Distributed Systems (H0N08A)

# *Report:* Java RMI

Bart van Bommel *(r0620692)*
Wouter Mertens *(r0746877)*

October 29, 2020

**How would a client complete one full cycle of the booking process, for both a successful and failed case? Base yourself on the example scenarios in Figure 1. Create sequence drawings to illustrate this.** See sequence diagrams: figure 1 & figure 2

**When do classes need to be serializable? You may illustrate this with an example class.** A class needs to be serializable whenever it's expected the class will be sent over the web as a parameter or return value. for example: CarType and Reservation are classes which will be returned to the client by value and as such need to be serializable. The other classes are used in calculations, but are never sent to the Client or the naming service, as such they don't need to be serializable and will only be used locally.

**When do classes need to be remotely accessible (Remote)? You may illustrate this with an example class.** A class should be remotely accessible when it is non-locally invoked and references are used between several distributed systems. For example: IReservationSession is used as a parameter in several Client methods while also being the interface for ReservationSession, which is remotely located.

**What data has to be transmitted between client and server and back when requesting the number of reservations of a specific renter?** First, The client needs to find the naming service in the registry. Once it's found, the client will request a manager session. The assignment left out user authentication, so no password or username is needed. When the client has been given the address of a manager session, it will request to invoke the getNrOfReservationsByUser() method. Once the method is requested, the manager session will look up all registered companies, which are saved in a list on the naming service and will iterate over each company. Every company will be asked for the reservations made by the user, so only the name is sent over the web to the companies. the Company then either sends an empty list or a list with all the reservations back to the manager session. The session will then count all reservations and return an integer.

**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). Create a component/deployment diagram to illustrate this: highlight where the client and server are.** To make adding/removing RentalCompanies as easy as possible and to remove the dependency of the NamingService on the data of the RentalCompanies, we locate the RentalCompanies and NamingService on different hosts. The Client is a separate host as there can be a high number of them and they only interact with the NamingService. The different sessions and the NamingService are located on the same host as the sessions are created by the NamingService and most of their methods work via the NamingService. (figure 3)

**How have you implemented the naming service, and what role does the built-in RMI registry play? Why did you take this approach?** The naming service fulfills the role of a server which allows companies to register themselves and clients to perform queries on the registered companies. The built-in RMI allows us to implement the message passing and to remotely invoke methods on objects that are on the server from the client or the company. It also allows the naming service to query the registered companies for information. Through RMI, the connection and the functionality can be implemented in a way similar to normal Java code.

**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.** The sessions are stored and created in the NamingService. The ManagerSession holds no client-specific data and therefor does not require a cleanup. The ReservationSession is statefull and has a method to clean itself, this method could be invoked by the server after a timer runs out, it is however not implemented.

**Why is a Java RMI application not thread-safe by default? How does your application of synchronization achieve thread-safety?** Java RMI implements at-most-once semantics. This means it is possible that calls on objects will not be performed as expected. Since objects may be visible to multiple threads, it is possible that multiple calls could potentially interfere with each other. This is where the "synchronized" keyword comes to play. When an object is synchronized, it will block calls to this object until the currently invoked method is completed. this prevents unexpected behaviour like performing calls on data that is still being changed.

**How does your solution to concurrency control affect the scalability of your design? Could synchronization become a bottleneck?** We labeled every method that edits data on the server as synchronized, this means that given a large number of users the system could slow down significantly due to wait times.

