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

import java.awt.BasicStroke;
import java.awt.Color;
import java.io.IOException;
import java.text.DecimalFormat;

import edu.rit.numeric.ListXYSeries;
import edu.rit.numeric.plot.Dots;
import edu.rit.numeric.plot.Plot;

/**
 * Class PlotHandler is the delegate for dealing with visualizing the data
 * generated by the "number crunching" program, MrPotatoHead.
 * Its purpose is to be instantiated in MrPotatoHead with the data to plot,
 * where the write() method should then be called.
 * <P>
 * 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.
 * </P>
 * @author Jimi Ford
 * @version 5-6-2015
 *
 */
public class PlotHandler {

    // private data members
    private final String rtTotalFile;
    private final String dfTotalFile;
    private final String rtLargeFile;
    private final String dfLargeFile;
    private final String rtSmallFile;
    private final String dfSmallFile;
    private final String routerDropFile;
    private final String reRouteFile;
    private final String primaryActivityFile;
    private final String secondaryActivityFile;
    private final ListXYSeries dfTotal;
    private final ListXYSeries rtTotal;
    private final ListXYSeries dfLarge;
    private final ListXYSeries rtLarge;
    private final ListXYSeries dfSmall;
    private final ListXYSeries rtSmall;
    private final ListXYSeries aDrop;
    private final ListXYSeries bDrop;
    private final ListXYSeries cDrop;
    private final ListXYSeries dDrop;
    private final ListXYSeries aReRoute;
    private final ListXYSeries bReRoute;
    private final ListXYSeries cReRoute;
    private final ListXYSeries dReRoute;
    private final ListXYSeries adActivity;
    private final ListXYSeries bdActivity;
    private final ListXYSeries cdActivity;
    private final ListXYSeries d2Activity;
    private final ListXYSeries abActivity;
    private final ListXYSeries acActivity;
    private final ListXYSeries baActivity;
    private final ListXYSeries bcActivity;
    private final ListXYSeries caActivity;
    private final ListXYSeries cbActivity;
    private final ListXYSeries daActivity;
    private final ListXYSeries dbActivity;
    private final ListXYSeries dcActivity;


    /**
     * Construct a new PlotHandler object
     *
     * @param prefix the prefix to use for saving the files
     * @param dfTotal the xy-series that contains the drop fraction info
     * @param rtTotal the xy-series that contains the response time info
     * @param dfLarge series containing the drop fraction for large packets
     * @param rtLarge series containing the response time for large packets
     * @param dfSmall series containing the drop fraction for small packets
     * @param rtSmall series containing the response time for small packets
     * @param aDrop series containing the drop fraction of router a
     * @param bDrop series containing the drop fraction of router b
     * @param cDrop series containing the drop fraction of router c
     * @param dDrop series containing the drop fraction of router d
     * @param aReRoute series containing the re-route fraction of router a
     * @param bReRoute series containing the re-route fraction of router b
     * @param cReRoute series containing the re-route fraction of router c
     * @param dReRoute series containing the re-route fraction of router d
     * @param adActivity series containing activity fraction of link ad
     * @param bdActivity series containing activity fraction of link bd
     * @param cdActivity series containing activity fraction of link cd
     * @param d2Activity series containing activity fraction of link d2
     * @param abActivity series containing activity fraction of link ab
     * @param acActivity series containing activity fraction of link ac
     * @param baActivity series containing activity fraction of link ba
     * @param bcActivity series containing activity fraction of link bc
     * @param caActivity series containing activity fraction of link ca
     * @param cbActivity series containing activity fraction of link cb
     * @param daActivity series containing activity fraction of link da
     * @param dbActivity series containing activity fraction of link db
     * @param dcActivity series containing activity fraction of link dc
     */
    public PlotHandler(String prefix,
        ListXYSeries dfTotal, ListXYSeries rtTotal,
        ListXYSeries dfLarge, ListXYSeries rtLarge,
        ListXYSeries dfSmall, ListXYSeries rtSmall,
        ListXYSeries aDrop, ListXYSeries bDrop, ListXYSeries cDrop,
        ListXYSeries dDrop, ListXYSeries aReRoute, ListXYSeries bReRoute,
        ListXYSeries cReRoute, ListXYSeries dReRoute,
        ListXYSeries adActivity, ListXYSeries bdActivity,
        ListXYSeries cdActivity, ListXYSeries d2Activity,
        ListXYSeries abActivity, ListXYSeries acActivity,
        ListXYSeries baActivity, ListXYSeries bcActivity,
        ListXYSeries caActivity, ListXYSeries cbActivity,
        ListXYSeries daActivity, ListXYSeries dbActivity,
        ListXYSeries dcActivity) {
        rtTotalFile = prefix + "-traversal-time.dwg";
        dfTotalFile = prefix + "-drop-fraction.dwg";
        rtLargeFile = prefix + "-traversal-time-large.dwg";
        rtSmallFile = prefix + "-traversal-time-small.dwg";
        dfLargeFile = prefix + "-drop-fraction-large.dwg";
        dfSmallFile = prefix + "-drop-fraction-small.dwg";
        routerDropFile = prefix + "-router-drop-fraction.dwg";
        reRouteFile = prefix + "-re-route-fraction.dwg";
        primaryActivityFile = prefix + "-primary-link-activity-fraction.dwg";
        secondaryActivityFile = prefix +
                "-secondary-link-activity-fraction.dwg";
        this.dfTotal = dfTotal;
        this.rtTotal = rtTotal;
        this.dfLarge = dfLarge;
        this.rtLarge = rtLarge;
        this.dfSmall = dfSmall;
        this.rtSmall = rtSmall;
        this.aDrop = aDrop;
        this.bDrop = bDrop;
        this.cDrop = cDrop;
        this.dDrop = dDrop;
        this.aReRoute = aReRoute;
        this.bReRoute = bReRoute;
        this.cReRoute = cReRoute;
        this.dReRoute = dReRoute;
        this.adActivity = adActivity;
        this.bdActivity = bdActivity;
        this.cdActivity = cdActivity;
        this.d2Activity = d2Activity;
        this.abActivity = abActivity;
        this.acActivity = acActivity;
        this.baActivity = baActivity;
        this.bcActivity = bcActivity;
        this.caActivity = caActivity;
        this.cbActivity = cbActivity;
        this.daActivity = daActivity;
        this.dbActivity = dbActivity;
        this.dcActivity = dcActivity;
    }

