COMP30220 Distributed Systems Practical

Lab 1: Simple Sockets

Work individually. Submit your code on csmoodle.ucd.ie.

Initial Steps:

- Download the socket sample code from the Moodle.

- Read through the client and server code and make sure you understand it.
  Try to work out what the code does purely from inspection and the last lecture.

- Run the server code then the client code. For example:
  ```
  java TCPSocketServer 1234
  java TCPSocketClient localhost 1234 "hello world"
  java TCPSocketClient 127.0.0.1 1234 "hello world"
  ```

  Does the code do what you expected? Ask a demonstrator if anything is unclear.

  Note: you can run the example code from a single machine (e. g. from different command shells) or try running the server code on one machine, and the client code on another. In the latter case, make sure both machines are connected to the same access point; otherwise you may have problems connecting to certain port ranges.

Task 1: Echo Server **Grade: D**

Alter the code so that the server sends back to the client "`SERVER ECHO: <message>`". The client should print this response out to the console.

Task 2: Multiplication Service **Grade: C**

Change the code to implement a "multiplication service", i. e. the client will send two numbers and the server will need to respond with the result of multiplying the first number by the second.
Example (testing on the same machine on port 1234):
```
java TCPSocketClient 127.0.0.1 1234 80 3
```
The client should output the following: "`The result of 80*3 is 240`"

Task 3: A Reusable Multiplication Service **Grade: B**

Change the client and the server code to handle 100 random multiplication requests (the client creates two random numbers and sends them to the server, which multiplies them together before sending the result).

Task 4: Efficiency Improvements **Grade: A**
Make the communication more efficient, by:

- creating the client socket only once
- use `DataOutputStream` and `DataInputStream` to transmit binary encodings of the numbers and the result.

+/- grades will be awarded depending on the overall quality (e. g. modular design separating networking code from functional aspects, correct handling of exceptions...) / mistakes impacting the efficiency and the modularity of the design.