Stacks and Queues

2-14-2006

Opening Discussion

- Do you have any questions about the quiz?
- What did we talk about last class?
- Do you have any questions about the reading?
- Do you have any questions about the assignment?

Thread Problems

- The primary problem one runs into with multithreaded programs is that threads share memory and more than one thread can access a piece of memory at once. This isn't a problem if they are just reading, but if any thread is writing you can have bad situations.
- An extreme condition would be to consider two threads operating on an array. Worst case is both are sorting the array at the same time. You could imagine one sorting while another tries to do a binary search and the results are similarly bad.
- The simplest (and most common) example is a bank account where a race condition occurs.

Synchronization

- The way to prevent two threads from accessing the same piece of memory at the same time is to synchronize the critical pieces of the code. You can put the synchronized keyword in front of methods or make synchronized blocks.
- Each object and class in Java can have a monitor that is locked when synchronized code is being executed. Only one thread can hold the lock on the monitor at a given time. This insures that you never have two threads executing critical code on a single object at the same time.

Wait and Notify

- We can get even more control over how threads behave with the wait and notify methods.
- The wait method will stop the execution of a thread until some other thread tells it to continue execution. The notify and notifyAll methods are how threads tell other threads that they are supposed to wake up.
- All of these must be called by a thread that holds the monitor to the object they are being invoked on. Typically that means that are called from inside synchronized code.

ADTs

- We now enter into the realm of the abstract data type. The term ADT in many ways describes the idea of an interface with documentation for what it should do.
- An ADT tells you that you have certain operations and tells you what those operations do. It does NOT tell you how those operations are done. This separation of interface from implementation is one of the hallmarks of good object-oriented code.
- We will learn about multiple ADTs this semester and you will learn about even more in Data Abstraction. Today we start with the simplest of them.

Stacks and Queues

- What is a stack? What operations can you do on a stack?
- What is a queue? What operations can you do on a queue?
- What is the primary difference between a stack and a queue?
- How do you implement a stack with an array?
- How do you implement a queue with an array?

Code

- I want you to write interfaces for a Stack and a Queue. Call them MyStack and MyQueue. Then implement a stack using an array based implementation.
- After we have done that we'll write a queue. I'm very displeased with the queue code in your book. I don't want you to mimic it. The reason is that the operations should be O(1) and their dequeue in O(n), much slower than it needs to be.

Minute Essay

- Can you think of ways that you could parallelize a sort algorithm? Would your method need any synchronization?
- Remember to read chapter #4 for next class.