    /**
     * Save the plot information into files and display the plots.
     *
     * @throws IOException if it can't write to the file specified
     */
    public void write() throws IOException {
        write("Total", "0.0", dfTotal, dfTotalFile, rtTotal, rtTotalFile);
        write("Large Pkt", "0.0", dfLarge, dfLargeFile, rtLarge, rtLargeFile);
        write("Small Pkt", "0.00", dfSmall, dfSmallFile, rtSmall, rtSmallFile);
        writeRouterDrop();
        writeRouterReRoute();
        writePrimaryLinkActivity();
        writeSecondaryLinkActivity();
    }


    /**
     * write the router drop fraction plot
     * @throws IOException if it can't write to the file specified
     */
    private void writeRouterDrop() throws IOException {
        Plot routerDropFraction = new Plot()
        .plotTitle("Router Drop Fraction")
        .xAxisTitle ("Mean arrival rate (pkt/sec)")
        .yAxisTitle ("Drop fraction")
        .yAxisStart (0.0)
        .yAxisEnd (1.0)
        .yAxisTickFormat (new DecimalFormat ("0.0"))
        .seriesDots(null)
        .seriesColor(Color.RED)
        .xySeries(aDrop)
        .seriesColor(Color.ORANGE)
        .seriesDots(Dots.circle(Color.ORANGE, new BasicStroke(),
                Color.ORANGE, 7))
        .xySeries(bDrop)
        .seriesDots(null)
        .seriesColor(Color.GREEN)
        .xySeries(cDrop)
        .seriesColor(Color.BLUE)
        .xySeries(dDrop)
        .labelColor(Color.RED)
        .label("<b>A</b>", 42.5, .85)
        .labelColor(Color.ORANGE)
        .label("<b>B</b>", 42.5, .75)
        .labelColor(Color.GREEN)
        .label("<b>C</b>", 42.5, .65)
        .labelColor(Color.BLUE)
        .label("<b>D</b>", 42.5, .55);
        Plot.write(routerDropFraction, routerDropFile);
    }

    /**
     * write the primary link activity plot
     * @throws IOException if it can't write to the file specified
     */
    private void writePrimaryLinkActivity() throws IOException {
        Plot linkActivity = new Plot()
        .plotTitle("Primary Link Activity")
        .xAxisTitle ("Mean arrival rate (pkt/sec)")
        .yAxisTitle ("Link Activity Fraction")
        .yAxisStart (0.0)
        .yAxisEnd (1.0)
        .yAxisTickFormat (new DecimalFormat ("0.0"))
        .seriesDots(null)
        .seriesColor(Color.RED)
        .xySeries(adActivity)
        .seriesColor(Color.ORANGE)
        .seriesDots(Dots.circle(Color.ORANGE, new BasicStroke(),
                Color.ORANGE, 7))
        .xySeries(bdActivity)
        .seriesDots(null)
        .seriesColor(Color.GREEN)
        .xySeries(cdActivity)
```

```
            .seriesColor(Color.BLUE)
            .xySeries(d2Activity)
            .labelColor(Color.RED)
            .label("<b>A</b>", 42.5, .65)
            .labelColor(Color.ORANGE)
            .label("<b>B</b>", 42.5, .55)
            .labelColor(Color.GREEN)
            .label("<b>C</b>", 42.5, .45)
            .labelColor(Color.BLUE)
            .label("<b>D</b>", 42.5, .35);
            Plot.write(linkActivity, primaryActivityFile);
        }

