CSE505 – Spring 2019

Assignment 3

Due Date: **April 15 (11:59 pm)**

You may work in pairs for this assignment.

There are two problems in this assignment. Problem 1 was assigned on April 2.

Problem 2: Lecture 16 presents the Readers-Writers problem with Writers priority using synchronized methods and the wait-notify constructs of Java, and Lecture 17 presents a *message-passing* approach to the Readers-Writers problem *without* priority. In this assignment, you are to develop a *message-passing* approach to the Readers-Writers problem *with* Priority.

Posted on Piazza is a file `RWP_MessagePassing.java` giving the outline of your solution. Apart from the `main` method and the process definitions for the `Database`, `Reader`, and `Writer` respectively, the focus of this assignment is on the definition of the `RWController` process.

In the *message passing* paradigm, processes communicate and exchange data with other processes using *send* and *receive* operations on *channels*. A library, `MessagePassing.jar`, is posted on Piazza under `Resources→Software and Programs`. It should be added to the 'Build Path' of your Eclipse project as an 'External Archive'.

In order to implement `Writers` priority, the `RWController` would need to know the number of waiting writers and waiting readers, respectively. This information is provided by two functions, `ww()` and `wr()`, and is given as part of the starter code for class `RWController`. They can be used in tests such as:

`if (ww() > 0) {` … *there are one or more waiting writers …* `}`
`if (wr() > 0) {` … *there are one ore more waiting readers …* `}`,
etc.

Note: The functions `ww()` and `wr()` also set the fields `ww` and `wr` in `RWController`. These fields are useful for drawing the second state diagram specified below.

Complete the definition of `RWP_MessagePassing.java`, run the program under JIVE, and save the execution trace in a file called `RWP.csv`. Note: Before running your program, add the entry `Scheduler.*` to the Exclusion Filter which can be found under the JIVE tab in the Debugging Configurations.

Load `RWP.csv` into the FSM plugin and obtain two state diagrams:

(i) For the first diagram, use `RWController:1->r` and `RWController:1->w`. Save the diagram in `safety.svg`.

The diagram should confirm the *safety* property, namely, that when `r > 0` for some state then `w == 0` in that state; and when `w == 1` in some state then `r == 0` in that state; and for every state `w == 0 || w == 1`.

(ii) For the second diagram, use `RWController:1->r`, `RWController:1->w` and `RWController:1->ww`. Save the diagram in `priority.svg`.

The diagram should confirm the *writers priority* property, namely, that when `ww > 0` for some state, either `r == 0` in the same state or else `r` should monotonically decrease in all paths emanating from this state until `r == 0`.

WHAT TO SUBMIT:

Make a directory called `A3_Problem2_UBITId` if working solo, or make a directory called `A3_Problem2_UBITId1_UBITId2` if working as a pair (give UBITId's in alphabetic order).

Put in this directory the files `RWP_MessagePassing.java, RWP.csv, safety.svg`, and `priority.svg`.

Compress the directory, and submit it using the `submit_cse505` command.

End of Assignment 3 Problem 2