Distributed Systems (H0N08A)

Report: Java RMI

Ruben Kindt *(r0656495)*
Yves Paquet *(r028549)*

October 29, 2020

How would a client complete one full cycle of the booking process, for both a successful and failed case?

See figure 1.

When do classes need to be serializable?

Classes need to be serializable when they or there data are send over the network, like the Quote class for example.

When do classes need to be remotely accessible(Remote)?

Classes need to be remote when they contain or share shared data, like the CentralRentalAgency, ReservationSession and the ManagerSession class.

What data has to be transmitted between client and server and back when requesting the number of reservations of a specific renter?

Only an integer, sensitive data is not meant to be transmitted.

What is the reasoning behind your distribution of remote objects over hosts? Show which hosts execute which classes, if run in a real distributed deployment (not a lab deployment where everything runs on the same machine).

The client and server are often not on the same physical network, In figure 2 you can see a client and the servers, the servers being the CenteralRentalAgenty and CarRentalCompany, with the CarRentalCompany being possibly multiple servers.

How have you implemented the naming service, and what role does the built-in RMI registry play?Why did you take this approach?

For the naming service we created one RMI registry where we store a remote object of the CentralRentalAgency which the client can pull from. We use the same RMI registry for storing our remote objects of the different CarRentalCompany's. We took this approach because in a real distributed deployment the CarRentalCompany's will not be on the same machine and will need to communicate with each other. For a real distributed deployment it would also be wise to use two different RMI registries, but for convenience we used one registry

Which approach did you take to achieve life cycle management of sessions? Indicate why you picked this approach, in particular where you store the sessions.

With our approach we expect that the client will ask for a creation and deletion of a session to the CenteralRentalAgenty, in the CenteralRentalAgenty the sessions are stored. The CenteralRentalAgenty will return a remote object to the client for the creation of the session. This approach requires less data to be transmitted, this approach is also commonly used when asking a session to a site where the site responds with a session cookie.

Why is a Java RMI application not thread-safe by default? How does your application of synchronization achieve thread-safety?

Java RMI is not thread-safe by default, because remote method invocation on the same remote object can execute concurrently and can cause a race-condition. Our application achieve thread-safety by implementing our race-condition prone objects with serializable, like the Car, CarType, Quote and the Reservation class.

How does your solution to concurrency control affect the scalability of your design? Could synchronization become a bottle neck?

Due to the implementation of serializable one of the two concurrent changes on the same object gets put on hold while the other preforms its changes. In the case that multiple changes all except one gets put on hold until that one is finished. Therefor our solution is not scalable and synchronization will be the bottle neck.

