CS 4345 (Operating Systems)

Project 1 [Spring 2023]

This is a pair-programming exercise where the students can work in pairs and have one submission. So, no individual submission is expected from a pair. However, if any of you want to work alone, you can do so. That is why the assignment submission box is created as an 'individual' type. Note, not more than two students are allowed to work together. It is mandatory to include name(s) of student(s) working in this assignment as authors in the comment lines at the top of the code. Without that the submission will not be graded and a temporary score (1 point) will be assigned until the authorship is established. Also, in the submission box, after uploading your file(s), write the student(s) name(s) in the comment box.

Purpose: Write a Java program to solve the following synchronization & deadlock problem.

Problem: Suppose a two-way (east-west), two-lane road contains a long one-lane bridge. A westbound (or eastbound) car can only use the bridge if there are no oncoming cars on the bridge. Because of accidents, a signaling system has been installed at each end of the bridge. When a car approaches the bridge, a sensor notifies the controller computer by calling a function **`arrive`** with the car's travel direction (east or west). When a car leaves the bridge, the sensor notifies the controller computer by calling **`passed`** with the car's travel direction. The traffic controller sets the signal lights to allow or prevent cars wanting to pass the bridge. Construct and implement an algorithm for controlling the lights such that they synchronize the traffic to pass the bridge correctly.

Points to consider:

a) You can assume that there will be steady stream of cars from each side. Hence, for easy referencing, you may assign odd numbers (1, 3, 5, 7, …) for eastbound cars and even numbers (2, 4, 8, 6…) for westbound cars.

b) The controller may allow more than one car passing through the bridge, but only in one direction.

c) Once a particular car arrives and passes the bridge, it does not come back again to cross the bridge. That is, same car number does not appear twice.

d) The synchronization should avoid deadlock. That is, the flow should not be such that cars from only one direction is keep crossing the bridge. If that happens due to JVM's preference of one particular thread, it is okay; but the flow should not be controlled that way through your code. That is, your code should not force all cars from one direction first, followed by all cars from other direction. In fact, the user can run the program multiple times, and this unintended serialization cannot happen all the time! If that happens in every run, then that indicates the code must have forced it.

e) Cars do not need to have same speed (the time they spend to pass the bridge).

f) **User is not expected to supply any input. The user will simply compile and run your program.**

g) Program should continue until the user presses CTRL-C in the shell to stop the program.

[***Hint: You may use any synchronization technique. For example, semaphores and mutex could be a good choice. You may create two threads for controlling eastbound and westbound cars. You may also use Thread.sleep() with different sleep times for eastbound and westbound cars. This would allow simulating different time taken to pass the bridge. You may also use synchronized controller method and use wait() and notify()/notifyAll() from Java concurrency package appropriately.***]