```
1  public class Stack<T> {
2    private AtomicInteger top;
3    private T[] items;
4    public Stack(int capacity) {
5      top = new AtomicInteger();
6      items = (T[]) new Object[capacity];
7    }
8    public void push(T x) throws FullException {
9      int i = top.getAndIncrement();
10     if (i >= items.length) { // stack is full
11       top.getAndDecrement(); // restore state
12       throw new FullException();
13     }
14     items[i] = x;
15   }
16   public T pop() throws EmptyException {
17     int i = top.getAndDecrement() - 1;
18     if (i < 0) {           // stack is empty
19       top.getAndIncrement(); // restore state
20       throw new EmptyException();
21     }
22     return items[i];
23   }
24 }
```

FIGURE 11.12 Unsynchronized concurrent stack.