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

import edu.rit.numeric.ListXYSeries;
import edu.rit.numeric.Series;
import edu.rit.sim.Simulation;
import edu.rit.util.Random;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Class MrPotatoHead is the hot potato simulation main program. It simulates
 * a network in which routers use hot potato routing and uses Prof. Alan
 * Kaminsky's pj2 library to aid in this discrete event simulation.
 *
 * @author  Alan Kaminsky
 * @author Jimi Ford (jhf3617)
 * @version 5-3-2015
 */
public class MrPotatoHead
{
    private static double rlb;
    private static double rub;
    private static double rdelta;
    private static int npkt;
    private static long seed;

    private static Random prng;
    private static Simulation sim;
    private static String prefix;
    private static Generator gen;

    /**
     * Main program to simulate hot-potato routing
     *
     * @param args command line arguments
     */
    public static void main(String[] args)
    {
        // Parse command line arguments.
        if (args.length != 5 && args.length != 6) usage();
        rlb = Double.parseDouble (args[0]);
        rub = Double.parseDouble (args[1]);
        rdelta = Double.parseDouble (args[2]);
        if(rlb <= 0) rlb = rdelta;
        npkt = Integer.parseInt (args[3]);
        seed = Long.parseLong (args[4]);
        prefix = args.length == 6 ? args[5] : "potato";
        // Set up pseudorandom number generator.
        prng = new Random (seed);

        // Set up plot data series.
        ListXYSeries respTimeSeries = new ListXYSeries();
```

```
59          ListXYSeries respTimeLargeSeries = new ListXYSeries();
60          ListXYSeries respTimeSmallSeries = new ListXYSeries();
61          ListXYSeries dropFracSeries = new ListXYSeries();
62          ListXYSeries dropFracLargeSeries = new ListXYSeries();
63          ListXYSeries dropFracSmallSeries = new ListXYSeries();
64
65          ListXYSeries aDrop = new ListXYSeries();
66          ListXYSeries bDrop = new ListXYSeries();
67          ListXYSeries cDrop = new ListXYSeries();
68          ListXYSeries dDrop = new ListXYSeries();
69
70          ListXYSeries aReRoute = new ListXYSeries();
71          ListXYSeries bReRoute = new ListXYSeries();
72          ListXYSeries cReRoute = new ListXYSeries();
73          ListXYSeries dReRoute = new ListXYSeries();
74
75          ListXYSeries adActivity = new ListXYSeries();
76          ListXYSeries bdActivity = new ListXYSeries();
77          ListXYSeries cdActivity = new ListXYSeries();
78          ListXYSeries d2Activity = new ListXYSeries();
79
80          ListXYSeries abActivity = new ListXYSeries();
81          ListXYSeries acActivity = new ListXYSeries();
82          ListXYSeries baActivity = new ListXYSeries();
83          ListXYSeries bcActivity = new ListXYSeries();
84          ListXYSeries caActivity = new ListXYSeries();
85          ListXYSeries cbActivity = new ListXYSeries();
86          ListXYSeries daActivity = new ListXYSeries();
87          ListXYSeries dbActivity = new ListXYSeries();
88          ListXYSeries dcActivity = new ListXYSeries();
89
90          // Sweep mean request rate.
91          System.out.printf ("Mean\tResp\tResp\tResp\tDrop\tDrop\tDrop%n");
92          System.out.printf ("Pkt\tTime\tTime\tTime\tFrac\tFrac\tFrac%n");
93          System.out.printf ("Rate\tTotal\tLarge\tSmall\tTotal\tLarge\tSmall%n");
94          StringBuilder builder = new StringBuilder();
95          builder.append(
96              String.format("Mean\tResp\tResp\tResp\tDrop\tDrop\tDrop%n"));
97          builder.append(
98              String.format("Pkt\tTime\tTime\tTime\tFrac\tFrac\tFrac%n"));
99          builder.append(
100             String.format("Rate\tTotal\tLarge\tSmall\tTotal\tLarge\tSmall%n"));
101         double rate;
102         for (int i = 0; (rate = rlb + i*rdelta) <= rub; ++ i)
103         {
104             // Set up simulation.
105             sim = new Simulation();
106             Host h1, h2;
107             Router a, b, c, d;
108
109             h1 = new Host(sim);
110             h2 = new Host(sim);
111             d = new Router(prng, sim);
112             a = new Router(prng, sim);
113             b = new Router(prng, sim);
114             c = new Router(prng, sim);
115             Link
116                 ab = new Link(sim, a, b),
```

