Faculty of
Computing, Engineering
and the Built Environment

Undergraduate Programme

Academic Year 2014-2015

Coursework: Team Project

Module:	*CMP2515 Software Design UG2*
School:	**Computing, Telecommunication and Networks**
Module Co-ordinator:	**Professor Zhiming Liu**
Setup Date:	**20/02/2015**
Submission Date:	**21/04/2015**
Team Number:	**M05**
Team Leader:	**Sophie Thompson**
Team Members:	**Anne Oboma, Hamza Ali, Ho Nam Tsang, Karan Badhen, Kevin Clopon, Olukayode Alatishe,**

Instruction to Students:

The final report to submit should contain this page as the cover page, with the above details of the team filled in.

As part of project management, each team should have a weekly meeting. A weekly project diary should be maintained to record the attendance of the meetings by team members, together with brief notes about the weekly project tasks allocations and how well individual team members meet the deadlines of their project tasks.

The final project report should include the weekly diary in the part of the project management.

Contents

Section 1 - The Initial Requirements Understanding

The nature of the Bvis Car Company, and why an Object-Oriented environment is applicable

The Bvis Car Company is a car hire company which currently operates with a paper-based manual system; managing the current physical functions of adding, removing and updating company records on the system. These operations also consist of producing and calculating costs and receipts based on customer payments once their hire is completed. It is evident from the problem description that a lot of data is recorded throughout the system, and it is blatant that this important data needs to be easily accessed, maintained and stored within the new computer-based system using an Object-Oriented development environment. The current files and concepts within the system are: a customer file, car file, hire agreement file, and a mechanic file, which all have various different field/attributes within them that contain the relevant data to store sufficient information for the company, concerning their customer car hires. All of these files can be easily transformed into object oriented classes containing properties and attributes for a particular concept. I.e. if the customer file was to be transformed into a class with objects, then the attributes of that class would be the customer name, telephone number and address, which would be the customer class' own private states to monitor and handle within the new computer-based system.

As the company deals with a wide range of customers on record, they currently have to manually add, remove and update customer details to their customer file, including: name, telephone number and address. It is essential to take into consideration that new appended records would also have to be manually sorted into alphanumeric order in their relevant file, to ensure sufficient data organisation alongside with fast and easy data access in a paper-based system. Additionally, the problem description states that if a customer is deemed as "non-regular" then they can be removed from the company's records, this in itself can be a time-consuming process as the staff in charge of files would have to manually search through the customer records, check to see if the customer regularly hires cars, and if not then remove them from the company's files. The process time for this operation can be significantly minimised with the use of a computer-based system, which could simply track a record and remove it from the company records in less than a second.

The problem description states that every car has to be sent off for two types of service depending on the mileage of the vehicle, which are performed by the company's on-site mechanics. A great deal of data needs to be recorded regarding the upkeep of the cars, how they are maintained by the company's mechanical staff, and how often they are hired by customers. Dates of services have to be manually recorded into the company's paper-based system, alongside with the name of the mechanic that performs the service. Here it is evident that there are various concepts that need to be considered once developing the system in an OO Environment, as this particular aspect of the system would require data being pulled from two separate classes, and different objects being created in order to store the data required for adding the service of a car to the new computerised system, it indicates that this project is eligible for transformation into an Object-Oriented application. As a lot of data needs to be recorded for the cars, this can once again be time-consuming for the staff managing the data, as all of these records need to regularly be updated and processed, as well as sorted back into order once being edited, which can be prone to human-error and can decrease the efficiency factor within the system.

As a lot of information is stored within the car file such as, the registration number, make, model, engine capacity, hire class and service data, this particular file holds very significant figures as each car has to have up-to-date data within it after each service and hire. Each car has to have the data concerning the mileage updated each time it is returned from a hire, and it has to have the data regarding each service added and updated to ensure that it meets the standard requirements for each customer hire. Handling this prolific scope of data for one file can be quite haphazard using a paper-based system, as it can be easily prone to human-error through the confusion and boredom of the persons managing the data. It can also be extremely time-consuming adding all of the appropriate data to the file, then repetitively updating the relevant fields to ensure data accuracy and consistence. Therefore, transforming the current paper-based system to an OO Environment will help control the size and scalability of files and the information stored within them. As the car file would be considered as a concept/class within the OO Environment, classes are in charge of their own private states, functions and methods, therefore this particular file which contains a lot of data could easily store all of the relevant information needed for the car class, through the use of instance variables/constants and the various methods used to store and display data to the user. These can also be manipulated for further use throughout the application, as data and variables can be accessed via different classes within the application, encouraging the distribution of data throughout the system.

In the current system, file updates on customer records being added and vehicle returns are manually altered by the staff in charge of managing the data in the paper-based system. When a vehicle is hired by an unknown customer their details have to be appended to the customer file, including their name, telephone number, address and driving licence number. Once in the customer file, one assumes that the record would need to be placed into alphanumerical order, which can inevitably be deemed as unnecessarily time-consuming when completing this process on a paper-based system, as it has to be done manually. It states that customers have to fill in a hire form once acquiring the hire service, which again adds excess time to the data processing procedure, the processing time on this function can significantly be reduced in the new computer-based system.

The problem description states that when a car is returned back to the company, the actual return date and the mileage are recorded in the hire agreement file, and the costs of the hire are manually calculated by members of staff based upon the daily hire rate. Customers must only make a cash payment and get a receipt before leaving the company. This particular attribute of the company's current paper-based system can easily be replaced with OO Environment functionality, as calculating the customers' payment for hire can be transformed into a method within the hire agreement class within the new computer-based application, which calculates a customer's total using the relevant arithmetic assignments in code. Another method could be used for displaying and printing the customer receipts using simple methods with ease. Completing these processes in an OO Environment would prove to be much easier and faster once handling the application, as all of these methods would automatically be done for the user once the appropriate data is inputted on entry. Whilst, performing these processes manually especially calculating the customer total costs can be prone to human error, therefore using an electronic-based system will decrease the potential for human-error on data input, increasing the accuracy and reliability of data within the company's records.

Staff are to regularly record every completed hire within the company, including: the details of the customer, the dates of the beginning and end of hire, alongside with the amount the customer has to pay for their service. This information once again could simply be implemented into an OO Environment application, as the information that needs to be recorded could easily be stored within instance variables declared inside the program; allowing them to be accessed by any class within the computer-based system if the variables are made public in the application. The same could be said for storing data on the hire class of vehicles, which has to be recorded also. As this data has to be manually added to the company's paper-based system, this once again can be very time-consuming and can cause annoyance for the staff managing the records as they regularly have to add, update, remove and maintain company records by hand, which can also cause demotivation for staff and a decrease in morale due to a high workload. An electronic-based OO Environment application would complete all of these processes with utmost speed and precision, applying objects, classes and attributes to help manage and maintain company data accurately, alongside with aiding in the storing of important data. The staff also have to manually add company mechanic details to the company's records such as, name, address, telephone number and they must hold a current driving licence. The same issues with the above system functionalities apply to this also.

An Object-Oriented Environment is applicable to this project due to the fact that the Bvis Car Company's current paper-based system has many concepts within its structure such as: car, customer, mechanic, hire agreement etc., these concepts represent an abstraction generalising the properties common to each of the concepts. In the OO framework these concepts such as car, customer etc. containing their properties/attributes including name, model etc. can be transformed into a class structure, offering an increase in data organisation and better storage capabilities.

Additionally, as the company's current system has many functions to consider such as, inserting, editing and removing data, alongside with calculating customer hire costs, a lot of these functions can be transformed into class methods. These class methods which perform differing operations, whether it be assigning values to a variable, or returning the result of an operation or assignment, the outcome of the method once it is been processed and performed gets returned to the main class of the application, once the method has been specifically called by a command written in code.

An OO Environment helps to separate functions and data based on the concepts in which they've been assigned, enabling more control over the behaviour of the system and the data which is processed and outputted as a result of the operations/procedures performed. This would be extremely helpful for the Bvis Car Company, as a lot of data needs to be added/grouped together from different aspects of the business, such as the data regarding the servicing of a car, has to have information containing the date of service and the mechanic that performed it, which would need data from two different sources combined. An OO Environment handles these types of procedures easily, and helps to organise the data in such a way that storage and data access shouldn't be a problem for the company with the future computer-based system. As the OO framework includes concepts/classes methods and functions within its structure, and the current paper-based system comprises of a lot of data/operations that need to be performed and maintained, an OO Environment can help to break down functions into suitable methods to be performed and converted into source code.

As in the OO Environment, on object simply represents a system entity, the Bvis Car Company's current system would contain a series of entities within its structure such as car, customer etc. These entities/objects are responsible for managing their own private states and variables, e.g. in the new car class of the computer-based system, the private states and functions would be declaring variables for all the different fields which are currently present in the car file such as registration number, make, model etc., which would carefully store the information required for each car within the company. Data and functions are encapsulated within an object; therefore all of the data concerned with that particular class/concept will be stored, updated and maintained specifically within that class. This particular feature of the OO Environment would prolifically be applicable to this project as many different operations and procedures need to be carried out within certain aspects of the company concerning differing entities within the business. For example, when collecting regular data for the cars within the company, many pieces of information need to be updated such as who's hired the car for a particular time, when it needs servicing due to the mileage value, alongside with the hire class of vehicles also.

A lot of things need to be recorded and functions need to be performed for specific entities, therefore adopting the class structure for this particular system would be applicable, as the differing entities could be in their own separate class, containing their own individual attributes and methods to be performed, making sure that the requirements for the system are being met adequately. This would enable the differing classes to possess their own private states, variables and functions to control the behaviour of the separate entities, ensuring that the new application runs smoothly for the company. Also, other classes can call for methods within other classes, which encourages the sharing and joining of data within the application, which is suitable for this system as information regarding cars and mechanics/ cars and customers would need to be accessed/stored and displayed simultaneously within the application for the car class and hire agreement file also.

Objects are known as instances of a class, therefore a class can have many objects; in essence, a class can have many records stored within its hierarchy also. Objects belonging to the same class would have common properties such as attributes and operations, for example: a record in the customer class otherwise known as an object in OO Environment terms, would consist of the same type of data stored in the instance variables in the class, alongside with the same functions which would operate on the data being inputted for the cars, as it would with a different customer object created from the same class. Therefore, it maintains data integrity, accuracy and maintainability amongst the records/data within the system; increasing the reliability of the data stored for the company.

批注 [ZL1]: This discussion helps in some way in the understanding of the problem. However, this part request the discussion of the attributes of this application in relation to the five attributes of complex system discussed in Chapter 3 of the course notes. This was actually clearly indicated in the project description.

System Functions

Overview of Project

The purpose of this project for the Bvis Car Company is to transform their current paper-based car hire system, into a computer-based Object-Oriented structured application. This application will be operated to control the monitoring of interactions and processes for customer car hires, and the general upkeep of the regularly used vehicles within the company.

The generic functions of the system are to securely store records based upon the customers that hire from the company, the cars within the business/ how they are serviced, the car hires that occur along with customer payments for hires, alongside with information regarding the company's mechanics also. The new computerised system should enable the user to add, update and remove records from any file/class stored within the application, to collect the relevant information needed to compile data for car hire agreements.

As the system functions have to take into account the information needed to store data on car hire agreements such as, the date of car hires and their returns, alongside with the mileages of the cars to see whether they're due for a service, a lot of required data needs to be accumulated for the cars to keep track on their maintenance and the amount of times they're hired etc. Therefore, a new computerised system could simply manage all of these simple procedural operations such as the storing of data in multiple files, alongside with the functions of adding, editing and deleting company records regarding customers, cars, hire agreements and mechanics, for specified system requirements.

Additionally, a key feature within the system is calculating the cost of payment for a customer once their hire is complete. This attribute can also be performed electronically through the new computer-based system, as software programs are facilitated with built in features that easily calculate and execute arithmetic assignments for any purpose, in this scenario we need this specific feature to calculate customer payments based on the daily hire rate for a car. Then a receipt needs to be produced and displayed for the customer to view to cease the care hire process.

This system will be able to interact and work with other systems through the fact that the finishing product, would be OO structured application, and the source code from the developed program would be able to compile and be distributed throughout company machines; this is due to the fact that all computer programs created using a programming language can be transformed into an executable file, which can easily be distributed and used on differing machines in/outside an organisation or establishment. Therefore, this would enable different users within the Bvis Car Company to use and benefit from the system. As this system would provide all of the essential functions for the desired end product, this application would be suitable for use in the entire company for the staff.

Goals

To generalise, the most important goals for this project are to increase the procedural speed of the functions performed throughout the system. Examples include, the appending of records to a file, the updating of a record once new data has been collected determining the values of the new updated record, and the removal of data once it is no longer of use. The new computer-based system should encourage an increase in efficiency, speed and customer-satisfaction. More specifically, these goals include:

- Fast and easy access to company records to add or manipulate data to files
- It will aid in the increased speed for processing data on new car hires
- It will help to store company data regarding customers and their payments
- This system will help to keep track of car maintenances such as service information, hire class data and the mileages consumed after each car hire with ease, enabling staff to keep control of important company data
- The system will become more efficient as it will be operated electronically, encouraging faster processing and access time, increasing customer-satisfaction
- It will help increase the morale and motivation of staff, as they will not have to manually add, edit, delete and sort company records due to the implementation of the new computer-based system. This can therefore increase staff productivity, resulting in staff performing better for the company
- An increase in staff productivity can result in an improvement in the quality customer care and handling, which can also cause an increase of customer satisfaction, which could potentially result in an increase in customers hiring cars for the business

System Functions

Reference	Function	Category
R1.1	Capture new customer information by inputting the name, address and telephone number to the company database	Evident
R1.2	Capture new car information by inputting registration number, make, model, engine capacity, hire class, date of registration, and date of services along with the mechanic who performed the services on the vehicle. Add this information to the company's database	Evident
R1.3	Capture new mechanic information by inputting the name, telephone number, address and driving licence number of the newly hired mechanic, to the company's database	Evident
R1.4	Remove a customer's information from the system database, making it no longer accessible to the user	Evident
R1.5	Ensure that the desired customer record is removed permanently from the company's database system	Hidden
R1.6	Remove a car's details from the system database, making it no longer accessible to the user	Evident
R1.7	Ensure that the desired car record is removed permanently from the company's database system	Hidden
R1.8	Remove a mechanic's information from the system database, making it no longer accessible to the user	Evident
R1.9	Ensure that the desired mechanic record is removed permanently from the company's database system	Hidden
R1.10	Record once a customer has completed their hire, the actual date of return and the consumed mileage of the car whilst it was being hired	Evident
R1.11	Once a customer hires a car, record the estimated return date for the hire, and the start date of the hire taking into the current mileage of the vehicle	Evident
R.1.12	If a customer is not already recorded in the company's database, then their details must be added to the file	Evident
R1.13	Record the car hire details before the hire is complete, containing an estimated return date, with the actual start date of the customer's car hire	Evident
R1.14	Store the car hire temporary information, until the actual return date and updated mileage values are received at the end of the hire process	Hidden
R1.15	Capture the car details from the car hire, such as the new actual return date for the hire, the mileage and the customer that proceeded with the hire	Evident
R1.16	Calculate the total cost of payment for the customer based upon the daily hire rate for that specific car	Hidden
R1.17	Log a completed car hire sale payment	Hidden
R1.18	Handle cash payments, and display/print a receipt showing the total cost of hire for the customer to view and take away	Evident
R1.19	Provide a sustainable storage mechanism	Hidden

R1.20	Provide serviceable processes and inter-system communication mechanisms	Hidden
R1.21	Capture the car servicing details from the management staff, declaring the date of the service, the mileage of the car before the service, alongside with the mechanic that performed the service	Evident
R1.22	Store the servicing data so that it can be accessed at any time throughout the system	Hidden
R1.23	Store information on cars and their hire class, specifying the particular hire rate for each car once they are hired	Hidden

System Attributes

Attribute	Constraints

Response Time	The software should be able to identify and process the information within 1 minute
Ease of Use	The user interface should be very simple so that everyone will be able to use it with ease, this could be done using simple drop boxes and check boxes
Operating system	Windows and Mac will be compatible
Retail Cost	The price of rental will depend on the information that users input, this will be decided upon the daily hire rate for the vehicle, the duration of the rental and the model of the specific car
Fault-tolerance	Creating accounts with usernames and passwords can be used to prevent faults and errors happening as the correct details and information will be checked and recorded
Accessibility	Should be able to access on various pcs throughout the company's building that are connected on a network, as the software installed should be executable and distributable
Backup/Storage	The information and data should be stored in separate servers and back up servers which are in different locations
Privacy/Security	As mentioned above, back up and storing data are to prevent either accident or human damages, and firewall and anti-virus can be used to provide internet security
Testability	Alpha and Beta testing is vital before this software can be released to the company
Portability	Once the software is completed it is highly easily portable as to access the software, as the application produced would be an executable file, which can easily be installed and uploaded onto any machine within the company offices

批注 [ZL2]: A quite analysis and understanding of core functions of the system

Essential Use Cases – High Level Use Cases

Use Case: Register a Customer

Actors: Staff (~~Initiator~~Direct Actor), Customer (Initiator)

Purpose: Collect new customer information to be stored on database

Overview: A customer arrives wanting to hire a car. Staff will initiate to collect the customers personal information. Upon Completion the customer is now eligible to hire a car

批注 [ZL3]: Avoid design details

Use Case: Car out for hire

Actors: Staff, Customer (Initiator)

Purpose: Record that a particular car has been hired

Overview: Customer enters and selects the car they would like to use. Staff will pull up their data from the database. Staff will record the date at which the car hire starts and when the hire will end (end of hire is estimated)

批注 [ZL4]: Hire a Car

Use Case: Car Return

Actors: Staff, Customer (Initiator)

Purpose: Record that a particular car has been returned

Overview: Customer returns car. Staff records actual return date and mileage.

批注 [ZL5]: Return a car

Use Case: Record a completed hire

Actors: Staff (Initiator), Customer

Purpose: Log a completed hire

Overview: Once the customer has returned the car a member of staff will log the completed hire by compiling a file of the details of the customer, dates of beginning and end of hire, and the amount of payment

Use Case: ~~Servicing~~Process a car service

Actors: Staff (Initiator), Mechanic

Purpose: Record a service for a particular car, together with the date of the service, the type of the service, and the name of the mechanic responsible

Overview: When a car is taken into the garage and has been used for 6,000 miles or 12,000 miles a mechanic is assigned to it. The date of the service is recorded.

Use Case: Removal of customer from database

Actors: Staff (Initiator)

Purpose: Remove a customer.

Overview: Staff will retrieve the customer file and records from their database, once this is done their records can be removed.

Use Case: Add car

Actors: Staff (Initiator), Management

Purpose: Add a new car to the fleet

Overview: Once a new car has been brought into the garage. The particular cars details such as model, registration number, make, engine capacity, hire class and date of registration will be added to the car file.

Use Case: Delete car

Actors: Staff (Initiator), Management

Purpose: Delete a car that is no longer in the hire fleet.

Overview: Once a car is deemed surplus to requirements a member of management will retrieve their records and delete the car from the fleet.

Use Case: Add mechanic

Actors: Staff (Initiator), Mechanic

Purpose: Add a mechanic who has joined the company

Overview: Once a mechanic has joined the company their details will be recorded such as name, address and telephone number. In addition it is a requirement for all mechanics to hold a current driving license.

Use Case: Delete former mechanic

Actors: Management (Initiator), Mechanic, Management

Purpose: Remove the details of a mechanic who has left the company

Overview: Once a mechanic leaves the company a member of management will retrieve their files and record and delete it from the system.

批注 [ZL6]: In general a quite good set of high level use case decriptions

Extended Use Cases

Registering a new customer

Actor Action	System Response
1. Customer comes into branch with the intention of hiring a car	
2. A member of staff will initiate the process of gathering the customers information	
	3. The system reads in the member of staff's input from the customer's information from the hire form
	4. The information gathered will now be stored on the company's database system
5. The member of staff may present the customer with their new registered details, enabling them view their details	6. The system will present the newly registered customer's details to the screen for display
7. Now the customer has been registered on the database they can now proceed to hiring a car	

批注 [ZL7]: One atomic action

批注 [ZL8]: Avoid GUI design details

Use Case

This use case focusses upon the process of where the actor (Customer) will arrive at the company, and will ask to hire a car from the company. The customer will need to register a new account with the company if their personal details are not already stored within the company's database. The customer will need to supply their personal details in order for the registration process to proceed.

Actors

The actors concerned for this particular use case are the Initiate Actor (Customer), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful customer registration is achieved. The initiate actor, which is the customer, is the one who will start off the registration process by approaching the cashier staff, asking to hire a car from the company. The direct actor, which is the member of staff, will be the one making direct contact with the system, enabling that the customer's registration to be performed by entering in their details to be stored on the company's database.

Purpose

The purpose of this analytical review is to enable customers to come into the Bvis Car Company to register for a future purchase from the business. The direct actor needs to have efficient communication between the initiate actor, to ensure that the exchange of data for the registration to be completely successful.

Overview

A customer arrives to the company wanting to hire a car, however they will first need to register with the company to enable them to hire a car. Staff will initiate to collect the customers personal information. Upon Completion the customer is now eligible to hire a car.

Car out for hire

Typical course of action:

Actor action	System Response
1. A customer will come into the branch to hire a car.	
2. The customer will select a car then inform a member of staff of their choice.	
3. A member of staff will begin the process by checking the customers details to see if they are registered	
	4. Once the customer has been found on the system the member of staff assigned to this transaction will then attempt
5. Staff will check to see if the car is available to be rented by the customer	
	6. The system will check for the car's availability within the system
7. Staff will assign the specified car to the customer with the car hire date that they will be able to hire the car from the company	
8. Staff will inform the customer on the estimated return date, so they know when to return the car back to the company	
	9. A record of the date that the car is hired, the customer and the expected return date.
10. The customer can now leave with the car	

批注 [ZL9]: More essential system actions needed here

Use Case

This use case focusses upon the process of where the actor (Customer) will arrive at the company, and will ask to hire a car from the company. The staff will need to check the customer's details to ensure that they are eligible for a car hire, and they will also need to check if the car they wish to hire is available. Once those details have been checked the customer will now be able to leave with the car they wish to hire, and the staff will record that a particular car has been hired on the company's system.

Actors

The actors concerned for this particular use case are the Initiate Actor (Customer), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful customer registration is achieved. The initiate actor, which is the customer, is the one who will start off the registration process by approaching the cashier staff, asking to hire a car from the company. The direct actor, which is the member of staff, will be the one making direct contact with the system, checking the customer's details and the car's availability to ensure that the customer can hire the car. Also, the staff will need to record the hire on the database.

Purpose

The purpose of this analytical review is to enable customers to come into the Bvis Car Company to purchase a hire car from the company, and for the hire to be recorded onto the company's database. The direct actor needs to have efficient communication between the initiate actor, to ensure that the exchange of data for the car hire to be completely successful, and for the details to be recorded accurately.

Overview

A customer enters and selects the car they would like to hire. Staff will pull up the customer's data from the database if they have already been registered, then they will check for the customer's selected car's availability. If the car is available then the customer will be able to hire the car, and the staff will record the date at the beginning of the hire, and the estimated return date.

