KU LEUVEN DEPARTMENT OF COMPUTER SCIENCE

Distributed Systems: Java RMI session 2 & 3

Xavier Dejager
Seppe Duwé

October 30, 2015

1 WHICH CLASSES ARE REMOTELY ACCESSIBLE AND WHY?

SessionFactory
Because we need to be able to create and retrieve a client and manager session from a remote location.

ReservationSession
This allows us to perform operations as a renter on different (remote) car rental companies, trough the central agency

ManagerSession
This allows us to perform operations as a manager on different (remote) car rental companies, trough the central agency

CarRentalCompany
Allows the agency to remotely execute the operations requested by clients and managers on the different car rental companies.

2 WHICH CLASSES ARE SERIALIZABLE AND WHY?

Car CarType Quote Reservation ReservationConstraints
These classes have to be serializable because their objects will need be transferred by wire. As they are part of a return value from the methods found in remoteCarRental-Company, and these can be remotely invoked.

3 Which remote objects are located at the same host (or not) and why?

On the rentalAgency server we will find all of the session objects for the renters and the managers. This is necessary to be able to synchronize the operations invoked by the different clients in an efficient way. On the other hand, the carRentalCompany object will be located on a separate machine, possibly a different machine for every carRentalCompany. The reason for distributing the carRentalCompany object is because companies don't want to have their data on the same server as their competition, for obvious data protection reasons. Furthermore, this allows for a flexible system.

4 Which remote objects are registered via the build-in RMI registry (or not) and why?

Only the objects of the remoteSessionFactory are registered using the built-in RMI registry. This is because we need a way of initializing our sessions out of the blue. Once we have this, the sessions keep track of new remote object references. The built-in RMI registry only tells us where to look for a particular object of the remoteSessionFactory.

5 Briefly explain the approach you applied to achieve life cycle management of sessions

The java RMI runtime environment has built-in features to take care of remote garbage collection. Thus if an object is no longer referenced by a particular client, the garbage collector will remove its local reference (on the server) and eventually also the object. However it remains possible to override this behaviour in purpose of gaining better control of object life cycle management.

6 At which places is synchronization necessary to achieve thread-safety? Will those places become a bottleneck by applying synchronisation?

Only when we commit the quotes into the actual reservations, we need to have absolute certainty that there are no conflicts. Conflicts as in possible double reservations. Therefore we apply a synchronization when we perform the commit.
In first place this does not imply a real bottleneck. Unless we would scale it to massive proportions. Then this would inherently become a bottleneck because of the single point of failure. All reservations have to go through the carRentalAgency in a synchronized way. It would be possible to redesign the system in such way that multiple reservations are handled at once as long as the do not share any of the companies. This way we avoid these double reservation conflicts.

