INFO212 - assignment 1

By:
Pål Hagen Størksen
Helge Hitland

Class diagram Explanation:

This class diagram represents the domain model for the car rental service, with classes for the branch, registering a car, the customer, sending in an order, agreed appointment, delivery of car, eventual fees, general keeping of history, and the employee which can take requests from certain individuals.

Regarding Customer class, attribute “score”: The score is assigned in the appointment class if he meets or does not meet, and this score is updated in the Fee class and Delivery class if the car has damage upon delivery or/and if the customer gets a fee. This score is kept by the History class (which is the general record of the customer), which some of the other classes have access to. The Employee class can also update the score.

Use case diagram Explanation:

This Use case diagram describes the way the customer interacts with the system for the online car rental service at different points of the time the rental service is being provided. And also how the different employees interact with the system to provide the rental service to the customer. The diagram also describes how the different actions interact with each other within the system.

Activity diagram Explanation:

This activity diagram describes the customer's process of renting a car from the online car rental company.

Regarding Request DL (e.g drivers license): In this case we have added pause order and request drivers license and in this case the driver's license is not registered, as this felt natural in this case.

Use Case Model: Online Rental Service (Website + mobile app)

Search for model, specify rental location etc.

Place order and insurance

Provide credit card information

Customer Registration

Log in to system, and confirm order

Customer signs contract, and receives keys

Pick up car

Misses appointment

Smoking detected

Deliver car

Damage detected

Full insurance

Inform insurance comp., and make claim

Not full insurance

Check insurance

Deduct fee or send invoice

Hands over car

Keep history of customer

Check history

Give discount

No discount

Create damage record, send invoice to customer

Review requests, from cust. with not good background

Car Registration (with ID, make, model, color, rental group etc

Transport car to branch after repair

Customer

Branch

Repair/workshop

Executive Employee

System

Admin

<<Include>>

<<extend>>

UML Class Diagram: domain model

Customer

+ driverslicenseNumber: int[1..*]
+ personalID: int[1] {readOnly}
+ fullName: String[1]
+ phoneNumber: int[1]
+ eMail: String[1], int[0..*]
+ address: String[1..*], int[1..*]
+ creditCard: String[1..*], int[1]
+ score: int[1]

+ register(): String, int
+ searchCar(): String, int
+ addcreditCard(): String, int
+ orderCar(): String, int
+ comfirmOrder(): String
+ logintoconfirmDL(): String, int
+ makeRequest(): String

Car

+ UniqueID: int[1] {readOnly}
+ make: String[1] {readOnly}
+ model: String[1] {readOnly}
+ color: String[1..*]
+ rentalGroup: String[1] {readOnly}

Car Rental Company

Branch

+ registerCar(): String, int
+ handoverCar(): String, int

* — 1

Employee

+ reviewHistory(): String, int
+ updateScore(): int[1]

History

+ driverslicenseNumber: int[1..*]
+ personalID: int[1] {readOnly}
+ fullName: String[1]
+ orderID: int[1]
+ damageRecord: String = "Empty" {readOnly}
+ contract: String, int
+ feeID: int[1] = "0" {readOnly}
+ score: int[1]

+ addHistory(): String, int

1 — *

Order

+ personalID: int[1] {readOnly}
+ UniqueID: int[1] {readOnly}
+ orderID: int[1]
+ rentalLocation: String[1]
+ returnLocation: String[1]
+ startTime: int[1]
+ rentalDuration: int[1]
+ insurancePol: String
+ insurancePrice: int[1]
+ creditCard: String[1..*], int[1]
+ driverslicenseNumber: int[1..
+ score: int[1]

+ confirmDL(): String, int
+ checkHistory(): boolean
+ giveDiscount(): int[1]
+ confirmOrder(): String

1 — 1

Appointment

+ personalID: int[1] {readOnly}
+ UniqueID: int[1] {readOnly}
+ orderID: int[1]
+ rentalLocation: String[1]
+ startTime: int[1]
+ contract: String, int

+ signContract(): String, int
+ noArrival(): boolean
+ assignScore(): int[1]

1 — 1

Delivery

+ personalID: int[1] {readOnly}
+ UniqueID: int[1] {readOnly}
+ orderID: int[1]
+ returnLocation: String[1]
+ rentalDuration: int[1]
+ insurancePol: String
+ insurancePrice: int[1]
+ damageRecord: String = "Empty" {readOnly}

+ cleaningPolicy(): boolean
+ checkDamage(): boolean
+ insuranceType(): String
+ createdamageRecord(): String
+ informinsuranceC(): String
+ makeClaim(): String, int
+ repairCar(): String
+ transferCar(): String
+ updateScore(): int[1]