Car return

Typical course of action:

Actor Action	System Response
1. The customer walks into the company, wanting to return their hired car on the agreed date	
2. The customer will need to show their registration and car hire data for proof of hire to enable the return	
3. Staff will need to check on the system whether the customer has an account, and if their hire has been recorded	
	4. The system will check to see if the customer is registered with the company, and if their car hire has been recorded
5. Staff will enter in the car's details that was hired, to ensure that the vehicle is from the company	
	6. The system will check to see if the car is from the car hire company from its registration number
7. Staff will inform the customer that the car has been successfully returned to the company	
8. Staff will log into the system to insert data concerning the date in which the car was returned, and the mileage on the car	
	9. The system will record that the car was returned from the customer, including the date and mileage of the car
10. Staff will issue a receipt for the customer	
	11. The system will generate and print a receipt for the customer
12. The customer will then pay the amount owed for the hire	
	13. The system will record that a payment was made by the customer for the car hire, and the transaction will be recorded with their personal details

批注 [ZL10]: The application logic is not quite right. When a car is being returned, the hire contract should be found, data and mileage should be updated, payment should be dealt with

Use Case

This use case focusses upon the process of where the actor (Customer) will arrive at the company, and will ask to return a hired car back to the company. The staff will need to check the customer's registration and car hire details to check when the car was hired and if it was hired from the company. Once those details have been checked the customer will now be able to leave the car with the company, and pay for the purchased car hire through the payment of cash. Staff will record that the car has been returned and that the customer has paid their owed amount for the hire.

Actors

The actors concerned for this particular use case are the Initiate Actor (Customer), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful customer car return and payment is achieved. The initiate actor, which is the customer, is the one who will start off the car return and payment process by approaching the cashier staff, asking to return a car back to the company, and then once the car is returned the customer will then pay the amount owed for the hire. The direct actor, which is the member of staff, will be the one making direct contact with the system, checking the customer and car details to ensure an effective car return and payment is achieved. Also, the staff will need to record the car return and customer payment on the database.

Purpose

The purpose of this analytical review is to enable customers to come into the Bvis Car Company to return a hired car back to the company, and to make official payment for the hire, also for the hire and the payment transaction to be recorded onto the company's database. The direct actor needs to have efficient communication between the initiate actor, to ensure that the exchanges of data for the car return and payment to be completely successful, and for the details to be recorded accurately.

Overview

A customer enters and returns the car in which they previously hired from the company. Staff will pull up the customer's hire data from the database if they have already been registered, then the return process will commence and the customer's payment transaction will happen. The staff will record the date at the beginning of the hire, and the actual return date, alongside with the mileage and the payment made by the customer for the hire.

Record a completed hire

Typical course of action:

Actor action	System Response
1. Once a customer has returned the car a member of staff will input specific information into the system	
	2. The system will log the customer who hired the car, the car type and model, the date the car was hired and when it was returned. In addition a completed record will include customer payment
3. Once this information has been recorded and stored a member of staff will double check to see everything matches up correctly.	

Use Case

This use case focusses upon the process of where the actor (Customer) will arrive at the company, and will ask to return a hired car back to the company. The staff will need to check the customer's registration and car hire details to check when the car was hired and if it was hired from the company. Once those details have been checked the customer will now be able to leave the car with the company, and pay for the purchased car hire through the payment of cash. Staff will record that the car has been returned and that the customer has paid their owed amount for the hire.

Actors

The actors concerned for this particular use case are the Initiate Actor (Customer), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful customer car return and payment is achieved. The initiate actor, which is the customer, is the one who will start off the car return and payment process by approaching the cashier staff, asking to return a car back to the company, and then once the car is returned the customer will then pay the amount owed for the hire. The direct actor, which is the member of staff, will be the one making direct contact with the system, checking the customer and car details to ensure an effective car return and payment is achieved. Also, the staff will need to record the car return and customer payment on the database.

Purpose

The purpose of this analytical review is to enable customers to come into the Bvis Car Company to return a hired car back to the company, and to make official payment for the hire, also for the hire and the payment transaction to be recorded onto the company's database. The direct actor needs to have efficient communication between the initiate actor, to ensure that the exchanges of data for the car return and payment to be completely successful, and for the details to be recorded accurately.

Overview

A customer enters and returns the car in which they previously hired from the company. Staff will pull up the customer's hire data from the database if they have already been registered, then the return process will commence and the customer's payment transaction will happen. The staff will record the date at the beginning of the hire, and the actual return date, alongside with the mileage and the payment made by the customer for the hire.

Servicing

Typical course of action:

Actors actions	System Response
1. Once a car has been returned from a customer hire, then their mileage will be checked and recorded onto the system	
2. Once a car has been used up to 6,000/12,00 miles the staff will make a note stating it's time for servicing	
3. Once staff has made a note on the particular car, a mechanic will be assigned to the job of servicing it	
	4. A selection of mechanics can be chosen to undertake the servicing process for the car.
5. A mechanic is chosen by staff to perform the car service, and the mechanic will be informed	
6. The mechanic will then perform the service on the car	
7. Staff will record the details of the car, mechanic and the date at which the service should commence.	

Use Case

This use case focusses upon the process of where the actor (Mechanic) will perform a service on a car that has been returned with a mileage of 6,000 or 12,000 miles in the company's own garage. Once the service has been performed by the designated mechanic then staff will record that the car has been serviced by a particular mechanic.

Actors

The actors concerned for this particular use case are the Initiate Actor (Mechanic), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful customer car return and payment is achieved. The initiate actor, which is the mechanic, is the one who will start off the car service process by performing the service on the car. However, a mechanic needs to be assigned to the service by management by selecting them from the system's database, and then they will be informed on the service. The direct actor, which is the member of staff, will be the one making direct contact with the system, checking the mechanic and car details to ensure that the car is due for a service, and selecting the mechanic to perform the service. Also, the staff will need to record that the car has been serviced, with the mechanic responsible for the service on the company's database.

Purpose

The purpose of this analytical review is to make sure that cars that are due for a service that have been returned are sent in for a mandatory service by the company's own mechanics. The date of the service alongside with the mechanic which performed the service are recorded into the database. The direct actor needs to have efficient communication between the initiate actor, to ensure that the exchanges of data for car servicing to be completely successful, and for the details to be recorded accurately.

Overview

A customer enters and returns the car in which they previously hired from the company. Staff will pull up the car's details to check the mileage of the car to see if it's due for servicing. If the car is due for servicing then staff will assign a mechanic to perform the service on the car. Once the mechanic performs the service, staff will record the date of the service, alongside with the mechanic responsible for the service.

Removal of customer from database

Typical course of action:

Actors actions	System Response
1. Staff want to remove a customer's record from the company database	
2. Staff will search the database for the designated customer by inserting their details	
	3. The system will check to see if the data entered matches a record on the system
4. Staff can now choose to delete the customer's record from the database	
	5. The system will remove the selected customer record which the member of staff have chosen to delete
6. Staff will notify the customer that their personal details have been removed from their database	
	7. The system will inform the customer via email or SMS

Use Case

This use case focusses upon the process of where the actor's (Customer) details will be removed from the company's database by the members of staff. The staff will need to check if the customer has been registered within the company's system by checking if the customer's personal details are on the system. If they are then the member of staff will then delete the selected customer's record from the database, and will alert higher management that such action has been taken, via email or SMS.

Actors

The actors concerned for this particular use case are the Initiate Actor (Customer), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful customer record deletion is achieved. The direct actor, which is the member of staff, will be the one making direct contact with the system, making sure that the customer's record is permanently removed from the company's database, and they will also alert management that such actions have been taken.

Purpose

The purpose of this analytical review is to enable members of staff to permanently delete customer records from the company's database. The direct actor needs to have efficient communication between the initiate actor, to ensure that the deletions of customer records are performed successfully.

Overview

A member of staff wishes to remove a customer's record from the database permanently from the existing system. They would have to search for the customer's details by inputting segments of it into the system to initiate a search for the record. Once the record is found by the system, the member of staff is able to select the record and delete it from the company's database. Once this is done, the members of staff are then required to inform members of management that a particular record has been removed from the company's database, via email or SMS.

Add car

Typical course of action:

Actors actions	System Response
1. Once the company has brought a new car into the garage it is the responsible of senior staff/ management to add the car to the database	
2. Staff will need to input the car's details onto the system	
	3. The system will check to see if the car's details are accurate to be stored on the system for hire
	4. The car details such as model, registration number, make, engine capacity, hire class and date of registration will all be stored on the database.
5. Management will be notified that a new car has been added to the fleet, ready for customer use	
	6. The system will inform management via email or SMS that a new car has been added to the company
7. Once the data has been successfully inputted onto the database the car can now be hired for customer usage.	

Use Case

This use case focusses upon the process of where the actor (Staff) will add a new car that has been added to the fleet onto the company's database. The member of staff will need to input all of the car's necessary details to ensure that the car is insured with the company, and that it is eligible for future customer hires. Once the car's details have been stored and added it will now be ready to be hired by a customer.

Actors

The actors concerned for this particular use case are the Initiate Actor (Mechanic), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The direct actor, which is the member of staff, will be the one making direct contact with the system, who will be adding the car's necessary details to the company's database, to ensure that it is fit for customer hires. The staff will need to ensure that new car details are successfully recorded to the company's database.

Purpose

The purpose of this analytical review is to make sure that new cars that are added to the fleet, have their details successfully recorded onto the company's database. The direct actor needs to ensure that all of the relevant details concerning the cars are accurately recorded to ensure that they are eligible for customer car hires.

Overview

A new car is added to the company's fleet and members of staff have to add the car's relevant details to the company's database. The system will check to see if the car's details are correct and if they are then the car's details will have been successfully added to the database. Once a new car has been added to the fleet and onto the company's database then staff must inform members of management of the action, via the use of email or SMS.

Delete car

Typical course of action:

Actors actions	System Response
1. Staff want to remove a particular car's record from the company database	
2. Once the company has brought an old car in the garage it is the responsible of senior staff/ management to delete the car on the database	
3. Staff will search the database for the designated vehicle by inserting its details	
	4. The system will check to see if the data entered matches a record on the system
5. Staff can now choose the car in which they wish to delete by selecting their details on the system	
	6. The car details such as model, registration number, make, engine capacity, hire class and date of registration will all be removed from the database.
7. Staff will notify management that a car has been removed from the company and the database records	
	8. The system will inform management via email or SMS that a car has been removed from the company
9. Once the data has been successfully deleted from the database the car can no longer be hired for customer usage.	

Use Case

This use case focusses upon the process of where a car's details will be removed from the company's database, and from the fleet by the members of staff. The staff will need to check if the car has been registered within the company's system by checking if the car's relevant details are on the system. If they are then the member of staff will then delete the selected car's record from the database, and will alert higher management that such action has been taken, via email or SMS.

Actors

The actors concerned for this particular use case is the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The direct actor needs to perform and interact with the system effectively for successful record deletion to take place. The direct actor, which is the member of staff, will be the one making direct contact with the system, making sure that the car's record is permanently removed from the company's database, and they will also alert management that such actions have been taken.

Purpose

The purpose of this analytical review is to enable members of staff to permanently delete car records from the company's database. The direct actor needs to have efficient communication with the system, to ensure that the deletions of car records from the fleet are performed successfully.

Overview

A member of staff wishes to remove a car record from the database permanently from the existing system. They would have to search for the car's details by inputting segments of it into the system to initiate a search for the record. Once the record is found by the system, the member of staff is able to select the record and delete it from the company's database. Once this is done, the members of staff are then required to inform members of management that a particular record has been removed from the company's database, via email or SMS.

Add mechanic

Typical course of action:

Actors actions	System Response
1. A new mechanic will come into the store but before he can start work he would have to get registered by a member of management.	
2. A member of staff will take specific information from the mechanic such as name, address, telephone number. In addition the mechanic must have a valid driver's license to complete this enrolment process.	
	3. The system will store the personal details of the new mechanic for the company
4. Staff will add some additional information for the mechanic such as the hours in which they will work, and their annual salary	
	5. The system will store this additional work hours / annual salary information on the company database for later use
	6. Once the relevant information has been entered on the system the mechanic is now an employee of the company. The system will again verify whether or not the mechanic has a valid driving license.

Use Case

This use case focusses upon the process of where the actor (Mechanic) has been newly employed by the company, and so their personal details are to be recorded into the company's database by members of staff. The mechanic will need to supply their personal details and state their current valid driving licence number, in order to be eligible to work for the company as a qualified mechanic. The members of staff will need to accurately record these details, alongside with additional data such as working hours and the annual salary of the mechanic also.

Actors

The actors concerned for this particular use case are the Initiate Actor (Mechanic), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful mechanic registration is achieved. The initiate actor, which is the mechanic, is the one who will start off the registration process by being newly employed by the company, in which case their personal details must be recorded by members of staff into the company database. The direct actor, which is the member of staff, will be the one making direct contact with the system, enabling that the mechanic's registration to be performed by entering in their details to be stored on the company's database.

Purpose

The purpose of this analytical review is to enable new mechanics that have been hired by the company to register their details, so that they are on the company's database system, to enable them to work for the Bvis Car Company. The direct actor needs to have efficient communication between the initiate actor, to ensure that the exchange of data for the registration to be completely successful.

Overview

A new mechanic has been employed by the Bvis Car Company, and their personal details including their current driving licence number needs to be added to the company's database. Members of staff then add the mechanic's personal details to the database, and the system will then check to see if the data entered is accurate. If it is then the mechanic's data has been successfully entered into the system, and the mechanic is now able to start working for the company.

Delete former mechanic

Typical course of action:

Actors actions	System Response
1. A mechanic will cease to be employed by the company requiring management team to delete the former employees records	
2. Staff will search the database for the designated mechanic by inserting their details	
	3. The system will check to see if the data entered matches a record on the system
4. Staff can now choose the mechanic in which they wish to remove from the database, by selecting their details on the system	
	5. Once the system locates the mechanic those members with the correct authority now have the option to delete the mechanics records
6. Management can now proceed and delete the former mechanic	
	7. The system will now remove the mechanic's personal details from the database
8. Staff will notify management that a mechanic has left the company and their personal details have been removed from the database records	
	9. The system will inform management via email or SMS that a mechanic has left the company

Use Case

This use case focusses upon the process of where the actor's (Mechanic) details will be removed from the company's database by the members of staff. The staff will need to check if the mechanic has been registered within the company's system by checking if the customer's personal details are on the system. If they are then the member of staff will then delete the selected mechanic's record from the database, and will alert higher management that such action has been taken, via email or SMS.

Actors

The actors concerned for this particular use case are the Initiate Actor (Mechanic), and the Direct Actor (Staff). The initiate actor is the one that commences the process of the use case, and the direct actor is the one that makes direct contact with the system to ensure that the process for the use case is performed. The two actors need to effectively communicate so that successful mechanic record deletion is achieved. The direct actor, which is the member of staff, will be the one making direct contact with the system, making sure that the mechanic's record is permanently removed from the company's database, and they will also alert management that such actions have been taken.

Purpose

The purpose of this analytical review is to enable members of staff to permanently delete former mechanic records from the company's database. The direct actor needs to have efficient communication between the initiate actor, to ensure that the deletions of customer records are performed successfully.

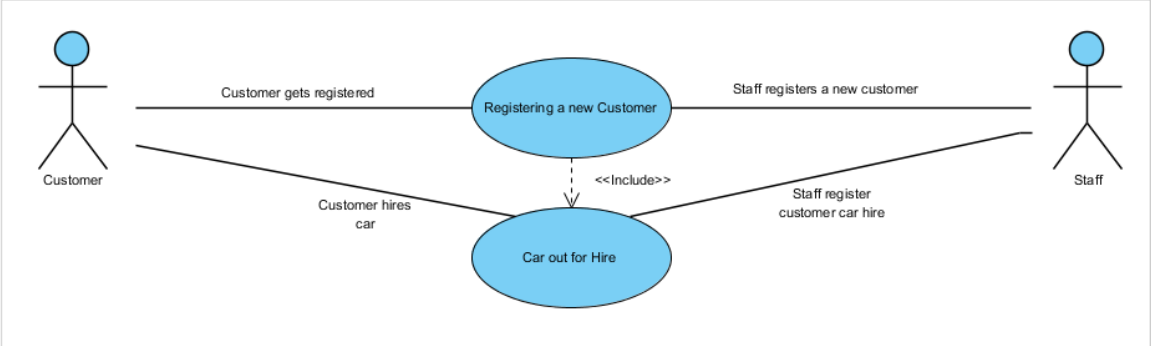
Overview

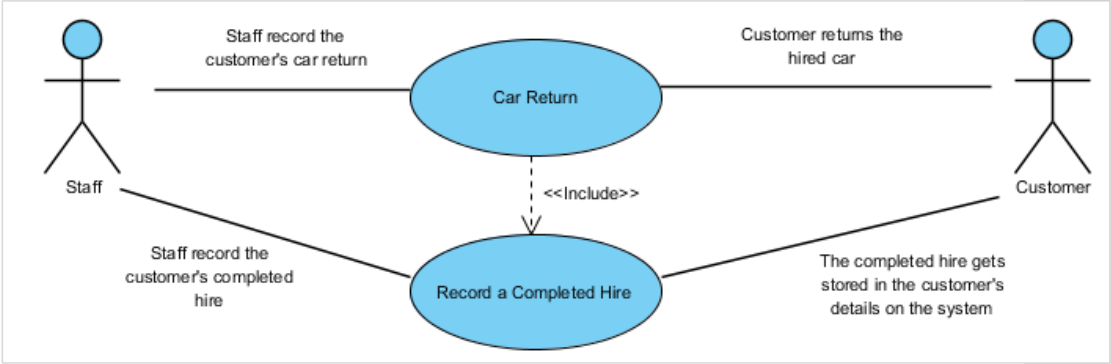
A member of staff wishes to remove a mechanic's record from the database permanently from the existing system. They would have to search for the mechanic's details by inputting segments of it into the system to initiate a search for the record. Once the record is found by the system, the member of staff is able to select the record and delete it from the company's database. Once this is done, the members of staff are then required to inform members of management that a particular record has been removed from the company's database, via email or SMS.

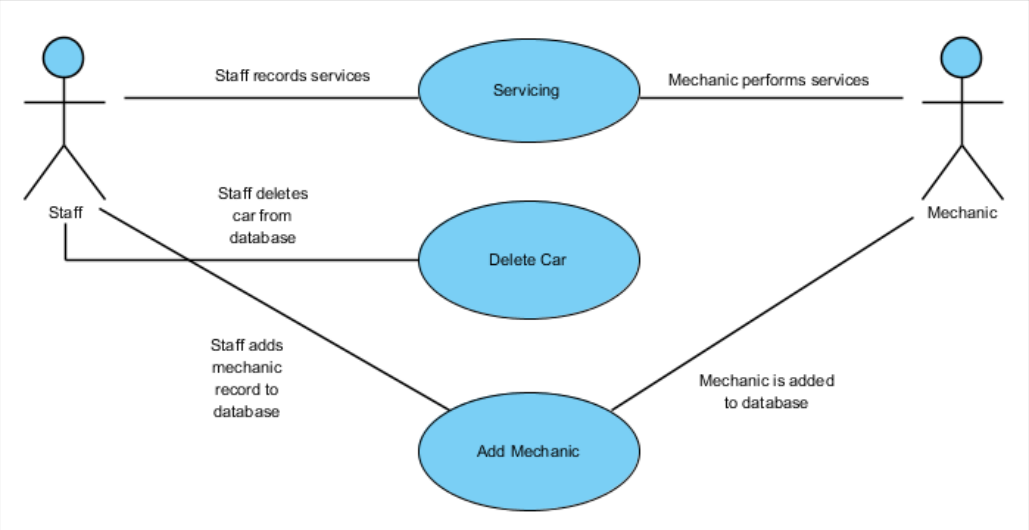
批注 [ZL11]: Show a good understanding of the interactive nature of use cases, and the format of expanded use cases, that is the typical of course of events. Weak ability of dealing with application logic, and no treatment of alternative course of events

Use Case Diagrams

批注 [ZL12]: It should be that “Hire a care” uses “register customer. The labels on the relation between actors and use cases are not mean to be association.

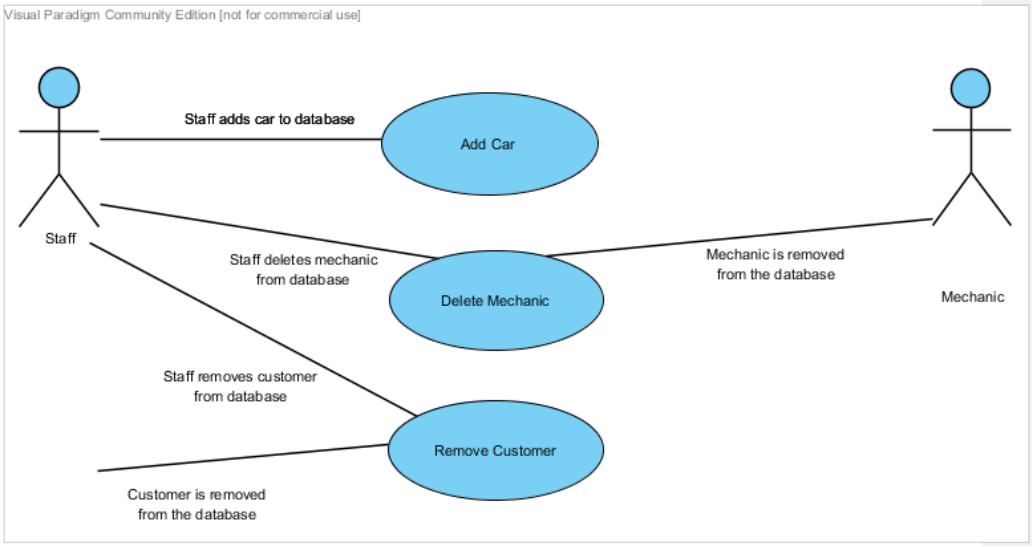






Classes (Concepts), Associations & Attributes in the Problem Domain

Here will be the featured classes, associations and attributes for each of the essential use cases provided from Q3 of the assignment. Alongside with that description will be a conceptual class diagram for each essential use case, displaying how each concept links with



one another using associations. I have chosen these concepts from the problem description due to the fact that, a lot of objects will be derived from them, such as the car class, the customer class and the mechanic class. Also, a lot of relating objects and classes link to these specific concepts such as car hire relates to the class car, and register customer relates to the class customer, therefore these classes can be singled out as primary classes where a lot of information and data can derive from and be inputted into. The attributes come directly from the problem description, as it states that particular aspects need to be recorded onto the system to store as information for a particular object coming from the system. For example: Name, address and driving licence number need to be stored in the mechanic file, as pre-determined by the problem description, therefore it will become an attribute for the mechanic class. Associations are what link the different classes together, particular tasks have to be done, and the system needs to complete specific operations for certain classes to be able to link. These associations are later identified when defining the classes below.

