Assignment 3

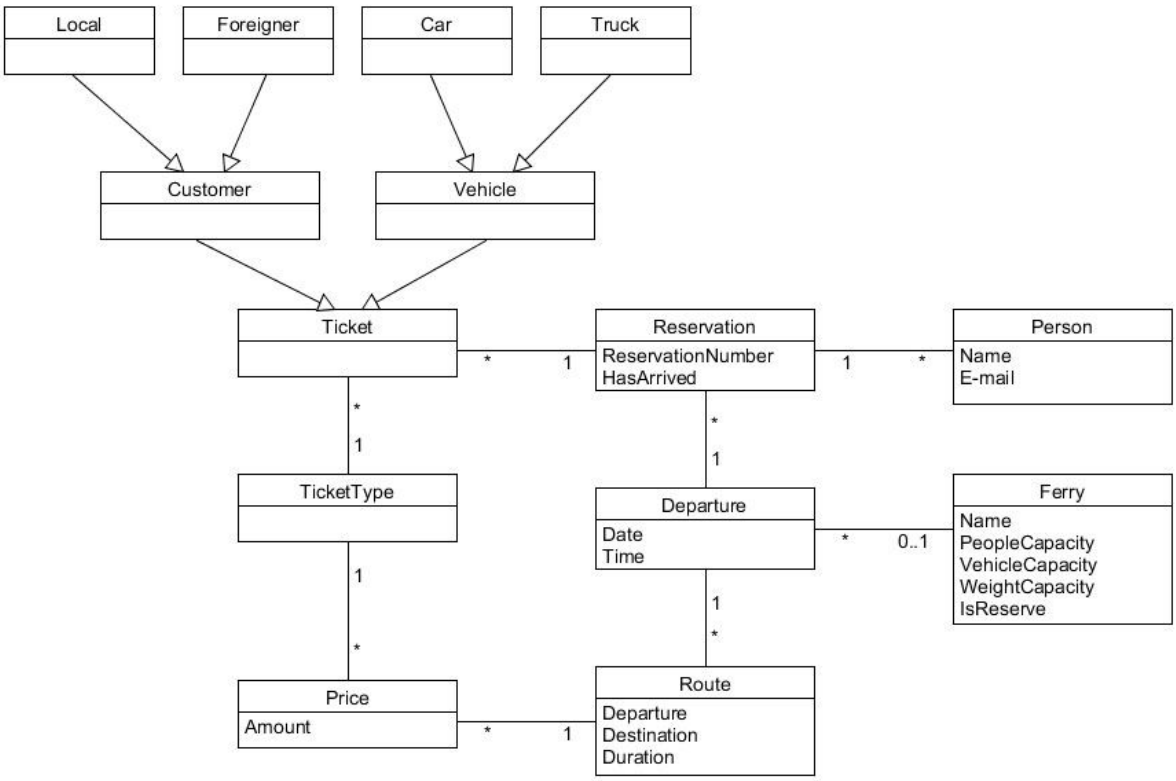
Contract operation assignment

Team Hundige: Søren BS, Casper J, Martin B
Team Legoklods: Søren E, Mads J, Michael N & Jonas P

This assignment and entire interface is written by team hundige, due to trouble with getting response from team legoklods.

