Computer Science Honours 2016

Distributed & Parallel Processing
Practical Three: RMI

A Simple Message Service

Use RMI to implement a simple message service. This should allow messages to be stored for a number of users (identified by a username), and for users to be able to retrieve their messages.

The "server" component of your system will need to store the messages. This can be done very easily using a hash table (see the `Hashtable` class in the `java.util` package) where the keys are the usernames, and the associated values are the messages, stored as lists of strings (the `java.util` package also contains a number of classes that can be used for the lists of messages).

The client side can be written either as a single program containing the logic for both storing and retrieving messages, or as separate programs for the two purposes. The client(s) can also be written either as GUI programs or as command-line applications, as you prefer.

Your starting point should be to define the interface that will specify the methods to be provided by the server for the client(s) to use.

Remember that you will need to run an RMI registry service too. As the registry will need access to the class files for your application, you should run it in the same directory as your application (or else, with appropriate Java `CLASSPATH` settings).

The source files for the Mandelbrot RMI example discussed in lectures are available on RUconnected — you might like to familiarise yourself with the use of RMI by compiling and running this.

To Hand In: You should submit the source files for your interface and the programs (server and client/s).