Joshua Nelson
CSCE 146

Process Queue Lab Report

Problem

Create a queue which can hold arbitrary objects and implements enque, deque, and peek. Create a class Process which has a name and time associated. Use the two to emulate a call stack, with the help of the provided driver.

Proposed Solution

Use a linked list to implement a queue. It will use a ListNode subclass to hold references to the next data object in the queue. It will have two major data points, head and tail. When queuing, place objects after tail; when dequeuing, remove object from head.

Tests and Results

```
PS C:\Users\jynelson\Documents\146\Labs\04\src> java ProcessSchedulerSimulator >
expected.log
WARNING: process given negative time -0.806. Using 0 instead
WARNING: process given negative time -0.649. Using 0 instead
WARNING: process given negative time -0.616. Using 0 instead
WARNING: process given negative time -0.258. Using 0 instead\
WARNING: process given negative time -0.962. Using 0 insteadPS
C:\Users\jynelson\Documents\146\Labs\04\src> diff log.txt expected.log
InputObject  SideIndicator
-----------  -------------
expected.log =>
log.txt      <=
```

Problems Encountered

The scheduler gave illegal times for processes. I had to decide how best to treat negative times. I decided to throw a warning and set the time to zero.

I also encountered difficulty implementing an ArrayQueue class – since arrays are not resizable, I had to either throw an exception or copy an array to a larger one any time it ran out of space. As I did not use the ArrayQueue in this project, I opted to throw an exception.

Joshua Nelson
CSCE 146

Conclusions and Discussion

This lab was much easier than expected, building smoothly on top of linked lists. It was fun to watch the process simulator run many processes at once. If this is at all similar to a call stack, I am much less intimidated than I was before this lab.

Additional Questions

1. Describe how a queue is structured.

   A queue is a first-in-first-out data structure. It can be imagined as a lunch line – objects arriving first leave first, objects arriving last leave last. Dequeuing is a person reaching the front of the line and exiting; queuing is waiting at the back of the line.

2. What are the differences between a queue and a stack?

   A stack is a first-in-last-out data structure. It can be imagined as a barrel – objects placed in first remain on the bottom until all the items on top have been removed.