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

/**
 * TreiberStack - Nonblocking stack using Treiber's algorithm
 * Code based on the original version by:
 * @author Brian Goetz and Tim Peierls
 */

public class TreiberStack <E> {
    AtomicReference<Node<E>> top = new AtomicReference<>();

    public void push(E item) {
        Node<E> newTop = new Node<>(item); //the new node is created
        Node<E> oldTop; //local var to host the value read from "top"

        do {
            oldTop = top.get();    //the value from "top" is read
            newTop.next = oldTop;  //the value for "next" in the new node is set
        } while (!top.compareAndSet(oldTop, newTop)); //attempt to finalize insertion
    }

    public E pop() {
        Node<E> oldTop;
        Node<E> newTop;

        do {
            oldTop = top.get();    //the value from "top" is read
            if (oldTop == null)    //empty stack!
                return null;
            newTop = oldTop.next;  //the possible new value for "top" is retrieved
        } while (!top.compareAndSet(oldTop, newTop)); //finalize only if setting unchanged

        return oldTop.item;
    }

    private static class Node <E> {
        public final E item;
        public Node<E> next;

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