AMCM608: Coursework 1

Graham White, after Kohei Honda

October 2013

The following exercises uses the Java code for the bank scenario, which is used in the lectures on Java RMI

What You Have to Do

First, a light warm-up

1. modify the client and server classes in the library so that they will both take command line arguments for the port, and so that the client will, in addition, take an argument for the IP address (or hostname) of the machine that the server is running on
2. Add the treatment of "deposit" and "withdraw" to `Client.java` (you can make user input part more robust if you like)
3. Compile and run `ServerImpl` and your new `Client` on different hosts, and check that everything works
4. Stop your `Client`, but keep `ServerImpl` running. Start the client again, and check that the previous data still remains in the account

Next some exercises on remote references

5. Extend `Server.java` with another method `int getAccountNo()` which returns an account number. Extend `ServerImpl.java` accordingly
6. Construct another remote object, say `accountFactory.java`, which has the following methods
   - `Server newAccount()`, which creates and returns a new `ServerImpl` (its account number will be assigned automatically), as well as storing it in a table (e.g. array)
   - `Server getAccount(int accNo)`, which retrieves a `ServerImpl` with the account number `accNo` from its table
7. Extend your `Client.java` so that it can also create a new account and retrieve an existing account (`deposit` etc. also should specify the account number) by interacting with the `AccountFactory`; once it has retrieved an account, it should go on to interact with it.
8. Finally compile and run your code

Finally we measure the bandwidth and latency of RMI

9. Construct an interface `testRMIserver.java`, toegher with its implementation `testRMIserverImpl.java`, with the following methods

    - `String sendString(int length)`, which returns a string of length `length` to the client
    - `int lengthOfString(String s)`, which returns the length of `s`
    - `char sendChar()`, which sends a character to the client

10. Construct a client `testRMI.java` which tests the bandwidth and latency of RMI, using the server

11. Consider how you can measure the performance of marshalling and unmarshalling. If you have solutions, implement them and summarise your observations based on them.

Deliverables

1. Your code

2. Any test data that you have generated

3. A short but adequate explanation of what you have done, together with any conclusions that you have drawn from your testing.

¡++¿