Quite logical groups of use cases in use case diagrams, though there are some misunderstanding about the meaning of the relations between use cases

批注 [ZL13]: By ZL

Identify Classes (Concepts), Attributes & Associations

Registering a new Customer

Classes:

Symbol: Customer

Intension: The initiate actor who is being newly registered by the company onto its database

Extension: e.g. Customers with names Tony Thompson, Layton Boyce, Jordan Smikle…

Symbol: Staff

Intension: The direct actor who completes the registration for the customer

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Register a Customer

Intension: The action taken to register a new customer with the company via Staff

Extension: e.g. Tony Thompson is registered as a new customer by the member of staff Chris Bosh

Attributes

Customer: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Register a Customer: Date of registration, Staff who registered, Customer who registered

Car out for Hire

Classes:

Symbol: Customer

Intension: The initiate actor is looking to hire a car from the company

Extension: e.g. Customers with names Tony Thompson, Layton Boyce, Jordan Smikle…

Symbol: Staff

Intension: The direct actor who registers and records the customer's car hire for the company

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Car

Intension: The car that is being hired by the customer from the company

Extension: e.g. Cars such as Citroen C1, Nissan Micra Activ, Honda Civic…

Symbol: Car Hired

Intension: The action of the customer hiring a car from the company

Extension: e.g. Layton Boyce hired the car Honda Civic from the company

Attributes:

Customer: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Car: Make, Model, Registration Number, Colour, Insurance Company, Breakdown Cover, Tax

Car Hired: Car Hire Date, Car Return Date

Car Return

Classes:

Symbol: Customer

Intension: The initiate actor is looking to return the car in which they hired from the company

Extension: e.g. Customers with names Tony Thompson, Layton Boyce, Jordan Smikle…

Symbol: Staff

Intension: The direct actor who records the return of the hired car from the customer

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Car

Intension: The car that is being returned from a hire by the customer

Extension: e.g. Cars such as Citroen C1, Nissan Micra Activ, Honda Civic…

Symbol: Car Return

Intension: The action of a customer returning a hired car back to the company

Extension: e.g. Jordan Smikle returns the hired car which is recorded by Dwayne Wade

Attributes:

Customer: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Car: Make, Model, Registration Number, Colour, Insurance Company, Breakdown Cover, Tax

Car Return: Date of Registration, Estimated return date, Actual Return Date, Mileage, Amount of Payment

Record a Completed Hire

Classes:

Symbol: Customer

Intension: The initiate actor has returned the car in which they hired from the company

Extension: e.g. Customers with names Tony Thompson, Layton Boyce, Jordan Smikle…

Symbol: Staff

Intension: The direct actor records the return of the hired car from the customer

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Car

Intension: The car that is being returned from a hire by the customer

Extension: e.g. Cars such as Citroen C1, Nissan Micra Activ, Honda Civic…

Symbol: Record Completed Hire

Intension: The action of a customer returning a hired car back to the company

Extension: e.g. Jordan Smikle returns the hired car which is recorded by Dwayne Wade

Attributes:

Customer: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Car: Make, Model, Registration Number, Colour, Hire Class, Insurance Company, Breakdown Cover, Tax

Record Completed Hire: Date of Registration, Estimated return date, Actual Return Date, Mileage, Amount of Payment

Servicing

Classes:

Symbol: Staff

Intension: The direct actor designates a mechanic to perform a service on one of the company's returned cars

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Mechanic

Intension: The initiate actor performs the service on the cars

Extension: e.g. Mechanics with names James Jones, Franklin Johnson, Walter James…

Symbol: Car

Intension: The car that is being serviced by the mechanic

Extension: e.g. Cars such as Citroen C1, Nissan Micra Activ, Honda Civic…

Symbol: Car Service

Intension: The action of a car undergoing a service by the company's mechanic

Extension: e.g. Nissan Micra Activ undergoes a service performed by James Jones

Attributes:

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Mechanic: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

Car: Make, Model, Engine Capacity, Registration Number, Colour, Mileage, Date of Registration, Tax

Car Service: Date of service, Mechanic Responsible, Mileage at Service

Removal of Customer from Database

Classes:

Symbol: Customer

Intension: The initiate actor that is being removed from the company's database

Extension: e.g. Customers with names Tony Thompson, Layton Boyce, Jordan Smikle…

Symbol: Staff

Intension: The direct actor that is performing the deletion of the current customer from the company's system

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Customer Removed from Database

Intension: The action of a customer being removed from the company's database via a member of staff

Extension: Member of staff Lebron James removes customer Tony Thompson from the company's database

Attributes:

Customer: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Customer Removed from Database: Date record was removed, Staff responsible, Customer Name

Add Car

Classes:

Symbol: Car

Intension: The car that is being added to the company's fleet

Extension: e.g. Cars such as Citroen C1, Nissan Micra Activ, Honda Civic…

Symbol: Staff

Intension: The direct actor who is adding the new car's record to the company's database

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: New Car Addition

Intension: The action of a new car being added to the company's fleet by a member of staff

Extension: e.g. Dwayne Wade adds the new car Citroen C1 to the company's car fleet

Attributes:

Car: Make, Model, Engine Capacity, Hire Class, Date of Registration, Mileage, Insurance Company, Breakdown Cover, Tax

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

New Car Addition: Car addition Date, Previous owner, Previous Owner Details

Delete Car

Classes:

Symbol: Car

Intension: The car that is being removed from the company's fleet

Extension: e.g. Cars such as Citroen C1, Nissan Micra Activ, Honda Civic…

Symbol: Staff

Intension: The direct actor who is removing the car's record from the company's database

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Car Removal from Fleet

Intension: The action of car being removed from the company's fleet by a member of staff

Extension: e.g. Lebron James removes the car Honda Civic from the company's car fleet

Attributes:

Car: Make, Model, Engine Capacity, Hire Class, Date of Registration, Mileage, Insurance Company, Breakdown Cover, Tax

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Car Removal from Fleet: Car Removal Date, Reason for Removal, Car Removed

Add Mechanic

Classes:

Symbol: Mechanic

Intension: The new initiate actor whose details are being registered to the company's database

Extension: e.g. Mechanics with names James Jones, Franklin Johnson, Walter James…

Symbol: Staff

Intension: The direct actor who's inputting the new mechanic's details into the company's database

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Addition of a new Mechanic

Intension: The action of a member of staff adding a new mechanic's details to the company's database that has currently joined the business

Extension: e.g. Chris Bosh add the new mechanic Franklin Johnson

Attributes:

Mechanic: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Addition of a new Mechanic: Date of registration, Staff who registered, Mechanic new to company

Delete Former Mechanic

Classes:

Symbol: Mechanic

Intension: The initiate actor that is being removed from the company's database

Extension: e.g. Mechanics with names James Jones, Franklin Johnson, Walter James…

Symbol: Staff

Intension: The direct actor that is removing the former mechanic from the company's database

Extension: e.g. Staff with names Lebron James, Dwayne Wade, Chris Bosh…

Symbol: Removal of Former Mechanic

Intension: The action of a member of staff removing a former mechanic from the company's database

Extension: e.g. Dwayne Wade removes the former mechanic Walter James

Attributes:

Mechanic: Name, Mobile Number, Home Telephone Number, Address, Driving Licence Number

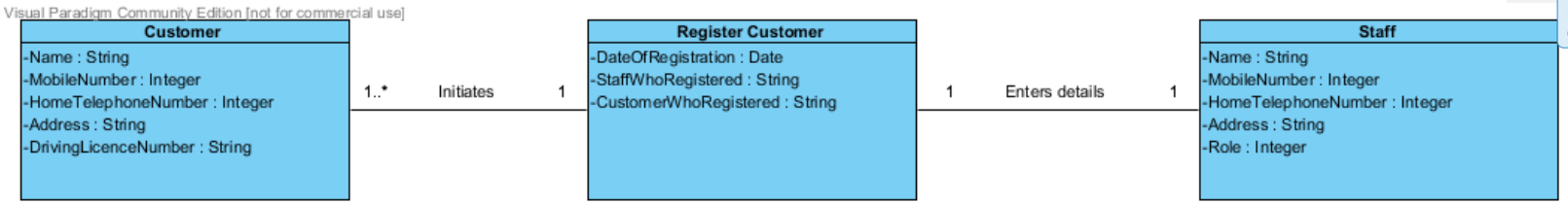
Staff: Name, Mobile Number, Home Telephone Number, Address, Role

Removal of Former Mechanic: Mechanic Removal Date, Reason for Removal, Mechanic Name

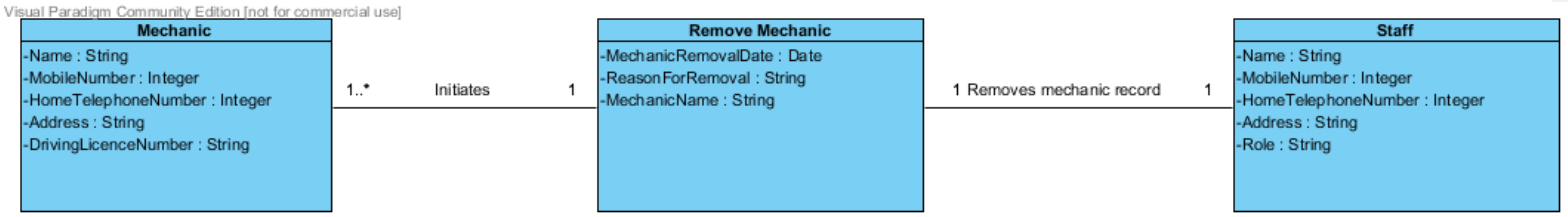
Conceptual Class Diagrams

批注 [ZL14]: In general, the overall class diagram is quite informative, but some important classes are missing, such as Bvis Company and Bvis System. This would lead to difficulties in later design, though it can be recovered when applying controller Pattern. However, in that case, conceptual class diagram should be modified too. Another problem is that many or the most of multiplicities of the associations are not right (though is is usually a hard problem for starters, especially those who are weak in mathematics)

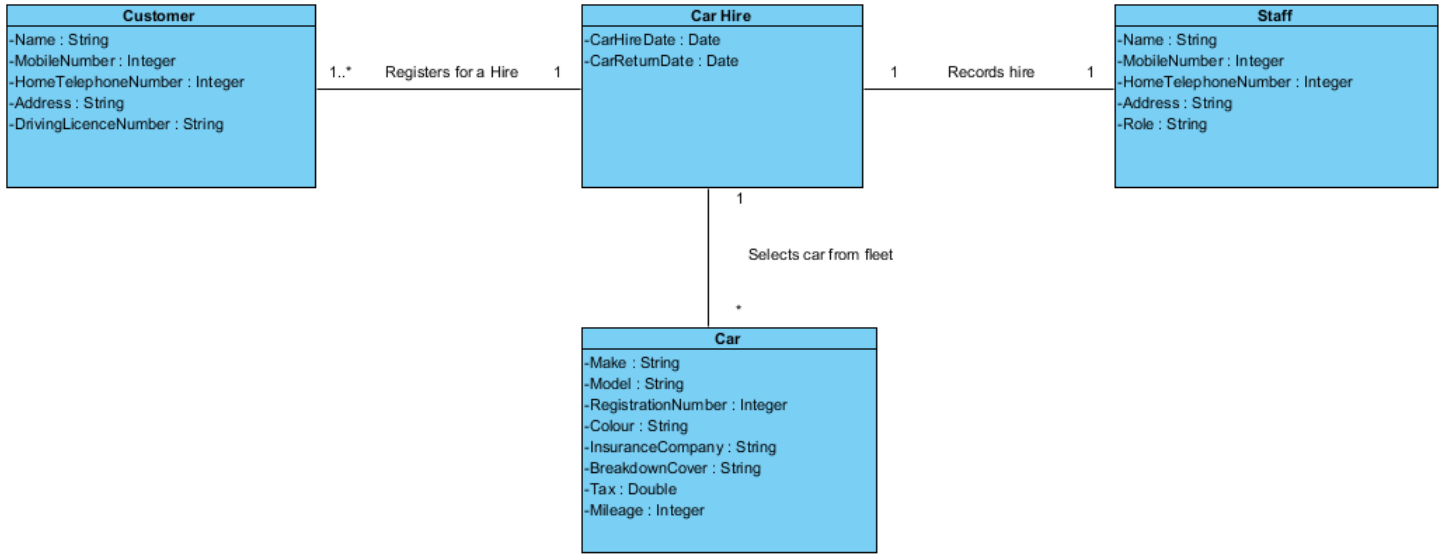
Registering a New Customer



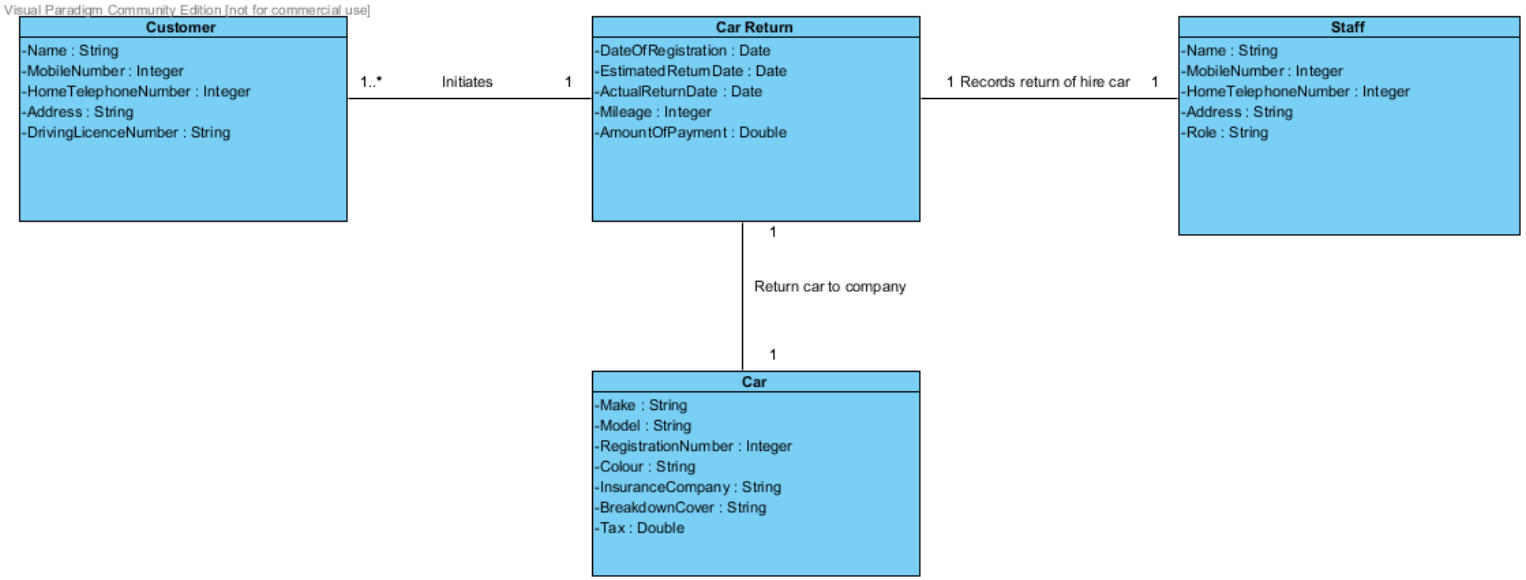
Delete Mechanic



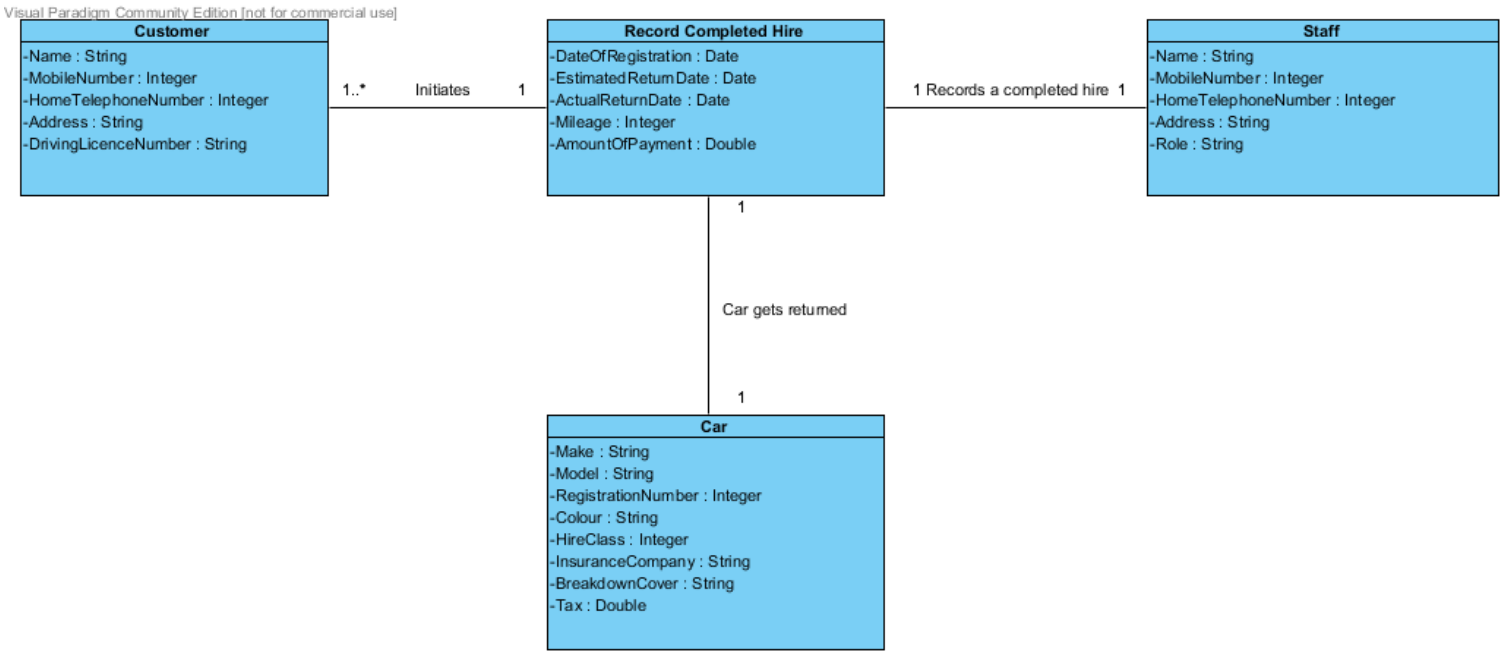
Car Our For Hire



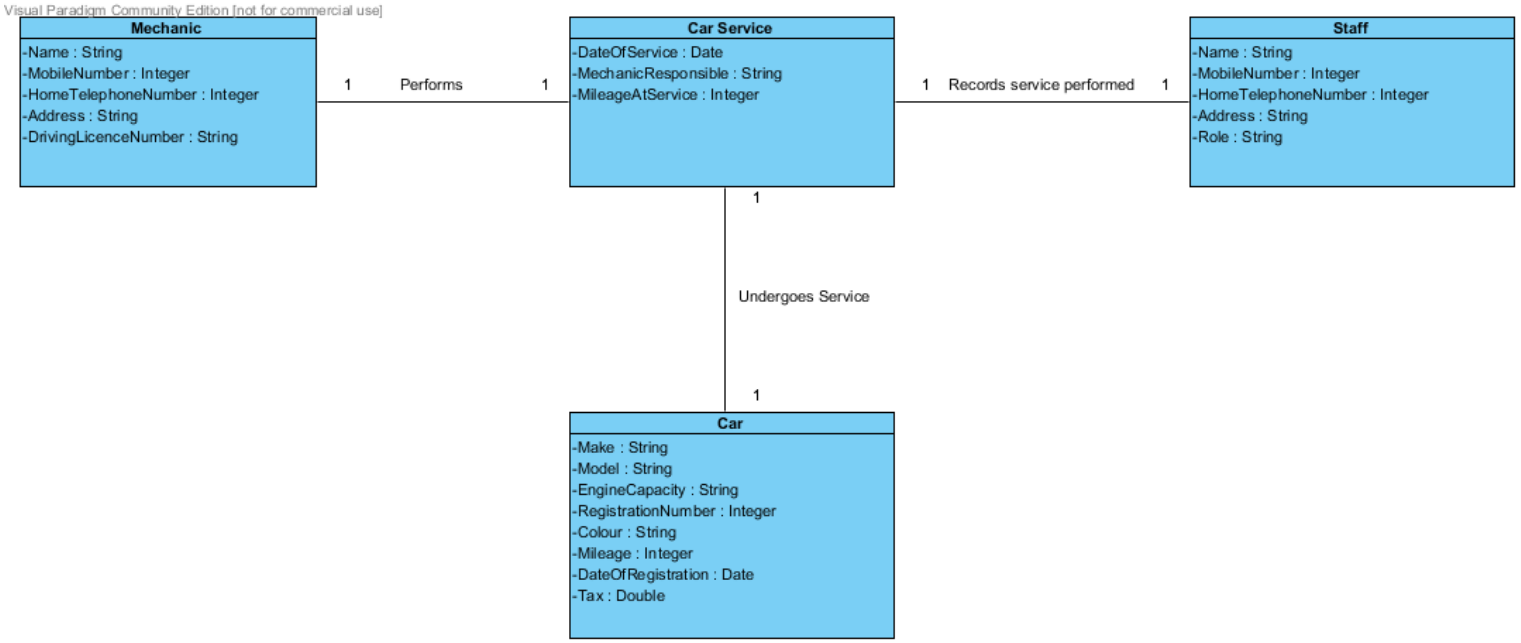
Car Return


Visual Paradigm Community Edition [not for commercial use]
Customer
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address : String
-DrivingLicenceNumber : String
1..*
Initiates
1
Car Return
-DateOfRegistration : Date
-EstimatedReturnDate : Date
-ActualReturnDate : Date
-Mileage : Integer
-AmountOfPayment : Double
1 Records return of hire car 1
Staff
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address : String
-Role : String
1
Return car to company
1
Car
-Make : String
-Model : String
-RegistrationNumber : Integer
-Colour : String
-InsuranceCompany : String
-BreakdownCover : String
-Tax : Double

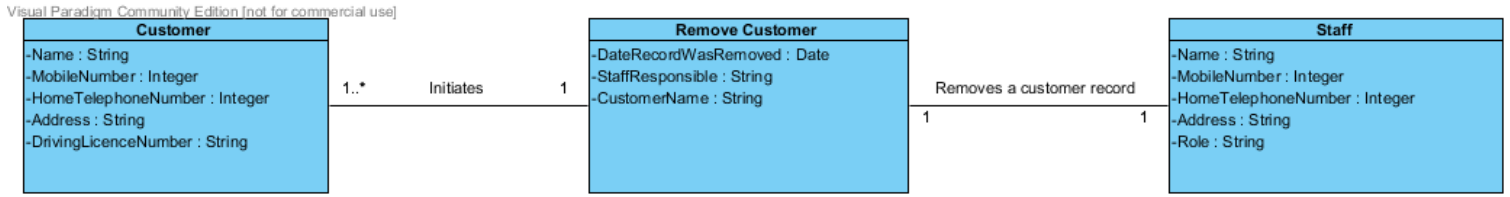
Record a Completed Hire



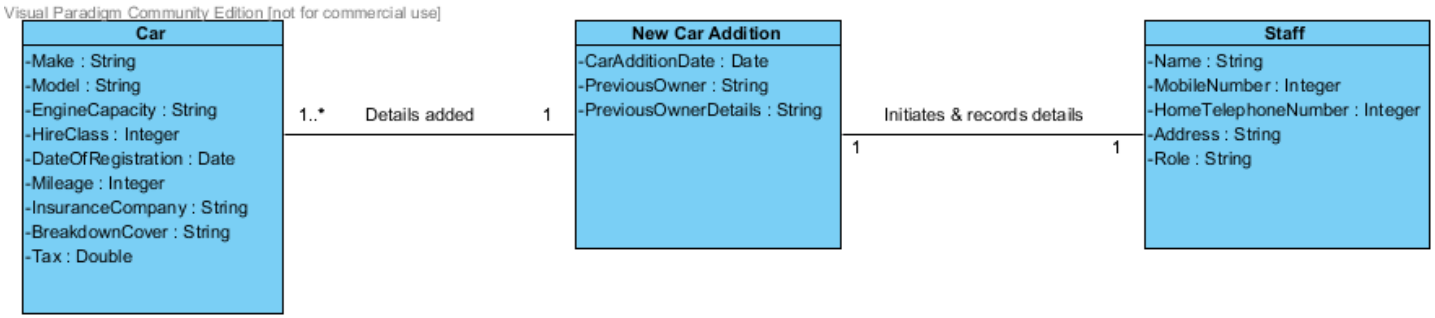
Servicing


Visual Paradigm Community Edition [not for commercial use]
Mechanic
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address : String
-DrivingLicenceNumber : String
1
Performs
1
Car Service
-DateOfService : Date
-MechanicResponsible : String
-MileageAtService : Integer
1
Records service performed
1
Staff
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address : String
-Role : String
1
Undergoes Service
1
Car
-Make : String
-Model : String
-EngineCapacity : String
-RegistrationNumber : Integer
-Colour : String
-Mileage : Integer
-DateOfRegistration : Date
-Tax : Double