```
117                 ac = new Link(sim, a, c),
118                 ad = new Link(sim, a, d),
119                 ba = new Link(sim, b, a),
120                 bc = new Link(sim, b, c),
121                 bd = new Link(sim, b, d),
122                 ca = new Link(sim, c, a),
123                 cb = new Link(sim, c, b),
124                 cd = new Link(sim, c, d),
125                 da = new Link(sim, d, a),
126                 db = new Link(sim, d, b),
127                 dc = new Link(sim, d, c),
128                 d2 = new Link(sim, d, h2);
129             // preferred link
130             a.setPrimary(ad);
131             b.setPrimary(bd);
132             c.setPrimary(cd);
133             d.setPrimary(d2);
134             // secondary links
135             a.addSecondary(ab);
136             a.addSecondary(ac);
137             b.addSecondary(ba);
138             b.addSecondary(bc);
139             c.addSecondary(ca);
140             c.addSecondary(cb);
141             d.addSecondary(da);
142             d.addSecondary(db);
143             d.addSecondary(dc);
144
145             // Set up request generator and generate first request.
146             gen = new Generator (sim, rate, npkt, prng, h1,
147                     new Link(sim, true, h1, a));
148
149             // Run the simulation.
150             sim.run();
151
152             // Print results.
153             Series.Stats totalStats = gen.responseTimeStats();
154             Series.Stats largeStats = gen.responseTimeLarge().stats();
155             Series.Stats smallStats = gen.responseTimeSmall().stats();
156             System.out.printf ("%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f%n",
157                     rate, totalStats.mean, largeStats.mean, smallStats.mean,
158                     gen.totalDropFraction(), gen.largePacketDropFraction(),
159                     gen.smallPacketDropFraction());
160             builder.append(String.format(
161                     "%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f%n",
162                     rate, totalStats.mean, largeStats.mean, smallStats.mean,
163                     gen.totalDropFraction(), gen.largePacketDropFraction(),
164                     gen.smallPacketDropFraction()));
165             // Record results for plot.
166             respTimeSeries.add (rate, totalStats.mean);
167             respTimeLargeSeries.add(rate, largeStats.mean);
168             respTimeSmallSeries.add(rate, smallStats.mean);
169             dropFracSeries.add (rate, gen.totalDropFraction());
170             dropFracLargeSeries.add(rate, gen.largePacketDropFraction());
171             dropFracSmallSeries.add(rate, gen.smallPacketDropFraction());
172             aDrop.add(rate, a.dropFraction(npkt));
173             bDrop.add(rate, b.dropFraction(npkt));
174             cDrop.add(rate, c.dropFraction(npkt));
```

```
175             dDrop.add(rate, d.dropFraction(npkt));
176             aReRoute.add(rate, a.reRouteFraction());
177             bReRoute.add(rate, b.reRouteFraction());
178             cReRoute.add(rate, c.reRouteFraction());
179             dReRoute.add(rate, d.reRouteFraction());
180             // primary link activity
181             adActivity.add(rate, ad.fractionClosed());
182             bdActivity.add(rate, bd.fractionClosed());
183             cdActivity.add(rate, cd.fractionClosed());
184             d2Activity.add(rate, d2.fractionClosed());
185             // secondary link activity
186             abActivity.add(rate, ab.fractionClosed());
187             acActivity.add(rate, ac.fractionClosed());
188             baActivity.add(rate, ba.fractionClosed());
189             bcActivity.add(rate, bc.fractionClosed());
190             caActivity.add(rate, ca.fractionClosed());
191             cbActivity.add(rate, cb.fractionClosed());
192             daActivity.add(rate, da.fractionClosed());
193             dbActivity.add(rate, db.fractionClosed());
194             dcActivity.add(rate, dc.fractionClosed());
195         }
196 
197         try {
198             new PlotHandler(prefix, dropFracSeries, respTimeSeries,
199                     dropFracLargeSeries, respTimeLargeSeries,
200                     dropFracSmallSeries, respTimeSmallSeries,
201                     aDrop, bDrop, cDrop, dDrop,
202                     aReRoute, bReRoute, cReRoute, dReRoute,
203                     adActivity, bdActivity, cdActivity, d2Activity,
204                     abActivity, acActivity, baActivity, bcActivity, caActivity,
205                     cbActivity, daActivity, dbActivity, dcActivity).write();
206             PrintWriter tableWriter = new PrintWriter(prefix + "-table.tsv");
207             tableWriter.print(builder.toString());
208             tableWriter.close();
209         } catch (IOException e) {
210             e.printStackTrace();
211         }
212     }
213 
214     /**
215      * Print a usage message and exit.
216      */
217     private static void usage()
218     {
219         System.err.println ("Usage: java MrPotatoHead <rlb> <rub> <rdelta> "
220                 + "<npkt> <seed> [<file-prefix> (optional)]");
221         System.err.println ("<rlb> = Mean packet rate lower bound");
222         System.err.println ("<rub> = Mean packet rate upper bound");
223         System.err.println ("<rdelta> = Mean packet rate delta");
224         System.err.println ("<npkt> = Number of packets");
225         System.err.println ("<seed> = Random seed");
226         System.err.println ("<file-prefix> = optional file prefix, "
227                 + "default = \"potato\"");
228         System.exit (1);
229     }
230 }
```