Distributed Systems: Java RMI session 2/3

Jago Gyselinck, Armin Halilovic

November 11, 2016

1 Overview

First, describe in 1 or 2 paragraphs the overview of your design. Which are the core parts/components and their responsibilities? This is not a sequential story! Next, at least the following design decisions should be discussed.

1.1 Serializable classes

The following classes are serializable:

- `Car, CarType, Quote, Reservation, ReservationConstraints`
- We made these classes serializable as data of their type has to be communicated between different distributed components.
- An example of this could be the `getAvailableCarTypes` method which returns `Set<CarType>` and is made available in the `CarRentalCompanyRemote` interface. The class `ReservationSession` makes use of this method, but resides on a different distributed component. For the method invocation on a remote reference of type `CarRentalCompanyRemote` to succeed, `CarType` must be serializable.

1.2 Remote classes

The following classes are remotely accessible:

- `CarRentalCompanyRemote, RentalAgencyRemote, ManagerSessionRemote, ReservationSessionRemote`
- We made these objects remotely accessible because their methods will be invoked from a non-local context, and remote references of their type will be passed along between different distributed components.

1.3 Remote Object Locations

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

- Only the sessions (Manager and Reservation) and the Rental Agency reside on the same host. This allows the client to request a remote reference for the `CarRentalAgency` via the rmiregistry, and remote references to sessions can be requested via that remote reference. Those session remote references are kept in the remote `CarRentalAgency` object so they can be removed later. This organisation also has the advantage that when a method is invoked on a `(Manager/Reservation)Session` through a remote reference, the `CarRentalAgency` it has to interact with is a static object on the same component, so no remote interaction is required, and synchronization is easily achieved.
- All other remote objects are located on different hosts. The car rental companies reside on their own server each

1.4 Registering of remote objects

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

1.5 Life cycle management

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

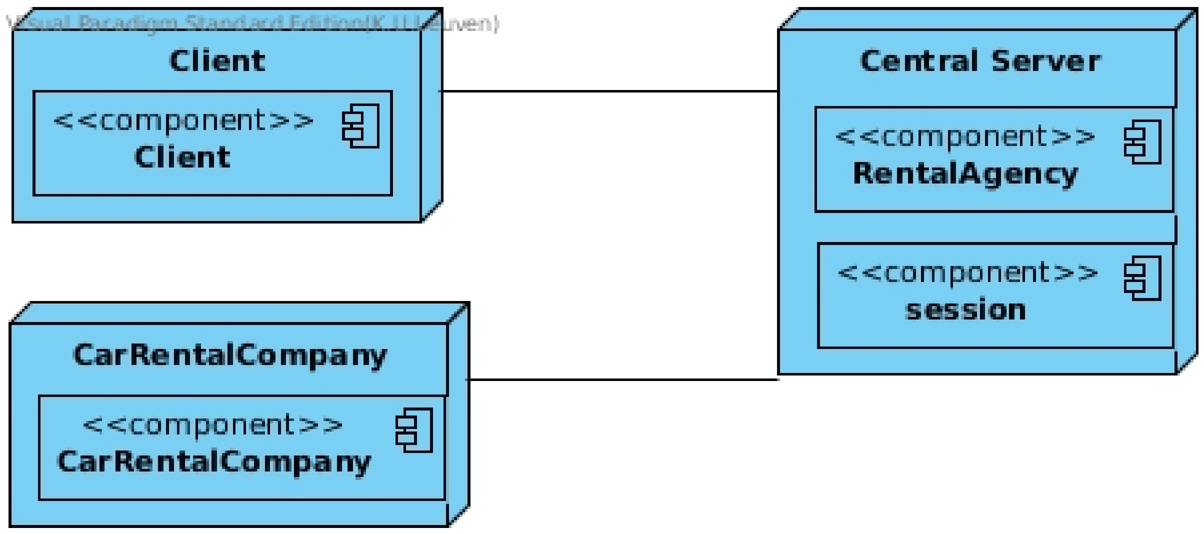
1.6 Synchronization

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

2 Full class diagram

See 'class-diagram.jpg'.

3 Deployment diagram



4 Sequence diagram

Sequence diagrams of the booking process have been included in the project. See 'sequence-diagram-success.jpg' and 'sequence-diagram-fail.jpg'.