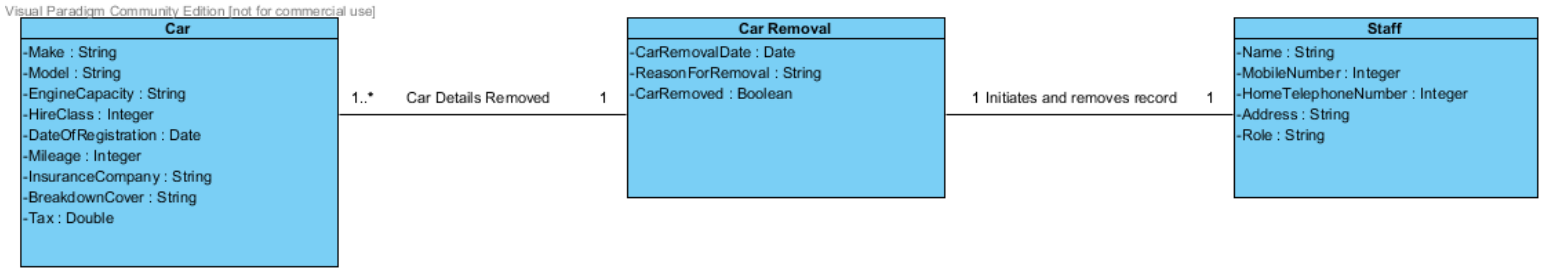
Removal of Customer from Database



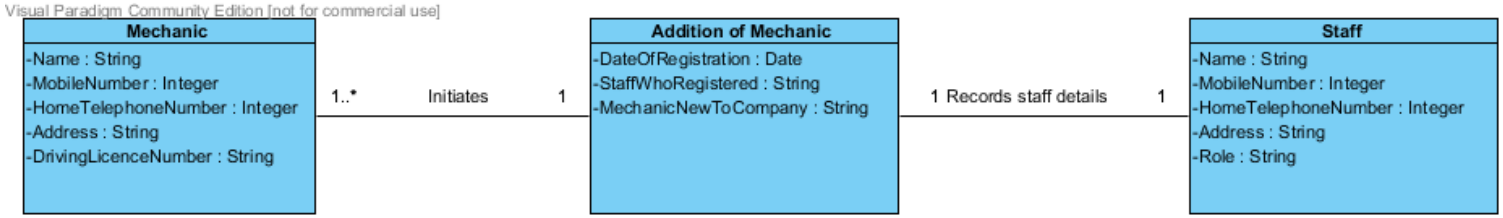
Add Car



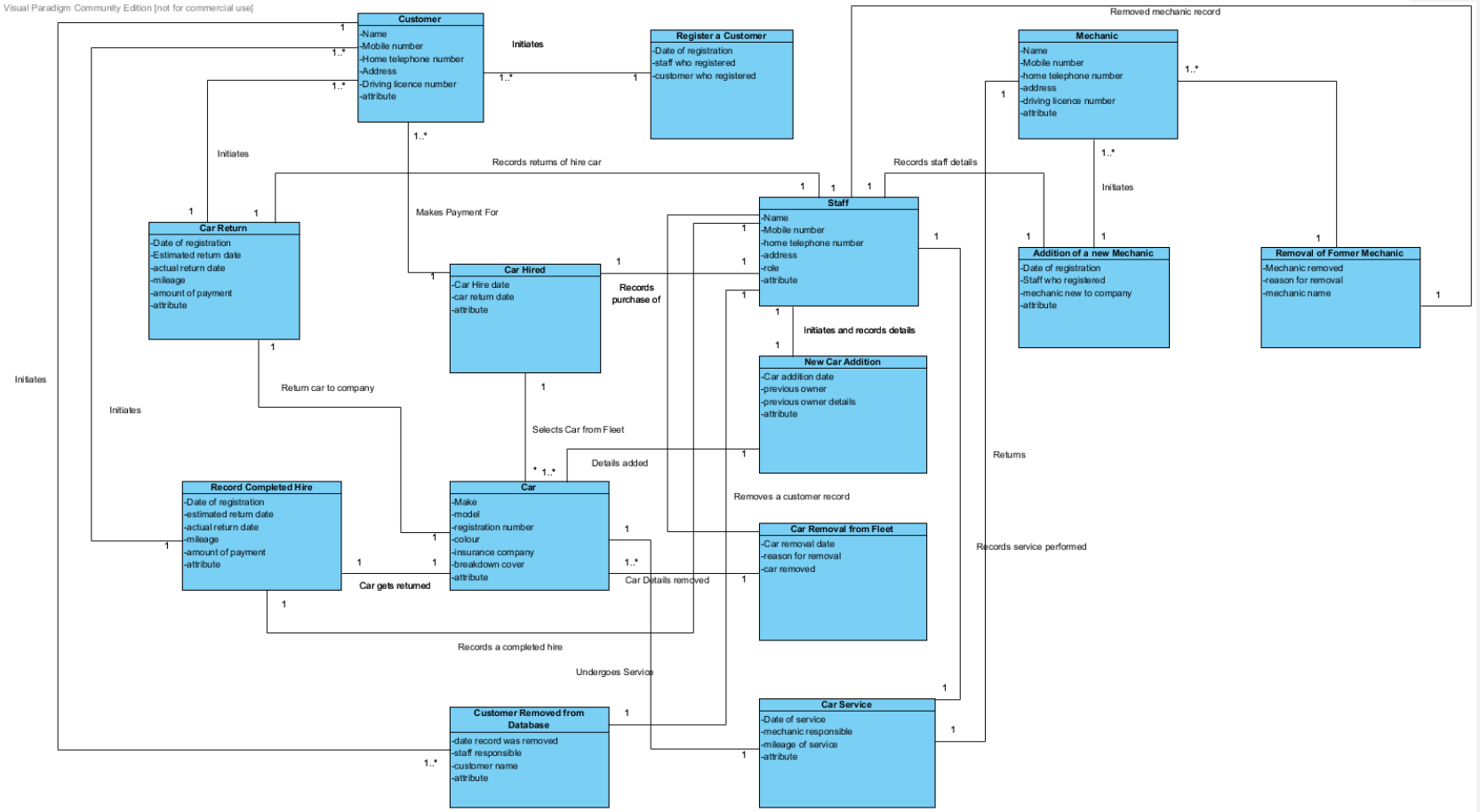
Delete Car



Add Mechanic



Conceptual Class Model



Section 2 – Analysis and Functionality of System Operations

System Operations & System Sequence Diagrams

System Operations

Register Customers

Input Events	Output Events
1. Customer arrives at cashier desk.	
2. Customer starts registrations	
3. Staff enters customers details	
4. Staff submits customer registrations	
	5. System confirms registrations
6. Staff confirms customers registrations	
7. Customer leaves cashier desk.	

批注 [ZL15]: No a system operation

批注 [ZL16]: Not a system operation

批注 [ZL17]: Not a system operation

批注 [ZL18]: What you are actually doing is repeating the typical course of events, and this is not needed

Car out for hire

Inputs	Outputs
1. Customer arrive at cash desk	
2. Staff check customer details on system	
	3. System then display customer details to staff
4. Customer then select a car	
5. Staff check availability of car selected by customer on system	
	6. System confirms car availability
7. Staff informs customer about the car	
8. Staff records hired car details on system	
9. Customer leaves cash desk	

Car return

Input events	**Output events**
1. Customer arrives at cash desk	
2. Customer shows their details to staff	
3. Staff check customer information on system	
	4. System display customer details to staff
5. Staff inputs car details to system	
	6. System check registration of hire car

Record completed hire

Input events	**Output events**
1. Customer arrives at cash desk	
2. Customer then give their details to staff	
3. Staff then inputs information on the system	
	4. System display customer information to staff
5. Staff informs customer about car return	
6. Customer pays amount for hire car	7. System prints receipt for staff
8. Customer leaves money on desk	

Servicing

Input events	Output events
1. Mechanics sends car information record to system	
	2. System then display car record to staff
3. Staff then records mileage information which is sent to the system	
4. Staff then picks mechanic	
5. Staff then records details on the system	

Remove Customer

Input Events	Output Events
1. Staff searches for existing customer on system	
	2. System returns existing customers records
3. Staff removes existing customers record	
	4. System confirms customer removal
5. Staff informs customer of their removal	
	6. System sends email to customer of removal

Add Car

Input events	**Output events**
1. Staff initiate a new car to add	
2. Staff then inputs new car	
3. Staff then verifies the car details on system	
	4. Systerm then returns the confirmation to staff
5. Staff then notify manager of the addition of the car	
	6. System then sends confirmation email to manager

Delete car

Input Events	Output Events
1. Staff initiate car removal	
2. Staff searches for car details on system	
3. Staff verify car details	
	4. System displays cars verified details
5. Staff removes details from system	
	6. System removes car details
	7. System sends removal notification to staff
8. Staff notifies management on car removal	
	9. System emails management of cars removal

Add mechanic

Input events	**Output events**
1. Mechanic initiate themselves to the system	
2. Staff enters mechanic details	
3. Staff submits mechanics details to system	
	4. System confirm mechanic registration
5. Staff shows confirmation to mechanic	
	6. System shows confirmed registration to mechanic

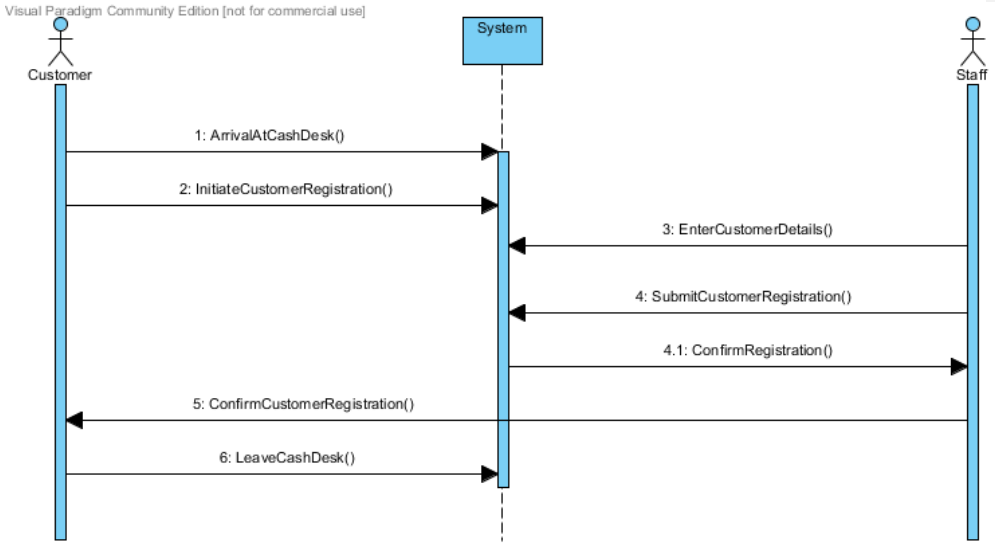
Delete mechanic

Input Events	Output Events
1. Staff initiate removal of mechanic	
2. Staff search for existing mechanic	
	3. The system returns existing mechanic data
4. Staff then deletes mechanics record	
	5. The system confirms mechanics removal
6. Staff notifies mechanic on removal	
7. Staff informs mechanic of removal	
	8. The system send email to mechanics of

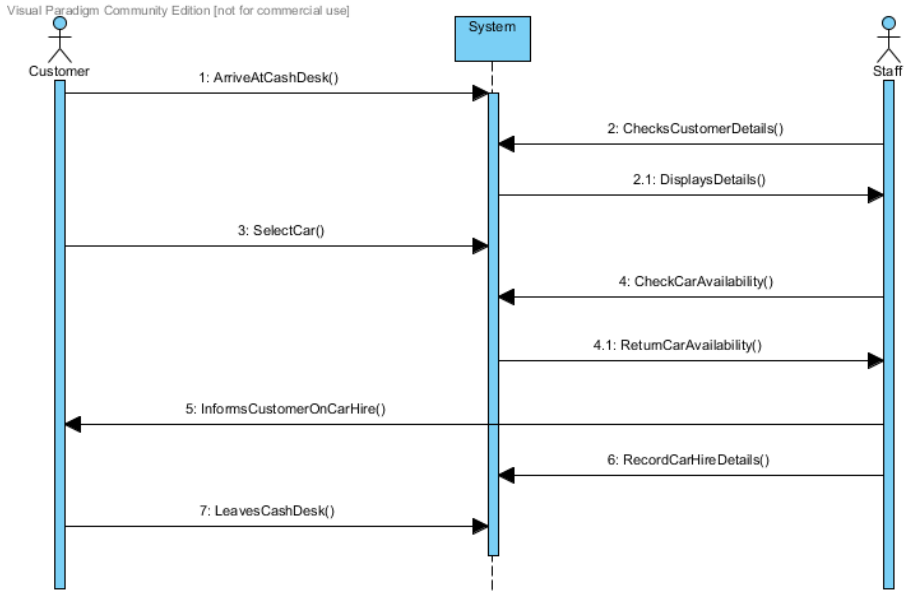
System Sequence Diagrams

Register a new Customer

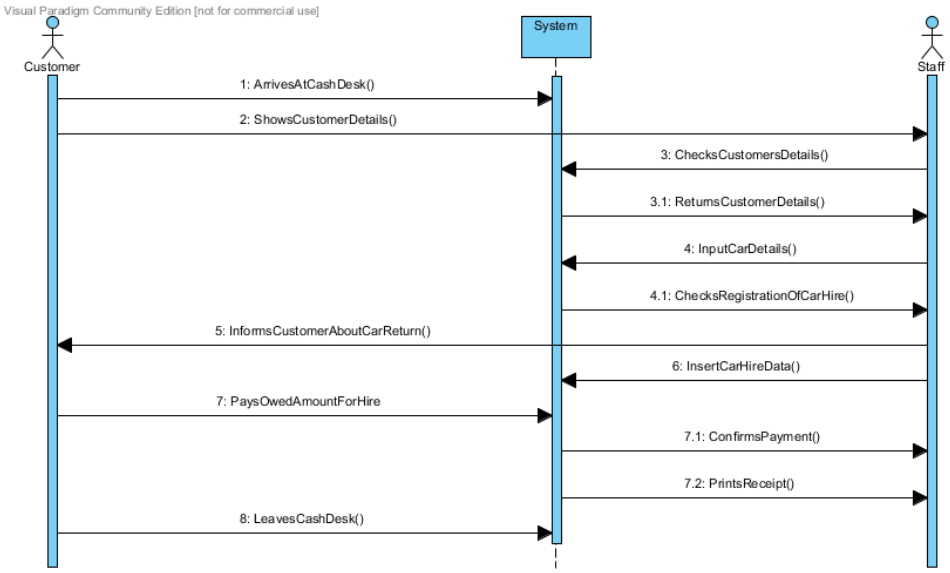
批注 [ZL19]: Message1, 1,2,4.1,5,6 should not be in the use sequence diagram



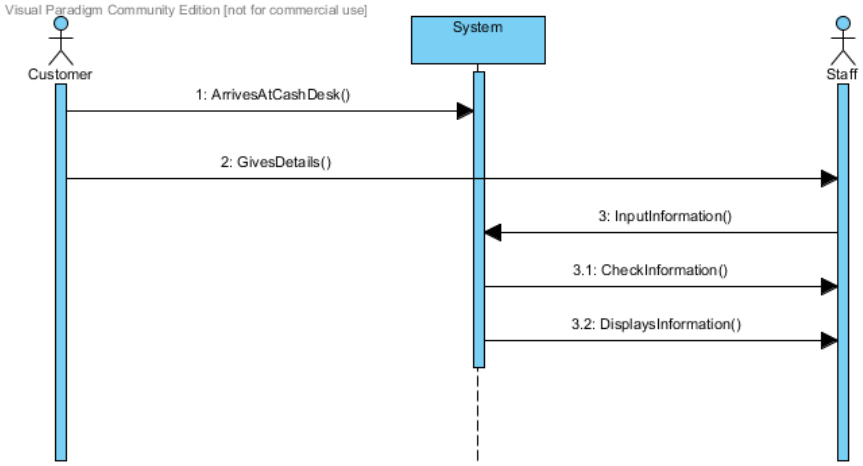
Car Out For Hire



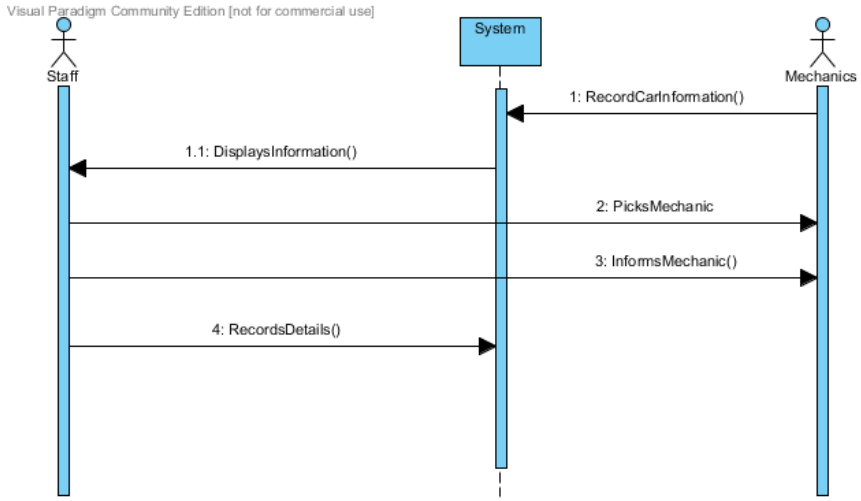
Car Return



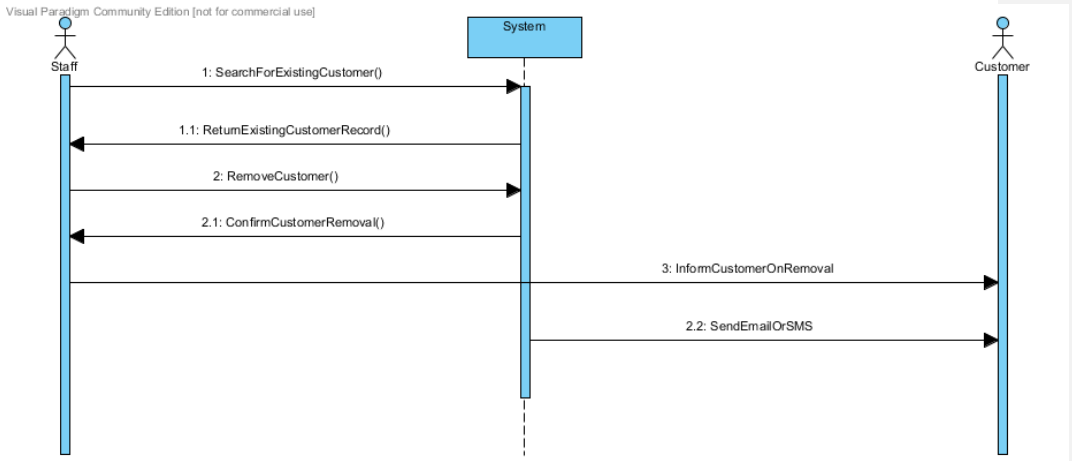
Record a Completed Hire



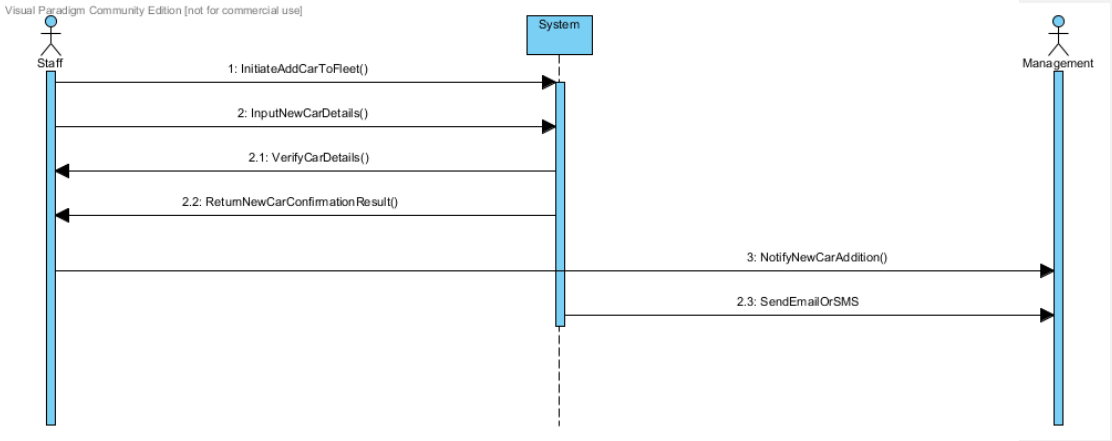
Servicing



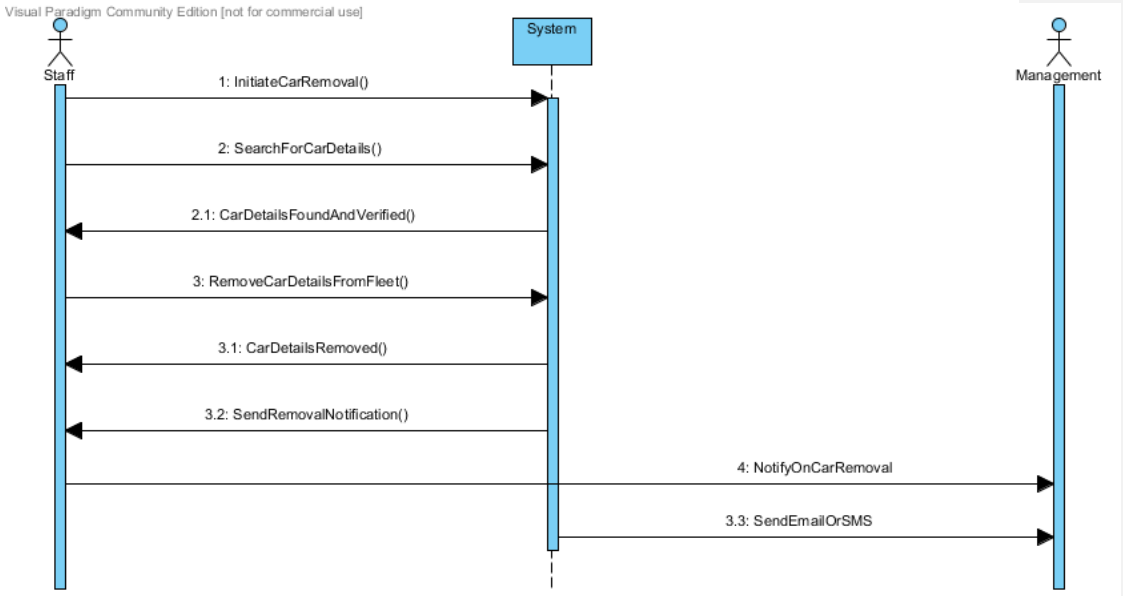
Removal of Customer from Database



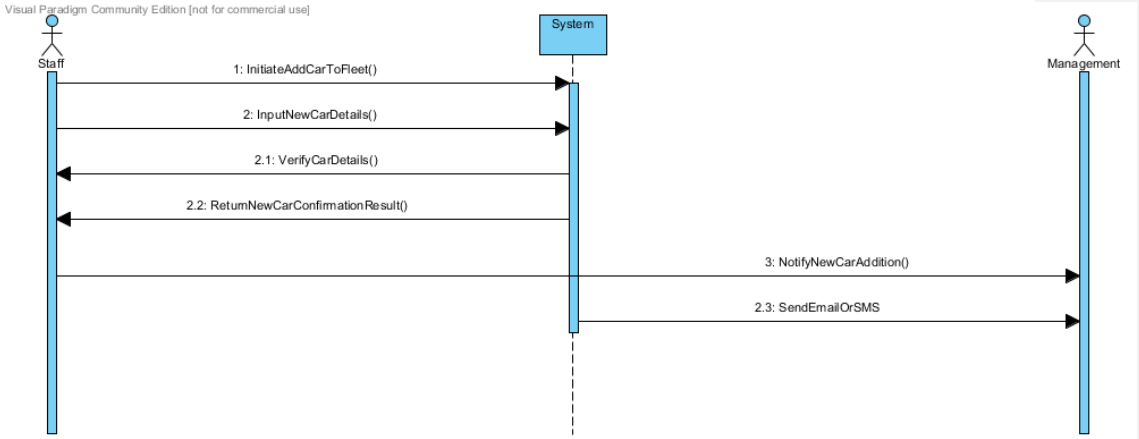
Add Car



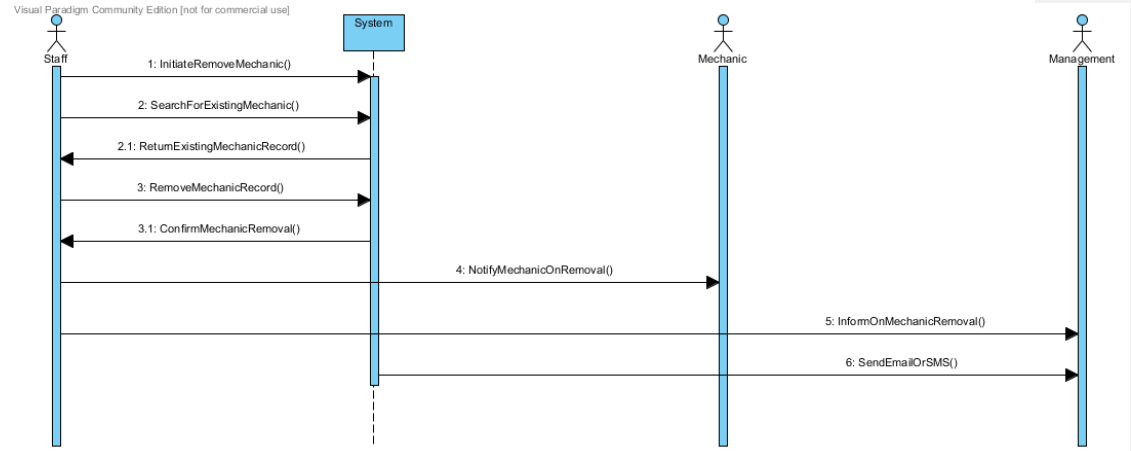
Delete Car



Add Mechanic



Delete Mechanic



批注 [ZL20]: Good understanding of the syntax (format of use case sequence diagrams) , but weak understanding of the abstraction needed. These sequence diagrams are useful, but also confusing if not cleared up before moving to the next step of the development

System Operations Contracts

Name: StartUp()

Responsibilities: Initialises/ boots up the system.