7 UML DIAGRAM

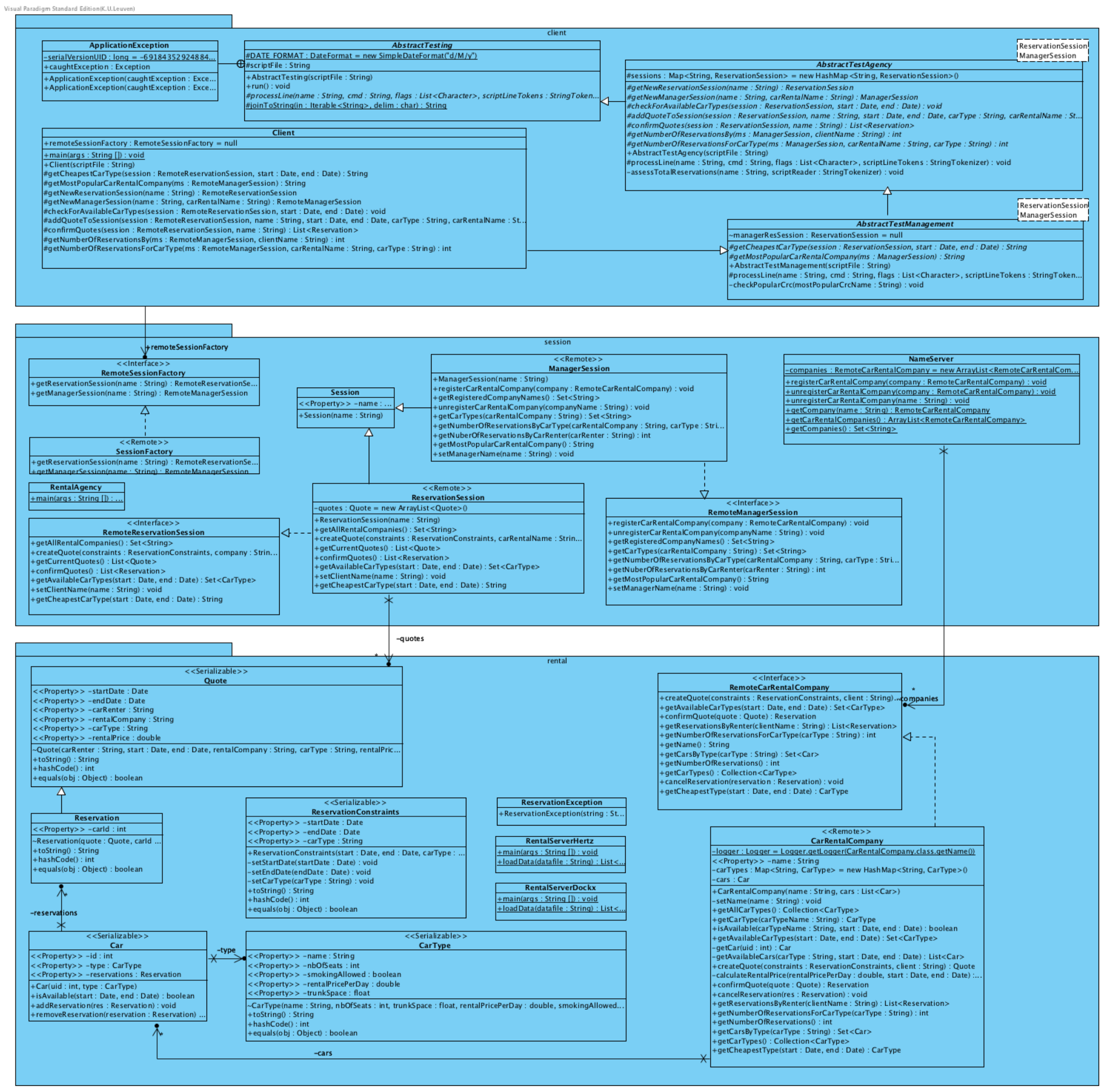

Figure 7.1: Complete UML diagram

8 Deployment diagram

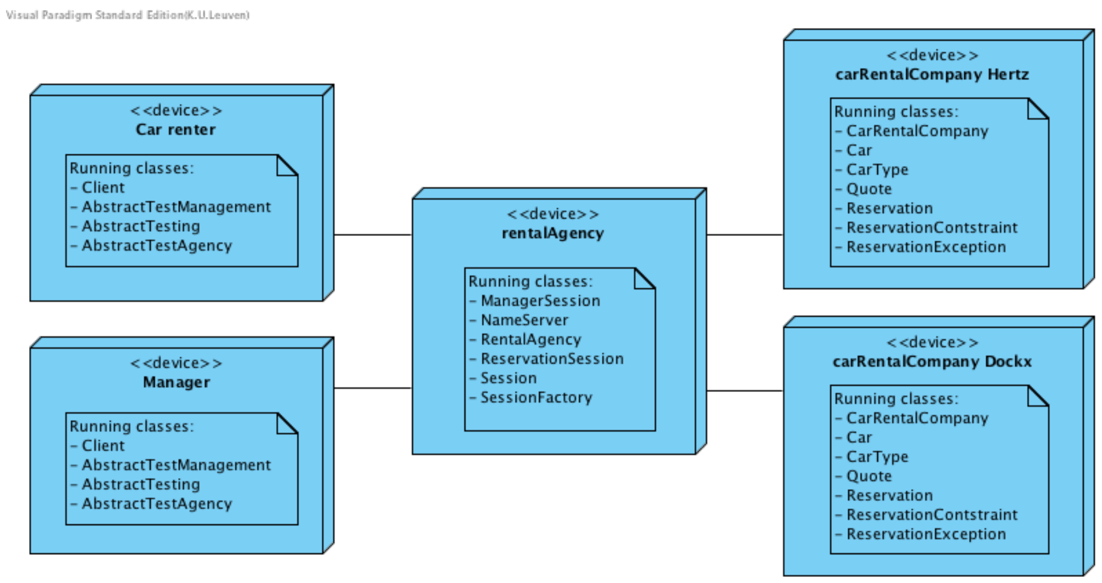

