questions

October 18, 2022

1. When many producer threads are waiting on a full queue, what happens when a thread `takes` a String and calls `notifyAll()`? Why is this thread safe?

As soon as the consumer thread `takes` an item `notifyAll()` will awake all sleeping threads. This operation is thread safe because the syncronized blocks ensures that only one of the awaken threads puts a new item in the queue.

2. Read the documentation for sychronized methods. Do synchronized methods synchronize on the object or the class? Why is this the right scope? (Consider that there may be multiple queues in an application.)

Synchronized methods synchronize on the object level. For this specific example, it is the right scope because all producers and consumers share a reference to the same queue. In the situation of having multiple queues, synchronizing on the object level is the best way to things because there is no danger of a race condition between threads that work in different queues. Since there is no danger of a race condition, it would not make sense to wait all the threads everytime a queue fills up or empties.

3. One might want the String copy operations to proceed in parallel because they are expensive (see below). This is not thread safe. Why? Give an example.

This would not be thread safe because all threads would be able to read and write to the `queue` without any constraints. A situation where this could be dangerous is if one of the threads is writing a long string to the array and in the middle of the operation another reads that same memory location. In this case, the reading thread would obtain a incomplete value.