```
      demonEnergy = 0;
      resetData();
   }

   public void resetData() {
      mcs = 0;
      systemEnergyAccumulator = 0;
      demonEnergyAccumulator = 0;
      acceptedMoves = 0;
   }

   public void doOneMCStep() {
      for(int j = 0;j<N;++j) {
         // choose particle at random
         int particleIndex = (int) (Math.random()*N);
         // random change in velocity
         double dv = (2.0*Math.random()-1.0)*delta;
         double trialVelocity = v[particleIndex]+dv;
         double dE = 0.5*(trialVelocity*trialVelocity-
              v[particleIndex]*v[particleIndex]);
         if(dE<=demonEnergy) {
            v[particleIndex] = trialVelocity;
            acceptedMoves++;
            systemEnergy += dE;
            demonEnergy -= dE;
         }
         systemEnergyAccumulator += systemEnergy;
         demonEnergyAccumulator += demonEnergy;
      }
      mcs++;
   }
}
```

Listing 15.2 The target class for the simulation of an ideal gas using the demon algorithm.

```
package org.opensourcephysics.sip.ch15;
import org.opensourcephysics.controls.*;

public class IdealDemonApp extends AbstractSimulation {
   IdealDemon idealGas = new IdealDemon();

   public void initialize() {
      idealGas.N = control.getInt("number of spins N");
      idealGas.systemEnergy =
           control.getDouble("desired total energy");
      idealGas.delta = control.getDouble("maximum velocity change");
      idealGas.initialize();
   }

   public void doStep() {
      idealGas.doOneMCStep();
   }

   public void stop() {
      double norm = 1.0/(idealGas.mcs*idealGas.N);
      control.println("mcs = "+idealGas.mcs);
```

```
      control.println("<Ed> = "+idealGas.demonEnergyAccumulator*norm);
      control.println("<E> = "+idealGas.systemEnergyAccumulator*norm);
      control.println("acceptance ratio = "+
           idealGas.acceptedMoves*norm);
   }

   public void reset() {
      control.setValue("number of spins N", 40);
      control.setValue("desired total energy", 40);
      control.setValue("maximum velocity change", 2.0);
   }

   public void resetData() {
      idealGas.resetData();
      idealGas.delta = control.getDouble("maximum velocity change");
      control.clearMessages();
   }

   public static void main(String[] args) {
      SimulationControl control =
           SimulationControl.createApp(new IdealDemonApp());
      control.addButton("resetData", "Reset Data");
   }
}
```

Problem 15.2 Monte Carlo simulation of an ideal gas

(a) Use the classes `IdealDemon` and `IdealDemonApp` to investigate the equilibrium properties of an ideal gas. Note that the mass of the particles has been set equal to unity and the initial demon energy is zero. For simplicity, the same initial velocity has been assigned to all the particles. Begin by using the default values given in the listing of `IdealDemonApp`. What is the mean value of the particle velocities after equilibrium has been reached?

(b) The configuration corresponding to all particles having the same velocity is not very likely, and it would be better to choose an initial configuration that is more likely to occur when the system is in equilibrium. In any case, we should let the system evolve until it has reached equilibrium before we accumulate data for the various averages. We call this time the equilibration or relaxation time. We can estimate the equilibration time from a plot of the demon energy versus the time. Alternatively, we can reset the data until the computed averages stop changing systematically. Clicking the Reset Data button sets the accumulated sums to zero without changing the configuration. Determine the mean demon energy $\langle E_d \rangle$ and the mean system energy per particle using the default values for the parameters.

(c) Compute the mean energy of the demon and the mean system energy per particle for $N = 100$ and $E = 10$ and $E = 20$, where E is the total energy of the system. Use your result from part (b) and obtain an approximate relation between the mean demon energy and the mean system energy per particle.

(d) In the microcanonical ensemble the total energy is fixed with no reference to the temperature. Define the kinetic temperature by the relation $\frac{1}{2}m\langle v^2\rangle = \frac{1}{2}kT_{\text{kinetic}}$, where $\frac{1}{2}m\langle v^2\rangle$ is the mean kinetic energy per particle of the system. Use this relation to