John Kucera

Prof. Suzanna Schmeelk

CMSC 412 Operating Systems

29 January 2020

Week 3: Homework 3 with Test Run

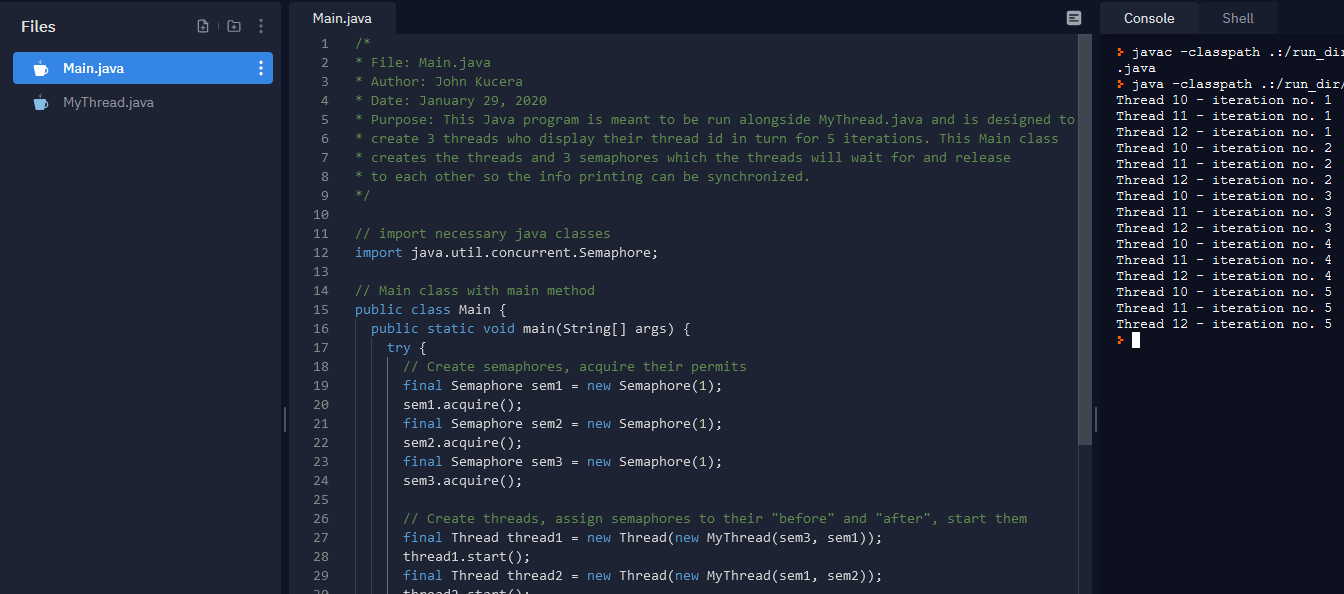
**Creating Threads and printing Thread IDs (Java)**

**Solution Description**: In my Java program, I used semaphores to coordinate the 3 threads and get them to print information in proper order and during each iteration. I initially considered using plain locks, but I found that more code and complexity was required to properly synchronize threads so they would acquire the exact lock. The core methods of semaphores, acquire() and release(), are (in my opinion) simpler than using wait/notify in synchronize blocks for locks in this case. Semaphores are more easily “assigned” to threads in a custom constructor.

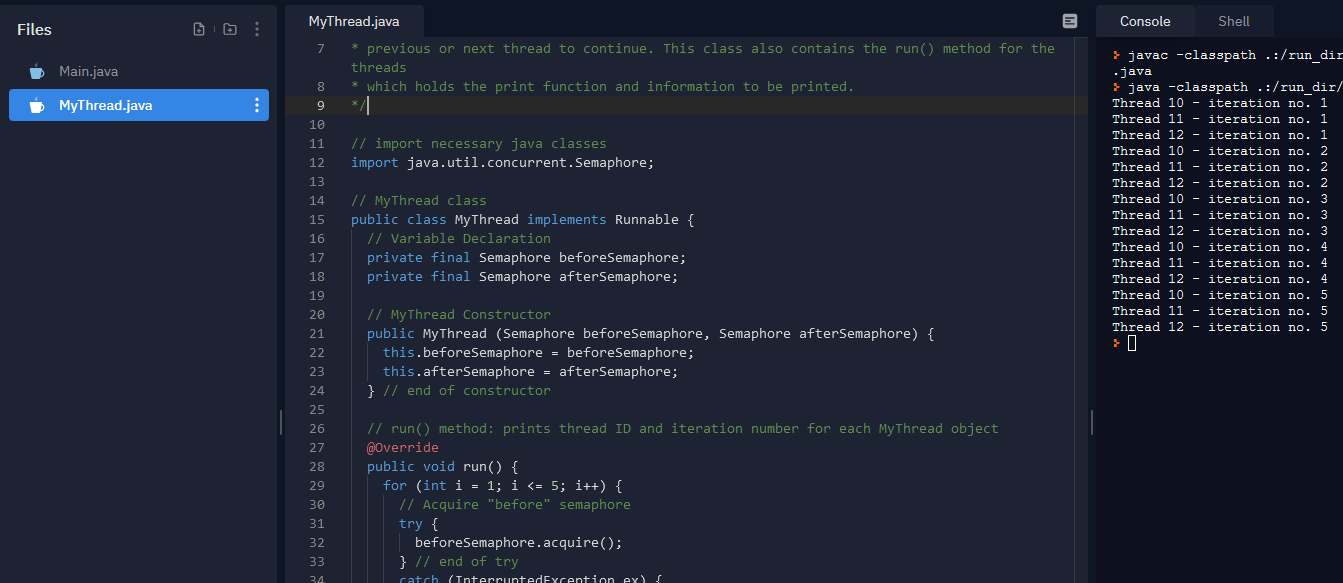
Main.java contains the code for creating the 3 threads and “assigning” semaphores to them so they acquire and release them in the correct order. For example, when Thread 1 has finished a print iteration, it releases semaphore 1, and Thread 2 is assigned to acquire that semaphore 1 immediately after. Then, Thread 2 prints its iteration information, and releases semaphore 2. Semaphore 2 gets acquired by Thread 3, and so on until this has happened for 5 iterations. MyThread.java contains the run() method and iteration information to be printed.

On the next pages are screencaps of the two java files running in repl.it and a test run.

**Main.java running in repl.it:**



**MyThread.java running in repl.it:**



**Test Run**

|  |  |  |
| --- | --- | --- |
| **Expected Output** | **Actual Output** | **Pass?** |
| (Applies to all test runs)  When “Run” is clicked, 3 threads are created and already have thread IDs. The following output will be printed in 5 iterations:  “Thread [id1] - iteration no. # Thread [id2] - iteration no. # Thread [id3] - iteration no. #”  It will show their thread ID and the number iteration it is on. |  | **YES** |