

```

1 //Robert Swartz
2 //Towers of Chicago
3 //Copyright 2024
4 //Fusion Power Applications
5
6
7 import java.awt.*;
8 import java.awt.datatransfer.*;
9 import java.awt.event.*;
10 import java.util.*;
11
12
13 public class Towers
14 {
15     public static void main(String[] args)
16     {
17         AppletImage cubs = new AppletImage();
18         int i, w = 1300, h = 1000, h2 = 90;
19         String[] instructions = {"# of Discs ", "# of Pegs ", "Initial Peg ",
20             "Final Peg ", "Speed ", "Random k Seed ", "Random Disc Seed ",
21             "Width ", "Height ", "Disc Colors ", "Peg Colors "},
22         defaults = {"40", "8", "1", "max", "250", ""+w, ""+h, "20", "6"};
23         cubs.powArray[0] = 1;
24         for (i = 1; i <= 62; i++)
25             cubs.powArray[i] = cubs.powArray[i-1]*2+1;
26         cubs.pF.setLocation(125, 100);
27         cubs.pF.setVisible(true);
28         cubs.pF.setSize(w, h);
29         cubs.pF.setMinimumSize(new Dimension(w, 3*h/5));
30         cubs.mwa = new MyWindowAdapter(cubs);
31         cubs.pF.addWindowListener(cubs.mwa);
32         cubs.mca = new MyComponentAdapter(cubs);
33         cubs.pF.addComponentListener(cubs.mca);
34         cubs.p0 = new Panel(new BorderLayout());
35         cubs.p0.setSize(w, h);
36         cubs.pF.add(cubs.p0);
37         cubs.start = new Button("    Start    ");
38         cubs.stop = new Button("    Stop    ");
39         cubs.move = new Button("    Move    ");
40         cubs.help = new Button("    Help    ");
41         cubs.copy = new Button(" Copy ");
42         cubs.resize = new Button(" Resize ");
43         cubs.colors = new Button(" Colors ");
44         cubs.shiftC[0] = new Button(">");
45         cubs.shiftC[1] = new Button(">");
46         cubs.shiftC[2] = new Button("<");
47         cubs.shiftC[3] = new Button("<");
48         cubs.mka1 = new MyKeyAdapter1(cubs);
49         cubs.mma1 = new MyMouseAdapter1(cubs);
50         cubs.mka2 = new MyKeyAdapter2(cubs);
51         cubs.mma2 = new MyMouseAdapter2(cubs);
52         cubs.mka3 = new MyKeyAdapter3();
53         cubs.mma3 = new MyMouseAdapter3();
54         cubs.mka4 = new MyKeyAdapter4(cubs);
55         cubs.mma4 = new MyMouseAdapter4(cubs);
56         cubs.mka5 = new MyKeyAdapter5(cubs);
57         cubs.mma5 = new MyMouseAdapter5(cubs);
58         cubs.mka6 = new MyKeyAdapter6(cubs);
59         cubs.mma6 = new MyMouseAdapter6(cubs);
60         cubs.mka7 = new MyKeyAdapter7(cubs);

```

```
61     cubs.mma7 = new MyMouseAdapter7(cubs);
62     cubs.mka8 = new MyKeyAdapter8(cubs);
63     cubs.mma8 = new MyMouseAdapter8(cubs);
64     cubs.mka9 = new MyKeyAdapter9(cubs);
65     cubs.mma9 = new MyMouseAdapter9(cubs);
66     cubs.mka10 = new MyKeyAdapter10(cubs);
67     cubs.mma10 = new MyMouseAdapter10(cubs);
68     cubs.mka11 = new MyKeyAdapter11(cubs);
69     cubs.mma11 = new MyMouseAdapter11(cubs);
70     cubs.mil1a = new MyItemListener1(cubs);
71     cubs.mil1b = new MyItemListener1(cubs);
72     cubs.mil2 = new MyItemListener2(cubs);
73     cubs.mal = new MyAdjustmentListener(cubs);
74     cubs.start.setBackground(new Color(0, 220, 0));
75     cubs.start.setFont(new Font("SansSerif", Font.BOLD, 13));
76     cubs.start.addKeyListener(cubs.mka1);
77     cubs.start.addMouseListener(cubs.mma1);
78     cubs.stop.setBackground(new Color(232, 0, 0));
