Universität Zürich
Institut für Informatik

Lecture **Distributed Systems**
Fall 2009

Prof. Abraham Bernstein, Ph.D.

ASSIGNMENT 1 - SOCKETS & RMI (20 POINTS)

Due date: Oct. 6, 2009, 10am (CET)

Rules

- Assumed programming language is *Java* in version 1.6
- Code that is handed in and does not compile will NOT be graded. So please make sure to test your implementation properly.
- Assignments have to be solved individually. It is ok and also desired to discuss problems with peers, whereas copying code is not. As a result, plagiarism will lead to 0 points for the particular assignment for both parties.
- The due date is a hard deadline. Emails that arrive after this deadline will be discarded and therefore the contained solution not graded.

All the above rules are final and no matter for further discussions!

Company Mysteria Solutions consists of ten departments with a central IT administration and therefore a central server. Information about all employees are stored in the central database. If it becomes necessary to retrieve data about an employee, then the server is queried and hands back the answer to the requester.

The data on the server is stored in a text file in the following form:

```
<department number>,<first name>,<last name>,<phone number>,<email address>
<department number>,<first name>,<last name>,<phone number>,<email address>
...
```

Find a sample data file for download under http://www.ifi.uzh.ch/ddis/fileadmin/teaching/Fall09/DistributedSystems/assignments/sampledb_a01.txt.

The following requests are necessary in every-day-life:

- `requestPhoneNumber(firstName, lastName, departmentNumber)`

This means the client send a request to the server with the employee‘s full name and the number of the department he works at. What the client receives back is a phone number, in case the employee exists and actually works in the given department, or an appropriate error message in case he does not.

- `requestEmail(firstName, lastName, departmentNumber)`

This means the client send a request to the server with the employee‘s full name and the number of the department he works at. What the client receives back is an email address, in case the employee exists and actually works in the given department, or an appropriate error message in case he does not.

- `requestEmployees(departmentNumber)`

This means the client send a request to the server with a department number. If the department number exists, the server sends back a list of all names of the department‘s employees, or an appropriate error message otherwise.

Now your tasks:

1) Implement a server/client system using Java Sockets and Threads that solves the above problem. You must not assume that a client necessarily asks for an existing user, so take care of appropriate error handling and feedback. The server system must be able to accept several client connections at a time (therefore the use of threading). For sake of simplicity, you can assume that the combination departmentNumber, firstName, lastName is unique, thus no two persons with the same name work at the same department. How you implement the protocol is up to you, but the system needs to work. (**8 Points**)

2) Implement the very same server/client system using Java RMI. (**6 Points**)

3) Create a short documentation in which you (**6 Points**)

   a) briefly describe your protocol and implementation and the reasons why you implemented it the way you did. In case some major design decisions changed between implementing 1) and 2), explain what changed and why.

   b) describe if you encountered any advantages of one variant over the other (RMI vs. sockets+threads). If you did, explain what these advantages are.

The whole documentation should not be longer than 1-2 pages.

Grading:

Grading will be based on

a) the correctness of your code, i.e. does it solve the given task?

b) readability/structure of your code (including appropriate comments)

c) clarity of your documentation, i.e. does it really describe what you implemented and how well can it be understood by somebody who has not written or read the code.

What to hand in and how:

- Create a zip file named <your_student_id>_<first name>_<last name>_assignment01.zip (e.g. 1234567_John_Doe_assignment01.zip). This zip file should contain two source code folders, one folder for the socket/thread part and one for the RMI part. It should also contain your documentation as *.pdf file.

In case your code requires any special treatment to compile, you have to enclose a README describing the necessary steps.

- Send this zip archive on time via email to Cathrin Weiss (*weiss|at|ifi.uzh.ch*) . The email subject should start with ***[DS 2009]*** . That way you should receive an automated reply indicating that your email was received. If you do not, check again whether your subject really started with [DS 2009], and then contact one of the course assistants.

Helpful reading

Java Sockets:

http://java.sun.com/docs/books/tutorial/networking/sockets/

Java Threads:

http://java.sun.com/docs/books/tutorial/essential/concurrency/

Java RMI:

http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp

http://java.sun.com/docs/books/tutorial/rmi/index.html