Object Design Document:
Car Rental Service

Group 4

CSCI 4050: Software Engineering
Instructor: Krys Kochut

Osama Mansour, Stephen Patton, Vincent Lee, and Minh Pham

Table of Contents

1. Introduction

The online car rental service will be implemented through various subsystem layers that interact with each other. In this object design document, we specify a complete blueprint of the system we will be implementing, finalizing intricate details of classes, methods, and subsystems.

1.1 Object Design Trade-Offs

1.1.1 Development Cost vs. Functionality

The system is designed to include a large amount of functionality. Vehicles can be checked in an out online, and system administrators can add, delete, and modify all vehicles, vehicle types, and

rental car store locations. Our vehicle search can be done based on many different parameters including location, vehicle type, and price. We feel that these features, although expensive to implement, are necessary to provide adequate functionality to the customer and administrators.

1.2 Interface Document Guidelines

1.2.1 Naming Conventions

Our implementation will follow standard Java naming conventions:

packages

lowercase

files

same name as the public class identifier

classes

capitalize the first letter and the first letter of any internal words

constants

all uppercase with underscores separating words

methods

first letter lowercase, remaining words begin with a capitol

comments

classes will begin with “/*.....*/” with a short description of the functionality of the class and instructions for usage. The following tags may be included for additional documentation:

1. @author author-name
2. @version version number of class
3. @see string
4. @see URL
5. @see classname #methodname

Within methods, // may also be used to give additional comments when needed.

methods

Javadoc conventions will be followed when outlining the purpose, usage, return values, parameters, exception, etc. of a method. This includes the following tags:

1. @param paramName description
2. @return description of return value
3. @exception exceptionName description
4. @see string
5. @see URL
6. @see classname#methodname

1.3 Definitions

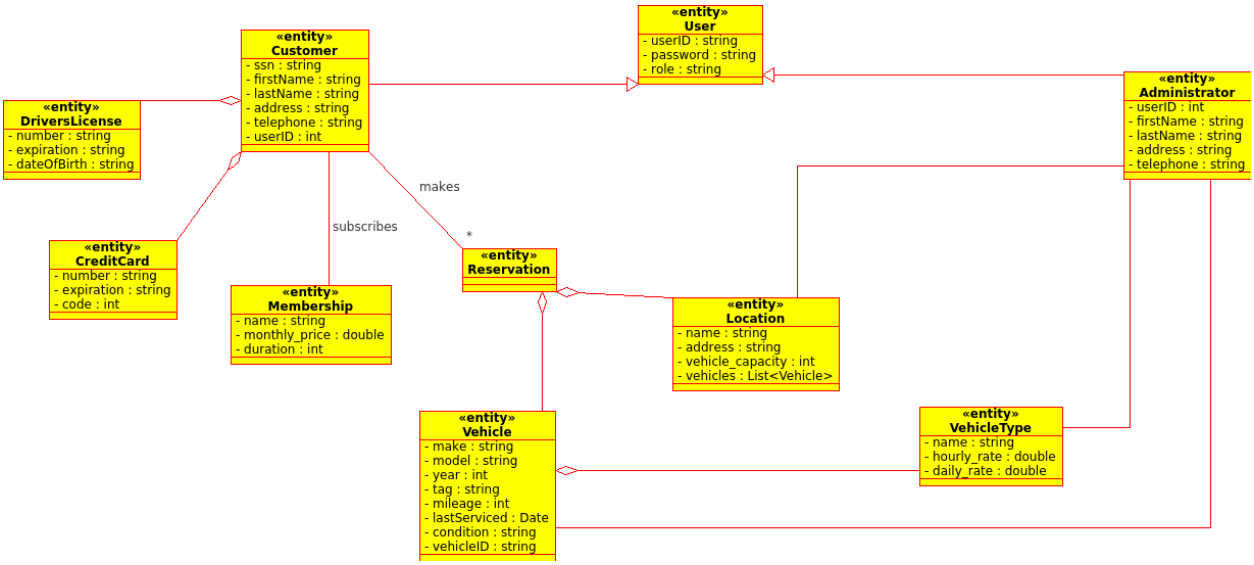
Vehicle type: a category of vehicle within the system where all vehicles in that categories share certain features (i.e. 7 seats), and share a common price. Examples: minivan, luxury car

