Homework 3, RMI And Databases

Network Programming, ID1212

1 Goal

- You can develop a distributed application which uses remote method invocation for inter-process communication.
- You can develop a distributed application which persists data in a database.

2 Grading

The grading is as follows:

0 points Not passed

1 point Passed, but **no points** for higher grades

2 points Passed, and **one point** for higher grades because an accepted solution was submitted before deadline.

3 points Passed, and **two points** for higher grades because an accepted solution including the optional higher grade task was submitted before deadline.

3 Auto-Generated Code and Copying

You must be able to explain and motivate every single part of your code. You are *not* allowed to copy entire files or classes from the example programs on the course web, even if you understand it and/or change it. However, you are allowed to write code which is very similar to the example programs on the course web. You are also allowed to use GUI builders and other tools that generate code.

4 Task, A File Catalog

Develop a client-server distributed application which allows storing, retrieving and removing files to/from a file catalog. A client is controlled by a user and provides a user interface. The server allows clients to perform the following actions:

- A user can register at the catalog, login and logout. To be allowed to upload or download files, a user must be registered and logged in.
- A user specifies username and password when registering at the file catalog server. On registration, the server verifies that the username is unique. If it is not, the user is asked to provide another username.
- When logging in to the server, a user provides username and password. The server verifies the specified username and password.
- A user can upload a local file to the catalog and download files from the catalog to the local file system. A file at the catalog is identified by its name, thus there can not be two files with the same name. A file has the following attributes: name; size; owner, i.e., the user uploading the file; write and read permissions. **It is sufficient to handle file metadata, to also upload/download file content is the optional higher grade task.**
- A file can be accessed by any user registered at the file catalog. The write and read permissions indicate whether the file is read-only for other users than the owner or if it can be modified, i.e. deleted or updated, by any user. Owners have all permissions for their files.
- All users can list all files in the catalog and their attributes.
- A file owner is notified when other users access one of the owner's files. When a file has been read or updated by another user, the server tells the owner who performed the action, and what action was taken. It is sufficient that this works as long as the user remains online as described by the following scenario. A user uploads a file, the file is accessed, the user is notified, the user goes offline, the file is accessed, the user is not notified, the user comes online again, the file is accessed, the user is still not notified.

Requirements on Your Program

All of the following requirements must be met in order for your solution to be accepted.

- Your solution must have an acceptable layered architecture and be well designed. This means it must follow the guidelines of the lecture on architecture, and of the programming examples on the course web. Your are, however, not required to use exactly the same layers as in those examples.
- Client and server communicate only using remote method invocation.
- Only the server is allowed to register itself in an RMI registry. When the server makes a call to a client, it must be via a remote reference passed to the server by the client.

- The server uses a database to keep records on each user (user name and password) and on each file in the catalog (name, size, owner, write/read permissions).
- The clients do not store any data. All data entered by a user must be sent to the server for processing, and all data displayed to a user must be received from the server. Here, *data*, means username, password and file information (name, size, owner and write/read permissions).
- The user interface must be handled entirely by the client. The server is not allowed to send for example strings with messages like "invalid username" or "the file has read permission".
- The user interface must be informative. The current state of the program must be clear to the user, and the user must understand what to do next.

What is NOT Required of Your Program

Below is an explanation of things that do not affect your score.

- Minor misunderstandings of the functionality are allowed, as long as your program does not become notably simpler than a program implementing the intended functionality.
- Your are not required to create a graphical user interface. A command line UI is sufficient.
- You are not required to make the client multithreaded.

5 Optional Higher Grade Task

File upload and download shall include also file content, not just file metadata. Uploading and downloading files shall be done over TCP sockets, not over RMI. You are free to choose whether to use blocking or non-blocking sockets. You are also free to choose how files are stored on both server and client. A suggestion, *not a requirement*, is to store files in a dedicated directory on the server, and to let the user choose where to store downloaded files.