CS313 – Computer Systems and Concurrency. Semester 2, 2019

Write a Java Client-Server application for the distribution of work.

**Part A: Plain old Java objects**

- Write a 'Client' object that creates some 'work'
  - This work should be in the form of a large set/list of data (probably Integers, but preferably generic).
  - You also need to provide at least one Function object (Function is a generic Java interface) that can be mapped onto each element in your data set.
- Write a 'Server' object that can accept the data and function from the client, map the function to each element of the data and return the result to the Client.
  - Use multiple 'Workers' to each handle a portion of the work.
- Your system must be multithreaded and thread-safe.
- Program to Interfaces! (You'll thank me for it eventually).

Things to consider:

- Multiple Clients
- Multiple Servers?
- Definitely multiple Workers
- What type(s) of data does your Server accept?
- Can your Server handle different Functions?

This part of the practical is marked out of 10.

**Part B: Using Sockets**

- Don't destroy your Part A work – make a new copy/branch for Part B.
- Your Server should have a welcome socket listening on your chosen port.
  - As discussed in lecture 6, the server should accept connections and configure a new Socket for each connected client.
  - These connections only handle streams of data. What about Objects?
  - If you are using Workers (you should be), these workers should also use Sockets. In theory, these 'Workers' could be on different hosts.
- Your Client should connect to your Server using a TCP Socket
  - You need to find a way of sending your Object (List<?>) over that connection.
  - How do you send the nice lambda Function you wrote?
    - You can't! What are you going to do about that?*
- Your system must be multithreaded and thread-safe.
  - You still need to ensure the right data goes to the right clients.
  - You still need to ensure the in-order nature of your data.
- Program to Interfaces by default, unless you have a solid reason not to.
  - This isn't specific to CS313.

Things to consider:

- Connection management – be sure to close any resources that are no longer needed.
- Because your client will be unable to send the desired function to the server it is no longer possible to unilaterally dictate what is to be done to the data.
  - The Server (and Workers) must know beforehand what is possible.
  - Perhaps your Server can advertise the functions it can do.
    - Simple, but a bit boring.
  - You could delve into the murky world of Remote Method Invocation (RMI).
    - This is more interesting, from my point of view, but Client, Server and Workers all need access to identical copies of the function.
  - If you use RMI, please split your work over separate projects (preferably different hosts) to show that you are using identical copies of classes, not the actual same classes.

This part of the practical is marked out of 10.

* Actually there is at least one very hacky way to do this but it isn't supported out-of-the-box, and for good reasons. These reasons I don't care about too much for this practical. If you have finished a simpler solution – and you are suitably insane – I'd love to see this happen. No extra marks but lots of Kudos.