```java
//******************************************************************************
//
// File:    PlotHandler.java
// Package: ---
// Unit:    Class PlotHandler
//
//******************************************************************************

import java.io.File;

/**
 * Class PlotHandler is the delegate for dealing with visualizing the data
 * generated by the "number crunching" program, MonteCarlo. Its purpose is to
 * be instantiated in MonteCarlo with the data to plot, where the write()
 * method should then be called. Running this program and specifying in
 * the command line arguments the plot files previously generated will
 * open a graphical representation of these plots for each file.
 *
 * @author Jimi Ford
 * @version 2-15-2015
 *
 */
public class PlotHandler {

    // private data members
    private final String fileName;
    private final int v;
    private final SimulationResultCollection collection;

    /**
     * Construct a new plot handler that plots average distances for a fixed
     * vertex count v, while varying the edge probability p
     *
     * @param   plotFilePrefix prefix to be used in the name of
     *          the plot file
     * @param   collection collection of results of the finished set of
     *          simulations.
     * @param v number of vertices used in each simulation
     */
    public PlotHandler(String plotFilePrefix,
            SimulationResultCollection collection, int v) {
        fileName = plotFilePrefix + "-V-" + v + ".dwg";
        this.v = v;
        this.collection = collection;
    }

    /**
     * Save the plot information into a file to visualize by running
     * the main method of this class
     *
     * @throws IOException if it can't write to the file specified
     */
    public void write() throws IOException {
        ListXYSeries results = new ListXYSeries();
        double[] values = collection.getAveragesForV(v);
        for(int i = 0, p = collection.pMin; i < values.length; i++,
                p += collection.pInc) {
            results.add(p / ((double) collection.pExp), values[i]);
```

```
64          }
65
66          Plot plot = new Plot()
67              .plotTitle (String.format
68                 ("Random Graphs, <I>V</I> = %1s", Integer.toString(v)))
69              .xAxisTitle ("Edge Probability <I>p</I>")
70              .xAxisTickFormat(new DecimalFormat("0.0"))
71              .yAxisTitle ("Average Distance <I>d</I>")
72              .yAxisTickFormat (new DecimalFormat ("0.0"))
73              .seriesDots (Dots.circle (5))
74              .seriesStroke (null)
75              .xySeries (results);
76          Plot.write(plot, new File(fileName));
77      }
78
79      /**
80       * Open a GUI for each plot in order to visualize the results of a
81       * previously run set of simulations.
82       *
83       * @param args each plot file generated that you wish to visualize
84       */
85      public static void main(String args[]) {
86          if(args.length < 1) {
87              System.err.println("Must specify at least 1 plot file.");
88              usage();
89          }
90
91          for(int i = 0; i < args.length; i++) {
92              try {
93                  Plot plot = Plot.read(args[i]);
94                  plot.getFrame().setVisible(true);
95              } catch (ClassNotFoundException e) {
96                  System.err.println("Could not deserialize " + args[i]);
97              } catch (IOException e) {
98                  System.err.println("Could not open " + args[i]);
99              }
100         }
101
102     }
103
104     /**
105      * Print the usage message for this program and gracefully exit.
106      */
107     private static void usage() {
108         System.err.println("usage: java PlotHandler <plot-file-1> "+
109                 "(<plot-file-2> <plot-file-3>... etc.)");
110         System.exit(1);
111     }
112 }
113
```