1 — 1

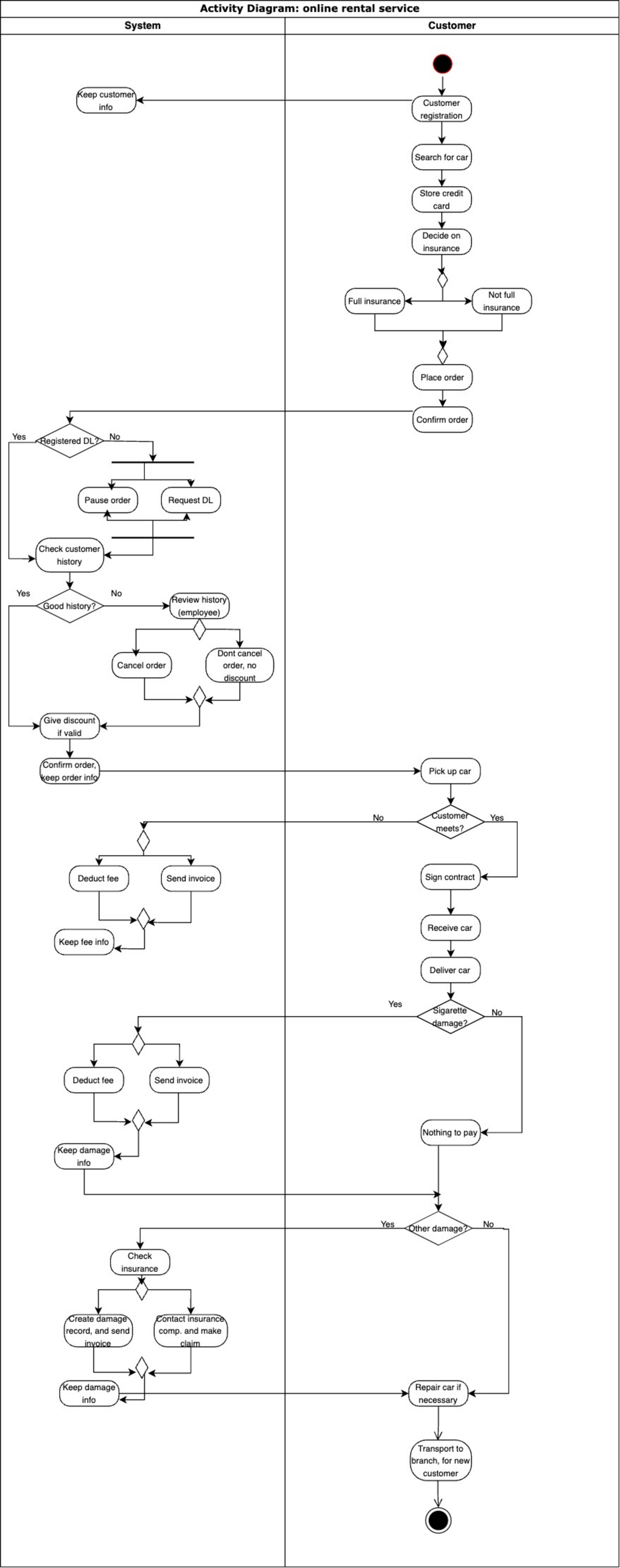
Fee

+ feeID: int[1] = "0" {readOnly}
+ orderID: int[1]
+ feeAppointment: int[*]
+ feeCleaning: int[3000]
+ creditCard: String[1..*], int[1]
+ fullName: String[1]
+ address: String[1..*], int[1..*]

+ deductFee(): int[0..*]
+ sendInvoice(): String, int
+ updateScore(): int[1]

1 — 1

{If Order.checkHistory() returns False, then History.feeID returns "0", and History.damageRecord returns "Empty".}


Activity Diagram: online rental service
System
Customer
Keep customer info
Customer registration
Search for car
Store credit card
Decide on insurance
Full insurance
Not full insurance
Place order
Confirm order
Yes
Registered DL?
No
Pause order
Request DL
Check customer history
Yes
Good history?
No
Review history (employee)
Cancel order
Dont cancel order, no discount
Give discount if valid
Confirm order, keep order info
Pick up car
No
Customer meets?
Yes
Deduct fee
Send invoice
Keep fee info
Sign contract
Receive car
Deliver car
Yes
Sigarette damage?
No
Deduct fee
Send invoice
Keep damage info
Nothing to pay
Yes
Other damage?
No
Check insurance
Create damage record, and send invoice
Contact insurance comp. and make claim
Keep damage info
Repair car if necessary
Transport to branch, for new customer