1.4 References

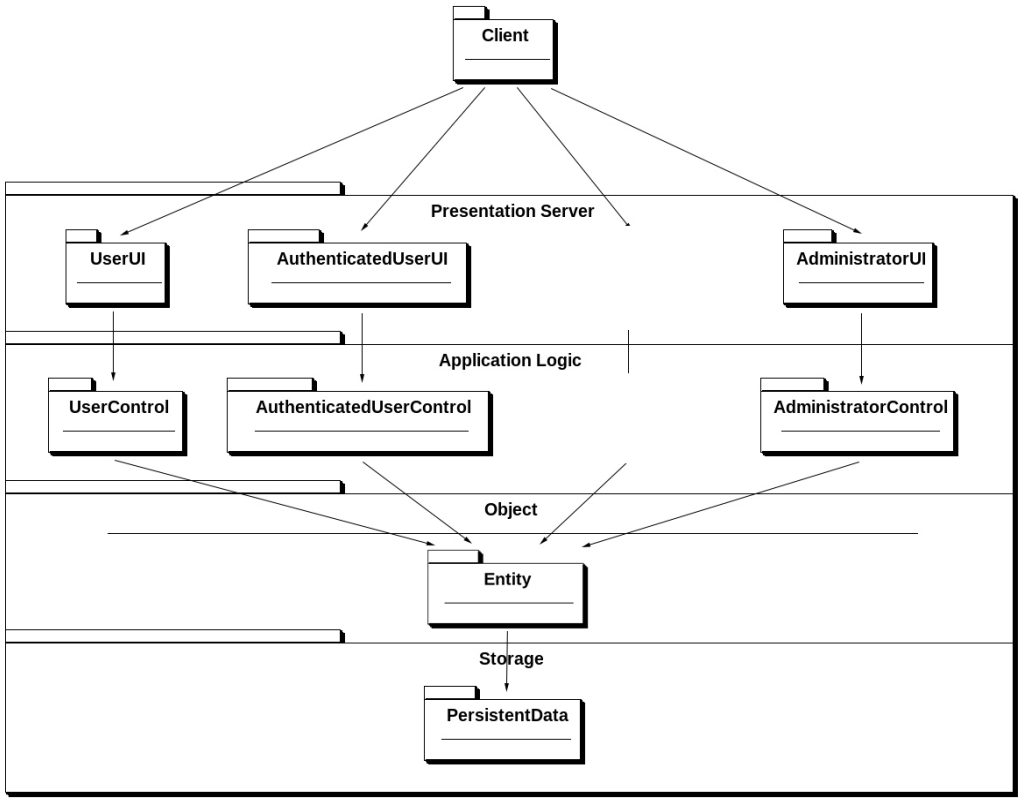
Java Enterprise Edition(Java EE) specifications

2. Packages

2.1 Modified class diagram



2.2 Packages, subsystems and layers



3. Class interfaces

3.1 Data dictionary

3.1.1 Entity Package

ClassName	User				
Purpose	The generalize entity involved in a Car rental system transaction				
SuperClass	None				
SubClass	Administrator, Customer				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
	private	userID	Integer	The ID of the User as defined by the System Registration	
	private	password	String	The encryption on the user profile to be accessed	
	private	role	String	The role of the user defined by the system	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	Login	userID password role		The login page allows users access to registered users within a system
	public	Logout			The logout page allows users currently login in to exit there current system profile
	public	updateProfile	profile		The update profile operation allows for user to change and save their profile information
Relationships	Super class of Administrator, Customer				
Constraints	**Context** User **inv**: userID <> nil and password.length >=6				
Exception	**login()** throws an **Exception** if **userID** and **password** do not match or no **userID** exists in the database				