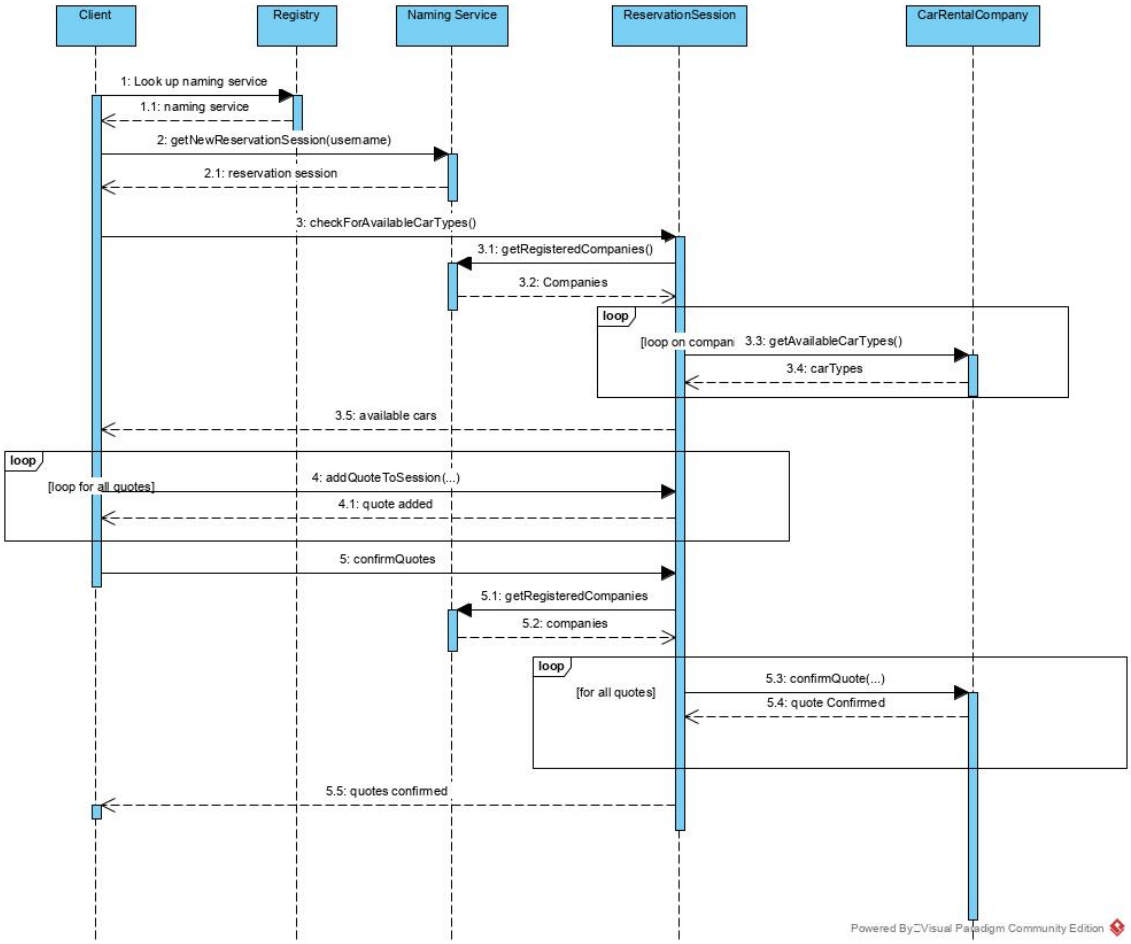


Figure 1: Successful sequence diagram

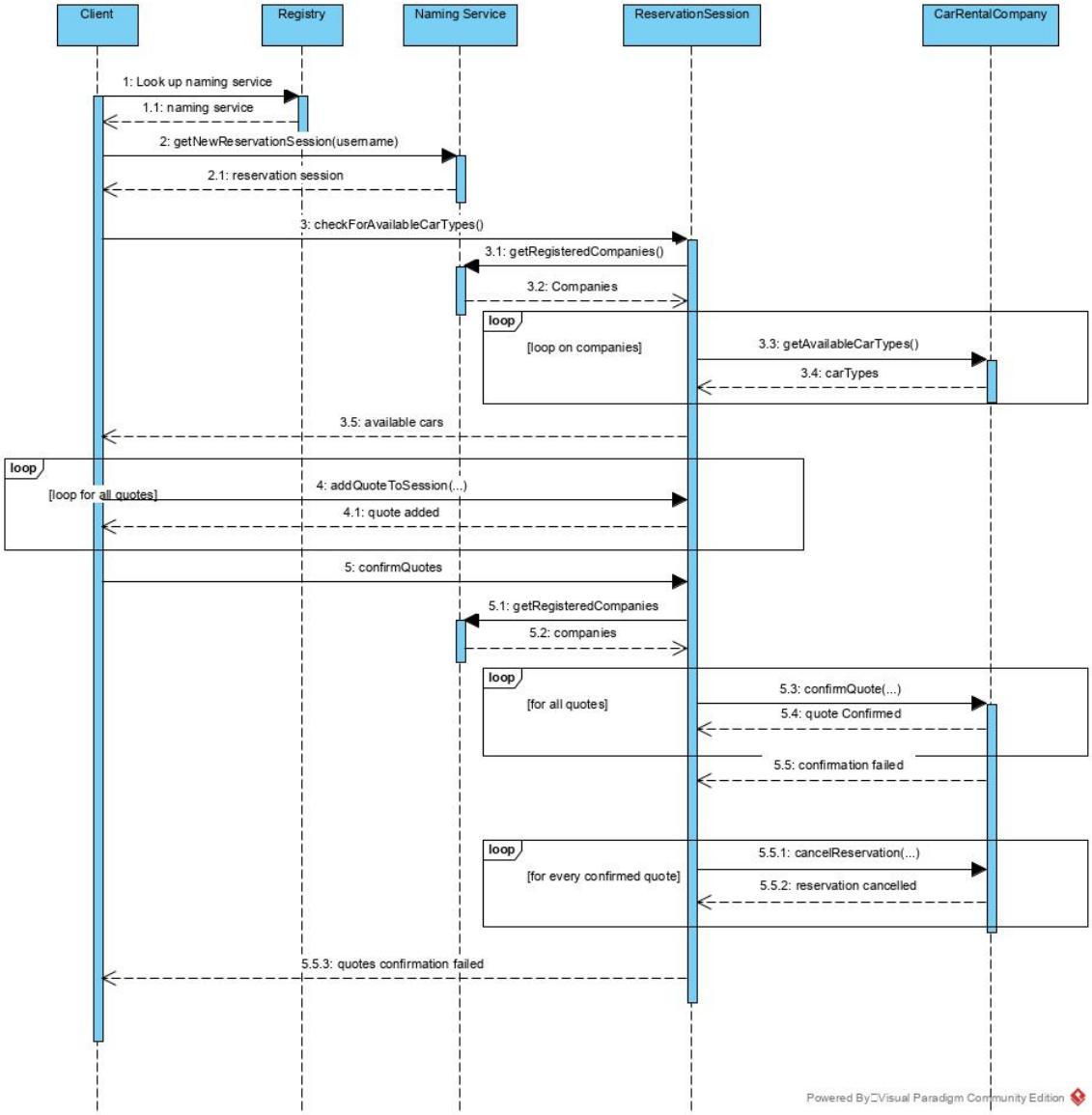


