**CSE 505 Spring 2017**

**Assignment # 3 – Concurrent Programming**
*(may be done by a team of two students)*

Assigned: March 28
revised March 30 (see prose with blue highlight)

Due: April 5 (11:59 pm)

The Readers-Writers problem is a classic example in the study of concurrency control. There are two types of concurrent threads, reader threads (readers) and writer threads (writers), and they concurrently access a database. Readers execute the database 'read' operation while writers execute the database 'write' operation. The basic requirement of concurrency control is that:

(i) a 'read' operation may be executed concurrently by two or more readers, but
(ii) a 'write' operation should not be executed concurrently with any 'read' or another 'write' operation;

There are different ways to program a solution to the Readers-Writers problem; we will explore one using Java threads and wait-notify constructs in this assignment. Finally, we will also consider a solution with Java semaphores.

**Part 1:** Refer to the program `Piazza:Resources→Homeworks→ReadWrite1.java`. Explain briefly with reference to JIVE's sequence and state diagrams why this program does not fully meet the basic requirement of concurrency control stated above.

Prepare a file, `Part1.pdf` in which you include your explanation as well as the JIVE state diagram with respect to the variables `Database:1->r` and `Database:1->w`, which stand for the number of active readers and writers respectively.

**Part 2:** Remedy the deficiency in ReadWrite1.java by writing a program ReadWrite2.java that meets both requirements of concurrency control. The outline of ReadWrite2.java has been posted at `Resources→Homeworks`. Complete this outline and run the program under JIVE, and save in file Part2.png the state diagram for the variables `Database:1->r` and `Database:1->w.`

**Part 3:** Enhance your solution in Part 2 by writing a program ReadWrite3.java that also incorporates a 'writers priority' requirement: when a writer tries to access the database and is made to wait because of one or more active readers, all subsequent read requests are delayed until this writer gets to access the database; however, active readers are not pre-empted but are allowed to complete their read operations before the writer begins its operation.

Run the program under JIVE and save in file Part3.png the state diagram for the variables `Database:1->r` and `Database:1->w.`

**Part 4 (added on March 28):** Systematically translate all synchronized methods and wait-notify constructs in your solution to Part 3 in terms Java Semaphores, using the methodology outlined in Lecture 13. Name the resulting program as ReadWrite4.java.

Run the program under JIVE and save in file Part4.png the state diagram for the variables `Database:1->r` and `Database:1->w`.

**ON-LINE SUBMISSION.**

Prepare a top-level directory named *A3_UBITId1_UBITId2* if the assignment is done by two students; otherwise, name it as *A3_UBITId* if the assignment is done solo. (Order the *UBITId*'s in alphabetic order, in the former case.) In this directory, place the files

`Part1.pdf, ReadWrite2.java, Part2.png, ReadWrite3.java, Part3.png, ReadWrite4.java, Part4.png`

Compress the directory and submit the resulting compressed file using the `submit_cse505` command. For more details regarding online submission, see

`Resources → Homeworks → Online_Submission_2017.pdf`.

**End of Assignment 3**