Maarten Sijmkens (r0616538)

Jonas Vervloet ()

**Distributed Systems: Java RMI**

Our system has five core components: the client, the rental server, the car rental company, the manager session and the reservation session. The client component enables clients to connect to the rental server and make quotes and reservations. The company component provides methods to create quotes for each company. The rental server is the central part of the system. It creates and manages sessions and it serves as communication between companies and clients. The clients and company only needs to locate the rental server, not each other.

The reservation session component creates and stores quotes for a client. The stored quotes can then be confirmed into a reservation. The manager session provides useful methods for managers like searching for the best customer or the most popular car type.

In our system, four classes are remotely accessible: the rental server, the reservation session, the manager session and the car rental company. The rental server is remote because clients need to be able to connect to it and start sessions. The reservation session is server side because if a client creates quotes etc., the server should be aware of this. The manager session is also server side because a manager should have up to date information. Finally, the car rental company needs to be remotely accessible because the rental server needs to be able to check whether or not a car is available for a given period. This information is stored in the car rental company class, but needs to be available to the rental server.

Three classes are serializable: reservations, quotes and car types. The client has to receive reservations, but shouldn’t modify them. It’s static data. The same holds for car types and quotes. This information should be transferred between the servers as static data.

The server hosts three remote objects: RentalServer, ReservationSession and ManagerSession. Each car rental company has its own host with a remote object of the CarRentalCompany class.

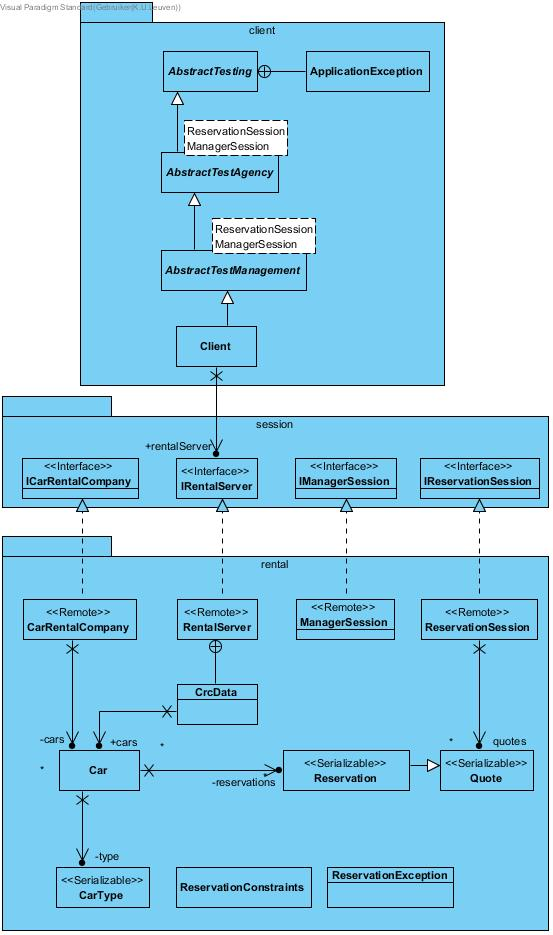
The RentalServer is registered in the rmi registry. This way the client can make first contact with the central server. Later the client will interact with the ReservationSession or ManagerSession. These are not in the registry because they shouldn’t be accessible to everyone. The RentalServer class will return them only to the client who created the session.

The companies all have a CarRentalCompany registered in the rmi registry so that the sessions can interact with them. But not any company who registers itself in the rmi registry will be approached by the RentalServer. A manager has to register the name of the companies in the RentalServer’s own registry. This way the server knows which companies to look up and which not.

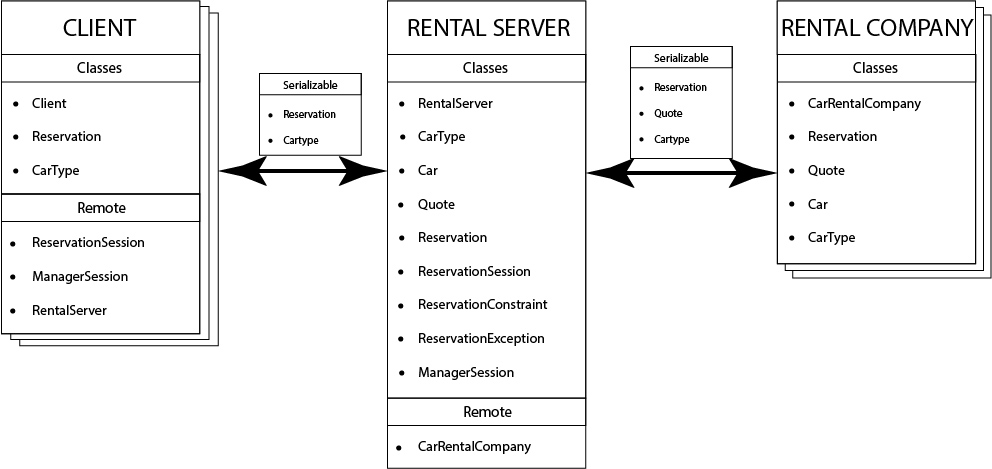
All created sessions are stored by the RentalServer with a date of creation. When a session is closed, the RentalServer will remove this object and unexport its stub. This way all references to it will be removed and the object will be deleted from memory by the java garbage collector. A session can be closed by the client who created the session. It can also be closed by the server if it sees the session is too old, using the date of creation. A better method, not implement though, would be to store a last-active field and close the session if the session has been idle for too long.

The confirmQuotes method in ReservationSession is synchronised. This way reservations won’t be made at the same time and no car can be double booked. Other methods, like createQuote are not synchronised and therefor a car might be double quoted, but this is no final reservation so not really an issue. One may argue to make this synchronised for a better user experience, but this may slow down the server when there is a lot of traffic.

Also the registerCompany method in RentalServer is synchronised, so that no companies can be registered with the same name.

Class diagram

Deployment diagram



Sequence diagrams

*note that Client, RentalServer, Company A and Company B indicates the hosts, not the* *classes*.