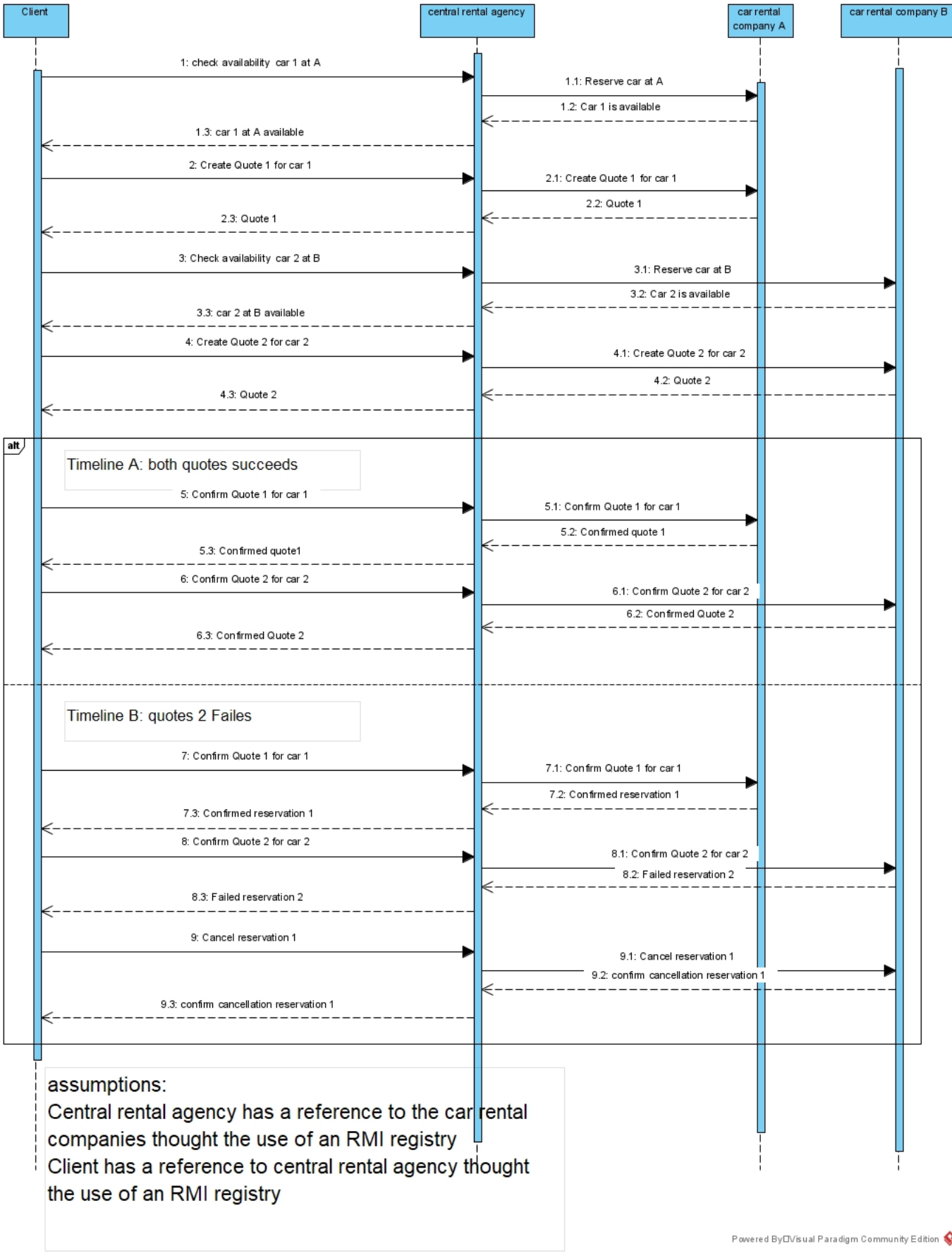

Figure 1: Sequence diagram for question 1

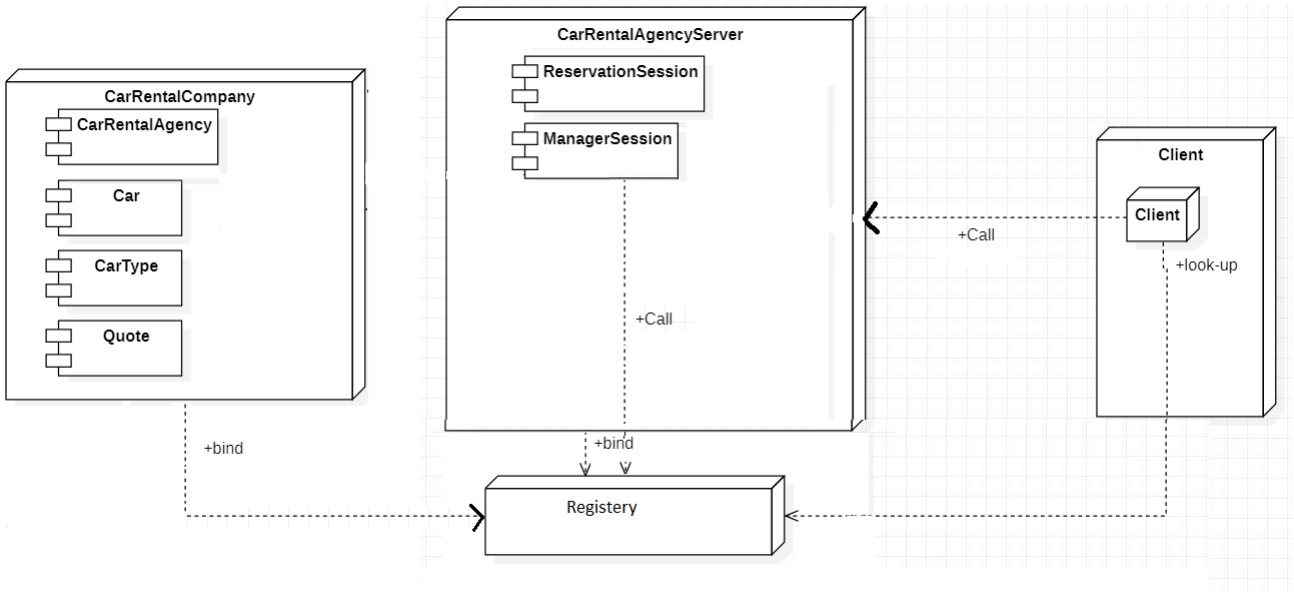

Figure 2: component/deployment diagram for question 5