79     cubs.stop.setFont(new Font("SansSerif", Font.BOLD, 13));
80     cubs.stop.addKeyListener(cubs.mka2);
81     cubs.stop.addMouseListener(cubs.mma2);
82     cubs.move.setBackground(new Color(64, 128, 255));
83     cubs.move.setFont(new Font("SansSerif", Font.BOLD, 13));
84     cubs.move.addKeyListener(cubs.mka3);
85     cubs.move.addMouseListener(cubs.mma3);
86     cubs.help.setBackground(new Color(240, 128, 64));
87     cubs.help.setFont(new Font("SansSerif", Font.BOLD, 13));
88     cubs.help.addKeyListener(cubs.mka4);
89     cubs.help.addMouseListener(cubs.mma4);
90     cubs.copy.setBackground(new Color(170, 170, 234));
91     cubs.copy.setFont(new Font("SansSerif", Font.BOLD, 13));
92     cubs.copy.addKeyListener(cubs.mka5);
93     cubs.copy.addMouseListener(cubs.mma5);
94     cubs.resize.setBackground(new Color(160, 112, 160));
95     cubs.resize.setFont(new Font("SansSerif", Font.BOLD, 13));
96     cubs.resize.addKeyListener(cubs.mka6);
97     cubs.resize.addMouseListener(cubs.mma6);
98     cubs.colors.setBackground(new Color(180, 125, 110));
99     cubs.colors.setFont(new Font("SansSerif", Font.BOLD, 13));
100    cubs.colors.addKeyListener(cubs.mka7);
101    cubs.colors.addMouseListener(cubs.mma7);
102    cubs.shiftC[0].setBackground(new Color(55, 166, 174));
103    cubs.shiftC[0].setFont(new Font("SansSerif", Font.BOLD, 13));
104    cubs.shiftC[0].addKeyListener(cubs.mka8);
105    cubs.shiftC[0].addMouseListener(cubs.mma8);
106    cubs.shiftC[1].setBackground(new Color(55, 166, 174));
107    cubs.shiftC[1].setFont(new Font("SansSerif", Font.BOLD, 13));
108    cubs.shiftC[1].addKeyListener(cubs.mka9);
109    cubs.shiftC[1].addMouseListener(cubs.mma9);
110    cubs.shiftC[2].setBackground(new Color(55, 166, 174));
111    cubs.shiftC[2].setFont(new Font("SansSerif", Font.BOLD, 13));
112    cubs.shiftC[2].addKeyListener(cubs.mka10);
113    cubs.shiftC[2].addMouseListener(cubs.mma10);
114    cubs.shiftC[3].setBackground(new Color(55, 166, 174));
115    cubs.shiftC[3].setFont(new Font("SansSerif", Font.BOLD, 13));
116    cubs.shiftC[3].addKeyListener(cubs.mka11);
117    cubs.shiftC[3].addMouseListener(cubs.mma11);
118    cubs.scramble1 = new Checkbox("Scramble k Values");
119    cubs.scramble1.setFont(new Font("SansSerif", Font.PLAIN, 13));
120    cubs.scramble2 = new Checkbox("Scramble Discs");
```

```

121 cubs.scramble2.setFont(new Font("SansSerif", Font.PLAIN, 13));
122 cubs.forward = new Checkbox("Forward", cubs.myCBG, true);
123 cubs.forward.setFont(new Font("SansSerif", Font.PLAIN, 13));
124 cubs.reverse = new Checkbox("Reverse", cubs.myCBG, false);
125 cubs.reverse.setFont(new Font("SansSerif", Font.PLAIN, 13));
126 cubs.pause = new Checkbox("Pause");
127 cubs.pause.setFont(new Font("SansSerif", Font.PLAIN, 13));
128 cubs.discNums = new Checkbox("Numbers");
129 cubs.discNums.setFont(new Font("SansSerif", Font.PLAIN, 13));
130 cubs.discNums.addItemListener(cubs.milla);
131 cubs.round = new Checkbox("Round");
132 cubs.round.setFont(new Font("SansSerif", Font.PLAIN, 13));
133 cubs.round.addItemListener(cubs.mil1b);
134 cubs.log = new Checkbox("Log", true);
135 cubs.log.setFont(new Font("SansSerif", Font.PLAIN, 13));
136 cubs.log.addItemListener(cubs.mil2);
137 cubs.speed.setSize(100, 20);
138 cubs.speed.setBackground(new Color(64, 224, 224));
139 cubs.speed.addAdjustmentListener(cubs.mal);
140 cubs.dial.setSize(25, 15);
