```
public class RevBarrier implements Barrier {
  int radix;
  ThreadLocal<Boolean> threadSense;
  int leaves;
  Node[] leaf;
  public RevBarrier(int mySize, int myRadix) {
    radix = myRadix;
    leaves = 0;
    leaf = new Node[mySize / myRadix];
    int depth = 0;
    threadSense = new ThreadLocal<Boolean>() {
      protected Boolean initialValue() { return true; };
    };
    // compute tree depth
    while (mySize > 1) {
      depth++;
      mySize = mySize / myRadix;
    }
    Node root = new Node();
    root.d = depth;
    build(root, depth - 1);
  }
  // recursive tree constructor
  void build(Node parent, int depth) {
    // are we at a leaf node?
    if (depth == 0) {
      leaf[leaves++] = parent;
    } else {
      for (int i = 0; i < radix; i++) {
        Node child = new Node(parent);
        child.d = depth;
        build(child, depth - 1);
      }
    }
  }
```

FIGURE 18.18 Reverse tree barrier part 1.