```
    //--------------
    public void Execute() {
        context.setPenColor(Color.red);     //set the color of the
                                              plot
        context.setLinePlot();              //set the kind of the
                                              plot
        context.readData("data.txt");       //read the data
        context.plot();                     //plot the data
    }
}
```

The Line and Bar Graph Strategies

The two Strategy classes are pretty much the same: They set up the window size for plotting and call a plot method specific for that display panel. Here is the Line graph Strategy:

```
public class LinePlotStrategy extends PlotStrategy {
    private LinePlotPanel lp;

    public LinePlotStrategy() {
        super("Line plot");
        lp = new LinePlotPanel();
        getContentPane().add(lp);
    }
//--------------
    public void plot(float[] xp, float[] yp) {
        x = xp; y = yp;             //copy in data
        findBounds();               //sets maxes and mins
        setSize(width, height);
        setVisible(true);
        setBackground(Color.white);
        lp.setBounds(minX, minY, maxX, maxY);
        lp.plot(xp, yp, color);     //set up the plot data
        repaint();                  //call paint to plot
    }
}
```

Drawing Plots in Java

The Java GUI is event-driven, so we don't actually write a routine that draws lines on the screen in the direct response to the plot command event. Instead, we provide a panel whose paint event carries out the plotting when that event is called. The previous *repaint* method ensures that it will be called right away.

We create a PlotPanel class based on JPanel and derive two classes from it for the actual line and bar plots (Figure 24.3).