Department of Computer Science
COMP212 - 2022 - CA Assignment 2
Distributed Communications
Java Socket Programming and Parallelism

Assessment Information

Assignment Number	2 (of 2)
Weighting	15%
Assignment Circulated	4th March 2022
Deadline	31st March 2022, 17:00 UK Time (UTC)
Submission Mode	Electronic via CANVAS
Learning outcome assessed	(1) An appreciation of the main principles underlying distributed systems: processes, communication, naming, synchronisation, consistency, fault tolerance, and security. (4) A sound knowledge of the criteria and mechanisms whereby traditional and distributed systems can be critically evaluated and analysed to determine the extent to which they meet the criteria defined for their current and future development. (5) An in depth understanding of the appropriate theory, practices, languages and tools that may be deployed for the specification, design, implementation and evaluation of both traditional and Internet related distributed computer systems.
Purpose of assessment	This assignment assesses the understanding of Java Socket Programming and Parallel Processing (via multithreading).
Marking criteria	Marks for each question are indicated under the corresponding question.
Submission necessary in order to satisfy Module requirements?	No
Late Submission Penalty	Standard UoL Policy.

1 Overall marking scheme

The coursework for COMP212 consists of two assignments contributing altogether 30% of the final mark. The contribution of the individual assignments is as follows:

Assignment 1	15%
Assignment 2	15%
TOTAL	30%

2 Objectives

This assignment requires you to write a Java program implementing client-server communication using **Java Sockets**.

> **Note that *no credit will be given for implementing any other form of client-server communication!***

3 Explanation of coursework

Consider the following network application. A server offers a small set of fundamental operations on natural numbers x, such as computing 2^x, x^2, $\sqrt{x}$, and $\log x$. Clients can connect to the server and request processing time. Each time a client is granted access, the client submits to the server a sequence S of natural numbers and a requested operation from those offered by the server and the server does the following. The server is equipped with k parallel threads, k being a pre-determined constant (and cannot ever use more than k threads). It allocates to each of these threads a part (i.e., a subsequence) of S and then all threads in parallel process their part. By "process", we mean applying to each natural number of the subsequence the operation requested by the client.

For example, if the server has access to $k = 3$ threads, the sequence provided by the client is $S = (1, 3, 2, 1, 5, 3, 4)$, the requested operation is 2^x, and the partitioning (arbitrarily here) is $(1, 3)$ allocated to thread 1, $(2, 1, 5, 3)$ to thread 2, and (4) to thread 3, then thread 1 will return $(2, 8)$, thread 2 will return $(4, 2, 32, 8)$, thread 3 will return (16), and after rejoining these to form again a single sequence, the server will return $(2, 8, 4, 2, 32, 8, 16)$ to the client.

3.1 Basic Java Socket Communication—30% of the assignment mark

As a first step, you are required to implement client-server communication only. At this stage you can ignore the program logic, but you should show some evidence that your program works and that communication between the client and server actually takes place. It is not a requirement that many clients can connect at the same time, but if you wish you could try to also achieve this functionality for a few concurrent clients. For this assignment it is sufficient that at any given time a single client establishes a "connection" to the server and executes its experiments. Moreover, you are allowed to have both the client and the server run on the same machine (connection to "localhost"), so that you don't have to use two distinct machines communicating over the network in order to test your software. On the CANVAS course of the module, alongside the assignment description, you can find a very basic example program discussed in the lectures that you could use as a template to get you started. Moreover, you are strongly encouraged to consult Oracle's tutorials on Java sockets and especially their "getting started" examples (Sockets Knock Knock example).

3.2 Processing—30% of the assignment mark

The next objective is to allow a client request processing from the server on a specific input sequence communicated over the network and to have the server correctly run the parallel computation on that sequence. In particular, the following functionality should be achieved:

- The server implements a small set of operations on natural numbers (like the example operations given above) and has at most k parallel threads ready for processing. This represents the parallelism available to the server. That is, the server cannot use more parallelism than k, which is expected to be a pre-defined constant number, e.g., 50 threads. **(10% of the assignment mark)**

- The client tries to connect to the server until a pre-determined timeout T. If it does not succeed within the time-interval, because the server is down or is serving other client(s), it prints a timeout message to the prompt and terminates. If it succeeds, it downloads from the server the available operations and then submits to the server a sequence S of natural numbers alongside its choice of an operation (from those offered) to be applied on the sequence. **(10% of the assignment mark)**

- The server receives the sequence and the client's choice and uses its k threads to apply the requested operation on all numbers of the sequence, thus, computing a new sequence S'. Once this is done, the server returns to the client the new sequence S' together with the measured server execution time from the point the server received the request up to the point it transmitted the outcome back to the client. **(10% of the assignment mark)**

3.3 Data Gathering—20% of the assignment mark

Make two versions of your server by adjusting the way the server utilizes its k threads. One version will be uniformly dividing the sequence S across the threads, while the other will be using $k-1$ threads for the first $k-1$ numbers of the sequence (one thread for each one of these numbers) and 1 thread for all the remaining numbers of S (if $|S| \leq k$, then the second version will use exactly one thread per number and $k-|S|$ threads will not be used at all). Focus on a single client and update it so that it requests processing on sequences of increasing size, for the purpose of data gathering. The numbers in those sequences could be randomly drawn numbers from a small subset of natural numbers. The client should collect processing time data (via the measured time returned by the server for each execution) for both of the aforementioned versions of the server. You (or the client automatically) should then plot the data against each other in a way that allows you to draw meaningful conclusions on the comparative behaviour of the two multi-threading methods offered by the server.

3.4 Report—20% of the assignment mark

It is a requirement that you also submit a concise report describing your design, including some testing of your software, the generated plots, and your drawn conclusions about the experimental evaluation performed.

4 Deadline and Submission Instructions

- The deadline for submitting this assignment is **Thursday, 31st March 2022, 17:00 UK time (UTC)**.
- Submit
  - (a) The Java source code for all your programs,
  - (b) A README file (plain text) describing how to compile/run your code to reproduce the various results required by the assignment, and
  - (c) A concise self-contained report (at most 5 pages including everything) on your design, testing, plots, and conclusions in PDF format.

  Compress all of the above files into **a single ZIP** file (the electronic submission system won't accept any other file formats) and **specify the filename** as *Surname-Name-ID*.zip. It is extremely important that you include in the archive all the files described above and not just the source code!
- Submission is via the "Assignments" tab of COMP212-202122 course on CANVAS.

Good luck to all!