CS117 (Fall 2015) Programming Assignment 7 25 pts. Due: Nov. 24 (T)

Practice for Concurrent programming with Java Threads

Task: Implement the monitor for managing serially reusable resources for concurrent processes.

Assume that there exist two serially reusable resources (SR1 and SR2) with 3 units and 2 units, respectively, and 6 concurrent processes in the system.
Each process acquires both SR1 and SR2 (one unit for each), does service, and releases SR1 and SR2 units after the service.

For the implementation, we need the following 4 classes:

Resource: The main class; responsible for creating multiple (6) process threads and making them run concurrently.
Note that the shared objects (serially reusable resources) should be passed as parameters to the constructor of each process thread.

SR1: The monitor that encapsulates data (avail_1) and operations on that, i.e., acquire/ release, which are implemented as synchronized methods for mutual exclusion.
For the synchronization, use wait() and notify() appropriately in those methods.

SR2: The monitor that encapsulates data (avail_2) and operations on that, i.e., acquire/ release, which are implemented as synchronized methods for mutual exclusion.
For the synchronization, use wait() and notify() appropriately in those methods.

Process: Class for the process, which acquires SR1 and SR2 (1 unit each), works (simply displays a message), and releases them.
For the purpose of tracing the behavior of the system, please use sleep( ) before each call for the synchorized methods, acquire and release.

For tracing the behavior of the system, the following messages should be generated at run time:
Thread for process-# created,
Process-# acquires SR#,
Process-# is waiting for SR#,
Process-# is working,
Process-# releases SR#,
(See attached sample testing output.)

Submit the hardcopy of your code and output – please do not edit your code/output.
Please include documentations in your code, i.e., global, class head, function head documentations.

Sample output (one possible output)

```
===== Thread for process-1 created
===== Thread for process-2 created
===== Thread for process-3 created
===== Thread for process-4 created
===== Thread for process-5 created
===== Thread for process-6 created
process-2 acquires sr1
process-4 acquires sr1
process-1 acquires sr1
process-1 acquires sr2
---- process-1 is working
process-2 acquires sr2
---- process-2 is working
+process-6 is waiting for sr1
process-1 releases sr1
process-6 acquires sr1
+process-5 is waiting for sr1
+process-3 is waiting for sr1
+process-4 is waiting for sr2
+process-6 is waiting for sr2
process-1 releases sr2
process-6 acquires sr2
---- process-6 is working
process-2 releases sr1
process-3 acquires sr1
+process-3 is waiting for sr2
process-6 releases sr1
process-5 acquires sr1
process-2 releases sr2
process-3 acquires sr2
---- process-3 is working
+process-5 is waiting for sr2
process-3 releases sr1
process-3 releases sr2
process-5 acquires sr2
---- process-5 is working
process-6 releases sr2
process-4 acquires sr2
---- process-4 is working
process-4 releases sr1
process-5 releases sr1
process-5 releases sr2
process-4 releases sr2
```