```
1  public class SimpleSnapshot<T> implements Snapshot<T> {
2    private StampedValue<T>[] a_table; // array of atomic MRSW registers
3    public SimpleSnapshot(int capacity, T init) {
4      a_table = (StampedValue<T>[]) new StampedValue[capacity];
5      for (int i = 0; i < capacity; i++) {
6        a_table[i] = new StampedValue<T>(init);
7      }
8    }
9    public void update(T value) {
10     int me = ThreadID.get();
11     StampedValue<T> oldValue = a_table[me];
12     StampedValue<T> newValue = new StampedValue<T>((oldValue.stamp)+1, value);
13     a_table[me] = newValue;
14   }
15   private StampedValue<T>[] collect() {
16     StampedValue<T>[] copy = (StampedValue<T>[]) new StampedValue[a_table.length];
17     for (int j = 0; j < a_table.length; j++)
18       copy[j] = a_table[j];
19     return copy;
20   }
21   public T[] scan() {
22     StampedValue<T>[] oldCopy, newCopy;
23     oldCopy = collect();
24     collect: while (true) {
25       newCopy = collect();
26       if (! Arrays.equals(oldCopy, newCopy)) {
27         oldCopy = newCopy;
28         continue collect;
29       }
30       T[] result = (T[]) new Object[a_table.length];
31       for (int j = 0; j < a_table.length; j++)
32         result[j] = newCopy[j].value;
33       return result;
34     }
35   }
36 }
```

FIGURE 4.17 Simple snapshot object.

Copyright © 2021 Elsevier Inc. All rights reserved.