        /**
         * write the secondary link activity plot
         * @throws IOException if it can't write to the file specified
         */
        private void writeSecondaryLinkActivity() throws IOException {
            Plot linkActivity = new Plot()
            .plotTitle("Secondary Link Activity")
            .xAxisTitle ("Mean arrival rate (pkt/sec)")
            .yAxisTitle ("Link Activity Fraction")
            .yAxisStart (0.0)
            .yAxisEnd (1.0)
            .yAxisTickFormat (new DecimalFormat ("0.0"))
            .seriesDots(null)
            .seriesColor(Color.RED)
            .xySeries(abActivity)
            .xySeries(acActivity)
            .seriesColor(Color.ORANGE)
            .seriesDots(Dots.circle(Color.ORANGE, new BasicStroke(),
                    Color.ORANGE, 7))
            .xySeries(bcActivity)
            .xySeries(baActivity)
            .seriesColor(Color.GREEN)
            .seriesDots(null)
            .xySeries(caActivity)
            .xySeries(cbActivity)
            .seriesColor(Color.BLUE)
            .xySeries(daActivity)
            .xySeries(dbActivity)
            .xySeries(dcActivity)
            .labelColor(Color.RED)
            .label("<b>A</b>", 42.5, .45)
            .labelColor(Color.ORANGE)
            .label("<b>B</b>", 42.5, .35)
            .labelColor(Color.GREEN)
            .label("<b>C</b>", 42.5, .25)
            .labelColor(Color.BLUE)
            .label("<b>D</b>", 42.5, .15);
            Plot.write(linkActivity, secondaryActivityFile);
        }

        /**
         * write the router re-route fraction plot
         *
         * @throws IOException if it can't write to the file specified
         */
```

```
    private void writeRouterReRoute() throws IOException {
        Plot reRouteFraction = new Plot()
        .plotTitle("Router Re-Route Fraction")
        .xAxisTitle ("Mean arrival rate (pkt/sec)")
        .yAxisTitle ("Re-Route fraction")
        .yAxisStart (0.0)
        .yAxisEnd (1.0)
        .yAxisTickFormat (new DecimalFormat ("0.0"))
        .seriesDots(null)
        .seriesColor(Color.RED)
        .xySeries(aReRoute)
        .seriesColor(Color.ORANGE)
        .seriesDots(Dots.circle(Color.ORANGE, new BasicStroke(),
                Color.ORANGE, 7))
        .xySeries(bReRoute)
        .seriesDots(null)
        .seriesColor(Color.GREEN)
        .xySeries(cReRoute)
        .seriesColor(Color.BLUE)
        .xySeries(dReRoute)
        .labelColor(Color.RED)
        .label("<b>A</b>", 42.5, .55)
        .labelColor(Color.ORANGE)
        .label("<b>B</b>", 42.5, .45)
        .labelColor(Color.GREEN)
        .label("<b>C</b>", 42.5, .35)
        .labelColor(Color.BLUE)
        .label("<b>D</b>", 42.5, .25);
        Plot.write(reRouteFraction, reRouteFile);
    }

    /**
     * Save the plot information into files.
     *
     * @param titlePrefix Prefix of the plot's title
     * @param yFormat decimal format of the traversal time y-axis labels
     * @param df drop fraction series
     * @param dfFile drop fraction file name
     * @param rt response time series
     * @param rtFile response time file
     * @throws IOException if it fails to write to any of the specified files
     */
    private void write(String titlePrefix, String yFormat, ListXYSeries df,
            String dfFile, ListXYSeries rt, String rtFile) throws IOException {
        Plot responseTime = new Plot()
        .plotTitle (titlePrefix+" Traversal Time")
        .xAxisTitle ("Mean arrival rate (pkt/sec)")
        .yAxisTitle ("Mean traversal time (sec)")
        .yAxisTickFormat (new DecimalFormat (yFormat))
        .seriesDots (null)
        .xySeries (rt);
        Plot dropFraction = new Plot()
        .plotTitle (titlePrefix+" Drop Fraction")
        .xAxisTitle ("Mean arrival rate (pkt/sec)")
        .yAxisTitle ("Drop fraction")
        .yAxisStart (0.0)
        .yAxisEnd (1.0)
        .yAxisTickFormat (new DecimalFormat ("0.0"))
```

```
            .seriesDots (null)
            .xySeries (df);
        Plot.write(responseTime, rtFile);
        Plot.write(dropFraction, dfFile);
    }

    /**
     * Open a GUI for each plot in order to visualize the results of a
     * previously run set of simulations.
     *
     * @param args each plot file generated that you wish to visualize
     */
    public static void main(String args[]) {
        if(args.length < 1) {
            System.err.println("Must specify at least 1 plot file.");
            usage();
        }

        for(int i = 0; i < args.length; i++) {
            try {
                Plot plot = Plot.read(args[i]);
                plot.getFrame().setVisible(true);
            } catch (ClassNotFoundException e) {
                System.err.println("Could not deserialize " + args[i]);
            } catch (IOException e) {
                System.err.println("Could not open " + args[i]);
            } catch (IllegalArgumentException e) {
                System.err.println("Error in file " + args[i]);
            }
        }
    }

    /**
     * Print the usage message for this program and gracefully exit.
     */
    private static void usage() {
        System.err.println("usage: java PlotHandler <plot-file-1> "+
                "(<plot-file-2> <plot-file-3>... etc.)");
        System.exit(1);
    }
}

```