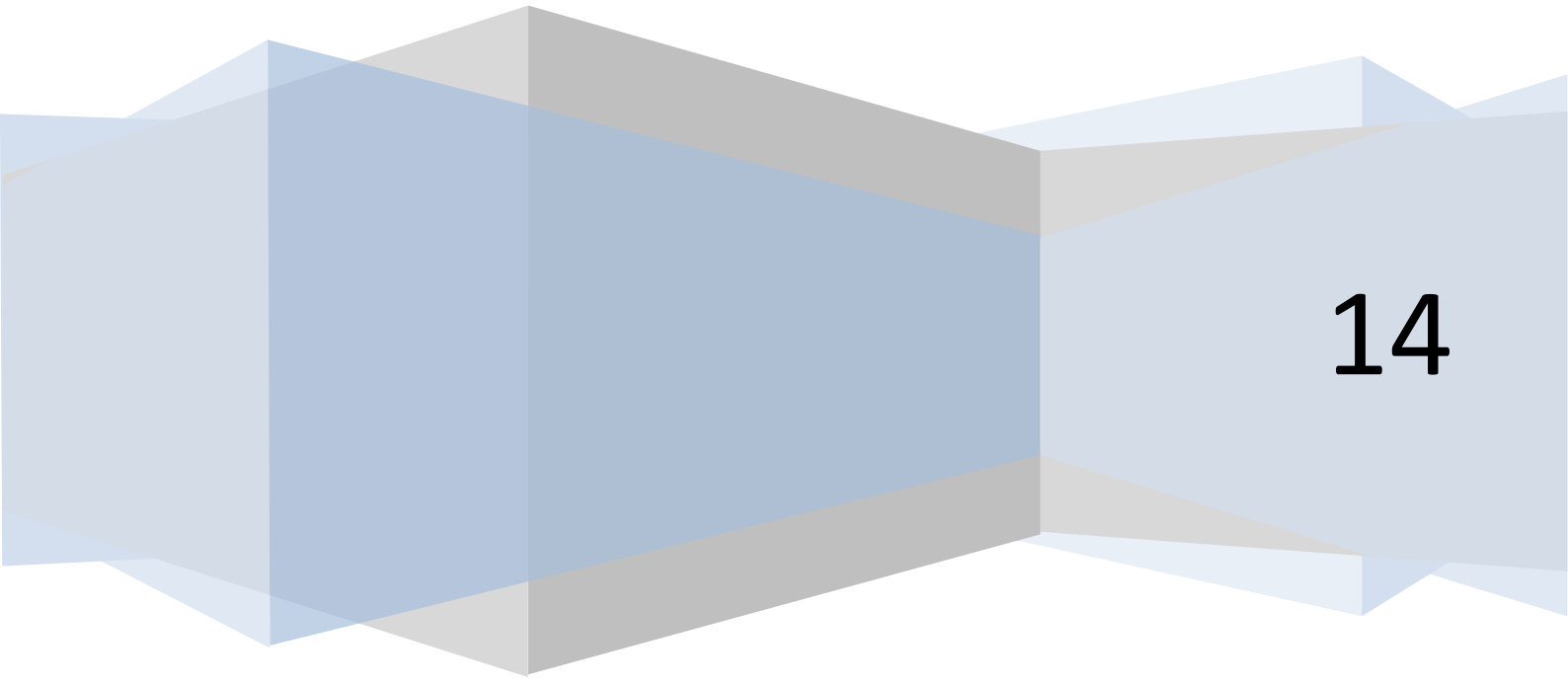
ID 1217 Programmering av parallella system

Banksystem med RMI

[Ange dokumentets underrubrik]

Oscar Nordén



14

Innehåll

Program description

I've made a Bank and ATM System where each ATM is connected to one specific bank. If the user wants to log in to an account that exists on a different bank, the bank itself will communicate with the other bank to log in and make withdrawals or deposits.

The banks run in one process and the ATMs I another, that together with the fact that I'm using RMI to communicate between banks and ATMs makes it possible to have the banks run on one computer and the ATMs on another. The only thing that would have to change is that the ATMs should ask for the IP address for the banks at startup, because in the current implementation they do not do that.

Each bank keeps track of it's accounts, which contains an account number, a pin number and balance, when an ATM that is connected to bank A wants to access an account on bank B, Bank A will "forward" all requests to bank A by calling the same method again on the appropriate bank instance, in this case bank B.

Environment and operation systems

I've written the code in Eclipse on windows 8.1, except for the ATM GUI which I generated with Netbeans GUI builder, and then exported to eclipse. When the GUI was back in eclipse I added the functionality and logics into the GUI methods.

Communication Mechanisms

I use a combination of RMI and reference passing.

All communication between an ATM and it's bank is done with RMI, for example the logIn() or withdraw() method is invoked on an stub of the remote object Bank.

The stub is retrieved from the registry that is created by the bankServer class before it creates and registers the bank instances.

the communication between banks is not done with RMI, but with reference passing. Each bank has a reference to a Vector that contains all the banks, so when bank A wants to log in to an account on bank B, all it has to do is call the logIn method with the appropriate account as parameter.

synchronization

Each RMI call will open in a new thread, so to make sure there are no mishaps all the functions of Account are Synchronized, which means that only one thread can access it at the same time.

Implementations

The project consists of 3 packages, a Client package, a Server package and a Common package,

Client package

The client package contains all the GUI files for the ATMs, and the AtmHandler class, which locates the registry, lookup the banks and then creates the ATMs.

Server package

This package contains the BankImpl class which is the real implementation of the banks. it implements the Bank interface(that is in the common package). Each bank has it's own list of accounts, and that list consists of Account objects.

The Account class is a small class that has 4 properties, an account number, a pin number, a balance, and a boolean inUse, that tells the banks if the account is logged in to allready or not.

The last thing in the server package is the BankServer, which works in similar ways as the AtmHandler. It creates all the banks, and the registry, and then registers the banks in the registry so the clients can get a hold of them.

Common package

The common package contains only the stuff that both clients and server need access to, in this case that is the Bank interface, which the client needs in order to know what methods are available on the banks, and the banks need it to know which methods they have to implement.

it also contains a class called Names, which just lists the names for all banks, so that both the server and the clients know under what name each bank should be registered or lookuped.