ClassName	Customer				
Purpose	The generalization of group of Users labeled Customers				
SuperClass	Users				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
	private	userID	Integer	The ID of the User as defined by the System Registration	
	private	ssn	String	Customer social security information for verification	
	private	firstName	String	Customer First name	
	private	lastName	String	Customer Last name	
	private	address	String	Customer Address	
	private	telephone	String	Customer Telephone number	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	ViewMyInfo		Customer	Displays all personal information of the current logged in Customer
	public	ViewAvailableCars	carName carLocation	List<Vehicle>	Displays the Cars available for rental by location
	public	ViewMyRentalHistory		List<Vehicle>	Displays a current logged Customer rental history
	public	ViewMyBillingInfo		billingReport	Displays Current logged

					Customer billing report
	public	ViewMyCarRental		Vehicle	Displays Customer past and present car rentals
	public	EditMyInfo	userID firstName lastName address telephone	Customer	Allows Customers to edit their information as needed
Relationships	Subclass of the User A Customer can only rent 1 Car at a time A Customer can only edit personal information, but not vehicle information				
Constraints	**Context** Customer **inv**: ssn <> nil **Context** Customer **inv**: firstName <> nil **Context** Customer **inv**: lastName <> nil **Context** Customer **inv**: address <> nil **Context** Customer **inv**: telephone <> nil				
Exception	EditMyInfo() throws an Exception if the information input is null, or no modifications were made.				

ClassName	Administrator
Purpose	The generalization of group of Users labeled Administrator within the Car rental System
SuperClass	Users
SubClass	None

Attributes	**Visibility**	**Name**	**Type**	**Description**	
	private	userID	Integer	ID identifying the Administrator within the car rental system	
	private	firstName	String	Admin First name	
	private	lastName	String	Admin Last name	
	private	address	String	Admin Address	
	private	telephone	String	Admin Telephone number	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	ViewMyInfo		Administrator	Displays all personal information of the current logged in admin
	public	ViewVehicles	CarName CarLocation	List<Vehicle>	Displays the Cars available within the Car rental system
	public	ViewRentalHistory	Customer Name	Customer	Displays a rental history for all customer by name
	public	ViewBillingHistory	Customer Name	Customer	Displays all Customer History Billing reports
	public	ViewCarRentals	Customer Name	Vehicle	Displays All Customer Car rental reports
	public	EditVehicleInfo	Vehicle Name	Vehicle	Edit Vehicle Information
	public	EditCustomerInfo	CustomerName	Customer	Allows Admin to edit customer

					information both adding and removing Customers
Relationships	Subclass of the User An Admin can edit Vehicle Information, and Customer information on Request				
Constraints	**Context** Administrator **inv**: userID <> nil **Context** Administrator **inv**: firstName <> nil **Context** Administrator **inv**: lastName <> nil **Context** Administrator **inv**: address <> nil **Context** Administrator **inv**: telephone <> nil				
Exception	None				

ClassName	DriversLicense				
Purpose	Provide driver's license information of Customer				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
	private	number	String	The number on the license	
	private	expiration	String	The expiration date	
	private	dateOfBirth	String	The date of birth on the license	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
Relationships	None				
Constraints	**Context** DriversLicense **inv**: number <> nil **Context** DriversLicense **inv**: expiration <> nil				

	Context DriversLicense **inv**: dateOfBirth <> nil
Exception	None

ClassName	CreditCard			
Purpose	Provide credit card information of Customer			
SuperClass	None			
SubClass	None			
Attributes	**Visibility**	**Name**	**Type**	**Description**
	private	number	String	The number on the card
	private	expiration	String	The expiration date
	private	code	Integer	The CVC code

Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
Relationships	None				
Constraints	**Context** CreditCard **inv**: number <> nil **Context** CreditCard **inv**: expiration <> nil **Context** CreditCard **inv**: code <> nil				
Exception	None				

ClassName	VehicleType			
Purpose	An entity class for storing information on vehicle types			
SuperClass	None			
SubClass	None			
Attributes	**Visibility**	**Name**	**Type**	**Description**

	private	name	String	The name of the vehicle type
	private	hourly_rate	Double	The hourly price of the vehicle type
	private	daily_rate	Double	The daily price of the vehicle type

Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**

Relationships	None
Constraints	**Context** VehicleType **inv**: name <> nil **Context** VehicleType **inv**: hourly_rate <> nil **Context** VehicleType **inv**: daily_rate <> nil
Exception	None

ClassName	Vehicle			
Purpose	An entity class for storing information on vehicles in the fleet			
SuperClass	None			
SubClass	None			
Attributes	**Visibility**	**Name**	**Type**	**Description**
	private	vehicleID	String	The ID of the vehicles
	private	make	String	The make of the vehicle
	private	model	String	The model of the vehicle
	private	year	Integer	The manufacture year of the vehicle
	private	tag	String	The registration tag of the vehicle
	private	mileage	Integer	The current mileage of the vehicle
	private	last_serviced	Date	The last date the vehicle was serviced
	private	condition	String	The condition of the vehicle

Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**

Relationships	A vehicle belongs to 1 location				
Constraints	**Context** Vehicle **inv**: vehicleID <> nil **Context** Vehicle **inv**: make <> nil **Context** Vehicle **inv**: model <> nil **Context** Vehicle **inv**: year <> nil **Context** Vehicle **inv**: tag <> nil **Context** Vehicle **inv**: mileage <> nil **Context** Vehicle **inv**: last_serviced <> nil **Context** Vehicle **inv**: condition <> nil				
Exception	None				

ClassName	Location				
Purpose	An entity class for storing information on point of presence POP locations				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
	private	name	String	The name of the location	
	private	address	String	The address of the location	
	private	vehicle_capacity	Integer	The capacity of the location	
	private	vehicles	List<Vehicle>	The list of vehicles assigned to the location	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
Relationships	None				

Constraints	**Context** Location **inv**: name <> nil **Context** Location **inv**: address <> nil **Context** Location **inv**: vehicle_capacity <> nil **Context** Location **inv**: vehicles <> nil
Exception	None

ClassName	Membership				
Purpose	An entity class for storing information on membership tiers				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
	private	name	String	The name of the membership tier	
	private	monthly_price	Double	The price of one month's service	
	private	duration	Integer	The length of membership in months	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
Relationships	A customer belongs to 1 membership				
Constraints	**Context** Membership **inv**: name <> nil **Context** Membership **inv**: monthly_price <> nil **Context** Membership **inv**: duration <> nil				
Exception	None				

3.1.2 user.control package

ClassName	User

Purpose	This control class allows a User to complete actions associated with login and profile modification				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	login	username password		Logs the user in with the appropriate permissions
	public	logout			Logs the user out
	public	loadProfile	username		Gets all relevant information related to the username
	public	updateProfile	profile	Boolean	Updates profile information and returns confirmation with Boolean
	public	registerCustomer	username password address creditcard driverLicense	Boolean	Allows a User to register as a customer
	public	vehicleSearch	name		Performs a search of vehicles in the fleet
Relationships	None				
Constraints	**Context** User::login(username, password) **pre**: username is not in the database **Context** User::login(username, password) **post**: username and password does not match				
Exception	updateProfile() throws **Exception** if fields are null or no information was				

	modified.

3.1.3 user.boundary package

ClassName	LoginUI				
Purpose	This boundary class allows a User to log in				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	login	username password		Verify the user and logs the user with appropriate permissions
Relationships	None				
Constraints	None				
Exception	None				

ClassName	LogoutUI				
Purpose	This boundary class allows a User to log out				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	logout			Logs the user out

Relationships	None
Constraints	None
Exception	None

ClassName	ChangeProfileUI				
Purpose	This boundary class allows a User to perform profile changes				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	updateProfile	profile		Updates the profile of a user
	public	saveProfile	profile	Boolean	Saves changes of profile change to data structure
Relationships	None				
Constraints	None				
Exception	None				

ClassName	RegisterUI			
Purpose	This boundary class allows a User to register as a customer			
SuperClass	None			
SubClass	None			
Attributes	**Visibility**	**Name**	**Type**	**Description**

Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	registerCustomer	username password address creditcard driverlicense	Boolean	Registers a customer
Relationships	None				
Constraints	None				
Exception	None				

ClassName	SearchUI				
Purpose	This boundary class allows a User to search the vehicle fleet				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	search	keyword	List<Vehicle>	Gets a list of available vehicles based on keyword
Relationships	None				
Constraints	None				
Exception	None				

3.1.4 administrator.control package

ClassName	UserAdd				
Purpose	This control class allows Administrator to add users of all permissions				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**		**Name**	**Type**	**Description**
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	addUser	username password	Boolean	Adds a user to the database with Boolean confirmation
Relationships	None				
Constraints	None				
Exception	addUser() thows **Exception** with username or password is null				

ClassName	VehicleRemove				
Purpose	This control class allows Administrator to remove a vehicle from the system				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**		**Name**	**Type**	**Description**
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	removeVehicle	vehicleProfile	Boolean	Removes a vehicle from the fleet with Boolean conformation
Relationships	None				
Constraints	None				

Exception	None

ClassName	VehicleTypeRemove						
Purpose	This control class allows an Administrator to remove a vehicle type from the system						
SuperClass	None						
SubClass	None						
Attributes	**Visibility**		**Name**		**Type**		**Description**
Operations	**Visibility**	**Name**		**Input**		**Output**	**Description**
	public	removeVehicleType		vehicleType		Boolean	Removes a vehicleType from the database with confirmation
Relationships	None						
Constraints	**Context** VehicleTypeRemove::removeVehicleType(vehicleType) **post**: vehicleType is not in the dataset						
Exception	removeVehicleType() throws an **Exception** if vehicleType is null						

3.1.5 administrator.boundary package

ClassName	UserAddUI						
Purpose	This boundary class allows an Administrator to add a user to the domain						
SuperClass	None						
SubClass	None						
Attributes	**Visibility**		**Name**		**Type**		**Description**
Operations	**Visibility**	**Name**		**Input**		**Output**	**Description**

	public	addUser	username password	Boolean	Adds a user to the system confirmation
Relationships	None				
Constraints	None				
Exception	None				

ClassName	RemoveVehicleUI				
Purpose	This boundary class allows Administrator to remove a vehicle from the fleet				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	removeVehicle	vehicleProfile	Boolean	Removes a vehicle from the fleet with conformation
Relationships	None				
Constraints	None				
Exception	None				

ClassName	RemoveVehicleTypeUI			
Purpose	This boundary class allows Administrator to remove a vehicleType from the system			
SuperClass	None			
SubClass	None			
Attributes	**Visibility**	**Name**	**Type**	**Description**

Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	removeVehicleType	vehicleType	Boolean	Removes a vehicleType from the site with confirmation
Relationships	None				
Constraints	None				
Exception	None				

3.1.6 customer.control package

ClassName	Membership				
Purpose	This control class allows a Customer to perform actions to their membership options				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	terminateMembership		Boolean	Terminates a membership of a customer with Boolean confirmation
Relationships	None				
Constraints	**Context MembershipCtrl::** terminateMemberShip(): Termination will be triggered by a button within the CustomerUI boundary				
Exception	None				

ClassName	VehicleReservation			
Purpose	This control class allows a Customer to preform actions related to renting a vehicle			
SuperClass	None			
SubClass	None			
Attributes	**Visibility**	**Name**	**Type**	**Description**
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	reserveVehicle	vehicleProfile	Boolean	Reserves a vehicle with Boolean confirmation
	public	returnVehicle	vehicleProfile	Boolean	Checks a vehicle back into the fleet of available vehicles with Boolean confirmation
Relationships	None				
Constraints	**Context ReservationCtrl::** reservation(vehicleProfile) pre: vehicleProfile is in the data set **Context ReservationCtrl::** returnVehicle(vehicleProile) pre: vehicleProfile is contained within the dataset				
Exception	None				

3.1.7 customer.boundary package

ClassName	MembershipUI			
Purpose	This boundary class allows a Customer to terminate their membership			
SuperClass	None			
SubClass	None			
Attributes	**Visibility**	**Name**	**Type**	**Description**

Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	terminateMembership		Boolean	Terminates a customer's membership with conformation
Relationships	None				
Constraints	None				
Exception	None				

ClassName	ReservationUI				
Purpose	This boundary class allows a Customer to reserve a vehicle				
SuperClass	None				
SubClass	None				
Attributes	**Visibility**	**Name**	**Type**	**Description**	
Operations	**Visibility**	**Name**	**Input**	**Output**	**Description**
	public	reserveVehicle	vehicleProfile	Boolean	Reserves a vehicle with Boolean confirmation
	public	returnVehicle	vehicleProfile	Boolean	Checks a vehicle back into the fleet of available vehicles with Boolean confirmation
Relationships	None				
Constraints	None				
Exception	None				

3.2 Factory methods for the persistent and object layers

3.2.1 entity

package uga.cs.x050.team4.entiy;

public interface EntityFactory {

public Customer createCustomer(Int userID, String ssn, String firstName, String lastName, String address, String telephone) throws CarRentalException;

public Administrator create Administrator(Int userID, String firstName, String lastName, String address, String telephone) throws CarRentalException;

}

3.2.2 Persistent

package uga.cs.x050.team4.persistent;

public interface EntityPersistentFactory {

public Customer storeCustomer(Int userID, String ssn, String firstName, String lastName, String address, String telephone) throws CarRentalException;

public Customer restoreCustomer(Int userID) throws CarRentalException;

public Iterator restoreCustomer() throws CarRentalException;

public Administrator storeAdministrator(Int userID, String firstName, String lastName, String address, String telephone) throws CarRentalException;

public Administrator restoreAdministrator(Int userID) throws CarRentalException;

public Iterator restoreAdministrator() throws CarRentalException;

public DriversLicense storeDriversLicense(String number, String expiration, String dateOfBirth) throws CarRentalException;

public DriversLicense restoreDriversLicense(String number) throws CarRentalException;

public Iterator restoreDriversLicense() throws CarRentalException;

public CreditCard storeCreditCard(String number, String expiration, Int code) throws CarRentalException;

public CreditCard restoreCreditCard(String number) throws CarRentalException;

public Iterator restoreCreditCard() throws CarRentalException;

public VehicleType storeVehicleType(String name, Double hourly_rate, Double daily_rate) throws CarRentalException;

public VehicleType restoreVehicleType(String name) throws CarRentalException;

public Iterator restoreVehicleType() throws CarRentalException;

public Vehicle storeVehicle(String vehicleID, String make, String model, Int year, String tag, Int mileage, Date last_serviced, String condition) throws CarRentalException;

public Vehicle restoreVehicle(String vehicleID) throws CarRentalException;

public Iterator restoreVehicle() throws CarRentalException;

public Location storeLocation(String name, String address, Int vehicle_capacity, List<Vehicle> vehicles) throws CarRentalException;

public Location restoreLocation(String name) throws CarRentalException;

public Iterator restoreLocation() throws CarRentalException;

public Membership storeMembership(String name, Double monthly_price, Int duration) throws CarRentalException;

public Membership restoreMembership(String name) throws CarRentalException;

public Iterator restoreMembership() throws CarRentalException;

}

3.2.3 Association

```
package uga.cs.x050.team4.associations;

public interface aggreagationModel {

        public aggreagationModel(long id) throws CarRentalException;
        public Iterator restoreAggreagationModels(Entity E) throws CarRentalException;

        public long storeAggregationModel(AggregationModel m) throws CarRentalException;

};


public interface reservationBy {

         public reservationBy     restoreReservationBy(long id) throws CarRentalException;

        public Iterator            restoreReservationBy(Customer C) throws CarRentalException;

       public  Iterator           restoreReservationBy(Vehcile V) throws CarRentalException;

      public long                storeReservationBy(reservationBy) throws CarRentalException;

};


public interface administratorMaintainsCustomer extends aggregationModel {

        public aggreagationModel(long id) throws CarRentalException;

        public Iterator restoreAdministratorMaintainsCustomer(Customer C) throws
        CarRentalException;


      public AdministratorMaintainsCustomer
restoreAdministratorMaintainsCustomer(Customer C)     throws CarRentalException;


        public long storeAggregationModel(CustomerProfile m) throws CarRentalException;


};


public interface adminMaintainsCustomer extends aggregationModel {
```