Figure 8.1: Deployment diagram

9 SEQUENCE DIAGRAM

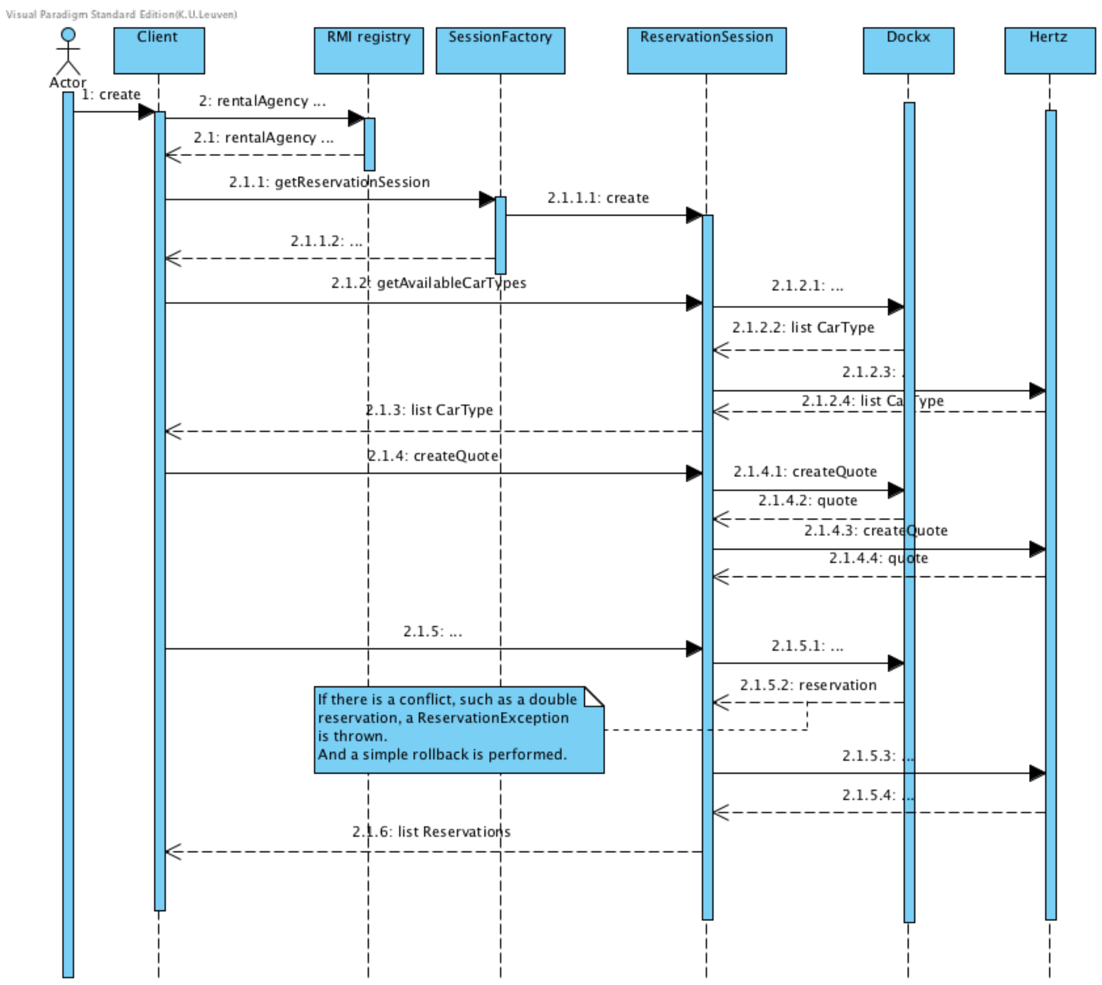

Figure 9.1: Deployment diagram