configuration, open the File menu in the control window and choose Save As.... A dialog box will open so that you can choose a name for the file, which will have the extension `xml`. To read a previously saved configuration, choose Read in the File menu. Notice that in `LJParticlesLoader`, the `loadObject` makes a call to the `computeAcceleration` and `resetAverages` methods of `LJParticles`, so that the simulation can be run starting from the newly loaded configuration by next clicking the Start button. The syntax for saving a model's configuration is described in more detail in Appendix 8A.

Listing 8.11 The `LJParticlesApp` target class.

```
package org.opensourcephysics.sip.ch08.md;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.*;
import org.opensourcephysics.display.GUIUtils;

public class LJParticlesApp extends AbstractSimulation {
   LJParticles md = new LJParticles();
   PlotFrame pressureData =
        new PlotFrame("time", "PA/NkT", "Mean pressure");
   PlotFrame temperatureData =
        new PlotFrame("time", "temperature", "Mean temperature");
   HistogramFrame xVelocityHistogram =
        new HistogramFrame("vx", "H(vx)", "Velocity histogram");
   DisplayFrame display =
        new DisplayFrame("x", "y", "Lennard-Jones system");

   public void initialize() {
      md.nx = control.getInt("nx"); // number of particles per row
      md.ny = control.getInt("ny"); // number of particles per column
      md.initialKineticEnergy =
           control.getDouble("initial kinetic energy per particle");
      md.Lx = control.getDouble("Lx");
      md.Ly = control.getDouble("Ly");
      md.initialConfiguration =
           control.getString("initial configuration");
      md.dt = control.getDouble("dt");
      md.initialize();
      display.addDrawable(md);
      // assumes vmax = 2*initalTemp and bin width = Vmax/N
      display.setPreferredMinMax(0, md.Lx, 0, md.Ly);
      xVelocityHistogram.setBinWidth(2*md.initialKineticEnergy/md.N);
   }

   public void doStep() {
      md.step(xVelocityHistogram);
      pressureData.append(0, md.t, md.getMeanPressure());
      temperatureData.append(0, md.t, md.getMeanTemperature());
   }

   public void stop() {
      control.println("Density = "+decimalFormat.format(md.rho));
      control.println("Number of time steps = "+md.steps);
      control.println("Time step dt = "+decimalFormat.format(md.dt));
      control.println("<T>= "
                      +decimalFormat.format(md.getMeanTemperature()));
      control.println("<E> = "
                      +decimalFormat.format(md.getMeanEnergy()));
      control.println("Heat capacity = "
                      +decimalFormat.format(md.getHeatCapacity()));
      control.println("<PA/NkT> = "
                      +decimalFormat.format(md.getMeanPressure()));
   }

   public void startRunning() {
      md.dt = control.getDouble("dt");
      double Lx = control.getDouble("Lx");
      double Ly = control.getDouble("Ly");
      if((Lx!=md.Lx)||(Ly!=md.Ly)) {
         md.Lx = Lx;
         md.Ly = Ly;
         md.computeAcceleration();
         display.setPreferredMinMax(0, Lx, 0, Ly);
         resetData();
      }
   }

   public void reset() {
      control.setValue("nx", 8);
      control.setValue("ny", 8);
      control.setAdjustableValue("Lx", 20.0);
      control.setAdjustableValue("Ly", 15.0);
      control.setValue("initial kinetic energy per particle", 1.0);
      control.setAdjustableValue("dt", 0.01);
      control.setValue("initial configuration", "rectangular");
      enableStepsPerDisplay(true);
      // draw configurations every 10 steps
      super.setStepsPerDisplay(10);
      // particles will appear as circular disks
      display.setSquareAspect(true);
   }

   public void resetData() {
      md.resetAverages();
      // clears old data from the plot frames
      GUIUtils.clearDrawingFrameData(false);
   }

   public static XML.ObjectLoader getLoader() {
      return new LJParticlesLoader();
   }

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


Listing 8.12 The `LJParticlesLoader` class for saving configurations.

```
package org.opensourcephysics.sip.ch08.md;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.display.GUIUtils;

public class LJParticlesLoader implements XML.ObjectLoader {
```