ESIR SPP – TP Introduction (non noté)

Exercise 1:

In Java write a multithreaded program that manipulates one shared long variable so that:

- 5 threads increment this variable 100,000 times;
- 15 other threads read this variable 100,000 times, and write out what they have read onto the console every 20,000 iterations (with their ID as prefix).

1 – Write a first version of your program without any synchronisation. What do you observe? Why do you think this is the case?

2 – Make your program thread-safe with a normal re-entrant lock. Measure the execution time taken by your program.

3 – Write a second version of your program, in which you replace the normal lock by a read/write lock. Measure the execution time of your program. What do you observe? Why do you think this is the case?

4 – We will now artificially increase the cost of the integer operations performed on the shared variable (both increment and read), by using `Thread.sleep(..)`, and reassess the effect of a read/write lock over a re-entrant lock. To this aim, write two new versions of your program which:

- only use 1000 iterations of each loop. Set the console printouts to occur every 200 iterations.
- artificially delay each operation on the shared variable by 1ms using `sleep(..)`.

One version should use standard re-entrant locks, the other read-write locks. Measure and compare the times obtain. How do you interpret them?