```java
1 //****************************************************************************
2 //
3 // File:    PlotHandler.java
4 // Package: ---
5 // Unit:    Class PlotHandler
6 //
7 //****************************************************************************
8
9 import java.io.File;
10 import java.io.IOException;
11 import java.text.DecimalFormat;
12 import edu.rit.numeric.ListXYSeries;
13 import edu.rit.numeric.plot.Dots;
14 import edu.rit.numeric.plot.Plot;
15
16 /**
17 * Class PlotHandler is the delegate for dealing with visualizing the data
18 * generated by the "number crunching" program, MonteCarlo. Its purpose is to
19 * be instantiated in MonteCarlo with the data to plot, where the write()
20 * method should then be called. Running this program and specifying in
21 * the command line arguments the plot files previously generated will
22 * open a graphical representation of these plots for each file.
23 *
24 * @author Jimi Ford
25 * @version 2-15-2015
26 *
27 */
28 public class PlotHandler {
29
30     // private data members
31     private final String fileName;
32     private final int v;
33     private final SimulationResultCollection collection;
34
35     /**
36      * Construct a new plot handler that plots average distances for a fixed
37      * vertex count v, while varying the edge probability p
38      *
39      * @param   plotFilePrefix prefix to be used in the name of
40      *          the plot file
41      * @param   collection collection of results of the finished set of
42      *          simulations.
43      * @param v number of vertices used in each simulation
44      */
45     public PlotHandler(String plotFilePrefix,
46             SimulationResultCollection collection, int v) {
47         fileName = plotFilePrefix + "-V-" + v + ".dwg";
48         this.v = v;
49         this.collection = collection;
50     }
51
52     /**
53      * Save the plot information into a file to visualize by running
54      * the main method of this class
55      *
56      * @throws IOException if it can't write to the file specified
57      */
58     public void write() throws IOException {
```

```java
 59          ListXYSeries results = new ListXYSeries();
 60          double[] values = collection.getAveragesForV(v);
 61          for(int i = 0, p = collection.pMin; i < values.length; i++,
 62                  p += collection.pInc) {
 63              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              } catch (IllegalArgumentException e) {
100                  System.err.println("Error in file " + args[i]);
101              }
102          }
103
104      }
105
106      /**
107       * Print the usage message for this program and gracefully exit.
108       */
109      private static void usage() {
110          System.err.println("usage: java PlotHandler <plot-file-1> "+
111                  "(<plot-file-2> <plot-file-3>... etc.)");
112          System.exit(1);
113      }
114 }
115
```