```java
public class StaticTreeBarrier implements Barrier {
  int radix;
  boolean sense;
  Node[] node;
  ThreadLocal<Boolean> threadSense;
  int nodes;

  public StaticTreeBarrier(int size, int myRadix) {
    radix = myRadix;
    nodes = 0;
    node = new Node[size];
    int depth = 0;
    while (size > 1) {
      depth++;
      size = size / radix;
    }
    build(null, depth);
    sense = false;
    threadSense = new ThreadLocal<Boolean>() {
      protected Boolean initialValue() { return !sense; };
    };
  }
  // recursive tree constructor
  void build(Node parent, int depth) {
    if (depth == 0) {
      node[nodes++] = new Node(parent, 0);
    } else {
      Node myNode = new Node(parent, radix);
      node[nodes++] = myNode;
      for (int i = 0; i < radix; i++) {
        build(myNode, depth - 1);
      }
    }
  }
  public void await() {
    node[ThreadID.get()].await();
  }
}
```

FIGURE 18.8 The StaticTreeBarrier class: Each thread indexes into a statically assigned tree node and calls that node's await() method.