CS 342: Software Design

Java Project #5

Due time: 5/2/2015 at 7:00 pm
Instructor: Ugo Buy
TAs: Chihua Ma and Weixiang Shao

For this project you must code a concurrent Traffic Light System (TLS) example in Java. The TLS consists of an intersection between two streets traveled by cars. The TLS must manage access to the intersection by cars travelling in perpendicular directions. Your job is to program and simulate the TLS using Java multi-threading code. The TLS manages the lights at an intersection between a North-South (NS) street and an East-West (EW) street. Of course, when the light is green in one direction, it must be red in the opposite direction and vice versa. The TLS would normally keep the light green in the NS direction, except when cars approach the intersection from the EW direction. When this happens, a sensor informs the TLS of the presence of a car in the EW direction. If the light has been green in the NS direction for at least 30 seconds, the light will turn to the amber color for 3 seconds, and then red in the NS direction and green in the EW direction. If, however, the light was green in the NS direction for less than 30 seconds, the TLS waits until the green cycle completes 30 seconds and then turns the light to amber and red. Finally, if the light was not green in the NS direction, input from the sensor in the EW direction is ignored. Once the light turns green in the EW direction, it stays that way for 20 seconds, then turns amber (for 3 seconds) and red in the EW direction (now green in the NS direction)

Your code must contain at least the following two threads, a *TLS Controller* that controls the lights, and an *EW Sensor* that randomly signals the presence of cars in the EW direction. Your *TLS Controller* thread must print appropriate statements on the console as each light changes color, or if a car is sensed in the EW direction.

General Implementation notes. Feel free to use synchronized methods to let the threads communicate with each other. However, make sure that the execution of a thread is quick; do not force a caller to wait until a synchronized method completes a sleep statement. You may use additional threads if necessary to execute the sleep statements needed to implement the delays in the above specification.

You must work alone on this project. Submit this project using the link in the Blackboard web page. Pack your Eclipse source code in .zip archive that includes also a README file with instructions on how to run your code. No late submissions will be accepted.