Distributed Systems (H0N08A)

Report: JAVA RMI

Dries Janse *(r0627054)*
Steven Ghekiere *(r0626062)*

October 29, 2019

1. 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.
Hier gaan we een sequence d

2. When do classes need to be serializable? You may illustrate this with an example class.
When a Java class implements the Serializable interface, an instance of this class can be passed as a result or an argument in Java RMI. These instances, just as all primitive types, are copied and passed by value. This means that the receiver creates a copy of the object. On this copied object, methods can be invoked but this will only change the local object. The state of the local object can be different from the state of the original object of the sender.
Classes need to be serializable when the value of the object of such a class is needed. The users of these classes are not allowed to modify the original objects. In our project, we made the following classes explicitly serializable:

- CarType
- ReservationConstraints
- Quote
- Reservation (subclass of Quote)

The following classes, used in our project, are already serializable: String, Date, HashSet, ArrayList and HashMap.
To illustrate this with an example: The client class can request a list of available car types. It requests this list by invoking the getAvailableCarTypes method on the remote object reference of his reservation session. This will return an ArrayList of car types. These types are passed by value, this is done because the client classes are not allowed to change the information of the actual car types.

3. When do classes need to be remotely accessible (Remote)? You may illustrate this with an example class.
Instances of classes which are remotely accessible (implement the java.rmi.Remote interface) are passed by remote object reference. The object which receives this remote object reference can make RMI calls on this remote object. Because no copy is made of the object, the original object of the sender is modified. For all the classes, which have to be remote, have to implement an interface specifying all the methods which can be invoked remotely. This interface directly implements the java.rmi.Remote interface and is extended by the remote class. In our project we created the following remote interfaces:

- INamingService

- ICarRentalCompany
- IManagerSession
- IReservationSession
- IRentalAgency

The INamingService is a remote interface used for registering, unregistering and requesting car rental companies to the naming service. The ICarRentalCompany is a remote interface used for requesting and manipulating car rental company data. It is used by both the reservation and manager session. The IReservationSession is a remote interface used for creating, requesting, confirming quotes and requesting car types. The IManagerSession is a remote interface used for registering/unregistering car rental companies and general information about the number of reservations, best customers and popular car types. The IRentalAgency is a remote interface used for creating and closing sessions.

4. What data has to be transmitted between client and server and back when requesting the number of reservations of a specific renter?
When calling the getNumberOfReservationsByRenter method in the Client class, two parameters are required: the client name of which the client wishes to receive the information and a remote object reference of the IManagerSession. This IManagerSession is stored on the RentalAgency. On this IManagerSession reference the client will (remotely) call the getNumberOfReservations method with the same clientName parameter. This clientName String will be marshalled and send by value to the RentalAgency. In RentalAgency we will loop over each ICarRentalCompany that the NameService has a remote reference of, calling the getReservationsByRenter() method on each ICarRentalCompany reference and add the size of the list to the total. These references of the ICarRentalCompanies are stored beforehand by the same ManagerSession. Finally the the RentalAgency will return the total amount of reservations of the renter back to the Client.

5. 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.

Answer 2

6. How have you implemented the naming service, and what role does the built-in RMI registry play? Why did you take this approach?
Answer 2

7. 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.
Answer 2

8. Why is a Java RMI application not thread-safe by default? How does your application of synchronization achieve thread-safety?
Answer 2

9. How does your solution to concurrency control aspect the scalability of your design? Could synchronization become a bottleneck?
Answer 2