COSC 311 Project #2 Priority Queue: Simulate Job Scheduling

Distributed: 3/8/3016 **Due**: 3/29/2016

(from Carrano)
Write a program to simulate job scheduling in an operating system. Jobs are generated at random times. Each job is given a unique id, a random priority from 1 to 4 – where 1 is the highest priority. Every job will require a fixed amount of time (10 units) to complete its execution.

Jobs do not begin execution and run to completion, but instead share a single processor. The operating system executes a job for a fixed unit of time called a time slice (or a quantum). At the end of the time slice, the current job's execution is suspended. The job is then placed on a priority queue, where it waits for its next share of processor time. The job having the highest priority is then removed from the priority queue and executed for a time slice.

To simplify this project, generate 6 jobs (all the jobs used in the simulation) during initialization phase. All are 'ready' to execute immediately. All are placed on the priority queue. Simulation time starts at `time = 0`, at which point the highest priority job is given the CPU.

When a job needs to be assigned to the CPU, take the highest priority job and let it 'run' for a time slice. After the job finishes its quantum, put it back on the priority queue if more time is required by the job. Obviously you will have to decrement the job's `time_to_completion` field. Repeat.

When a job completes, do not place it back on the priority queue.

The length of a time slice is 3 – thus each job will require 4 context switches. A job may (will) complete during its last time slice. As soon as the job terminates, start a new quantum with the highest priority job.

A run of this program should output a title line with current date and time. For example:

```
Job Scheduling Simulation  Tue Mar 08 2016 09:21:31 GMT-0500
```

At each context switch (i.e., give CPU to job), output the following:

```
Time         Job_id       priority         time_to_completion.
```

"Time" refers to simulated clock time.

We expect there to be 6 jobs * 4 context switch/job = 24 lines tracing the simulation.

Turn in:

Hard copy of source code, with javadoc comments,
UML (showing classes, methods, data) and relationships between classes,
Screen shot of executing code.

Grade based on:

Meets specs 75%
(includes UML, screen shot, …)
Elegance, readability 25%
(includes javadoc, use of white space, names, good design, …)