Post-conditions:

- A MemberOfStaff, Garage, CashDesk, CarList, ServiceList and ServiceSpecification were created
- ServiceSpecification was associated with ServiceList
- ServiceList was associated with Garage
- CashDesk was associated with Garage
- ServiceList was associated with CashDesk
- ServiceSpecification was associated with CarList

批注 [ZL21]: Most of these objects have no classes in the class diagram. I do understand why you have this problem here.

批注 [ZL22]: No class in the class diagram

批注 [ZL23]: You do not have CashDesk in the class diagrm

Register Customer

Name: ArrivalAtCashDesk()

Responsibilities: The customer arrives at the cash desk, which initiates the operation by communicating with the MemberOfStaff.

Type: Concept

Cross-Reference: System Sequence Diagrams: Register Customer 1

Post-conditions: Customer begins InitiateCustomerRegistration() contract.

批注 [ZL24]: This is not a system operation. Thus, no contract should be defined for it.

Name: InitiateCustomerRegistration()

Responsibilities: Customer informs cashier that they want to be registered with the company.

Type: Concept

Cross-Reference: System Sequence Diagrams: Register Customer 2

Pre-conditions: ArrivalAtCashDesk() has been completed

Post-conditions: MemberOfStaff begins EnterCustomerDetails() contracts

Name: EnterCustomerDetails()

Responsibilities: Member of Staff inputs in the new customer's details into the system to register with the company.

Type: Interface

Cross-Reference: System Sequence Diagrams: Register Customer 3

Pre-conditions: MemberOfStaff must enter in the customer's details including: Name, Mobile Number, Home Telephone Number, Address, and Driving Licence Number

Post-conditions:

- RegisterCustomer was selected from ServiceList
- Details entered by MemberOfStaff used for SubmitCustomerRegistration() contract

Name: SubmitCustomerRegistration()

Responsibilities: MemberOfStaff sends the customer's details through to the system to complete the registration process

Type: System

Cross-Reference: System Sequence Diagrams: Register Customer 4

Pre-conditions: EnterCustomerDetails() has been completed

Post-conditions:

- RegisterCustomer was selected from ServiceList
- Details entered by MemberOfStaff used for SubmitCustomerRegistration() contract

Name: ConfirmRegistration()

Responsibilities: The MemberOfStaff will receive confirmation that the new customer has been registered with the company

Type: Interface

Cross-Reference: System Sequence Diagrams: Register Customer 4.1

Output: MemberOfStaff receives confirmation that the new customer's details have been registered with the company

Pre-conditions: SubmitCustomerRegistration() has been completed

Post-conditions: RegisterCustomer was de-selected from the ServiceList

Name: ConfirmCustomerRegistration()

Responsibilities: Customer receives confirmation that they have been registered with the company

Type: Interface

Cross-Reference: System Sequence Diagrams: Register Customer 5

Output: Customer receives confirmation that their details have been registered with the company

Pre-conditions: ConfirmRegistration() has been completed

Post-conditions: RegisterCustomer was de-selected from the ServiceList

Name: LeaveCashDesk()

Responsibilities: Customer leaves the cash desk/ building or initiates another process

Type: Concept

Cross-Reference: System Sequence Diagrams: Register Customer 6

Output: Customer leaves the cash desk or they inform the member of staff on another query

Pre-conditions: ConfirmCustomerRegistration() has been completed

Car Out For Hire

Name: ArriveAtCashDesk()

Responsibilities: Customer informs member of staff that they wish to hire a car from the company

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Out For Hire 1

Post-conditions: MemberOfStaff begins ChecksCustomerDetails() contract

Name: ChecksCustomerDetails()

Responsibilities: MemberOfStaff inputs customer details into the system to check if they have already been registered with the company

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Out For Hire 2

Exceptions: Customer's details do not match any rows within the company's database

Pre-conditions: ArriveAtCashDesk has been completed

Post-conditions:

- If customer exists, Begin DisplaysDetails() contract
- If the customer's details do not display, MemberOfStaff will begin the function: Register Customer

Name: DisplaysDetails()

Responsibilities: MemberOfStaff receives confirmation that the customer exists from the system pulling up the customer's record

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Out For Hire 2.1

Output: MemberOfStaff receives confirmation of the customer's record existing in the database

Pre-conditions: ChecksCustomerDetails() has been completed

Post-conditions: Customer can now begin SelectCar() contract

Name: SelectCar()

Responsibilities: The customer selects an available car from the company's garage for them to hire

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Out For Hire 3

Exceptions: The make and model of the car that the customer chooses is not available or identified on the database

Pre-conditions: DisplayDetails() has been completed

Post-conditions: Begin CheckCarAvailability() contract

Name: CheckCarAvailability()

Responsibilities: MemberOfStaff inputs the car details that the customer has chosen to hire to see if it is available

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Out For Hire 4

Exceptions: The make and model of the car that the customer chooses is not available or identified on the database

Pre-conditions: SelectCar() has been completed

Post-conditions:

- If car exists, begin ReturnCarAvailability() contract
- If the car's details do not display, then Customer will have to return back to the SelectCar() contract

Name: ReturnCarAvailability()

Responsibilities: The system will verify that the make and model of the car that the customer wants to hire exists, and the number of available vehicles there are

Type: Interface and System

Cross-Reference: System Sequence Diagrams: Car Out For Hire 4.1

Output: Returns the car's make and model that the customer wants, as well as the amount available in the garage

Pre-conditions: CheckCarAvailability() has been completed

Post-conditions:

- If car's make and model is found then continue onto InformCustomerOnHire() contract
- If the car's details do not display, then Customer will have to return back to the SelectCar() contract

Name: InformCustomerOnHire()

Responsibilities: MemberOfStaff informs the customer that they are now able to hire their desired car from the company

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Out For Hire 5

Pre-conditions: ReturnCarAvailability() has been completed

Post-conditions: MemberOfStaff begins RecordCarHireDetails() contract

Name: RecordCarHireDetails()

Responsibilities: MemberOfStaff inputs the car hire details from the customer into the company's database

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Out For Hire 6

Pre-conditions: InformCustomerOnHire() has been completed

Post-conditions: Customer begins LeavesCashDesk() contract

Name: LeavesCashDesk()

Responsibilities: The customer leaves the cash desk or starts another process by asking a query

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Out For Hire 7

Output: The customer either leaves the cash desk or starts another process by asking a query

Pre-conditions: RecordCarHireDetails() has been completed

Car Return

Name: ArrivesAtCashDesk()

Responsibilities: The customer arrives at the cash desk, stating that they wish to return their hired car

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Return 1

Post-conditions: Customer begins ShowsCustomerDetails() contract

Name: ShowsCustomerDetails()

Responsibilities: The customer shows their customer details, along with their car hire details to initiate the return

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Return 2

Pre-conditions: ArrivalAtCashDesk() has been completed

Post-conditions: MemberOfStaff begins ChecksCustomerDetails() contract

Name: ChecksCustomerDetails()

Responsibilities: MemberOfStaff inputs the customer's personal details as well as their car hire details to see if the hire has been recorded in the database on the day beginning of hire

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Return 3

Exceptions: The customer's car hire details do not match any company records on the database

Pre-conditions: ShowsCustomerDetails() has been completed

Post-conditions:

- If customer car hire details exist, begin ReturnsCustomerDetails() contract
- If the car hire details do not display, then end the function: Car Return, and tell them to register with the company

Name: ReturnsCustomerDetails()

Responsibilities: The system returns a result with the customer's car hire details, along with their personal details

Type: System

Cross-Reference: System Sequence Diagrams: Car Return 3.1

Output: Returns the customer's car hire details to the MemberOfStaff so they can confirm that the customer hired a car, so it can be returned

Pre-conditions: CheckCustomerDetails() has been completed

Post-conditions:

- If the customer's car hire details and their personal details exist, begin InputCarDetails() contract
- If they don't exist, then end the function: Car Return, and tell them to register with the company

Name: InputCarDetails()

Responsibilities: MemberOfStaff inputs the car details into the system to verify that the car hired was hired by the allocated customer

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Return 4

Pre-conditions: ReturnsCustomerDetails() has been completed

Post-conditions: Begins ChecksRegistrationOfCarHire() contract

Name: ChecksRegistrationOfCarHire()

Responsibilities: The system checks to see whether the car has been hired by the customer, so it can be returned, and then it displays the information to the MemberOfStaff

Type: System

Cross-Reference: System Sequence Diagrams: Car Return 4.1

Output: The system will output the car's details showing that it was hired by the designated customer, and that it is ready to be returned

Pre-conditions: InputCarDetails() has been completed

Post-conditions:

- If the details of the car that was hired are correct, begin InformCustomerAboutCarReturn() contract
- If the details returned are not correct, return back to contract: InputCarDetails()

Name: InformCustomerAboutCarReturn()

Responsibilities: The MemberOfStaff informs the customer that their details are correct and that the car has now been returned

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Return 5

Output: MemberOfStaff informs customer that their details have been found, and now the car has been returned

Pre-conditions: ChecksRegistrationOfCarHire() has been completed

Post-conditions: MemberOfStaff begins InsertCarHireDetails() contract

Name: InsertCarHireDetails()

Responsibilities: MemberOfStaff inputs the final details of the customer's car hire onto the system, including the actual return date and the amount owed from the customer.

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Return 6

Pre-conditions: InformCustomerAboutCarReturn() has been completed

Post-conditions: Customer begins PaysOwedAmountForHire() contract

Name: PaysOwedAmountForHire()

Responsibilities: The customer confirms a method a payment for the car hire service as well as the total cost, taking into account the daily hire rate for the car for the total cost

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Return 7

Pre-conditions: InsertCarHireDetails() has been completed

Post-conditions: System begins ConfirmsPayment() contract

Name: ConfirmsPayment()

Responsibilities: The customer confirms a method a payment for the car hire service as well as the total cost, taking into account the daily hire rate for the car for the total cost

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Return 7.1

Output: The customer receives confirmation that their payment for their car hire services has been completed, with their total amount information and hire details

Pre-conditions: PaysOwedAmountForHire() has been completed

Post-conditions: System begins PrintsReceipt() contract once customer payment and confirmation is complete

Name: PrintsReceipt()

Responsibilities: The system prints a receipt for the MemberOfStaff to give to the customer as evidence of customer payment and confirmation

Type: Interface

Cross-Reference: System Sequence Diagrams: Car Return 7.2

Output: A printed receipt is given to the MemberOfStaff to give to the customer

Post-conditions: ConfirmsPayment() has been completed

Pre-conditions: Customer begins LeavesCashDesk() contract, or selects another service from the company on request

Name: LeavesCashDesk()

Responsibilities: Customer leaves the cash desk, or selects another service upon request

Type: Concept

Cross-Reference: System Sequence Diagrams: Car Return 8

Output: Customer leaves the cash desk, or selects another service upon request

Pre-conditions: PrintsReceipt() has been completed

Record Completed Hire

Name: ArrivesAtCashDesk()

Responsibilities: The customer arrives at the cash desk, and then their car hire details are recorded

Type: Concept

Cross-Reference: System Sequence Diagrams: Record Completed Hire 1

Post-conditions: Customer begins GivesDetails() contract

Name: GivesDetails()

Responsibilities: The customer shows their customer details, so the MemberOfStaff can record their hire details

Type: Concept

Cross-References: System Sequence Diagrams: Record Completed Hire 2

Pre-conditions: ArrivesAtCashDesk() has been completed

Post-conditions: MemberOfStaff begins InputInformation() contract

Name: InputInformation()

Responsibilities: MemberOfStaff inputs the customer's car hire details into the system, for it to be checked and recorded

Type: Interface

Cross-Reference: System Sequence Diagrams: Record Completed Hire 3

Pre-conditions: GivesDetails() has been completed

Post-conditions: Begins CheckInformation() contract

Name: CheckInformation()

Responsibilities: The system checks to see whether the car hire information has been successfully entered into the system

Type: System

Cross-Reference: System Sequence Diagrams: Record Completed Hire 3.1

Exceptions: The system does not show up with the customer's car hire details

Pre-conditions: InputInformation() has been completed

Post-conditions: Begins DisplaysInformation() contract

Name: DisplaysInformation()

Responsibilities: The system returns the customer's car hire information to the MemberOfStaff

Type: Interface

Cross-Reference: System Sequence Diagrams: Record Completed Hire 3.2

Output: The system returns back the customer's car hire details to confirm that their details have been inputted into the system successfully

Pre-conditions: CheckInformation() has been completed

Servicing

Name: RecordCarInformation()

Responsibilities: MemberOfStaff records the car information when it is returned from a hire, including the mileage to see if it is ready for a service by the mechanic

Type: Interface

Cross-Reference: System Sequence Diagrams: Servicing 1

Post-conditions: Begin DisplaysInformation() contract

Name: DisplaysInformation()

Responsibilities: The system returns back the car information recorded by the member of staff, as confirmation that the record has been added before it is due for servicing

Type: Interface

Cross-Reference: System Sequence Diagrams: Servicing 1.1

Output: The MemberOfStaff is able to see the car hire details of the returned car before its service

Pre-conditions: RecordCarInformation() has been completed

Post-conditions: MemberOfStaff begins PicksMechanic() contract

Name: PicksMechanic()

Responsibilities: The MemberOfStaff picks the mechanic that is due to perform the service of a returned vehicle

Type: Concept

Cross-Reference: System Sequence Diagrams: Servicing 2

Output: The MemberOfStaff informs the mechanic that they have to perform a service on a designated vehicle that has been returned

Pre-conditions: DisplaysInformation() has been completed

Post-conditions: If the desired mechanic's details appear, begin InformsMechanic()

Name: InformsMechanic()

Responsibilities: The MemberOfStaff informs the designated mechanic that they will be performing a particular service for a car that has been returned

Type: Concept

Cross-Reference: System Sequence Diagrams: Servicing 3

Output: The MemberOfStaff informs the mechanic that they will be performing a service on a car

Pre-conditions: PicksMechanic() has been completed

Post-conditions: MemberOfStaff begins RecordDetails() contract

Name: RecordsDetails()

Responsibilities: The MemberOfStaff records the car service details on the system

Type: Interface

Cross-Reference: System Sequence Diagrams: Servicing 4

Pre-conditions: InformsMechanic() has been completed

Remove Customer

Name: SearchForExistingCustomer()

Responsibilities: The MemberOfStaff search for a customer's details on the company's database, by inputting the customer's details for them to remove it

Type: Interface

Cross-Reference: System Sequence Diagrams: Remove Customer 1

Post-conditions: Begin ReturnExistingCustomerRecord() contract

Name: ReturnExistingCustomerRecord()

Responsibilities: The system returns and displays back the desired customer record and displays the option for deletion

Type: System

Cross-Reference: System Sequence Diagrams: Remove Customer 1.1

Exception: If the system states that the customer that is getting deleted does not match any records on the system

Pre-conditions: SearchForExistingCustomer() has been completed

Post-conditions:

- If the system returns back the desired customer record, MemberOfStaff begins RemoveCustomer() contract
- If no record is found, return back to the contract: SearchForExistingCustomer()

Name: RemoveCustomer()

Responsibilities: The MemberOfStaff now selects the option to remove the desired customer's record from the company's database

Type: Interface

Cross-Reference: System Sequence Diagrams: Remove Customer 2

Pre-conditions: ReturnExistingCustomerRecord() has been completed

Post-conditions: Begin ConfirmCustomerRemoval() contract

Name: ConfirmCustomerRemoval()

Responsibilities: The system confirms to the MemberOfStaff that the designated customer has been removed from the system

Type: System

Cross-Reference: System Sequence Diagrams: Remove Customer 2.1

Output: The system displays confirmation to the MemberOfStaff that the desired customer details have been removed

Pre-conditions: RemoveCustomer() has been completed

Post-conditions: MemberOfStaff begins InformCustomerOnRemoval() contract

Name: InformCustomerOnRemoval()

Responsibilities: The MemberOfStaff informs the customer that their details have now been removed from the company's database

Type: Concept

Cross-Reference: System Sequence Diagrams: Remove Customer 3

Pre-conditions: ConfirmCustomerRemoval() has been completed

Post-conditions: Begin SendEmailOrSMS() contract

Name: SendEmailOrSMS()

Responsibilities: The system sends an email or an SMS to the customer to inform them that their details have been removed from the company's database

Type: System

Cross-Reference: System Sequence Diagrams: Remove Customer 2.2

Pre-conditions: InformCustomerOnRemoval() has been completed

Add Car

Name: InitiateAddCarToFleet()

Responsibilities: The MemberOfStaff begins the process of adding a new car to the fleet and to the database

Type: Concept

Cross-Reference: System Sequence Diagrams: Add Car 1

Post-conditions: MemberOfStaff begins InputNewCarDetails() contract

Name: InputNewCarDetails()

Responsibilities: The MemberOfStaff inputs the new car's details into the company's database

Type: Interface

Cross-Reference: System Sequence Diagrams: Add Car 2

Pre-conditions: InitiateAddCarToFleet() has been completed

Post-conditions: Begin VerifyCarDetails() contract

Name: VerifyCarDetails()

Responsibilities: The system verifies and checks whether the car details have been correctly entered

Type: System

Cross-Reference: System Sequence Diagrams: Add Car 2.1

Exception: The system shows that the car details entered are incorrect

Pre-conditions: InputNewCarDetails() has been completed

Post-conditions:

- If the car details have been correctly verified, begin ReturnNewCarConfirmationResult() contract
- If not, return to the contract: InputNewCarDetails()

Name: ReturnNewCarConfirmationResult()

Responsibilities: The system returns the new car's information to the MemberOfStaff to verify that the car's details have successfully been added

Type: System

Cross-References: System Sequence Diagrams: Add Car 2.2

Output: Displays the new car's details to the MemberOfStaff to verify that its details have successfully been added to the database

Pre-conditions: VerifyCarDetails() has been completed

Post-conditions: MemberOfStaff begins NotifyNewCarAddition()

Name: NotifyNewCarAddition()

Responsibilities: The MemberOfStaff informs management that a new car has been added to the fleet, and its details has been added to the company's database

Type: Concept

Cross-References: System Sequence Diagrams: Add Car 3

Pre-conditions: ReturnNewCarConfirmationResult() has been completed

Post-conditions: Begin SendEmailOrSMS() contract

Name: SendEmailOrSMS()

Responsibilities: The system sends an email or an SMS to the management team to inform them that a new car has been added to the fleet and its details have been added to the company's database

Type: System

Cross-Reference: System Sequence Diagrams: Add Car 2.3

Pre-conditions: NotifyNewCarAddition() has been completed

Delete Car

Name: InitiateCarRemoval()

Responsibilities: The MemberOfStaff starts the process of removing a car from the fleet and its details from the company's database

Type: Concept

Cross-Reference: System Sequence Diagrams: Delete Car 1

Post-conditions: MemberOfStaff begins SearchForCarDetails() contract

Name: SearchForCarDetails()

Responsibilities: The MemberOfStaff search for a car's details on the company's database, by inputting the car's details for them to remove it

Type: Interface

Cross-Reference: System Sequence Diagrams: Delete Car 2

Pre-conditions: InitiateCarRemoval() has been completed

Post-conditions: Begin CarDetailsFoundAndVerified() contract

Name: CarDetailsFoundAndVerified()

Responsibilities: The system returns and displays back the desired car record, it has been verified and displays the option for deletion

Type: System

Cross-Reference: System Sequence Diagrams: Delete Car 2.1

Exception: The car details are not found on the system

Pre-conditions: SearchForCarDetails() has been completed

Post-conditions:

- If the system returns back the desired car record, MemberOfStaff begins RemoveCarDetailsFromFleet() contract
- If no record is found, return back to the contract: SearchForCarDetails()

Name: RemoveCarDetailsFromFleet()

Responsibilities: The MemberOfStaff now selects the option to remove the desired car record from the company's database

Type: Interface

Cross-Reference: System Sequence Diagrams: Delete Car 3

Pre-conditions: CarDetailsFoundAndVerified() has been completed

Post-conditions: Begin CarDetailsRemoved() contract

Name: CarDetailsRemoved()

Responsibilities: The system removes the car's details from the system, as it is no longer in the fleet

Type: System

Cross-Reference: System Sequence Diagrams: Delete Car 3.1

Pre-conditions: RemoveCarDetailsFromFleet() has been completed

Post-conditions: Begin SendRemovalNotification() contract

Name: SendRemovalNotification()

Responsibilities: The system informs shows the MemberOfStaff that a car's details have now been removed from the company's database and from the fleet

Type: System

Cross-Reference: System Sequence Diagrams: Delete Car 3.2

Output: The system displays to the MemberOfStaff that the car's record has now been removed from the company's database

