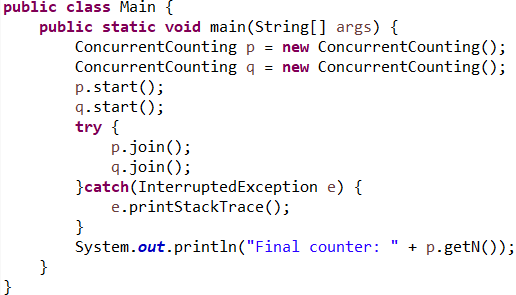
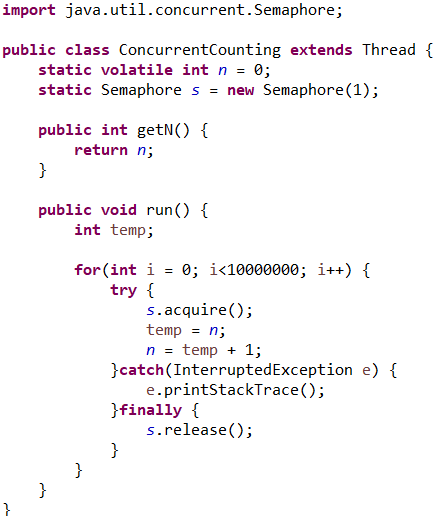
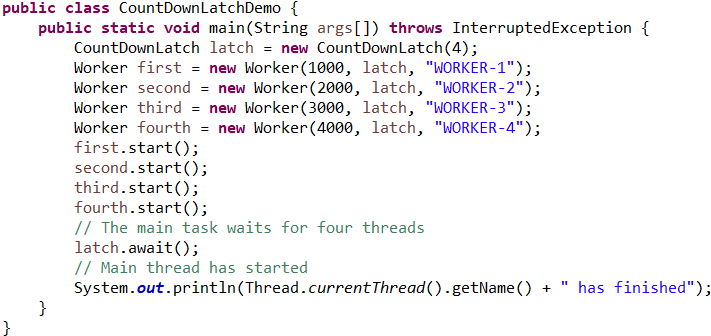
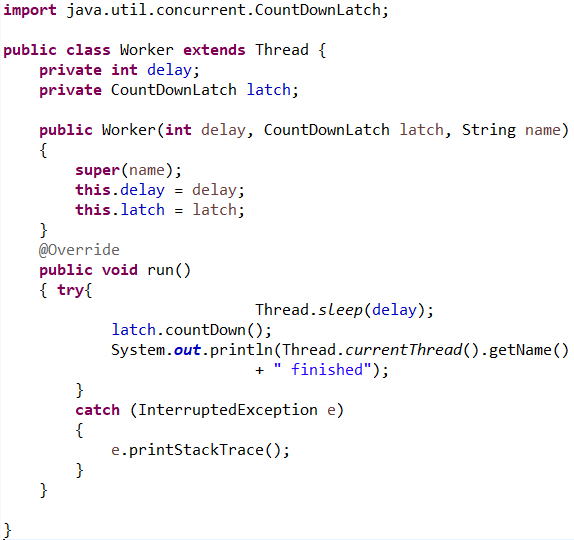
**Laboratory 6**

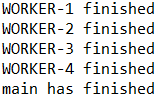
1. I implemented the ConcurrentCountering example as follows:

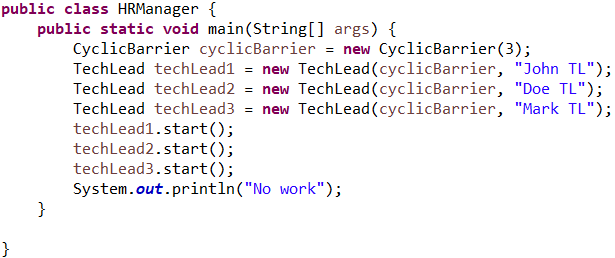
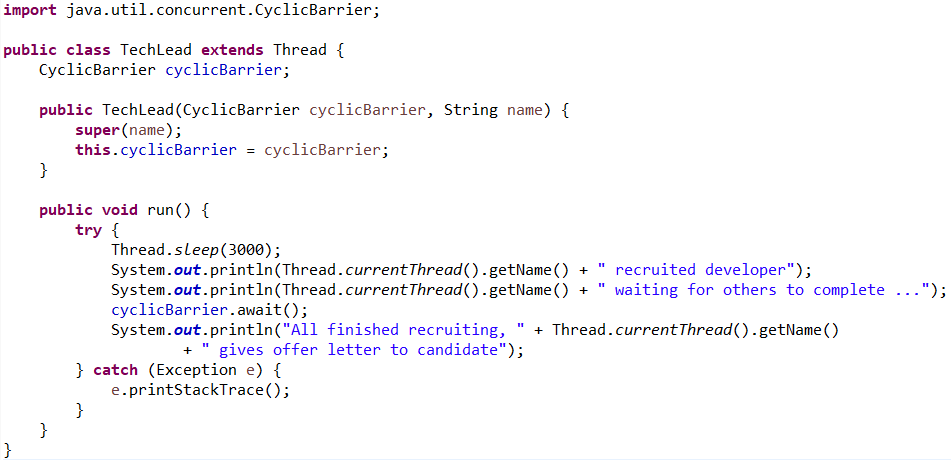


The result of running this code is: 

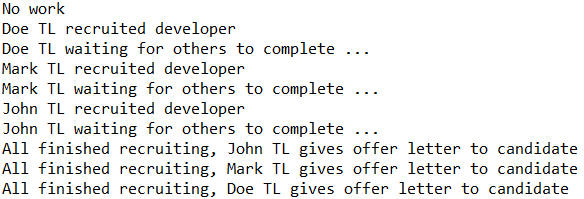
1. I implemented the CountDownLatch example as follows:



The result of running this code is: 

1. I implemented CyclicBarrier as follows: 

The result of running this code is:



1. a. If we set Semaphore “s” as non-static, then 2 instances of semaphore will be created (one for each thread), so the acquire() method will be run on different semaphores, and in result both threads will have access simultaneously to the variable “n”, this creating an undetermined output. 

4. b. If we set the “n” variable as non-static, then each thread will have his own variable “n” which will be incremented. In this case, at the end of execution, each thread will have the counter equal to 10000000. 

4. c. If we acquire before the for loop, will have a different result depending where we release: in the for loop of after it.

If we release the for loop in the for loop will result in having accessed the variable n simultaneously and creating an undetermined result. 

If we release after the for loop, will result in first thread executing integrally, and after and only after releasing the semaphore in order to be acquired by the second thread to start its execution.



4. d. If we release either before, or not at all, the result will be the same, the program will be blocked, because at each step in the for loop, it will try to acquire again, but there will be no permission till after the for loop finish execution. So the thread will be blocked in the for loop at second step indefinetely.