IN4330 Oblig 5 report

Intro

For this assignment we were tasked with making a wait and swap program, by synchronizing threads by only using semaphores. The idea is that the first thread waits, then the second passes through, and then the third one waits. This pattern then repeats itself.

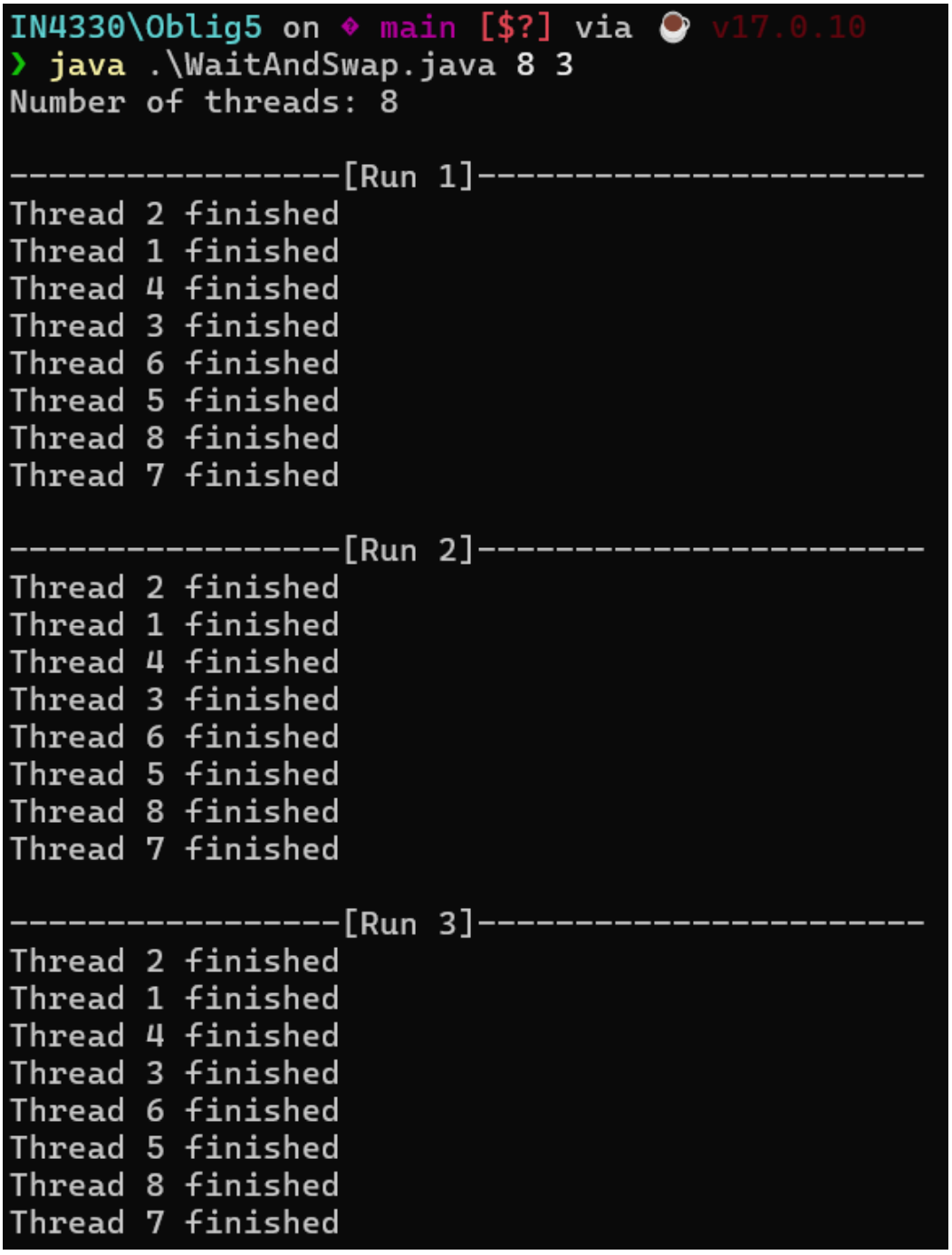
Solution

I took many approaches to this problem but I ended up with a solution that is fairly simple. It simple checklist if the id number of the thread is odd and then has that one acquire a permit causing the thread 1 to wait. Then thread 2 passes through and releases the semaphore allowing thread 1 to pass through. Lastly thread 3 tries then to acquire permit and waits, then the cycle repeats.

The second solution i didn't end up finishing does what is the same principle but using 2 semaphores. It starts with the semaphore 1 having a starting permit of 1 and the semaphore having a starting permit of 0. It checks if the thread number is even, thread 1 isn't even and attempts wait at semaphore 2 after attempting to acquire. Then for thread 2 semaphore 1 acquires permit, finishes, and releases semaphore 2, causing thread 1 to finish. This pattern is then repeated giving the our desired wait and swap output.

Results

The output for multiple runs of both solutions gives the desired output the oblig described. The program is set to run WaitAndSwap a specified amount of times, to ensure the results are consistent.The current default is set two 3 but is changeable through arguments as seen in the run.



Run WaitAndSwap with 8 threads, and 3 runs

Conclusion

Through use of semaphores we can synchronize running threads in various ways to control the running of a program that solves a problem in parallel. That being said using them to solve this challenge presented a couple of problems.I found the details of the problem i bit difficult to follow, but i ended up building what i believe is the correct solution given the desired output described in the Oblig text.