Pre-conditions: CarDetailsRemoved() has been completed

Post-conditions: MemberOfStaff begins NotifyOnCarRemoval() contract

Name: NotifyOnCarRemoval()

Responsibilities: The MemberOfStaff informs management that a car has been added from the fleet, and its details have been removed from the company's database

Type: Concept

Cross-Reference: System Sequence Diagrams: Delete Car 4

Pre-conditions: SendRemovalNotification() has been completed

Post-conditions: Begin SendEmailOrSMS() contract

Name: SendEmailOrSMS()

Responsibilities: The system sends an email or an SMS to the management team to inform them that a car has been removed from the fleet and its details have been removed from the company's database

Type: System

Cross-Reference: System Sequence Diagrams: Delete Car 3.3

Pre-conditions: NotifyOnCarRemoval() has been completed

Add Mechanic

Name: InitiateAddMechanic()

Responsibilities: The Mechanic begins the process by stating that they have joined the company, and their details need to be recorded

Type: Concept

Cross-Reference: System Sequence Diagrams: Add Mechanic 1

Post-conditions: MemberOfStaff begins EnterMechanicDetails() contract

Name: EnterMechanicDetails()

Responsibilities: The MemberOfStaff inputs the new Mechanic's personal details into the company's database

Type: Interface

Cross-Reference: System Sequence Diagrams: Add Mechanic 2

Pre-conditions: InitiateAddMechanic()

Post-conditions: MemberOfStaff begins SubmitMechanicDetails()

Name: SubmitMechanicDetails()

Responsibilities: MemberOfStaff sends the new mechanic's details through to the system

Type: Interface

Cross-Reference: System Sequence Diagrams: Add Mechanic 3

Pre-conditions: EnterMechanicDetails() has been completed

Post-conditions: Begin ConfirmMechanicRegistration() contract

Name: ConfirmMechanicRegistration()

Responsibilities: The system returns the new mechanic's information to the MemberOfStaff to verify that the mechanic's details have successfully been added

Type: System

Cross-References: System Sequence Diagrams: Add Mechanic 3.1

Output: Displays the new mechanic's details to the MemberOfStaff to verify that its details have successfully been added to the database

Pre-conditions: SubmitMechanicDetails() has been completed

Post-conditions: MemberOfStaff begins ConfirmMechanicRegistration() contract

Name: ConfirmMechanicRegistration()

Responsibilities: The MemberOfStaff informs the new mechanic that their details have been added to the company's database

Type: Concept

Cross-Reference: System Sequence Diagrams: Add Mechanic 4

Pre-conditions: ConfirmMechanicRegistration() has been completed

Post-conditions: Begin ConfirmAddedToSystem() contract

Name: ConfirmAddedToSystem()

Responsibilities: The system informs the mechanic that their new details have been added to the company's database, via the use of email of SMS

Type: System

Cross-Reference: System Sequence Diagrams: Add Mechanic 3.2

Pre-conditions: ConfirmMechanicRegistration() has been completed

Delete Mechanic

Name: InitiateRemoveMechanic()

Responsibilities: The MemberOfStaff starts the process of removing a former mechanic from the company's database

Type: Concept

Cross-Reference: System Sequence Diagrams: Delete Mechanic 1

Post-conditions: MemberOfStaff begins SearchForExistingMechanic() contract

Name: SearchForExistingMechanic()

Responsibilities: The MemberOfStaff search for the mechanic's details on the company's database, by inputting the mechanic's details for them to remove it

Type: Interface

Cross-Reference: System Sequence Diagrams: Delete Mechanic 2

Pre-conditions: InitiateRemoveMechanic() has been completed

Post-conditions: Begin ReturnExistingMechanicRecord() contract

Name: ReturnExistingMechanicRecord()

Responsibilities: The system returns and displays back the desired former mechanic record, it has been verified and displays the option for deletion

Type: System

Cross-Reference: System Sequence Diagrams: Delete Mechanic 2.1

Exception: The system doesn't find the mechanic's details on the database

Pre-conditions: SearchForExistingMechanic() has been completed

Post-conditions:

- If the system returns back the desired mechanic record, MemberOfStaff begins RemoveMechanicRecord() contract
- If no record is found, return back to the contract: SearchForExistingMechanic()

Name: RemoveMechanicRecord()

Responsibilities: The MemberOfStaff now selects the option to remove the desired mechanic record from the company's database

Type: Interface

Cross-Reference: System Sequence Diagrams: Delete Mechanic 3

Pre-conditions: ReturnExistingMechanicRecord() has been completed

Post-conditions: Begin ConfirmMechanicRemoval() contract

Name: ConfirmMechanicRemoval()

Responsibilities: The system removes the former mechanic's details from the system, as they are no longer an employee in the company

Type: System

Cross-Reference: System Sequence Diagrams: Delete Mechanic 3.1

Output: The system displays to the MemberOfStaff that the mechanic's record has now been removed from the company's database

Pre-conditions: RemoveMechanicRecord() has been completed

Post-conditions: MemberOfStaff begins NotifyMechanicOnRemoval() contract

Name: NotifyMechanicOnRemoval()

Responsibilities: The MemberOfStaff informs the former mechanic that their details have been removed from the company's database as they are no longer a part of the company

Type: Concept

Cross-Reference: System Sequence Diagrams: Delete Mechanic 4

Pre-conditions: ConfirmMechanicRemoval() has been completed

Post-conditions: MemberOfStaff begins InformOnMechanicRemoval() contract

Name: InformOnMechanicRemoval()

Responsibilities: The MemberOfStaff informs management that a former mechanic's details has been removed from the company's database

Type: Concept

Cross-Reference: System Sequence Diagrams: Delete Mechanic 5

Pre-conditions: NotifyMechanicOnRemoval() has been completed

Post-conditions: Begin SendEmailOrSMS() contract

Name: SendEmailOrSMS()

Responsibilities: The system sends an email or an SMS to the management team to inform them that a car has been removed from the fleet and its details have been removed from the company's database

Type: System

Cross-Reference: System Sequence Diagrams: Delete Mechanic 6

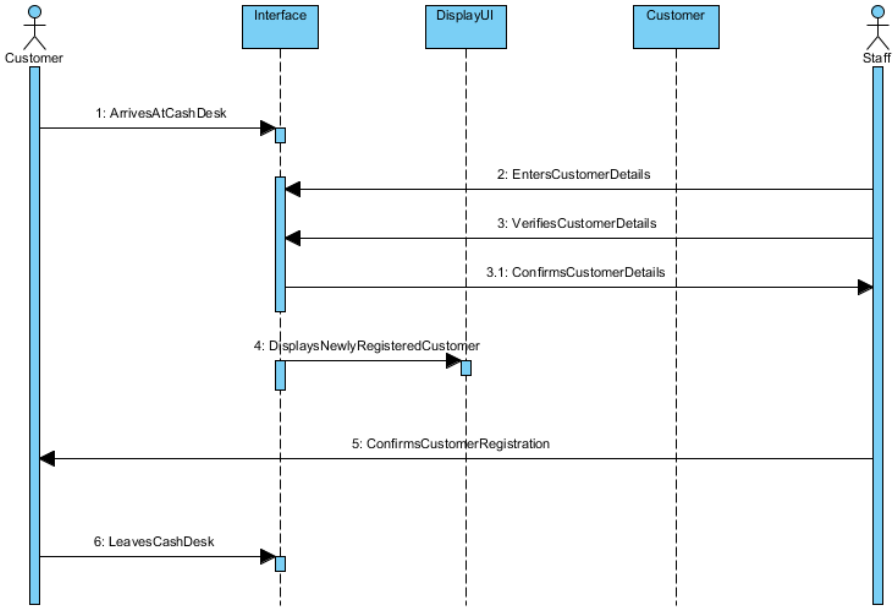
Pre-conditions: InformOnMechanicRemoval() has been completed

批注 [ZL25]: In general the report show a vague understanding about contracts and knows vaguely how to describe it. However, It also shows that the student do not understand their usage well. There is a weak understanding of the relation between contracts and the conceptual class diagram

Section 3 – Use Case Design

Object Sequence Diagrams

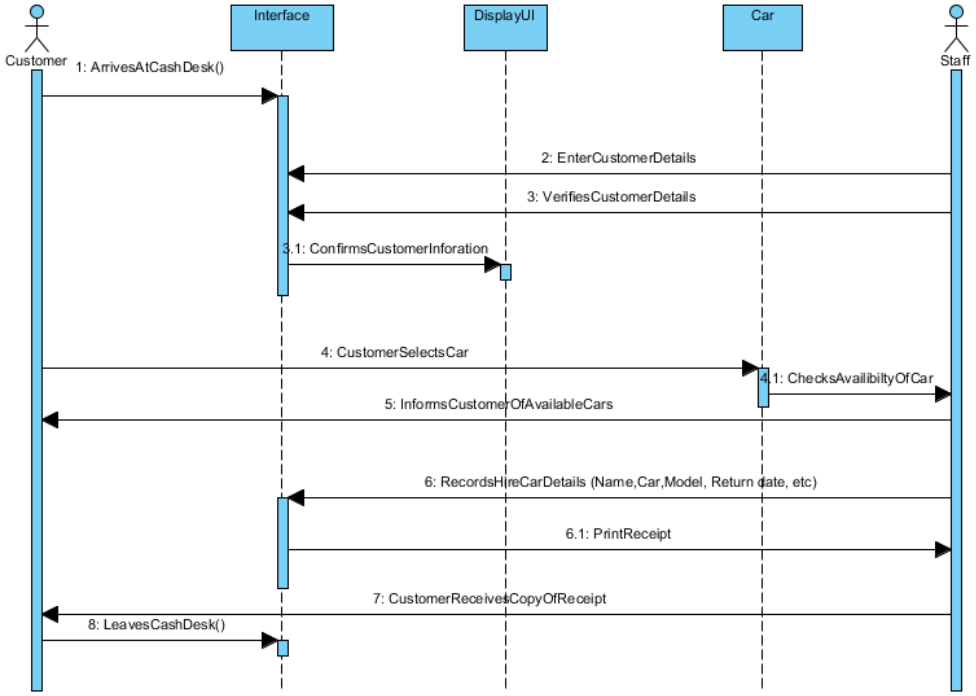
Registering a new Customer



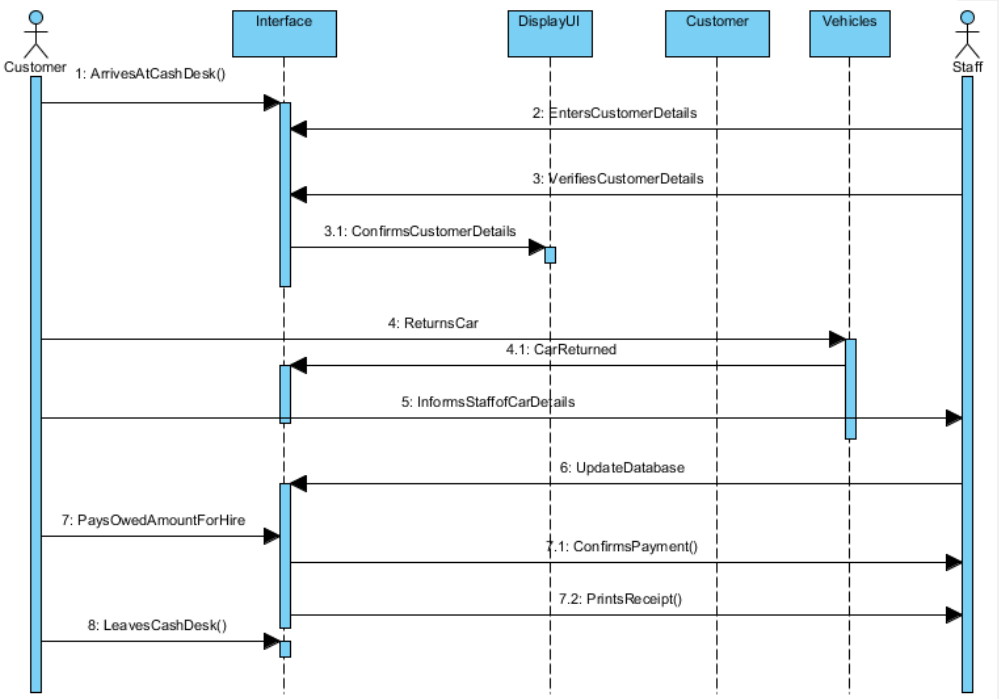
批注 [ZL26]: These diagrams show basic understanding of using sequence diagrams to describe interaction and message passing. However the student have a weak understanding of OO design, possibly because of the lack of OO programing experience.

The relation between object sequence diagrams and use case sequence diagrams , and the relation between object sequence diagrams conceptual diagrams are poor.

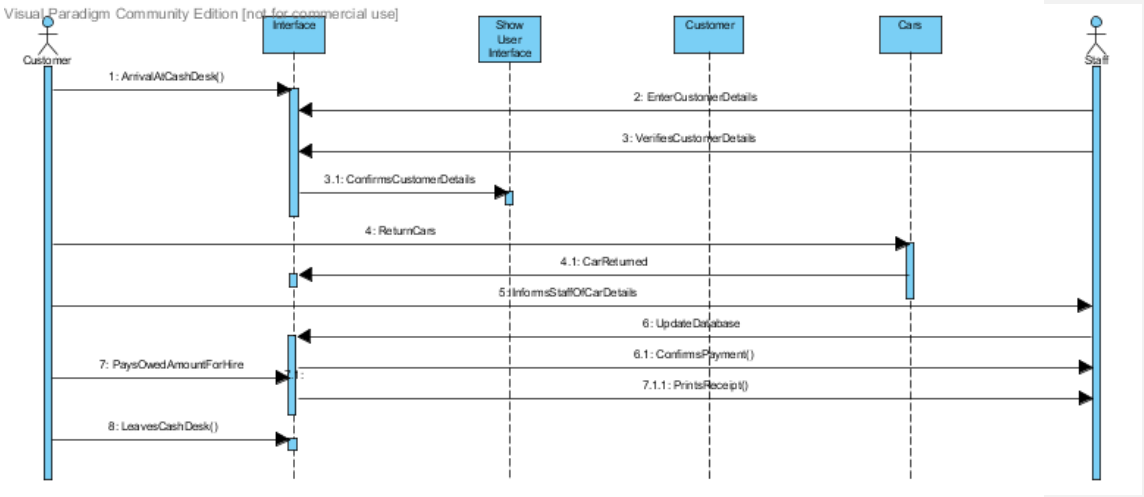
Car Out For Hire



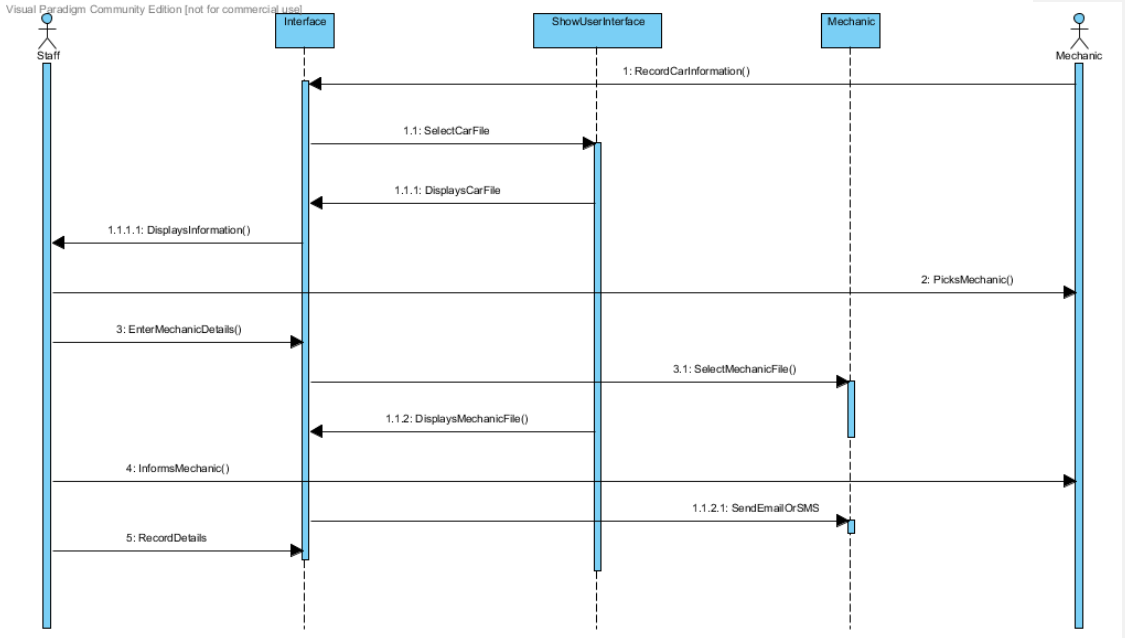
Car Return


Customer
Interface
DisplayUI
Customer
Vehicles
Staff
1: ArrivesAtCashDesk()
2: EntersCustomerDetails
3: VerifiesCustomerDetails
3.1: ConfirmsCustomerDetails
4: ReturnsCar
4.1: CarReturned
5: InformsStaffofCarDetails
6: UpdateDatabase
7: PaysOwedAmountForHire
7.1: ConfirmsPayment()
7.2: PrintsReceipt()
8: LeavesCashDesk()

