005 to 2016
* 7 numbers -profile id - Any digit should not contain 0

 eg:**MH1420116662821**is a valid licence number.

b) Write a method isExperiencedDriver(String licenceNumber) which takes a license number(String) and returns a boolean.  
This function checks for experience of a driver with respect to licence Number. The year of the licence issue starts from 5 th chacter (yyyy determines the year).  
If year of license issued is equal to or greater than 5 years from current year then return true ,else return false. Assume today’s date as 28-11-2017. If the method returns true, then the driver is termed as experienced. If its less than 5 years, then print as Not Experienced Driver.

For Experienced Driver print “Experienced Driver”, else print “Not Experienced Driver”.

Menu:

1) Validate licence Number  
2) Check Driver Experience

**Sample Input and Output 1:**

Enter license number:

**MH1420110062821**

Menu:

1) Validate licence Number

2) Check Driver Experience

Enter choice:

**1**

License number is not valid

**Sample Input and Output 2:**

Enter license number:

**tn482018547585**

Menu:

1) Validate licence Number

2) Check Driver Experience

Enter choice:

**1**

License number is not valid

**Sample Input and Output 3:**

Enter license number:

**TN5920151122556**

Menu:

1) Validate licence Number

2) Check Driver Experience

Enter choice:

**1**

License number is not valid

**Sample Input and Output 4:**

Enter license number:

**KL2620044451236**

Menu:

1) Validate licence Number

2) Check Driver Experience

Enter choice:

**1**

License number is not valid

**Sample Input and Output 5:**

Enter license number:

**TN4520124482563**

Menu:

1) Validate licence Number

2) Check Driver Experience

Enter choice:

**1**

License number is valid

**Sample Input and Output 6:**

Enter license number:

**KA1220112235647**

Menu:

1) Validate licence Number

2) Check Driver Experience

Enter choice:

**2**

Experienced Driver

**Sample Input and Output 7:**

Enter license number:

**TN5420154485263**

Menu:

1) Validate licence Number

2) Check Driver Experience

Enter choice:

**2**

Not Experienced Driver

**Car-Service Management System** **- Requirement 4**

**Requirement 4:**

1. Create a **Booking**Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_bookingId | long |
| \_dateTimeOfService | DateTime |
| \_paymentMode | string |
| \_customerIns | Customer |
| \_carIns | Car |
| \_amount | double |
| \_serviceEngineer | string |

        Create / generate appropriate Properties and constructors.  
        Create Classes **Car**and **Customer**as given in the previous requirement.While reading the input, only \_customerId and car \_licenseNumber is provided in the input. You need to create \_customerIns and \_carIns objects and assign only these values. Other details about customer and car can be safely ignored. (Already given in template code)

1. Create an abstract class **PaymentMode**with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_type | string |

Include a single argument constructor as follows,  PaymentMode(string \_type)

c) Create a class **CreditCardPayment**which extends PaymentMode class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_creditcardNumber | string |
| \_holderName | string |
| \_cardNumber | string |
| \_dateOfExpiry | DateTime |

Include a 5-argument constructor with parameters CreditCardPayment(string \_creditcardNumber, string \_holderName, string \_cardNumber, DateTime \_dateOfExpiry, string \_type)  
  
d. Create a class **WalletPayment**which extends PaymentMode class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_walletNumber | string |

Include a 2-argument constructor with parameters WalletPayment(string \_walletNumber, string \_type)  
  
e. Mark the access modifiers appropriately, and create constructors. Include appropriate properties for all the classes.  
  
f.  Create an **abstract**method **MakePayment(Booking booking)** in PaymentMode class which takes a booking object as parameter and returns the payment in Rupees as double. Calculate the total amount with additional cost or discount as per the payment mode and return the value. Override the MakePayment method in CreditCardPayment and WalletPayment classes. Round off the final amount to zero decimal values.

|  |  |
| --- | --- |
| **PaymentMode** | **Cost(Rs)** |
| CreditCardPayment | 2% additional cost |
| WalletPayment | 5% discount |

**Sample Input & Output 1:**

bookingId

**1**

dateTimeOfService

**12-04-2016**

paymentMode

**creditcardpayment**

customer id

**1**

licence number

**MH1420110062821**

amount

**1000**

service engineer

**subash**

creditcard number

**1546 2563 5896 5879**

holder name

**krishna**

card number

**123**

date of expiry

**12-04-2018**

Cost is Rs 1020

**Sample Input & Output 2:**

booking id

**2**

date time of service

**11-04-2016**

payment mode

**walletpayment**

customer id

**1**

licence number

**MH1420110062821**

amount

**3000**

service engineer

**allwin**

wallet number

**128975**

Cost is Rs 2850

**Car-Service Management System** **- Requirement 5**

**Requirement 5:**

a) Create a **Customer**Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_customerId | long |
| \_firstName | string |
| \_lastName | string |
| \_gender | string |
| \_email | string |
| \_phoneNumber | string |
| \_address | string |

b) Mark all the attributes as private

c) Create / Generate appropriate Properties.

d) Add a default constructor and a parameterized constructor to take the following arguments: Customer(long \_customerId, string \_firstName, string \_lastName, string \_gender, string \_email, string \_phoneNumber, string \_address)

e) A valid email has an @ and ends with “.com" or " .org”.

