```
      double meanVirial = virialAccumulator/steps;
      // quantity PA/NkT
      return 1.0+0.5*meanVirial/(N*getMeanTemperature());
   }

   public double getHeatCapacity() {
      double meanTemperature = getMeanTemperature();
      double meanTemperatureSquared =
           totalKineticEnergySquaredAccumulator/steps;
      // heat capacity related to fluctuations of kinetic energy
      double sigma2 =
           meanTemperatureSquared-meanTemperature*meanTemperature;
      double denom = sigma2/(N*meanTemperature*meanTemperature) - 1.0;
      return N/denom;
   }

   public void resetAverages() {
      steps = 0;
      virialAccumulator = 0;
      totalPotentialEnergyAccumulator = 0;
      totalKineticEnergyAccumulator = 0;
      totalKineticEnergySquaredAccumulator = 0;
   }
```

The resetAverages method is used to set the accumulated averages to zero so that the initial transient behavior can be removed from the computed averages.

We use the Drawable interface to display the trajectories of the individual particles.

Listing 8.9 The draw method.

```
   public void draw(DrawingPanel panel, Graphics g) {
      if(state==null) {
         return;
      }
      int pxRadius = Math.abs(panel.xToPix(radius)-panel.xToPix(0));
      int pyRadius = Math.abs(panel.yToPix(radius)-panel.yToPix(0));
      g.setColor(Color.red);
      for(int i = 0;i<N;i++) {
         int xpix = panel.xToPix(state[4*i])-pxRadius;
         int ypix = panel.yToPix(state[4*i+2])-pyRadius;
         g.fillOval(xpix, ypix, 2*pxRadius, 2*pyRadius);
      }
      // draw central cell boundary
      g.setColor(Color.black);
      int xpix = panel.xToPix(0);
      int ypix = panel.yToPix(Ly);
      int lx = panel.xToPix(Lx)-panel.xToPix(0);
      int ly = panel.yToPix(0)-panel.yToPix(Ly);
      g.drawRect(xpix, ypix, lx, ly);
   }
```

The beginning of the LJParticles class includes the usual import statements, instance variables, and the initialize method. If you include all of the code we have discussed in a single file, you will have a working LJparticles class. Alternatively, you can download the source code from the ch08 directory.

Listing 8.10 Beginning of class LJParticles.

```
package org.opensourcephysics.sip.ch08.md;
import java.awt.*;
import org.opensourcephysics.display.*;
import org.opensourcephysics.frames.HistogramFrame;
import org.opensourcephysics.numerics.*;

public class LJParticles implements Drawable, ODE {
   public double state[];
   public double ax[], ay[];
   // number of particles, number per row, number per column
   public int N, nx, ny;
   public double Lx, Ly;
   public double rho = N/(Lx*Ly);
   public double initialKineticEnergy;
   public int steps = 0;
   public double dt = 0.01;
   public double t;
   public double totalPotentialEnergyAccumulator;
   public double totalKineticEnergyAccumulator;
   Public double totalKineticEnergySquaredAccumulator;
   public double virialAccumulator;
   public String initialConfiguration;
   public double radius = 0.5;    // radius of particles on screen
   ODESolver odeSolver = new Verlet(this);

   public void initialize() {
      N = nx*ny;
      t = 0;
      rho = N/(Lx*Ly);
      resetAverages();
      state = new double[1+4*N];
      ax = new double[N];
      ay = new double[N];
      if(initialConfiguration.equals("triangular")) {
         setTriangularLattice();
      } else if(initialConfiguration.equals("rectangular")) {
         setRectangularLattice();
      } else {
         setRandomPositions();
      }
      setVelocities();
      computeAcceleration();
      odeSolver.setStepSize(dt);
   }
```

The target class is given in Listing 8.11. When the user presses the Stop button, various thermodynamic averages are displayed in the message area of the control window. As you will find in Problem 8.8, a time consuming part of a molecular dynamics simulation is equilibrating the system, especially at high densities. The quickest way to do so is to start with a configuration that is typical of the configurations at the desired energy and density. Hence, we will want to be able to read the positions and velocities of the particles from a previously saved file. The class LJParticlesLoader allows us to save a configuration. This class is used in the getLoader method in class LJParticlesApp. To save a given