Java RMI

Kenny Op de Beeck, Jonathan Langens

November 2, 2014

1 Design overview

The following is an overview of the core parts and their responsibilities.

1.1 CarRentalCompany

A CarRentalCompany contains a collection of cars. It offers methods to check for available cars in a certain period, and for creating, confirming or cancelling quotes for a certain carType.

1.2 IRentalManager

IRentalManager is an interface for the bridge between user sessions and the CarRentalCompanies. Implementing classes must offer methods for creating new sessions, for (un)registering carRentalCompanies, and for placing or cancelling quotes at one of the registered companies.

1.3 Session

Session is an abstract class representing a user session. It contains a username, a unique ID and a reference to an IRentalManager instance. Subclasses of Session are ReservationSession and ManagerSession. ReservationSession is the session type for a client who can place reservations at the CarRentalCompanies. It keeps a collection of quotes that were created during the session, and offers a method to confirm these quotes. ManagerSession is the session type for a manager, who can register or unregister carRentalCompanies.

2 Remotely accessible and serializable classes

Only one class is remotely accessible: the (I)RentalManager. Clients of any type should look up this RentalManager in the RMI registry and ask for a new Reservation- or ManagerSession. Session is therefore Serializable. other Serializable classes are Reservation, CarType and ReservationConstraints. These are all classes that are needed at the client-side, but might as well be 'just copies' of the instances at the server-side.

3 Different hosts

All objects are located on no more than two different hosts: the client and the server. This makes it easy for the client to communicate with the car rental companies, since they only need to look up one object via the RMI registry. However, this also means that all CarRentalCompy instances are located on the same server. Avoiding this problem could be achieved by registering each carRentalCompany on the RMI registry. The central server, which now contains all companies, would then only serve as a name server, containing the names of the registered companies.

4 Synchronization

It is necessary that, for a specific carRentalCompany, only one quote can be confirmed at a time. If this were not the case, thread-safety would be compromised: two (or more) reservations for the same car could be placed without resulting in an exception. This thread-safety can be achieved by simply adding a 'synchronized' keyword to the confirmQuote method in CarRentalCompany.

Due to the difference in creating and confirming quotes, it is unlikely that this synchronized method will present itself as a bottleneck. Creation of quotes and requesting of statistics are not synchronized processes, meaning they can be performed by multiple threads simultaneously. Invalid quotes could therefore be created by accident, though they would not compromise the thread-safety, since they're not yet confirmed.

In order to ensure a unique ID for every session, it is also necessary to declare the method creating these ID's as synchronized. This is a very small method and is very unlikely to become a bottleneck.