```
import java.util.concurrent.atomic.*;

/**
 * MSQueue - Insertion in the Michael-Scott nonblocking queue algorithm
 * Code based on the original version by:
 * @author Brian Goetz and Tim Peierls
 */

public class MSQueue<E> {

    private final MSQueue.Node<E> dummy
            = new MSQueue.Node<>(null);               //the initial "dummy" node
    private final AtomicReference<MSQueue.Node<E>> head
            = new AtomicReference<>(dummy);           //reference to queue head
    private final AtomicReference<MSQueue.Node<E>> tail
            = new AtomicReference<>(dummy);           //reference to queue tail

    public boolean put(E item) {
        MSQueue.Node<E> newNode = new MSQueue.Node<>(item); //new node created

        while (true) {
            MSQueue.Node<E> curTail = tail.get();         //"tail" value is read
            MSQueue.Node<E> tailNext = curTail.next.get();//next of "tail"

            if (curTail == tail.get()) {
                if (tailNext != null) {  // queue in intermediate state...
                    tail.compareAndSet(curTail, tailNext);   //advance tail
                } else {                 // queue in quiescent state...
                    // try inserting the new node
                    if (curTail.next.compareAndSet(null, newNode)) {
                        // insertion succeeded, try advancing tail
                        tail.compareAndSet(curTail, newNode);
                        return true;
                    }
                }
            }
        }
    }

    private static class Node<E> {  //inner class to support a "node"
        final E item;
        AtomicReference<MSQueue.Node<E>> next; //this ref. must be "atomic"!

        public Node(E item) {
            this.item = item;
        }
    }

}
```