Logical data model

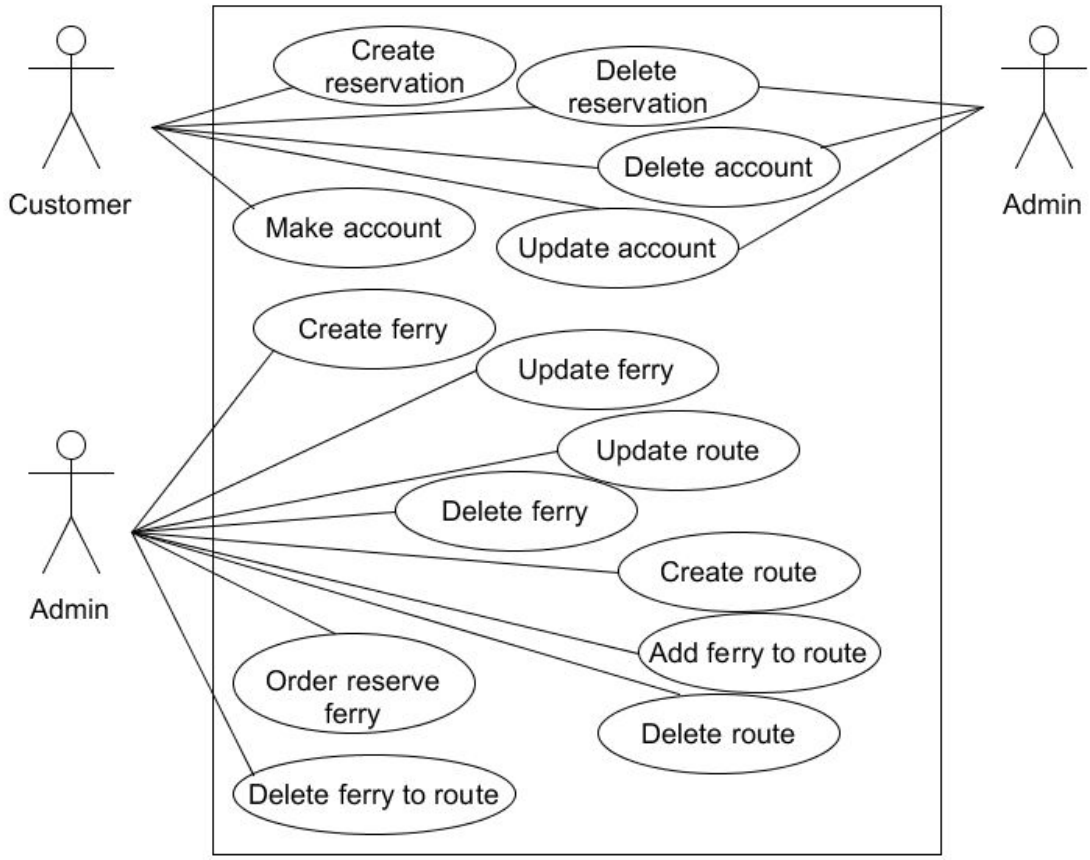
Our logical data model is based on our two clusters communication around our previous assignments, along with the discussion in our class and the examples given. We have a travel entity that inherits from customer and vehicle, which in term has a price through the traveling entity type. To avoid the issue revolving around making reservations for several vehicles, we've made it so that one reservation can contain several ticket, fixing it that way.



The reservation is for a specific departure, which has one or more ferries available and a route to go by. The price is based on the details of the ticket.

Use case model

In our use case model we've got two actors: the customer and the administrator.



Use cases

Create reservation Delete reservation	Create ferry Delete ferry Update ferry configuration Order reserve ferry	Create route Delete route Update route
Add ferry to route Delete ferry from route	Create account Delete account Update account information	

