Assessment 2

Name: Sahand Sabour

Student ID Number: 1614650

E-mail: [Sahand.sabour16@student.xjtlu.edu.cn](mailto:Sahand.sabour16@student.xjtlu.edu.cn)

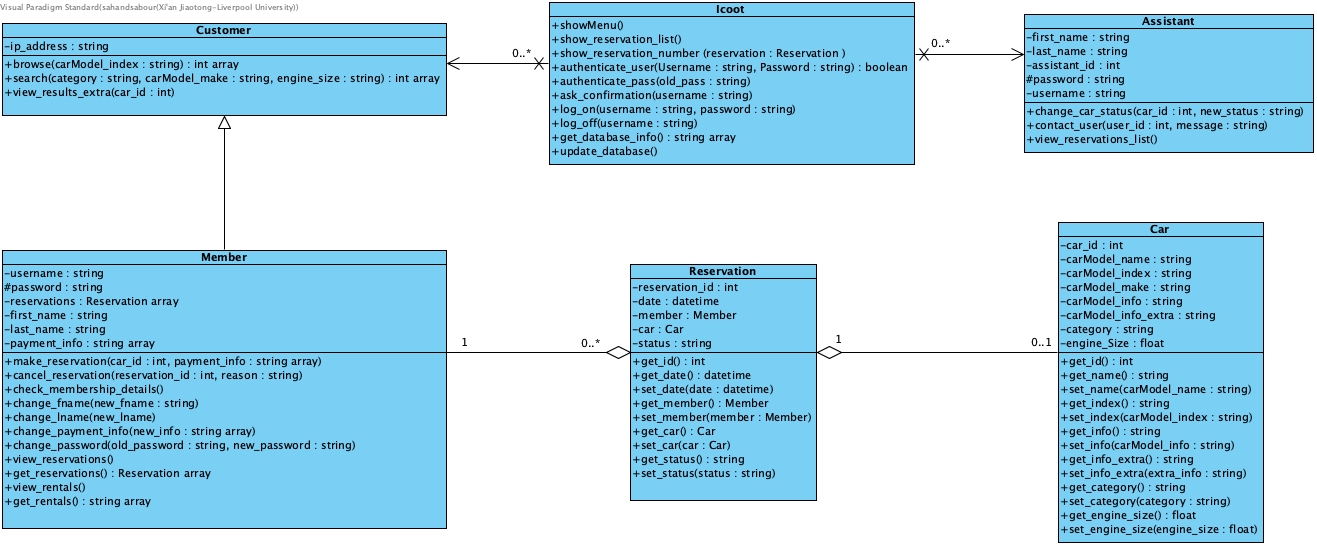
Computer Science and Technology (CST)

Software Engineering I – CSE207

Lecturer: Kaiyu Wan

Submission deadline: FRI 2018/11/30 6PM

d) Design a class diagram for the Coot software.



|  |  |
| --- | --- |
| Class name: Icoot |  |
| Major functionality | The major functionality of the Icoot class can be regarded as keeping the system in balance. This class is implemented in order to do necessary authentications and changes of information while creating other objects based on the type of demand. Therefore, it can be considered as both a connection between different classes and also an assistant to the flow of the program. |
| Important attributes | There are no important attributes as this class only possesses functions (methods). |
| Important methods | **showMenu () -->** shows the menu and its options to user.  **show\_reservations\_list () -->** displays the list of reservations.  **show\_reservation\_number () -->** displays number of a made reservation.  **authenticate\_user (username, password) -->** checks whether the user is a member, an assistant or neither of the two.  **authenticate\_pass (old\_pass) -->** checks whether the entered password matches the user’s password (for changing pass).  **ask\_confirmation (username) -->** ask the corresponding user for confirmation.  **log\_on (username, password) -->** create corresponding user objects. A user object can be from Member or Assistant class.  **log\_off (username)-->** delete corresponding user objects. A user object can be from Member or Assistant class.  **get\_database\_info()-->** gets the necessary data from database.  **update\_database()-->** identifies the changes in the system information and updates the database accordingly. |
| Comments | Although the functions for this class could be implemented regardless of an Icoot object, it was believed that creating a class for the system would benefit the system from the future perspective, both in clarity and performance. |