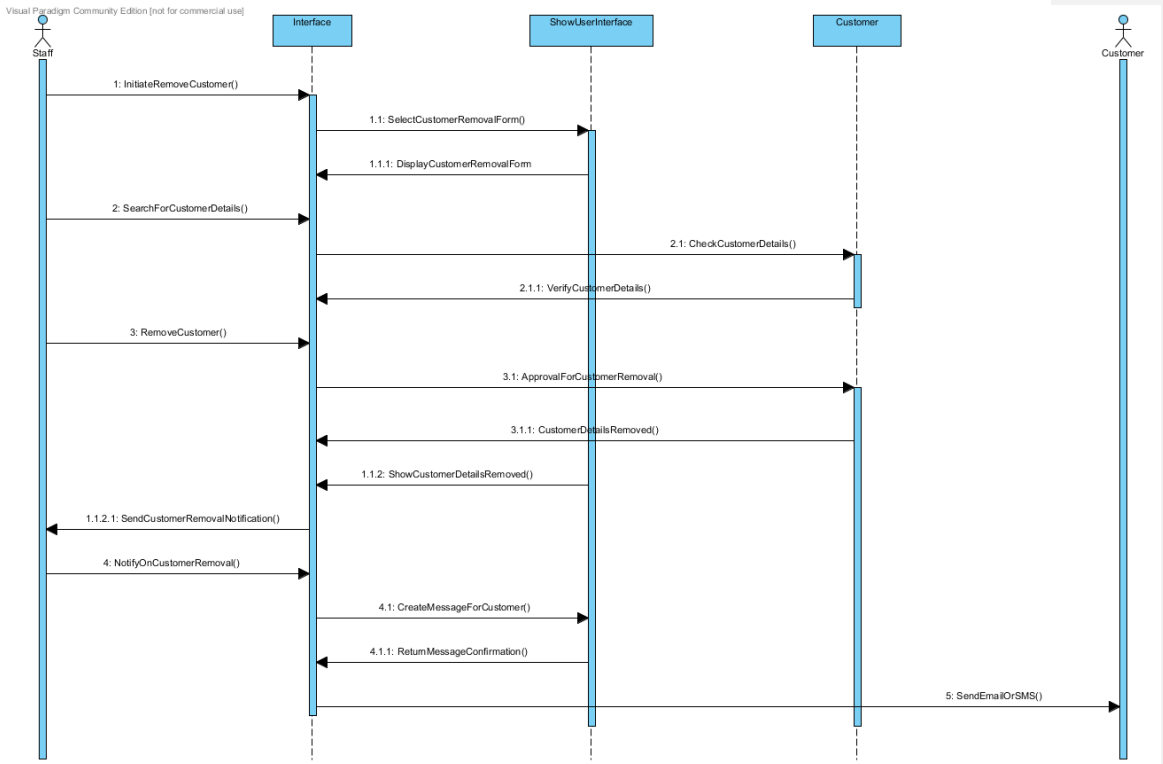
Record a Completed Hire



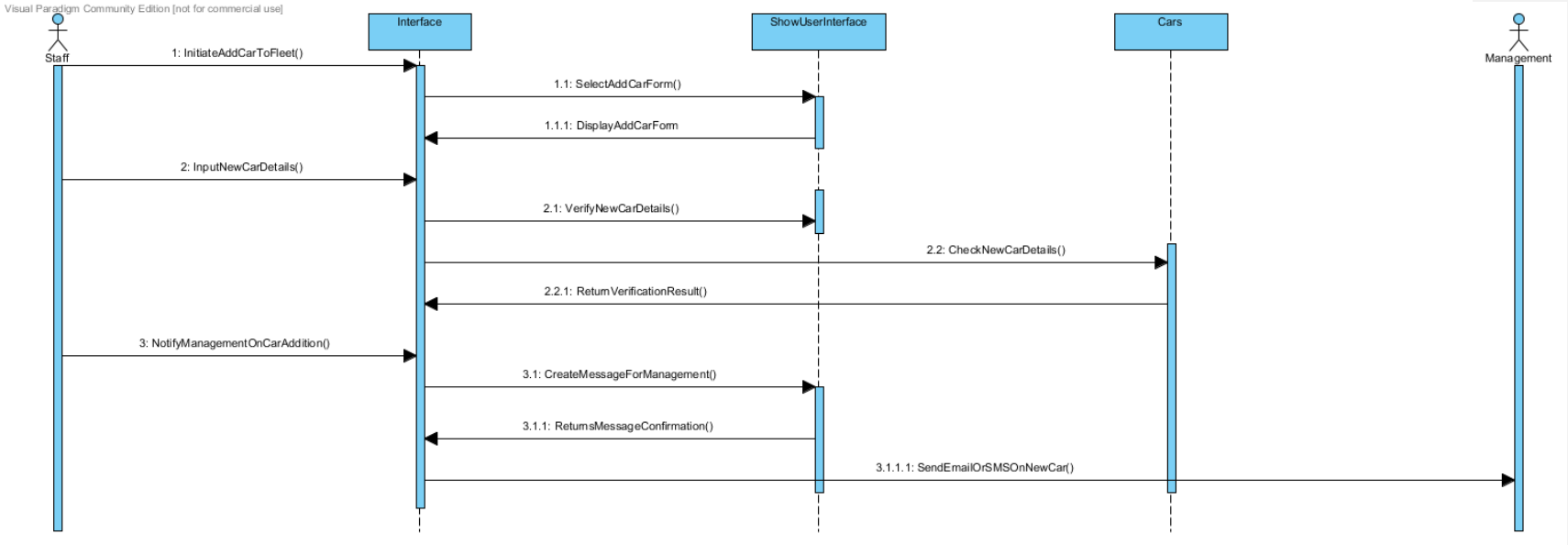
Servicing



Removal of Customer from Database

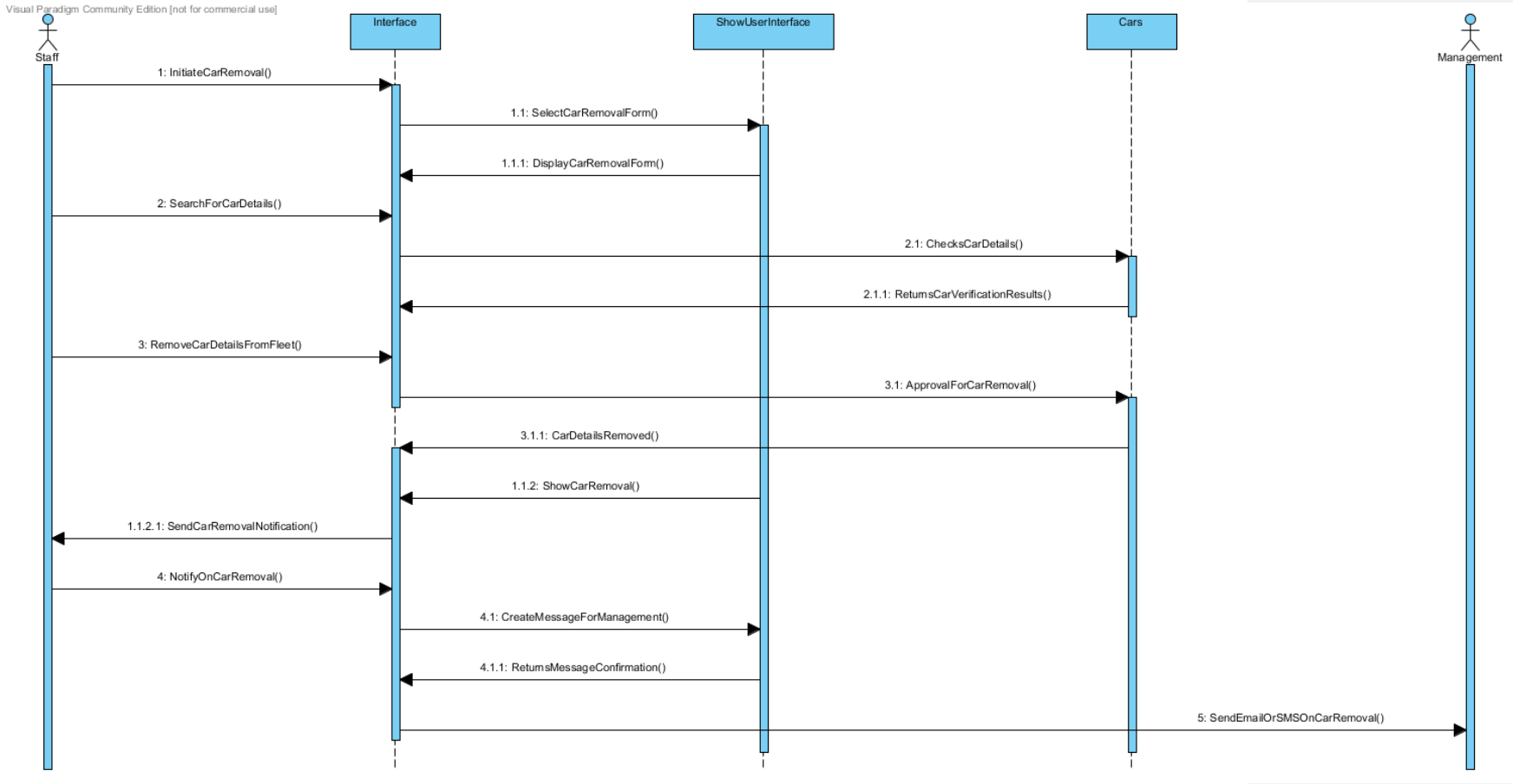


Add Car

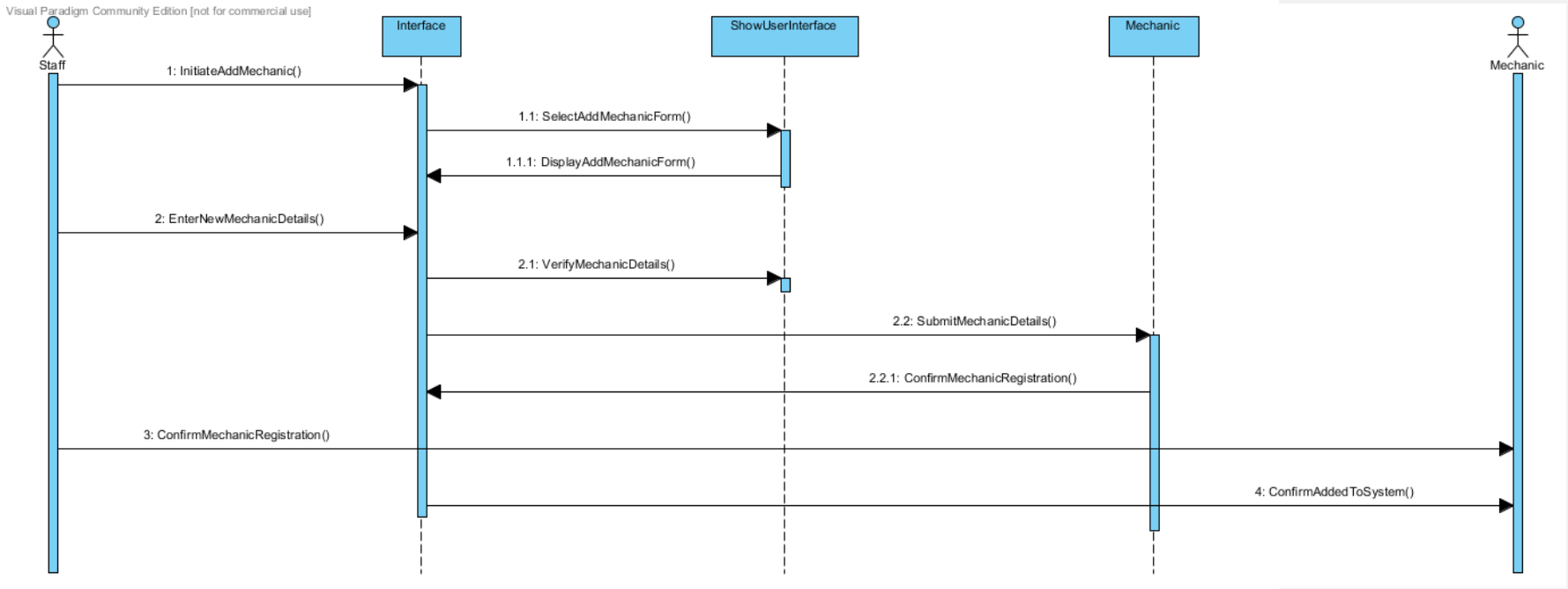

Visual Paradigm Community Edition [not for commercial use]
Staff
Interface
ShowUserInterface
Cars
Management
1: InitiateAddCarToFleet()
1.1: SelectAddCarForm()
1.1.1: DisplayAddCarForm
2: InputNewCarDetails()
2.1: VerifyNewCarDetails()
2.2: CheckNewCarDetails()
2.2.1: ReturnVerificationResult()
3: NotifyManagementOnCarAddition()
3.1: CreateMessageForManagement()
3.1.1: ReturnsMessageConfirmation()
3.1.1.1: SendEmailOrSMSOnNewCar()

Delete Car

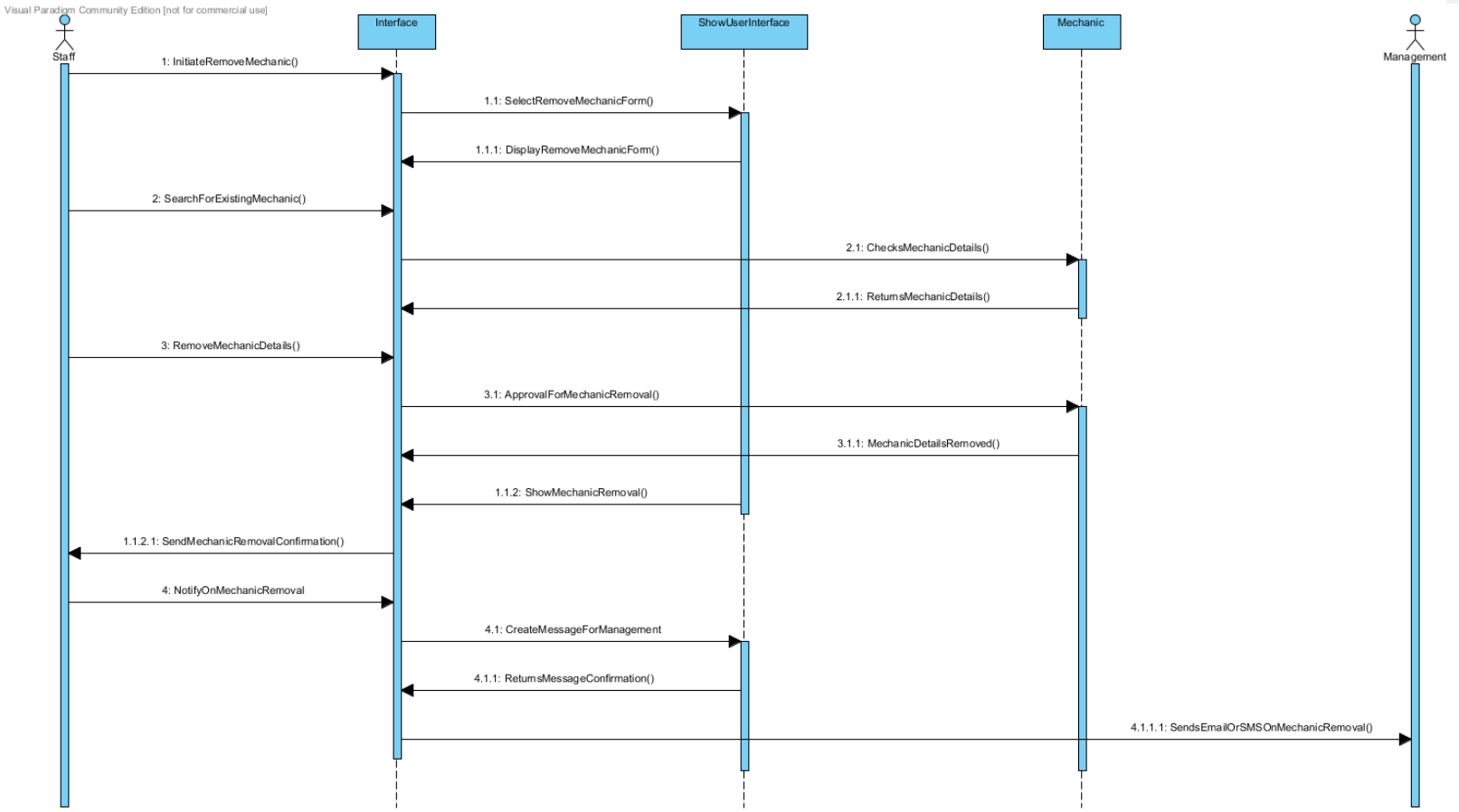
Add



Mechanic


Visual Paradigm Community Edition [not for commercial use]
Staff
Interface
ShowUserInterface
Mechanic
Mechanic
1: InitiateAddMechanic()
1.1: SelectAddMechanicForm()
1.1.1: DisplayAddMechanicForm()
2: EnterNewMechanicDetails()
2.1: VerifyMechanicDetails()
2.2: SubmitMechanicDetails()
2.2.1: ConfirmMechanicRegistration()
3: ConfirmMechanicRegistration()
4: ConfirmAddedToSystem()

Delete Mechanic



Use of Patterns

Controller Pattern

Pattern Name: Controller Pattern

Solution: Assign the responsibility of inputting data into the system to a class that has a role associated to it, represented by one of the following:

- Represents the role of a person, i.e. a person that may be involved in the task of inputting system data
- Represents an artificial handler of the whole system input
- Represents the entire system

Problem: Who/what is going to be responsible for inputting concept details and data into the system, as the controller pattern implies that the controller is a non-user interface object with the responsibility of controlling user input into the system. Therefore, for this operation to be completed, the responsibility needs to be given to a controller that takes into consideration user-input, so data can actually be added into the system.

The controller pattern applies to this assignment due to the fact that, there are various controller concepts throughout the project's entirety. For example: Staff, Customer and Car are all examples of controllers using the controller pattern, as the system revolves a lot around user input into the system, a controller based pattern is essential for defining he method of user input for the system. As a controller pattern states that the controller doesn't exactly carry out the system operations, they just input data into the system to be manipulated, the controller pattern is definitely evident within this system, as the user inputters would be the members of Staff of the company, so therefore they would be classed as the controllers, as they are simply inputting data and then delegating the system operations onto the actual interface to deal with whatever was inputted into the system via the controller.

Here the controller classes, such as Staff and Customer are used for all of the system inputs, such as recording Customer, Mechanic and Car data, as well as for removing company data also, which all align with the relevant use cases concerning: Registering a Customer, Add Mechanic, Add Car, Car Hire, Car Return, Remove Car, Remove Customer, Remove Mechanic etc. There is a single controller class which is Staff that is receiving all system inputs, and various other classes that are receiving system outputs sent by the Staff controller. Additionally, the Staff controller is able to handle most of the system operations itself such as inputting data and manipulating details, without the need to delegate work to other classes, reinforcing high-cohesion.

There are also some artificial handlers for some of the operations using controllers such as: Car Hired, Car Return, Register Customer, Remove Customer etc. These represent the system operating on certain processes within the program, such as performing these operations.

Creator Pattern

Pattern: Creator Pattern

Solution: Assign the instance of the main class responsibility for creating an object or instance of the main class, for example:

- Class B contains Class A objects
- Class B produces instances of Class A objects
- Class B uses Class A objects
- Class B contains data that will be passed onto Class A once it is created

Problem: Who/what is responsible for creating a new instance of a class, or an object of a class. As with creator patterns, it assumes that one of the classes within the system, has instances of itself with relational operations, making another class an instance of that class, which would therefore make the main class the creator of the relational instances of that class, with different methods associating themselves with the main creator class.

The creator pattern applies to this assignment due to the fact that, there are some classes which aggregate objects from different classes, such as the Car class. The car class has many other objects derived from it, such as Car Hired, Car Return, Servicing and Add/Remove Car. As all of these objects derive from the class Car, it would suggest that Car would be a suitable creator for these instances. The creator pattern is also demonstrated with the Customer and Mechanic classes, as they each have classes that are related to them, or aggregate them. For example: The Customer has objects derived from it such as Register Customer and Remove customer, all of which relate back to the main Customer class, which could indicate that Customer could potentially be a suitable creator class. Similarly, with the Mechanic class, it also has objects derived from it such as, Add Mechanic and Delete Mechanic, all of which relate and directly correlate back to the Mechanic class. Which could indicate that they may be sub-classes that are in relation to their main classes, Customer and Mechanic.

This indicates that certain methods created within the derived objects must relate or pass on information to the creator class, such as Add Mechanic would have to pass on information to the Mechanic class, informing that a new record has been created and therefore placed within the class. Another example would be, that the classes Car Return, Car Hire etc. these classes pass information onto the Car class once changes have been made or information has been added, which would indicate that the Car class is a creator.

Low Coupling Pattern

Pattern: Low Coupling Pattern

Solution: Assign a responsibility to classes so that coupling and dependency stays low.

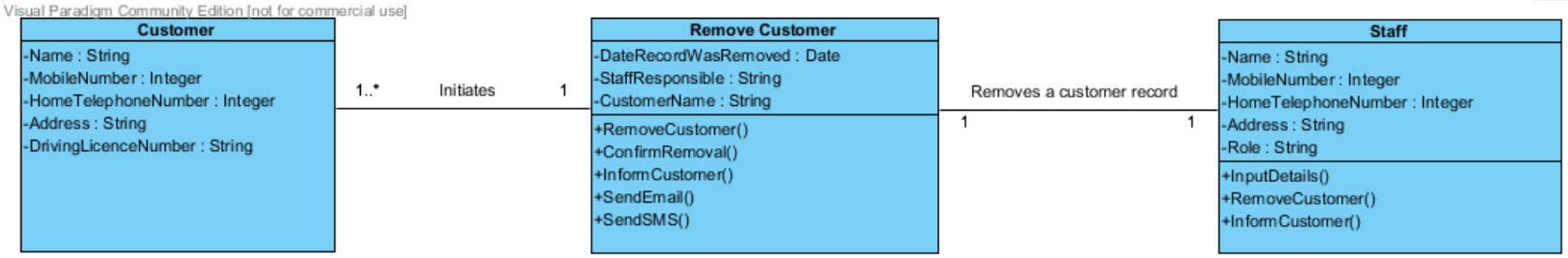
Problem: How to support low dependency classes?

The Low Coupling Pattern applies to this assignment due to the fact that, there are some classes within the system that do highly relate to one another, has knowledge of or relies upon one another. The Register Customer class heavily relies upon and is related to the Customer class, as any changes made to the Customer file on the system, via Register Customer or Remove Customer, must be updated and changed within the Customer class also. This also applies to the Mechanic and Car classes also. As each of the main classes have objects derived from them, i.e. Car has Add Car, Remove Car, Car Hire and Car Return deriving from it, all of these classes are highly related back to the Car class, as each of them contain relevant information or processes that will change or update the objects within the Car class, therefore the Car Class is a suitable example of how this assignment uses the Low Coupling Pattern. The assignment of responsibilities to particular classes partner those of their main classes. For example: the Register Customer class has the responsibilities to add a customer to the database, creating a Customer class. It is clear that the responsibilities of the Register Customer class complement those of the Customer class.

