```
            addGrain(x+1, y);
         }
         if(x>0) {
            addGrain(x-1, y);
         }
         if(y+1<L) {
            addGrain(x, y+1);
         }
         if(y>0) {
            addGrain(x, y-1);
         }
         if(numberOfSitesToTopple>0) {
            // next site to topple
            siteToTopple[0] = toppleSiteX[numberOfSitesToTopple-1];
            siteToTopple[1] = toppleSiteY[numberOfSitesToTopple-1];
            return true;
         } else {
            return false;
         }
      }

      public void addGrain(int x, int y) {
         int h = height.getValue(x, y)+1;
         height.setValue(x, y, h); // add grain to site
         height.render();
         if(h==4) { // new site to topple
            toppleSiteX[numberOfSitesToTopple] = x;
            toppleSiteY[numberOfSitesToTopple] = y;
            numberOfSitesToTopple++;
         }
      }

      public void resetAverages() {
         distribution = new int[numberToppledMax];
         numberOfGrains = 0;
      }
   }
```

Listing 14.7 The target class for the two-dimensional sandpile model.

```
package org.opensourcephysics.sip.ch14.sandpile;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.*;

public class SandpileApp extends AbstractSimulation {
   Sandpile sandpile = new Sandpile();
   LatticeFrame height = new LatticeFrame("x", "y", "Sandpile");
   PlotFrame plotFrame = new PlotFrame("ln s", "ln N",
        "Distribution of toppled sites");

   public SandpileApp() {
      height.setIndexedColor(0, java.awt.Color.WHITE);
      height.setIndexedColor(1, java.awt.Color.BLUE);
      height.setIndexedColor(2, java.awt.Color.GREEN);
      height.setIndexedColor(3, java.awt.Color.RED);
      height.setIndexedColor(4, java.awt.Color.BLACK);
   }
```



```
   public void initialize() {
      sandpile.L = control.getInt("L");
      height.setPreferredMinMax(0, sandpile.L, 0, sandpile.L);
      sandpile.initialize(height);
   }

   public void doStep() {
      sandpile.step();
   }

   public void stop() {
      plotFrame.clearData();
      for(int s = 1;s<sandpile.distribution.length;s++) {
         double f = (double)sandpile.distribution[s];
         double N = (double)sandpile.numberOfGrains;
         if(f>0) {
            plotFrame.append(0, Math.log(s), Math.log(f/N));
         }
      }
      plotFrame.render();
   }

   public void reset() {
      control.setValue("L", 10);
      enableStepsPerDisplay(true);
   }

   public void resetAverages() {
      sandpile.resetAverages();
   }

   public static void main(String[] args) {
      SimulationControl control =
            SimulationControl.createApp(new SandpileApp());
      control.addButton("resetAverages", "resetAverages");
   }
}
```

Problem 14.7 A two-dimensional sandpile model

(a) Use the classes `Sandpile` and `SandpileApp` to simulate a two-dimensional sandpile with linear dimension L. Run the simulation with $L = 10$ and stop it once toppling starts to occur. When this behavior occurs, black cells (with four grains) will momentarily appear. Use the Step button to watch individual toppling events and obtain a qualitative sense of the dynamics of the sandpile model.

(b) Comment out the `height.render()` statements in `Sandpile` and add a statement to `SandPileApp` so that the number of grains added to the system is displayed. (The number of grains added is a measure of the number of configurations that are included in the various averages.) Now you will not be able to see individual toppling events, but you can more quickly collect data on the toppling distribution, the frequency of the number of sites that topple when a grain is added. The program outputs a log-log plot of the distribution. Estimate the slope of the log-log distribution from the part of the plot that is linear and thus determine the power law exponent α. Reset the