Preston Stosur-Bassett

CS210 (Data Structure) - Howser

8.10.14

KQueue Mini-Lab

Pre-Lab:

I expect to learn not only how to implement a Queue using a linked list, but also to implement Java Interfaces and all that is involved with that. Being new to Java, this will be my first time implementing and extending Interfaces and Classes, and I am a little concerned on what the best practices are. I believe the lab itself will look past implementation and try and convey more on what is the best way to implement a queue (using what type of list) for each different task that could come up in our programming careers. I expect to write an all new Java class called LLQueue that will implement KQueue and java.util.LinkedList.

Post-Lab:

What I learned from this lab is how to use a queue, more than how to implement a queue, which is what I had originally expected to learn. Because so many of the features of a queue are already provided for in java.util.LinkedList it makes sense to learn best how to manage a queue using the LinkedList already provided. I now feel that I have a very good understanding of how Queues work and how to use them when writing programs of my own (regardless of the language). The only issue I came across when writing this was type errors. At first my program would not compile because I had not Typed the class that I had made with type <T>. After I fixed that I got another error that when I dequeued an element from the queue, I would have to deal with the type, and although my Queue had been initialized with type String, I had to initialize a new Object type when dequeuing.

Analysis Questions:

* Consider the following data structures. Which ones could be used as the internal data representation for a KQueue implementation with the performance characteristics described above?
  + a Java array?
  + an ArrayList object?
  + a LinkedList object?
* Which do you think would be easiest to implement?
  + I believe that implementing a LinkedList object queue would be easiest when trying to achieve a performance of O(1). When implementing using a Java array or ArrayList you have to shift elements or keep a front pointer that will cycle around the array, however in an ArrayList you could grow the array, but that would be O(n).
* What happens when you ask for the queues size? Should there be a size method? If so, where would you put it?
  + Currently, you can not find out the size of a queue, even though in my implementation there is a size variable keeping track of the size, there is not a method that will return that size to the user of the queue. If I were to add a size method to my queue I would add it to my LLQueue class and have it return the size int that is already keeping track of the size of the queue. I’m not entirely sure why a size method would be useful, because it a user wants to dequeue all the elements in the queue they can just use a while statement and use the boolean that is returned from the isEmpty method.