For example, if the initial displacement of every particle is zero, and the initial velocity of every particle is zero except for $v_1(0) = 1$, we find $A_n = 0$ for all n, and

$$B_n = \frac{C}{\omega_n} \sin q_n a. \tag{9.16}$$

The corresponding solution for $u_j(t)$ is

$$u_j(t) = \frac{2}{N+1} \sum_{n=1}^{N} \frac{1}{\omega_n} \cos \omega_n t \sin q_n a \sin q_n j a. \tag{9.17}$$

What is the solution if the particles start in a normal mode, that is, $u_j(t = 0) \propto \sin q_2 j a$?

The `Oscillators` class in Listing 9.1 displays the analytic solution (9.11) of the oscillator displacements. The `draw` method uses a single circle that is repeatedly set equal to the appropriate world coordinates. The initial positions are calculated and stored in the `y` array in the `Oscillators` constructor. When an oscillator is drawn, the position array is multiplied by the given mode's sinusoidal (phase) factor to produce a time-dependent displacement.

Listing 9.1 The `Oscillators` class models the evolution of a normal mode of a chain of coupled oscillators.

```
package org.opensourcephysics.sip.ch09;
import java.awt.Graphics;
import org.opensourcephysics.display.*;

public class Oscillators implements Drawable {
   OscillatorsMode normalMode;
   Circle circle = new Circle();
   double[] x; // drawing position
   double[] u; // displacement
   double time = 0;

   public Oscillators(int mode, int N) {
      u = new double[N+2]; // includes the two ends of the chain
      x = new double[N+2]; // includes the two ends of the chain
      normalMode = new OscillatorsMode(mode, N);
      double xi = 0;
      for(int i = 0;i<N+2;i++) {
         x[i] = xi;
         u[i] = normalMode.evaluate(xi); // initial displacement
         // increment x[i] by lattice spacing of one
         xi++;
      }
   }

   public void step(double dt) {
      time += dt;
   }

   public void draw(DrawingPanel drawingPanel, Graphics g) {
      normalMode.draw(drawingPanel, g); // draw initial condition
      double phase = Math.cos(time*normalMode.omega);
```



```
      for(int i = 0, n = x.length;i<n;i++) {
         circle.setXY(x[i], u[i]*phase);
         circle.draw(drawingPanel, g);
      }
   }
}
```

The `OscillatorsMode` class in Listing 9.2 instantiates a normal mode. This class stores the mode frequency and implements the `Function` interface to evaluate the analytic solution. It draws a light gray outline of the initial analytic solution using a `FunctionDrawer`.

Listing 9.2 The `OscillatorsMode` class models a normal mode of a chain of coupled oscillators.

```
package org.opensourcephysics.sip.ch09;
import java.awt.*;
import org.opensourcephysics.display.*;
import org.opensourcephysics.numerics.*;

public class OscillatorsMode implements Drawable, Function {
   static final double OMEGA_SQUARED = 1; // equals k/m
   FunctionDrawer functionDrawer;      // draws the initial condition
   double omega;                       // oscillation frequency of mode
   double wavenumber;                  // wavenumber = 2*pi/wavelength
   double amplitude;

   OscillatorsMode(int mode, int N) {
      amplitude = Math.sqrt(2.0/(N+1));
      omega = 2*Math.sqrt(OMEGA_SQUARED)
              *Math.abs(Math.sin(mode*Math.PI/N/2));
      wavenumber = Math.PI*mode/(N+1);
      functionDrawer = new FunctionDrawer(this);
      // draws the initial displacement
      functionDrawer.initialize(0, N+1, 300, false);
      functionDrawer.color = Color.LIGHT_GRAY;
   }

   public double evaluate(double x) {
      return amplitude*Math.sin(x*wavenumber);
   }

   public void draw(DrawingPanel panel, Graphics g) {
      functionDrawer.draw(panel, g);
   }
}
```

The `OscillatorsApp` target class extends `AbstractSimulation`, creates an object `Oscillators`, and displays the particle displacements as transverse oscillations. The complete listing is available in the ch09 code package, but it is not given here because it is similar to other animations.