Figure 2: Failing sequence diagram

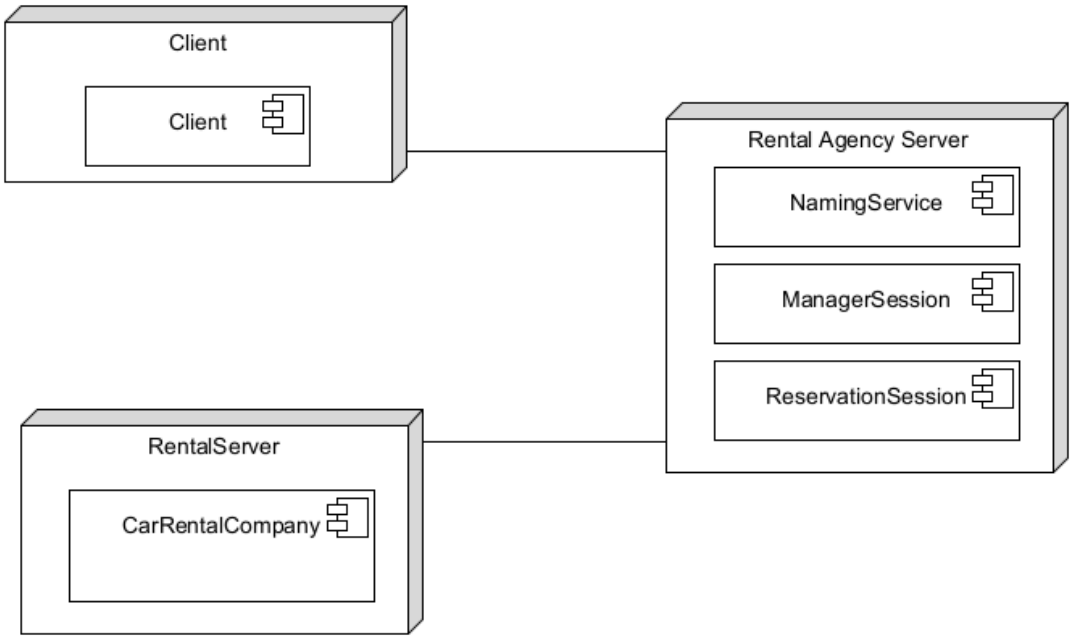


Figure 3: Deployment Diagram