Create a method ValidateEmail in customer class with return type as void and check for valid email. If an email id is invalid, raise the custom exception(InvalidEmailException) and do not add the entity into the list. Print the messege "Invalid Email for the user".

f)Sort the list of customer based on firstname ( hint: Use IComparable).  
In the Customer class extend the **IComparable**interface and implete the **CompareTo(Customer obj)** method to compare the first name of the objects.  
  
**InvalidEmailException**Class that extends the Exception class is added as a part of template code. Add a constructor for InvalidEmailException class that takes errorMessage of type string as argument and extends base errorMessage.  
  
Program class to read the customer details in the form of csv is also added as a part of template code. In the **Program**class, create a list of type Customer. Use Split() function to split the csv data obtained as input.  
  
Call **ValidateEmail**() method to check if the email is valid. If the email is valid, add the customer object to the list. If the email is invalid , print the exception message.  
  
Call **Sort**() method to sort the customer list.   
  
The output format should be Console.WriteLine("{0}{1,25}{2,25}{3,25}{4,25}{5,25}", "Id", "First Name", "Last Name", "Gender", "Email", "Phone Number");

**Sample Input and Output:**

Enter customer details:  
**1,Vel,Murugan,Male,vel@mail.com,9876543210,Coimbatore**  
Do you want to continue?  
**yes**  
Enter customer details:  
**2,Mani,Gandan,male,mani@mail.org,9873216540,CBE**  
Do you want to continue?  
**yes**  
Enter customer details:  
**3,Thana,Rathanam,male,thana@mail.in,9783210456,Karur**  
InvalidEmailException: Invalid Email for the user  
Do you want to continue?  
**yes**  
Enter customer details:  
**4,Karthi,Keyan,male,keyan@mail.edu,9632587410,Tirupur**  
InvalidEmailException: Invalid Email for the user  
Do you want to continue?  
**yes**  
Enter customer details:  
**5,Soori,yaa,male,yaa@yaa.co.in,9875321460,Chennnai**  
InvalidEmailException: Invalid Email for the user  
Do you want to continue?  
**no**  
Id    First Name      Last Name       Gender          Email                           Phone  
2      Mani                    Gandan              male                mani@mail.org     9873216540  
1      Vel                         Murugan            Male                vel@mail.com        9876543210

**Car-Service Management System - Requirement 6**

**Requirement 6:**

Given the list of bookings, you would like to calculate the amount of revenue generated by each service engineer. For convenience, the customer and car references in the booking class is replaced by their respective ids alone.   
  
a)Create a **Booking**Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_bookingId | long |
| \_dateTimeOfService | DateTime |
| \_paymentMode | string |
| \_customerId | long |
| \_carId | long |
| \_amount | double |
| \_serviceEngineer | string |

B )Mark all the attributes as private. Create / Generate appropriate Properties. Add a default constructor and a parameterized constructor to take the arguments in the following order Booking(long \_bookingId, DateTime \_dateTimeOfService, string \_paymentMode, long \_customerId, long \_carId, double \_amount, string \_serviceEngineer). 

c)The input format  consists of all booking details seperated by comma in the below order,  
(bookingId,dateTimeOfService,paymentMode,customerId,carId,amount,serviceEngineer)

d) Create static method **OrganizeBookings** in **Booking** class which takes **List<Booking>** as input parameter and return a **SortedDictionary<string,List<Booking>>.**  "serviceEngineer" is the key and the value is the list of all the bookings handled by the service engineer. This method should iterate through the list of bookings and create multiple smaller lists of booking for each service engineer (value of the dictionary). Add each of the smaller lists to thedictionary and return the same.

e) Create a static method **FindBestServiceEngineer** in **Booking** class which takes**SortedDictionary<string,List<Booking>>** as parameters and **List<string>** as return type.   
This method takes up the organisedbooking for each service engineer and computes the total amount generated by each engineer. The list is sorted based on the maximum amount generated by engineer. The list of the service engineer names are returned from the method.

**Sample Input and Output:**  
Enter a booking detail:  
**1,06-01-2015,CC,11,1001,100,John**  
Do you want to add another booking detail:  
**yes**  
Enter a booking detail:  
**2,01-06-2017,CC,12,1002,200,Peter**  
Do you want to add another booking detail:  
**yes**  
Enter a booking detail:  
**3,28-02-2015,DC,13,1003,150,Peter**  
Do you want to add another booking detail:  
**yes**  
Enter a booking detail:  
**4,24-06-2015,CC,14,1004,250,John**  
Do you want to add another booking detail:  
**yes**  
Enter a booking detail:  
**5,14-06-2016,DC,15,1005,75,John**  
Do you want to add another booking detail:  
**yes**  
Enter a booking detail:  
**6,12-05-2017,DC,16,1006,125,Peter**  
Do you want to add another booking detail:  
**yes**  
Enter a booking detail:  
**7,24-12-2011,CC,17,1007,185,Peter**  
Do you want to add another booking detail:  
**no**  
Name - No of Booking  
Peter - 4  
John - 3