CSCI 6401 Programming Project #1 Dr. Tu
Threads and Synchronization by Monitor

Due Date Tuesday, October 21, 2014

Problem Description

The description of the classic single-elevator problem can be found in Section 2.2.5 of Donald Knuth's "*The Art of Computer Programming, Vol. I, Fundamental Algorithms*". On one hand, we are to ignore the details concerning the physical properties of the elevator such as velocity, acceleration of the elevator, and the time delays for doors to open and close. On the other hand, we are to consider the multiple-elevator scheduling problem. Your task is to write a simulator that enforce the chosen scheduling algorithm, can display/visualize the operations of the *m* elevators, and to produce statistics of the simulation results. The configuration of the system and the services of the elevators are specified as the following.

- Each elevator has *n* ($n > 5$) buttons, one for each floor. [*Fancy features* for extra points: These buttons illuminate when pressed and cause the elevator to stop at the corresponding floor. The illumination is cancelled when the corresponding floor is visited by the elevator.]

- At each floor, except the ground and the top, there two buttons -- one for up and the other for down. [*Fancy features* for extra points: These buttons illuminate when pressed. The illumination is cancelled when an elevator visits the floor.]

- When an elevator has no pending requests, it travels to a specified "parking zone" and remains there with its door closed. The parking zone is set to be at Floor 1 during night and the morning rush hours (6 to 9:30 am), Floors *n*-3 to *n* during the leaving rush hours (4:30 to 7 pm). No parking zone is set during other time; so the elevators stay at the last serving stop when they are idle. A wall clock time is adjustable in the simulator's interface.

- Each elevator has a shield panel which requires a password to open. It contains two buttons: *go-alone* and *cancel*. The *go-alone* button will separate this elevator from the multiple-elevator system and serve the requests made by the buttons in this elevator only. The *cancel* button will cancel all the requests made by the buttons in this elevator. In an urgent situation, an emergency responder may press *go-alone*, then *cancel*, then the destination floor number. This sequence is offered as an option.

The goal of the elevator scheduling is to minimize the distance that all the elevators travel, and guarantee the customer's waiting time not to exceed the maximum round trip time of an elevator in this building.

The elevator algorithm (SCAN) has been adopted to the computer disc scheduling problem. More references can be found in the Wiki page http://en.wikipedia.org/wiki/Elevator_algorithm. Using the acronyms in that page, the algorithm given in our text book is actually the C-LOOK version[1]. While the real-world elevator scheduling could be more complex[2], your simulator is to enforce a variation of the LOOK algorithm, N-Step-SCAN[3] in order to guarantee the limit to waiting time.

Monitor Design

The core of your project is to design a monitors that enforce the C-LOOK scheduling. Each elevator should be simulated by a thread which synchronizes its movement through the monitor. Our focus will be on the synchronization structures and mechanisms of a multithread implementation.

---

[1] http://www.cs.iit.edu/~cs561/cs450/disksched/disksched.html
[2] http://www.columbia.edu/~cs2035/courses/ieor4405.S13/p14.pdf
[3] http://en.wikipedia.org/wiki/N-Step-SCAN

In addition, you are required to develop user interfaces for initializing the simulator, that is to take the user's configuration data such as the number of floors, the number of elevators, the parking zone floors and time frames, as well as the starting wall-clock time. Another interfaces for collecting statistics should also be made, in which the configure data include the simulation time period and request arrival rate at floors, the choices of the statistical items include the average waiting time and average service time, as well as operation log.

Technical Design and Implementation

Java threads have equipped the programmers with many handy frameworks to implement monitors. In addition to the basic thread features, you are strongly encouraged to use Java's built-in classes in packages java.util.concurrent, java.util.concurrent.atomic, and java.util.concurrent.locks. Before you write your Java code, take time to design your monitor in the pseudo-code used in our textbook. Then you should design the structure of threads of your system, indicating what are represented by active threads and their synchronization mechanism (who wait who for what condition) in text and diagrams. Your third step is to choose the proper classes provided in the Java API. If you choose to use some (you definitely should do so), study the introduction to your chosen Class in the Java API. A 2-page (or longer) write-up about these three steps are required in your project submission.

If you implement some graphical simulation (strongly encouraged), you can limit *n t*o 12 and *m* to 4.

Test Plans and Test Records

Adequate test cases should be prepared and documented.