|  |  |
| --- | --- |
| Class name: Customer |  |
| Major functionality | To represent ordinary users of the system that do not have an account, which is registered in the system’s database. As it was mentioned in the system requirements, ordinary customers were provided with limited functionality and therefore, this class was design accordingly. |
| Important attributes | **ip-address: string -->** acts as an identification attribute, which is used to distinguish the users. |
| Important methods | **browse (carModel\_index) -->** browses the database based on the provided index and returns a list of corresponding car\_ids.  **search (category, carModel\_make, engine\_size) -->** searchds the database based on the provided category, make and engine size and returns a list of corresponding car\_ids.  **view\_results\_extra (car\_id) -->** displays extra information for a specific car (each car has unique car\_id). |
| Comments | An object of this class is instantiated as the user opens the webpage. |

|  |  |
| --- | --- |
| Class name: Member |  |
| Major functionality | To represent customers who have become members and have obtained usernames and passwords of their choice. This class was designed to allow the user to access the features that were addressed as ‘extra services’ in the requirements section. |
| Important attributes | **username: string -->** it is unique for each user and it is used for member identification.  **password: string -->** used for user authentication.  **reservations: Reservation array -->** indicates the reservations that the user has made by an array of Reservation objects. |
| Important methods | **make\_reservation (car\_id, payment\_info) -->** to submit a reservation in a database using the id of the car, which is user is viewing at the moment, and also valid payment info.  **cancel\_reservation (reservation\_id, reason) -->** to cancel an already made reservation. This function searches the database based on the provided id to find the corresponding reservation and removes it consequently.  **check\_membership\_details () -->** displays the membership details for the current user (current member).  **view\_reservations () -->** displays the items in the ‘reservations’ (check important attributes for details of this attribute).  **get\_reservations() -->** gets ‘reservations’ as a variable (array).  **view\_rentals() -->** displays the rentals with ‘collected’ status.  **get\_rentals() -->** gets the rentals with ‘collected’ status as an array. |
| Comments | Objects from this class are instantiated once the user has passed the authentication function. |

|  |  |
| --- | --- |
| Class name: Assistant |  |
| Major functionality | To represent users of the system who are currently working in the company as an assistant. This class was designed to provide the user with the functionality that is only available to working assistants. Although this class does not contain a rather large number of attributes and methods, it plays an essential role in the functionality of the system. |
| Important attributes | **assistant\_id: int -->** a unique id that is used for distinction between assistants.  **password: string -->** used for assistant authentication.  **username: string -->** additionally, a username was to be added for each assistant with the aim of adding to the identity of each assistant in the system. |
| Important methods | **change\_car\_status (car\_id, new\_status) -->** in order to change the status of a car, identified by car\_id, to either ‘Collectable’, ‘Displayable’ or ‘Notifiable’.  **contact\_user (user\_id, message) -->** sends an arbitrary the user, identified by user\_id.  **view\_reservations\_list () -->** displays a list of reservations made by the members of the system. |
| Comments | Objects from this class are instantiated once the user has passed the authentication function. |

|  |  |
| --- | --- |
| Class name: Reservation |  |
| Major functionality | To represent reservations, which are made through the system. The objects of this class can be considered to be similar to entries in a list of reservations. |
| Important attributes | **reservation\_id: int -->** a unique id for identifying a reservation.  **date: datetime -->** the date which the reservation was made on. The format of the date is based on the system standards.  **member: Member -->** the member responsible for making the reservation.  **car: Car -->** the car which corresponds to the reservation. It could be none if no car is still found for the reservation.  **status: string -->** current status of the reservation which could be ‘Collectable’, ‘Notifiable’ or ‘Concluded’. |
| Important methods | The methods for this class consist of getter and setter methods which are believed to not require any further description. Every method that starts with ‘get\_’ would return the requested variable and methods starting with ‘set\_’ would change the value of the mentioned variable to the provided value.  It should be noted that the absence of ‘set\_id’ method is due to the id being unique for each reservation and is used for its identification. Therefore, it should not be subjected to change. |

|  |  |
| --- | --- |
| Class name: Car |  |
| Major functionality | To represent cars that are registered within the database. This class would provide the system with the ability to distinguish different vehicles in order to improve data management. |
| Important attributes | **car\_id: int -->** a unique id for identifying a car  **carModel\_name: string -->** the name of the car  **carModel\_index: string -->** the index of the car  **carModel\_make: string -->** the make of the car  **carModel\_info: string -->** information regarding the car  **carModel\_info\_extra: string -->** Extra information for the car  **category: string -->** the category for the car  **engine\_size: string -->** the engine size of the car |
| Important methods | The methods for this class consist of getter and setter methods which are believed to not require any further description. Every method that starts with ‘get\_’ would return the requested variable and methods starting with ‘set\_’ would change the value of the mentioned variable to the provided value.  Similar to the ‘Reservation’ class, the absence of ‘set\_id’ method is due to id being unmodifiable in the system design. |

**Note:**

in the descriptions above, in case any attribute and/or method that existed in the class diagram was not mentioned in the above tables, it would be due to its clarity and low chance of causing misunderstanding. Furthermore, the names chosen for these attributes/methods was believed to display a clear description of their functionality and further description was not seen as a necessity.

