```
public class LockFreeStack<T> {
  AtomicReference<Node> top = new AtomicReference<Node>(null);
  static final int MIN_DELAY = ...;
  static final int MAX_DELAY = ...;
  Backoff backoff = new Backoff(MIN_DELAY, MAX_DELAY);

  protected boolean tryPush(Node node){
    Node oldTop = top.get();
    node.next = oldTop;
    return(top.compareAndSet(oldTop, node));
  }
  public void push(T value) {
    Node node = new Node(value);
    while (true) {
      if (tryPush(node)) {
        return;
      } else {
        backoff.backoff();
      }
    }
  }
  ...
}
```

FIGURE 11.2 The LockFreeStack<T> class: In the push() method, threads alternate between trying to alter the top reference by calling tryPush(), and backing off using the Backoff class from Fig. 7.5.