Listing 5.9 A class that models the dynamics of the three-body problem.

```
package org.opensourcephysics.sip.ch05;
import java.awt.*;
import org.opensourcephysics.display.*;
import org.opensourcephysics.numerics.*;

public class ThreeBody implements Drawable, ODE {
   int n = 3; // number of interacting bodies
   // state= {x1, vx1, y1, vy1, x2, vx2, y2, vy2, x3, vx3, y3, vy3, t}
   double[] state = new double[4*n+1];
   double[] force = new double[2*n];
   double[] zeros = new double[2*n];
   ODESolver odeSolver = new RK45MultiStep(this);
   Mass mass1 = new Mass(), mass2 = new Mass(), mass3 = new Mass();

   public void draw(DrawingPanel panel, Graphics g) {
      mass1.draw(panel, g);
      mass2.draw(panel, g);
      mass3.draw(panel, g);
   }

   public void doStep() {
      odeSolver.step();
      mass1.setXY(state[0], state[2]);
      mass2.setXY(state[4], state[6]);
      mass3.setXY(state[8], state[10]);
   }

   void initialize(double[] initState) {
      // copies initState to state
      System.arraycopy(initState, 0, state, 0, 13);
      mass1.clear(); // clears data from old trail
      mass2.clear();
      mass3.clear();
      mass1.setXY(state[0], state[2]);
      mass2.setXY(state[4], state[6]);
      mass3.setXY(state[8], state[10]);
   }

   void computeForce(double[] state) {
      // sets force array elements to 0
      System.arraycopy(zeros, 0, force, 0, force.length);
      for(int i = 0;i<n;i++) {
         for(int j = i+1;j<n;j++) {
            double dx = state[4*i]-state[4*j];
            double dy = state[4*i+2]-state[4*j+2];
            double r2 = dx*dx+dy*dy;
            double r3 = r2*Math.sqrt(r2);
            double fx = dx/r3;
            double fy = dy/r3;
            force[2*i] -= fx;
            force[2*i+1] -= fy;
            force[2*j] += fx;
            force[2*j+1] += fy;
         }
      }
   }
```

```
   public void getRate(double[] state, double[] rate) {
      computeForce(state); // force array alternates fx and fy
      for(int i = 0;i<n;i++) {
         int i4 = 4*i;
         rate[i4] = state[i4+1];     // x rate is vx
         rate[i4+1] = force[2*i];    // vx rate is fx
         rate[i4+2] = state[i4+3];   // y rate is vy
         rate[i4+3] = force[2*i+1];  // vy rate is fy
      }
      rate[state.length-1] = 1; // time rate is last
   }

   public double[] getState() {
      return state;
   }

   class Mass extends Circle {
      Trail trail = new Trail();

      // draws the mass
      public void draw(DrawingPanel panel, Graphics g) {
         trail.draw(panel, g);
         super.draw(panel, g);
      }

      // clears trail
      void clear() {
         trail.clear();
      }

      // sets postion and adds to trail
      public void setXY(double x, double y) {
         super.setXY(x, y);
         trail.addPoint(x, y);
      }
   }
}
```

The initial conditions for our examples are in the ThreeBodyInitialConditions class. This file is available in the ch05 package but is not listed here because it contains mostly numeric data.

In 1765 Euler discovered an analytic solution in which three masses start on a line and rotate so that the central mass stays fixed. The EULER array in ThreeBodyInitialConditions initializes the model to produce this type of solution. The first mass is placed at the center, and the other two masses are placed on opposite sides with velocities that are equal but opposite. Because of the symmetry, the trajectories are ellipses with a common focus at the center.

A second analytic solution to the unrestricted three-body problem was found by Lagrange in 1772. This solution starts with three masses at the corners of an equilateral triangle. Each mass moves in an ellipse in such a way that the triangle formed by the masses remains equilateral. The LAGRANGE array initializes this solution.

A spectacular new solution that adds to the sparse list of analytic three-body solutions was first discovered numerically by Moore and proven to be stable by Chenciner and Montgomery. The MONTGOMERY array contains the initial conditions for this solution.