public aggreagationModel(long id) throws CarRentalException;

public Iterator restoreAdministratorMaintainsCustomer(Customer C) throws CarRentalException;

public AdministratorMaintainsCustomer restoreAdministratorMaintainsCustomer(Customer E) throws CarRentalException;

public long storeAggregationModel(CustomerProfile m) throws CarRentalException;

};

public interface Locations extends aggregationModel {

public aggreagationModel(long id) throws CarRentalException;

public Iterator Location restoreLocation(Location L) throws CarRentalException;

public Location restoreAdministratorMaintainsCustomer(Location L) throws CarRentalException;

public long storeAggregationModel(Location L) throws CarRentalException;

};

public interface maintainVehicles {

public maintainVehicle restoreMaintainVehicle(long id) throws CarRentalException;

public Iterator restoreMaintainVehicle (Vehicle V) throwCarRentalException;

public Iterator restoreMaintainVehicle (VehcileInfo D) throws CarRentalException;

public long storeReservationBy(maintainVehicles) throws CarRentalException;

};

public interface maintainCustomerInfo {

public maintainVehicle restoreMaintainVehicle(long id) throws CarRentalException;

public Iterator restoreMaintainVehicle (Customer C) throwCarRentalException;

public Iterator restoreMaintainVehicle (CustomerInfo D) throws CarRentalException;

public long storeReservationBy(maintainCustomerInfo) throws CarRentalException;

};

4. Glossary

Abstract Factory pattern: provides an abstract class for each object that can be substituted and provides an interface for creating groups of objects

Adapter pattern: provides a different interface to en existing component, used to convert the interface of an existing piece of code into an interface that a calling subsystem expects

Analysis object model: describes the entity, boundary, and control objects that are visible to the user

Boundary use cases: describe, from the user's point of view, administrative and exceptional cases that the system handles

Contracts: constraints on a class that enable class users, implementers, and extenders to share the same assumption about the class. Contracts include three types of constraints: invariant, precondition, and postcondition.

Delegation: the alternative to implementation inheritance that should be used when reuse is desired.

Design pattern: a template solution that developers have refined over time to solve a range of recurring problems. A design pattern includes a name, a problem description, a solution, and a set of consequences. There are three types of patterns: creational, structural, and behavioral patterns

Development Cost: cost of developing the initial system.

Façade pattern: allows developers to further reduce dependencies between classes by encapsulating a subsystem with a simple, unified interface

Invariant: a predicate that is always true for all instances of a class. Invariants are constraints associated with classes or interface. Invariants are used to specify consistency constraints among class attributes

Object Constraint Language (OCL): a formal language defined as part of the UML used for expressing constraint

Object Design Document (ODD): a document describing the object design model. The object design model is often generated from comments embedded in the source code. There are three main approached to document object design: Self-contained ODD generate from model, ODD as extension of the RAD, and ODD embedded into source code

Object model restructuring: transform the object design model to improve its understandability and extensibility.

Object model optimization: transform the object design model to address performance criteria such as response time or memory utilization.

Postcondition: a predicate that must be true after an operation is invoked, used t specify constraints that the class implementor and the class extender must ensure after the invocation of the operation

Precondition: a predicate that must be true before an operation is invoked associated with a specific operation. Preconditions are used t specify constraints that a class user must meet before calling the operation

Reuse: indentify off-the shelf components and design pattern to make use of existing solution.

Scalability: a system's ability to process more workload, with a proportional increase in system resource usage.

Service specification: describe each class interface.

Simplicity: a system should be well-designed, system, and tools are usually reliable, easy to use and maintain, and simple in concept.

Singleton pattern: ensure a class only has one instance, and provide a global point of access to it