APIT, Lab Book 3

Dr. Simon Rogers

Jan 2019

Introduction and aims

This lab book covers the material in the concurrency section of the course. As before, do them at your own pace.

Exercise 1: The motivation system

The University feels that its staff are not suitably motivated. So, they've decided to build a computer system that will randomly send motivational quotes to staff members every 5 seconds. Your task is to build a Server and Client in Java to test this out.

Tasks

1. The `QuoteLoader` class and `quotes.txt` (both available from Moodle) will provide you with an `Arraylist` of *amazing* motivational quotes. Download them and motivate yourself.
2. Create classes for a `client` and a `server` that, when the client connects, allows a single (randomly chosen) quote to be sent from the server to the client before both classes finish. This is all possible within the `main` methods of the two classes. If this feels like a big chunk, start with one of the examples from the lectures.
3. Modify your solution to part 2 so that the server sends a new (randomly chosen) quote every 5 seconds.
4. Modify your solution again so that the `Server` can allow connections from multiple clients. Note: you'll need a new `Thread` on the server for each client.
5. After a while, the University realises this system is a bit weird. So, they want to modify such that a quote is only sent when the client sends the message `MOTIVATEME`.
6. Staff would now like to add their own quotes. If they send a message of the form: `ADDQUOTE:a quote` then `a quote` should be added to the global list of amazing quotes.

Exercise 2: Sending and receiving objects

In this exercise you will create a distributed system for finding the maximum value in a large matrix. You did this in the threading labs. The difference now is that instead of the. max for each row being found by a new thread, it will be found by a different client.

I'll leave this one with you to determine the individual tasks. But here are some hints:

1. Note that there are two ways to approach this problem. Either you assume that there will be enough clients to take a row of the matrix each or clients can be given multiple rows. It's up to you. The latter is trickier, but the former will require you to start up lots of clients.
2. An array of Doubles is already seralizable, so can be sent direct to the client via an `ObjectOutputStream` and received via an `ObjectInputStream`

3. Whichever way you choose (see point 1) take some time to think about the communication protocol. E.g. what will send what to whom. A simple protocol would be: client requests a row with some kind of message, Server sends one back (if there is one left to process). Once processed, client sends back a single Double value representing the max.

This is probably the most complex programme you will create on this course, so spend some time designing!