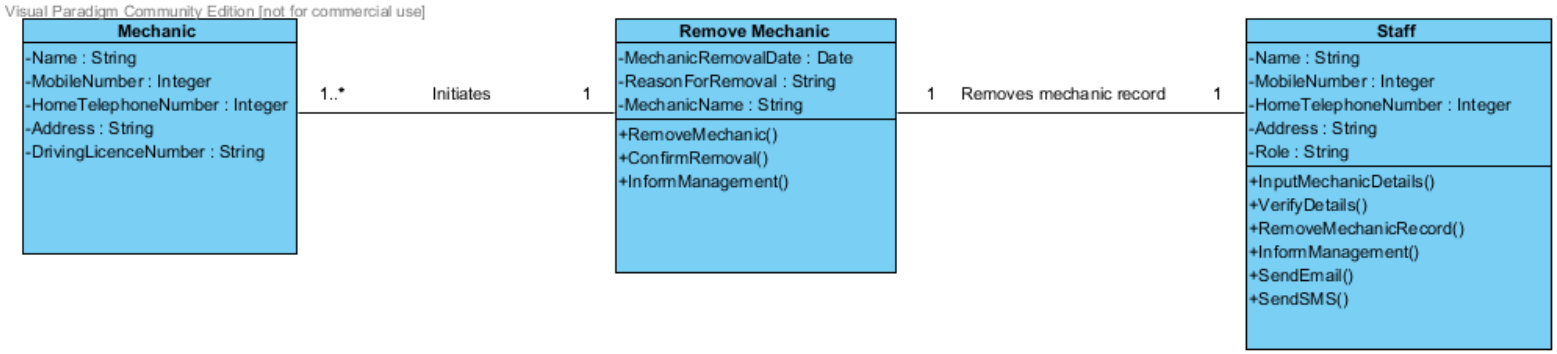
Design Class Diagrams

批注 [ZL27]: The overall design class diagram is required

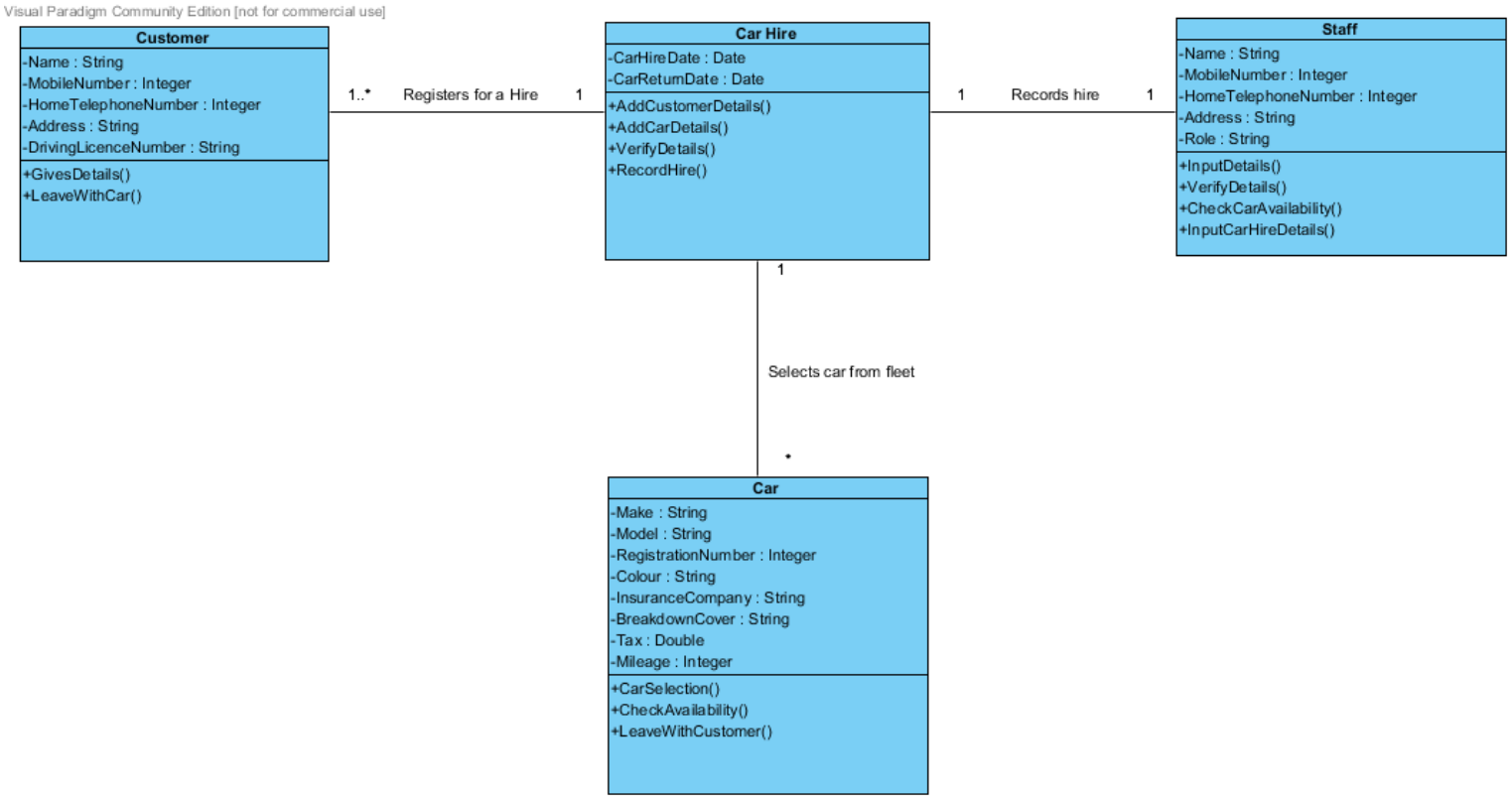
Registering a new Customer



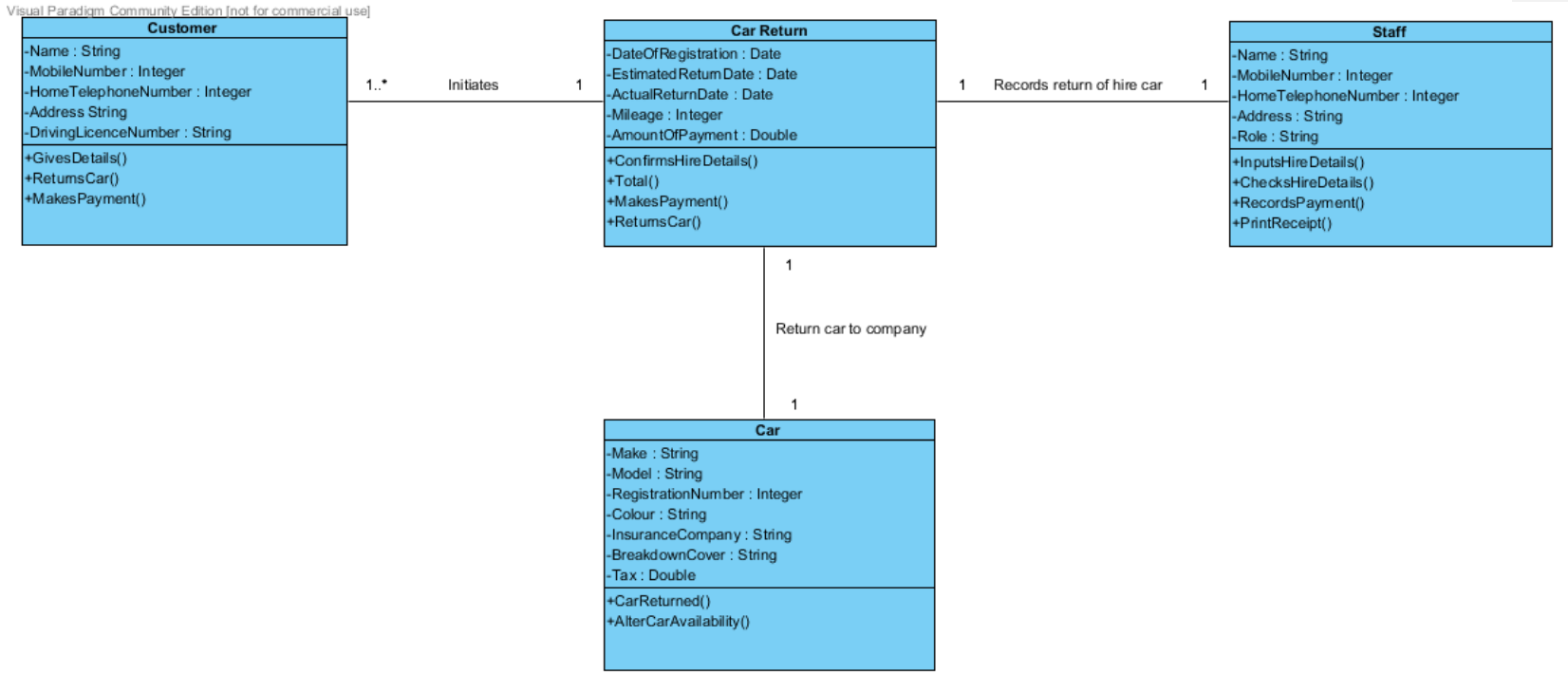
Delete Mechanic



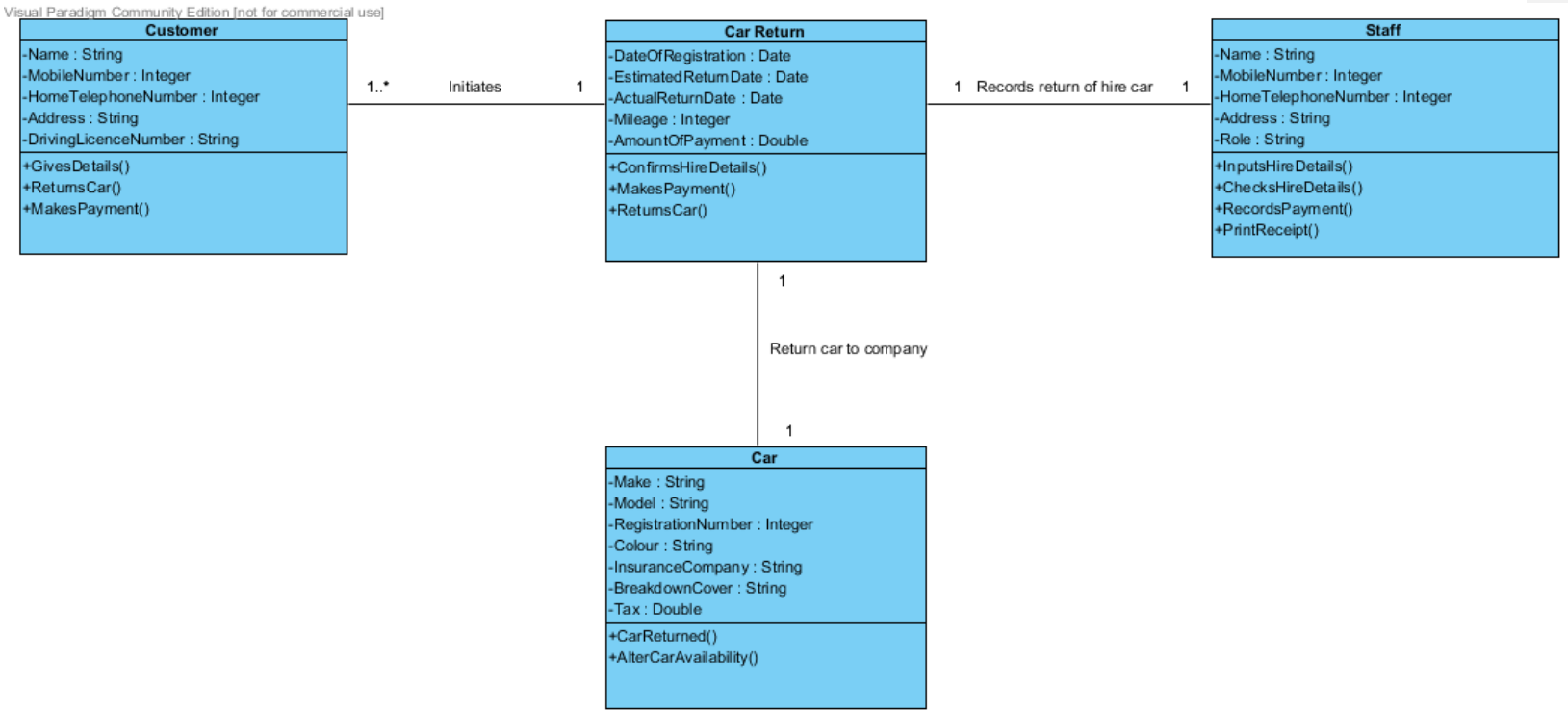
Car Out For Hire



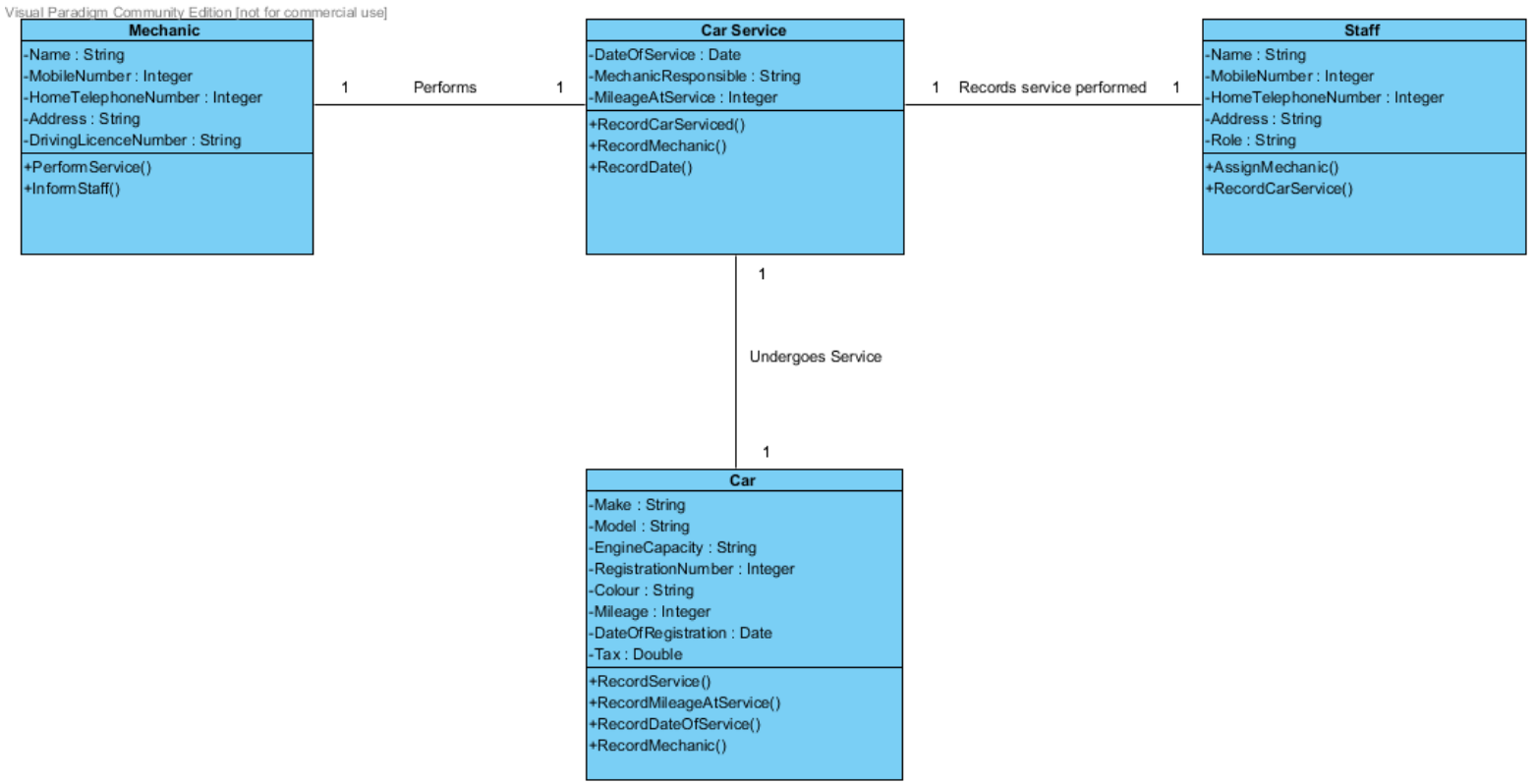
Car Return


Visual Paradigm Community Edition [not for commercial use]
Customer
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address String
-DrivingLicenceNumber : String
+GivesDetails()
+ReturnsCar()
+MakesPayment()
1..*
Initiates
1
Car Return
-DateOfRegistration : Date
-EstimatedReturnDate : Date
-ActualReturnDate : Date
-Mileage : Integer
-AmountOfPayment : Double
+ConfirmsHireDetails()
+Total()
+MakesPayment()
+ReturnsCar()
1
Records return of hire car
1
Staff
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address : String
-Role : String
+InputsHireDetails()
+ChecksHireDetails()
+RecordsPayment()
+PrintReceipt()
1
Return car to company
1
Car
-Make : String
-Model : String
-RegistrationNumber : Integer
-Colour : String
-InsuranceCompany : String
-BreakdownCover : String
-Tax : Double
+CarReturned()
+AlterCarAvailability()

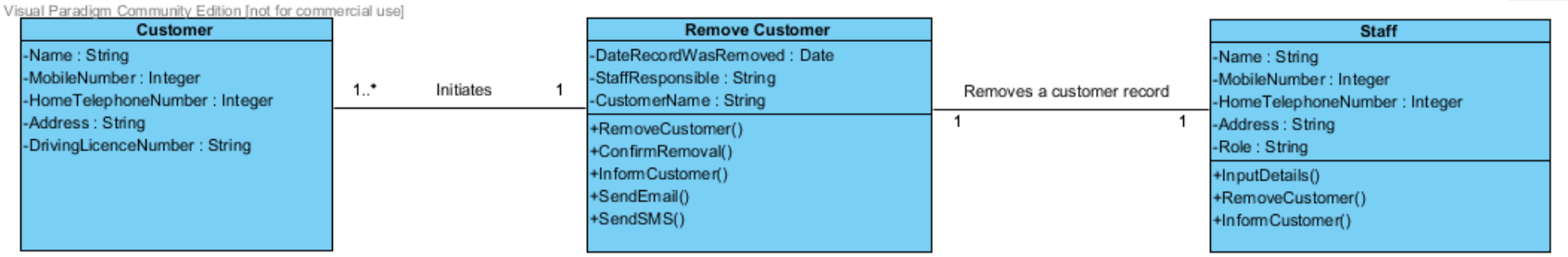
Record a Completed Hire



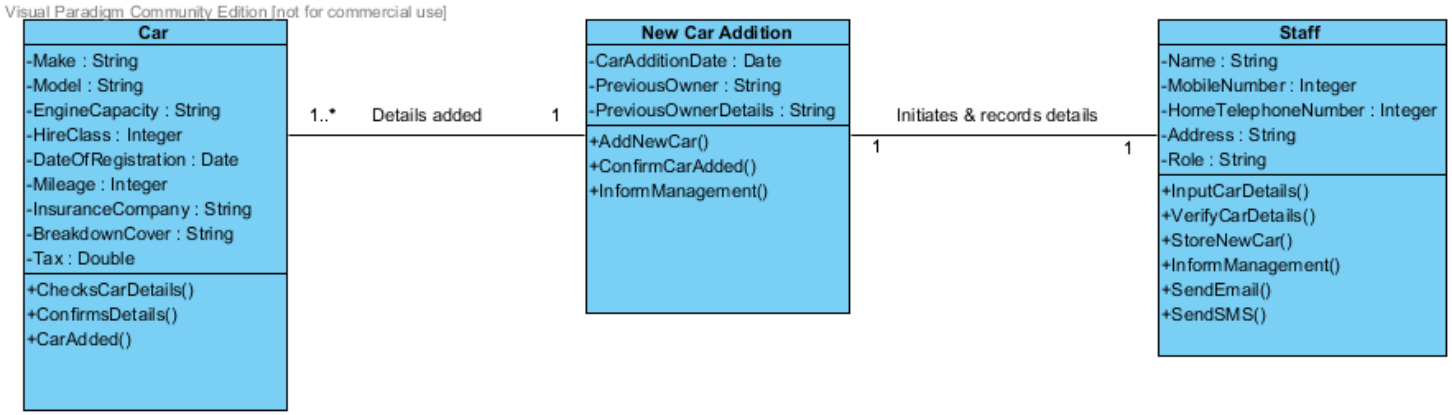
Servicing


Visual Paradigm Community Edition [not for commercial use]
Mechanic
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address : String
-DrivingLicenceNumber : String
+PerformService()
+InformStaff()
1
Performs
1
Car Service
-DateOfService : Date
-MechanicResponsible : String
-MileageAtService : Integer
+RecordCarServiced()
+RecordMechanic()
+RecordDate()
1
Records service performed
1
Staff
-Name : String
-MobileNumber : Integer
-HomeTelephoneNumber : Integer
-Address : String
-Role : String
+AssignMechanic()
+RecordCarService()
1
Undergoes Service
1
Car
-Make : String
-Model : String
-EngineCapacity : String
-RegistrationNumber : Integer
-Colour : String
-Mileage : Integer
-DateOfRegistration : Date
-Tax : Double
+RecordService()
+RecordMileageAtService()
+RecordDateOfService()
+RecordMechanic()

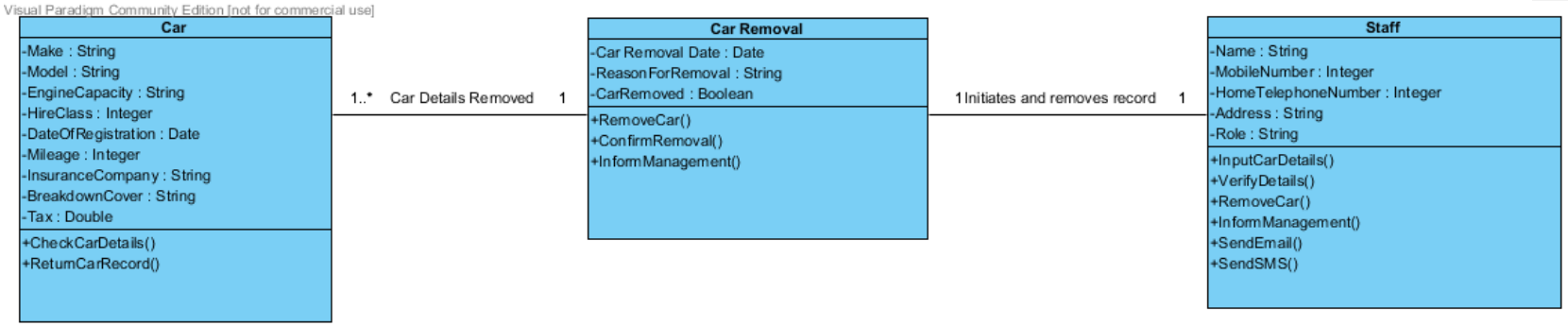
Removal of Customer from Database



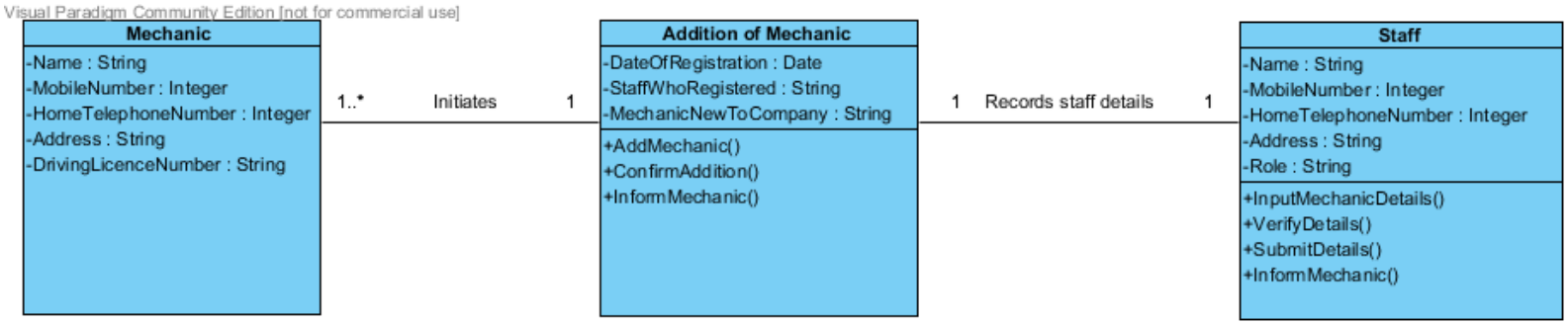
Add Car



Delete Car



Add Mechanic



Appendix

Group Meeting Attendance Diary

*Green represents the group member that was present for the meeting on that particular week

Names in Attendance							Week Commencing	Meeting Discussion
Sophie Thompson	**Anne Oboma**	**Hamza Ali**	**Ho Nam Tsang**	**Karan Badhen**	**Kevin Clopon**	**Olukayode Alatishe**		
							26th January	Begin Project, allocate roles to each member for Section 1.
							2nd February	Continue progress on Section 1.
							9th February	Continue progress on Section 1.
							16th February	Continue progress on Section 1.
							23rd February	Confirm roles for section 2 and 3 of project.
							1st March	Start progress on section 2 of the project
							8th March	Continue progress on section 2 of the project.
							15th March	Continue progress on section 2 of the project.
							22nd March	
							29th March	

Work Allocation

Section 1: The Initial Requirements Understanding

Task name	Description	Assigned team members	Start date	Estimated date of completion	Completed by:	Actual date of completion
Situation	Discuss the nature of the Bvis Car Hire Company described above and justify why an object-oriented development applicable to the project. You need to read and refer points to ideas discussed Chapter 3 of the course notes. Elaborate the problem description as necessary to support your analysis.	Sophie Thompson	26th January	10th February	Sophie	24th February
System Functions	Present the system functions according to the guidelines in Section 4.1 of the course notes	Sophie Thompson Ho Nam Tsang	26th January	10th February	Sophie Thompson Ho Nam Tsang	3rd February
Essential use cases	Identify the essential use cases, which cover and support the understanding of the required functions in the problem description. Write an expanded version for each of these use cases.	Olukayode Alatishe Kevin Clopon Hamza Ali	26th January	10th February	Olukayode Alatishe Kevin Clopon Hamza Ali	10th February
Use case diagrams	Create a use case diagram for the use cases that you identified in the previous question to show the relationships between the actors and the use cases, and the relationships between the use cases.	Olukayode Alatishe Kevin Clopon Hamza Ali	10th February	17th February	Olukayode Alatishe Kevin Clopon Hamza Ali	17th February

Identifying classes, associations and attributes within the application domain	Using the guidelines, strategies, and notation discussed in the course notes, work through the problem statement and the use cases that you have identified in item 3 to identify classes (concepts), associations, and attributes in the application domain. You should give enough discussion to support your identification. Draw a conceptual class diagram, which includes, the classes, associations, and attributes that you have identified. Again, you only have to consider the functions and the use cases that you considered for item 3. You should give enough discussion to support your identification. Draw a conceptual model, which includes, the classes, associations, and attributes that you have identified. You may find that you need to refine or modify your use cases.	Sophie Thompson Karan Badhen	10th February	17th February	Sophie Thompson Karan Badhen	17th February

Section 2: Analysis and Functionality of System Operations

Task name	Description	Assigned team members	Start date	Estimated date of completion	Completed by:	Actual date of completion
Identifying System Operations	Use the techniques discussed in Chapter 6 of the course notes to identify the system operations from the typical course of events of the use cases that you have produced. Create system sequence diagrams for the typical course of events of the use cases that you think most significant for the development of the system. You may find that you need to refine or modify your use cases and conceptual model that you have produced.	Sophie Thompson Anne Oboma Kevin Clopon	24th February	10th March	Sophie Thompson Anne Oboma Kevin Clopon	24th March
System Operation Contracts identified	Based on your use-case model and conceptual model that you have produced write the contracts for the system operations that you have identified. You may find that you need to refine or modify your use-case model and conceptual model while you are working on the contracts.	Sophie Thompson	17th March	31st March	Sophie Thompson	7th April

Section 3: Use Case Design

Task name	**Description**	**Assigned team members**	**Start date**	**Estimated date of completion**	**Completed By:**	**Actual date of completion**
Collaboration Diagrams/ Object Sequence Diagrams	The collaboration diagrams or object sequence diagrams (not both) which show the assignment of responsibilities to classes of objects.	Sophie Thompson Karan Badhen Olukayode Alatishe	7th April	15th April	Sophie Thompson Karan Badhen Olukayode Alatishe	14th April
Use of patterns regarding the responsibilities of classes of objects	Enough discussion about the use of the patterns in your assignment of responsibilities to classes of objects.	Sophie Thompson Ho Nam Tsang	14th April	16th April	Sophie Thompson Ho Nam Tsang	
The design class diagrams	The design class diagrams, which shows the methods/operations of classes.	Sophie Thompson Hamza Ali	16th April	18th April	Sophie Thompson Hamza Ali	

Section 4: The Report

Task Name	Description	Assigned team members	Start date	Estimated date of completion	Actual date of completion
Construct the final documentation	Add the completed work of the first three sections and add them together	Sophie Thompson	19th April	15th April	19th April
Proof-Read the final document	Proof-read and check the document so that there aren't any errors in there	All members	15th April	17th April	19th April
Finalise & Verify the document	Everyone decides whether the final project report is ready for submission	All members	19th April	17th April	19th April
Submit Project Document	Submit the final document to Moodle.	Sophie Thompson	19th April	21st April	19th April

Glossary

Concept: A concept can be represented in terms of its symbols, intensions and extensions to show what that concept is.

Class: A class is part of the Object Oriented design framework used to separate differing concepts so that they are all grouped together, and their attributes and methods can control the actions of an object instantiated from that class.

Object: An object is an instance of a class, containing its attributes and associations.

Method: A method is a behavioural control mechanism used by a class, to control how specific objects of that class behave

Attribute: An attribute is a characteristic of a class.

Association: An association is a link between classes that show how they are connected with each other

Use Case: A use case is a case or a story that occurs when the system is in operation, shows how certain processes happen throughout the system.

Actor: A person who interacts with the system in one way or another.

Use Case Sequence Diagram: Shows the operations taken place once the system is in use, and shows how the actors interact with the system.

Patterns: Shows how the users and the system work.