**Justification:**

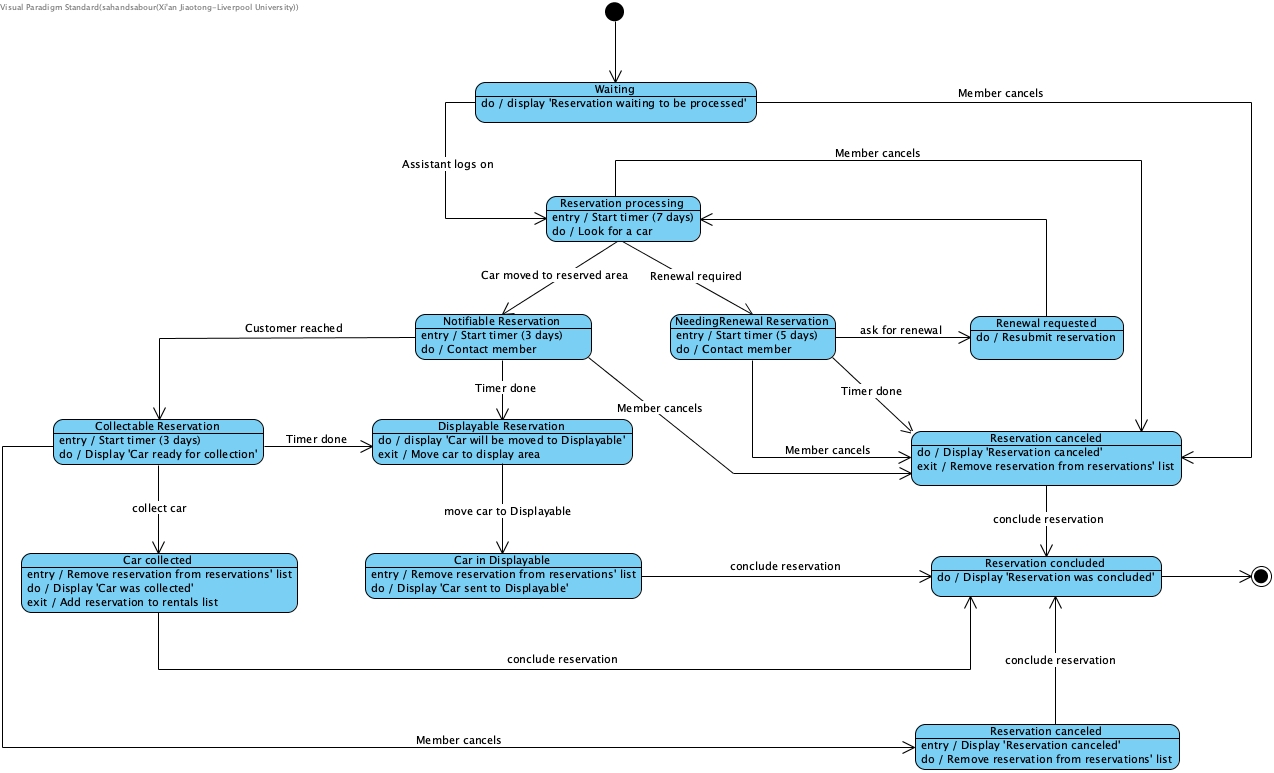
The functionality of the system mainly depends on the user and the user demand. There were three classes dedicated to the system user (‘Customer’, ‘Assistant’ and ‘Member’) as these were the three types of users that were defined for the system in the requirements. As for the user demand, it was concluded that each demand, in Icoot system in particular, corresponded to a reservation as well as a user and a vehicle. Moreover, ‘Reservation’ and ‘Car’ classes were designed in order to highlight this relation.

Although the design could be sufficient for development, an additional class (‘Icoot’) was designed in order to gather the rest of the necessary functions as the functions of one collective class. It was believed that by doing so, further changes in the system would be made with less effort and also less risk of damaging the system.