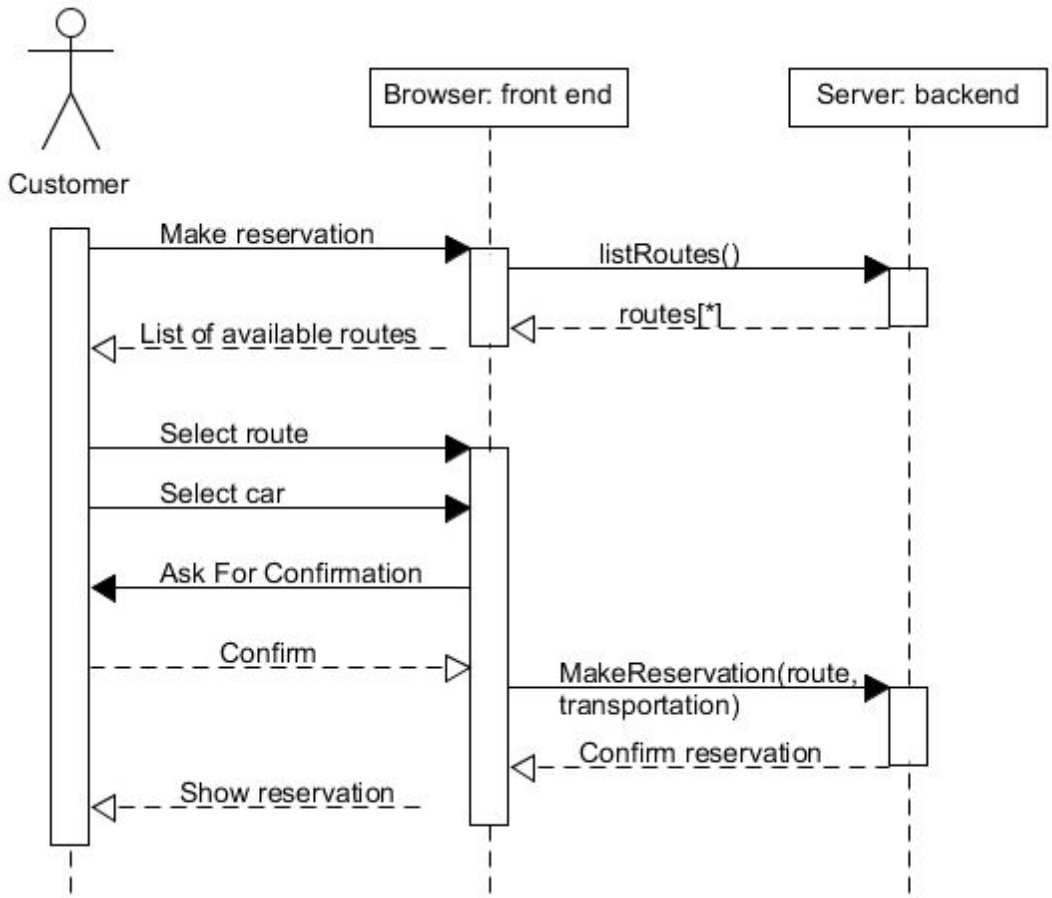
Fully dressed use cases

Use case - Create reservation
Primary actor: Customer
Goal: The customer makes a successful ticket(s) reservation
Scope: System under Development (SuD)
Stakeholders & Interests: Stakeholders = Ferry Company, Interests = Customer
PreCondition: Custom have created an account and is logged in to the system
Minimal Guarantees: Error message: Reservation failed
Success Guarantees: Success message: Reservation completed
Trigger:
Main Success Scenario: Customer logs in, Customer initiate ticket reservation, Customer fills out reservation data, Customer receives “Reservation completed” message.

Use case - Delete reservation
Primary actor: Customer
Goal: The customer deletes a ticket reservation
Scope: System under Development (SuD)
Stakeholders & Interests: Stakeholders = Ferry Company, Interests = Customer
PreCondition: Custom have created an account and is logged in to the system, and have a reservation
Minimal Guarantees: Error message: Deletion of reservation failed
Success Guarantees: Success message: Deletion of reservation completed
Trigger:
Main Success Scenario: Customer logs in, Customer initiate deletion of reservation, Customer approve deletion, Customer receives “Deletion of reservation completed” message.

System sequence diagram

We've decided to make the "Make reservation" our highest priorities use case, and will use from here on.



Operation contract

Operation : Make reservation - CreateReservation(Departure departure);

Cross Reference:

Preconditions: - User is logged into system

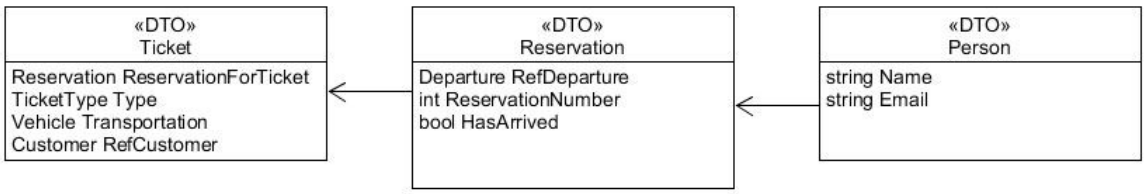
Postconditions: - ListRoutes() instant is requested for specific day

- Return list of available departures of specified day
- Desired departure is selected +
  and necessary vehicle information is filled
- Confirm reservation
- Reservation is made
- Reservation is associated with customer and confirmed

- Visual confirmation message is shown to customer

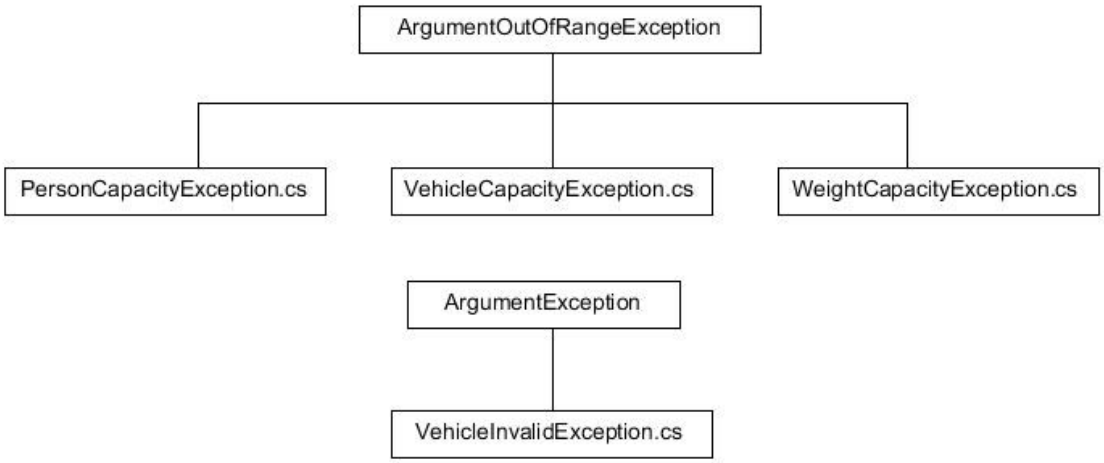
Design transfer object diagram:

Based on the Use case “Make reservation” this is how our DTO looks like.



Exception transfer objects

For our exception, we looked at how capacity and vehicle type was important for the ferry.



Communications Model

Git repository containing Common Interface and DTO/ETO Library can be read at:
https://github.com/xxblancxx/FerryContract2.0.git