```
   double xmin, xmax;
   Polynomial p;

   public void reset() {
      control.setValue("coefficients", "-2,0,1");
      control.setValue("minimum x", -10);
      control.setValue("maximum x", 10);
   }

   public void calculate() {
      xmin = control.getDouble("minimum x");
      xmax = control.getDouble("maximum x");
      String[] coefficients =
            control.getString("coefficients").split(",");
      p = new Polynomial(coefficients);
      plotAndCalculateRoots();
   }

   void plotAndCalculateRoots() {
      plotFrame.clearDrawables();
      plotFrame.addDrawable(new FunctionDrawer(p));
      double[] range = Util.getRange(p, xmin, xmax, 100);
      plotFrame.setPreferredMinMax(xmin, xmax, range[0], range[1]);
      plotFrame.repaint();
      double[] roots = p.roots(0.001);
      control.clearMessages();
      control.println("polynomial = "+p);
      for(int i = 0, n = roots.length;i<n;i++) {
         control.println("root = "+roots[i]);
      }
   }

   public void derivative() {
      p = p.derivative();
      plotAndCalculateRoots();
   }

   public static void main(String[] args) {
      CalculationControl control =
            CalculationControl.createApp(new PolynomialApp());
      control.addButton("derivative", "Derivative",
            "The derivative of the polynomial.");
   }
}
```

Exercise 11.23 Taylor series
Use the `PolynomialApp` class to plot the first three nonzero terms of the Taylor series expansion of the sine function. How accurate is this expansion in the interval $|x| < \pi/2$? ■

Exercise 11.24 Polynomials
Write a test program to do the following:

(a) Create the polynomial $x^4 - 5x^3 + 5x^2 + 5x - 6$ and divide this polynomial by $x - 2$. Is 2 a root of the original polynomial?
(b) Find the roots of $x^5 - 6x^4 + x^3 - 7x^2 - 7x + 12$. ■

Problem 11.25 Chebyshev polynomials
Orthogonal polynomials often can be written in terms of simple recurrence relations. For example, the Chebyshev polynomials of the first kind $T_n(x)$ can be written as

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \tag{11.83}$$

where $T_0(x) = 1$ and $T_1(x) = x$. Write and test a class using a static method that creates the Chebyshev polynomials. To improve efficiency, your class should store the polynomials as they are created during recursion. ■

It is always possible to construct a polynomial that passes through a set of n data points (x_i, y_i) by creating a *Lagrange interpolating polynomial* as follows:

$$p(x) = \sum_{i=0}^{n} \frac{\prod_{i\neq j}(x - x_j)}{\prod_{i\neq j}(x_i - x_j)} y_i. \tag{11.84}$$

For example, three data points generate the second-degree polynomial (see (11.5)):

$$p(x) = \frac{(x-x_1)(x_0-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_3. \tag{11.85}$$

Note that terms multiplying the y values are zero at the sample data points except for the term multiplying the sample data point's abscissa y_i. Various computational tricks can be used to speed the evaluation of (11.84), but these will not be discussed here (see Besset or Press et al.). We have implemented Lagrange's polynomial interpolation formula using a generalized Horner expansion in the `LagrangeInterpolator` class in the numerics package. Listing 11.6 tests this class.

Listing 11.6 `LagrangeInterpolatorApp` tests `LagrangeInterpolator` by sampling an arbitrary function and fitting the samples by a polynomial.

```
package org.opensourcephysics.sip.ch11;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.display.*;
import org.opensourcephysics.frames.*;
import org.opensourcephysics.numerics.*;

public class LagrangeInterpolatorApp extends AbstractCalculation {
   PlotFrame plotFrame = new PlotFrame("x", "f(x)",
         "Lagrange interpolation");

   public void reset() {
      control.setValue("f(x)", "sin(x)");
      control.setValue("minimum x", -2);
      control.setValue("maximum x", 2);
      control.setValue("n", 5);
      control.setValue("random y-error", 0);
      calculate();
   }

   public void calculate() {
      String fstring = control.getString("f(x)");
      double a = control.getDouble("minimum x");
      double b = control.getDouble("maximum x");
```