CS 2003 Fundamentals of Algorithms and Computer Applications

Simulation of task arrival and processing in queue(s)

In this lab you will write a program that will simulate the arrival and processing of tasks in a queue over time. The task description will contain the *id*, *arrival time*, and *transaction time* for the task. These will be read in from a file called `sim.in`. You will have to find out the average wait times for the tasks from the time they arrive to the time their processing starts. Copy `sim.in`, `Lab8.java`, `TaskQueue.java`, `Task.java`, `Node.java`, `QueueReferenceBased.java`, `QueueException.java`, and `QueueInterface.java` from the the `~class_sandip/2003/` directory on `linux.ens.utulsa.edu` to your account.

In the main function first a queue is created which is used to store a series of task descriptions from the file `sim.in`. An instance of a TaskQueue is created which simulates the processing of tasks in a queue. Each task is then inserted into the TaskQueue when that task's arrival time is reached in the simulation. Each call to the *process* method of the TaskQueue object simulates one time unit of processing of the task queue and increments the *Time* variable which is initialized to 0 in the constructor. A new task is enqueued to the task queue when its *arrivalTime* equals *Time*. When a task arrives at the front of the queue, i.e., when a task is inserted into an empty queue or the task before it finishes processing at the front of the queue, its *startTime* is set to *Time* and the *dequeueTime* for the task queue is set to the time when this task will be completed (*startTime* + *transactionTime* of this task). The wait time for the task is the difference between its *startTime* and *arrivalTime*. When *Time* equals *dequeueTime*, the task at the front of the queue is removed.

For each time tick, print out which task, if any, is being processed, started, finished, enqueued. At the end of the simulation, print out the number of tasks processed and the average wait times for all tasks processed. You need to add code only at the places marked with `* TO COMPLETE *` in the `TaskQueue.java` file.

NOTE: We believe that it will greatly help your understanding of the lab if you used pen and paper to simulate the arrival and processing of the tasks in `sim.in` on a timeline, i.e., draw the state of the task queue for each time tick.

Extra points: You can get upto 30% bonus points for this lab if you add the following functionality by the due date:

> Increase the number of queues in the `TaskQueue` class from 1 to 4 and list the average wait time in each case. Use an array of queues and a corresponding array of `dequeueTime`s. Note that there will still only be a single `Time` variable, a single `tasks` variable, and `totalWaitTime` variable. When there are multiple queues, a newly arriving task should be inserted to one of the queues with minimum number of tasks in it.