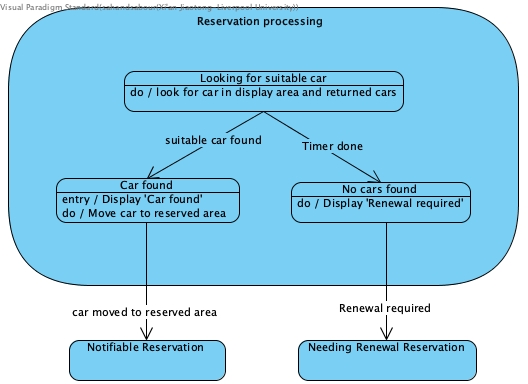
It should also be noted that the absence of interfaces in the design of this system is believed to be due to python being the chosen programming language for system development. Furthermore, it is acknowledged that in the case of using programming languages such as JAVA and/or C++, interfaces would also be implemented as a part of the design.

It is also acknowledged that the system design may be changed during the development of the system. As it was mentioned in the suggested template for this assessment, extra objects may be added to the system in the future. However, it was concluded that the system development with the current design must first be completed and plans for adding new objects would be initiated accordingly.

e) Design a state machine to describe “Reservation” status in system.



The following diagram describes the ‘Reservation processing’ state in detail:



**The states description:**

|  |  |
| --- | --- |
| States | Description |
| Waiting | The system is waiting for an assistant to log on so that the reservation could be processed. |
| Reservation Processing | The assistant is looking for a suitable car in the displayable area. A suitable car could be an unreserved car or a car returned by a customer. The 7-day timer for finding a car is running in this state. |
| Notifiable Reservation | The suitable car for the reservation is found and the car has been moved to the reserved area. The member is notified either by phone or in person. The 3-day timer for car collection is running in this state. |
| Needing Renewal Reservation | The deadline for finding a car is due and the reservation needs to be renewed. The member is notified either by phone or in person. The 5-day timer for reservation renewal is running in this state. |
| Renewal requested | Renewal has been requested and the reservation is resubmitted in the system. |
| Reservation canceled | The reservation is canceled and removed from the list of reservations. A message indicating that the mentioned cancelation is displayed. |
| Displayable Reservation | The car is moved to the displayable area. A message indicating that the car is moved to the displayable area is shown. |
| Collectable Reservation | The car is ready for collection. A message indicating the mentioned is displayed. The 3-day timer for car collection is running. |
| Car collected | The car is collected. A message indicating the mentioned is displayed. The reservation is removed from the list of reservations (outstanding Reservations). The reservation is added to the list of rentals (outstanding Rentals). |
| Car in Displayable | The car is moved to the displayable area. The reservation is removed from the list of reservations (outstanding Reservations). A message indicating the mentioned is displayed accordingly. |
| Reservation concluded | The reservation was concluded. |

**The events description:**

|  |  |
| --- | --- |
| Stimulus | Description |
| Assistant logs in | The user (assistant) has logged on. |
| Car moved to reserved area | The assistant has moved the car to the reserved area. |
| Move car to Displayable | The assistant has moved the car to the displayable area. |
| Renewal required | The system has concluded that a renewal is required for the reservation. |
| Ask for renewal | The member has requested reservation renewal. |
| Customer reached | The assistant has successfully contacted the member. |
| Collect car | The member has successfully collected the car. |
| Member cancels | The member has cancelled their reservation. |
| Timer done | The timer for the previous state has reached the mentioned amount (either seven or five or three days). |
| Conclude reservation | The system has concluded the reservation. |

f) According to the state machine you designed for task b, design test cases to cover all states and transition testing.

* Waiting -> Reservation canceled -> Reservation concluded
* Waiting -> Reservation processing -> Reservation canceled -> Reservation concluded
* Waiting -> Reservation processing -> Needing Renewal Reservation -> Reservation canceled -> Reservation concluded
* Waiting -> Reservation processing -> Needing Renewal Reservation -> Renewal requested -> Reservation processing
* Waiting -> Reservation processing -> Notifiable Reservation -> Reservation canceled -> Reservation concluded
* Waiting -> Reservation processing -> Notifiable Reservation -> Displayable Reservation -> Car in Displayable -> Reservation concluded
* Waiting -> Reservation processing -> Notifiable Reservation -> Collectable Reservation -> Reservation canceled -> Reservation concluded
* Waiting -> Reservation processing -> Notifiable Reservation -> Collectable Reservation -> Car collected -> Reservation concluded