141 cubs.logBox.setEditable(false);
142 cubs.logBox.setSize((int)(.4*w), h2);
143 cubs.logBox.setBackground(new Color(110, 255, 110));
144 cubs.logBox.setFont(new Font("SansSerif", Font.PLAIN, 13));
145 cubs.helpBox = new TextArea(cubs.helpMsg, 100, 100,
146     TextArea.SCROLLBARS_VERTICAL_ONLY);
147 cubs.helpBox.setEditable(false);
148 cubs.helpBox.setBackground(new Color(160, 224, 255));
149 cubs.helpBox.setFont(new Font("SansSerif", Font.PLAIN, 17));
150 cubs.p1 = new MyPanel(new BorderLayout());
151 cubs.p1.setSize(w, h-h2);
152 cubs.p2 = new Panel(new GridLayout(4, 1));
153 cubs.p2.setSize((int)(.6*w), h2);
154 cubs.p3 = new Panel(new FlowLayout(FlowLayout.CENTER, 6, 12));
155 cubs.p3.setSize(w, h2);
156 cubs.p3.setBackground(Color.lightGray);
157 GridBagLayout gbl = new GridBagLayout();
158 GridBagConstraints gbc = new GridBagConstraints();
159 cubs.p2a = new Panel(gbl);
160 cubs.p2a.setSize((int)(.6*w), h2/3);
161 cubs.p2b = new Panel(new FlowLayout(FlowLayout.CENTER, 1, 7));
162 cubs.p2b.setSize((int)(.6*w), h2/3);
163 cubs.p2c = new Panel(new FlowLayout(FlowLayout.CENTER, 10, 7));
164 cubs.p2c.setSize((int)(.6*w), h2/3);
165 cubs.p2d = new Panel(new FlowLayout(FlowLayout.CENTER, 1, 7));
166 cubs.p2d.setSize((int)(.6*w), h2/3);
167 cubs.p2.add(cubs.p2a);
168 cubs.p2.add(cubs.p2b);
169 cubs.p2.add(cubs.p2c);
170 cubs.p2.add(cubs.p2d);
171 Canvas[] cs = new Canvas[17];
172 for (i = 0; i < 17; i++)
173 {
174     cs[i] = new Canvas();
175     cs[i].setSize(1, 5);
176 }
177 gbc.gridx = 0;
178 gbc.gridy = 0;
179 cs[0].setSize(13, 5);
180 gbl.setConstraints(cs[0], gbc);

```

```

181     cubs.p2a.add(cs[0]);
182     gbc.weightx = 2;
183     for (i = 0; i < 4; i++)
184     {
185         gbc.gridx = 2*i+1;
186         gbc.anchor = GridBagConstraints.EAST;
187         cubs.lbls[i] = new Label(instructions[i], Label.RIGHT);
188         cubs.lbls[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
189         gbl.setConstraints(cubs.lbls[i], gbc);
190         cubs.p2a.add(cubs.lbls[i]);
191         gbc.gridx = 2*i+2;
192         gbc.anchor = GridBagConstraints.WEST;
193         cubs.entries[i] = new TextField(defaults[i], 3);
194         cubs.entries[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
195         gbl.setConstraints(cubs.entries[i], gbc);
196         cubs.p2a.add(cubs.entries[i]);
197     }
198     cubs.lbls[4] = new Label(instructions[4], Label.RIGHT);
199     cubs.lbls[4].setFont(new Font("SansSerif", Font.PLAIN, 13));
200     gbc.gridx = 9;
201     gbc.anchor = GridBagConstraints.EAST;
202     gbl.setConstraints(cubs.lbls[4], gbc);
203     cubs.p2a.add(cubs.lbls[4]);
204     gbc.gridx = 10;
205     gbc.weightx = 60;
206     gbc.anchor = GridBagConstraints.CENTER;
207     gbc.fill = GridBagConstraints.HORIZONTAL;
208     gbl.setConstraints(cubs.speed, gbc);
209     cubs.p2a.add(cubs.speed);
210     gbc.gridx = 11;
211     gbc.weightx = 3;
212     gbl.setConstraints(cubs.dial, gbc);
213     cubs.p2a.add(cubs.dial);
214     gbc.gridx = 12;
215     cs[1].setSize(7, 5);
216     gbl.setConstraints(cs[1], gbc);
217     cubs.p2a.add(cs[1]);
218     cs[2].setSize(30, 5);
219     cubs.p2b.add(cs[2]);
220     cubs.p2b.add(cubs.scramble1);
221     cubs.lbls[5] = new Label(instructions[5], Label.RIGHT);
222     cubs.lbls[5].setFont(new Font("SansSerif", Font.PLAIN, 13));
223     cubs.entries[4] = new TextField(defaults[4], 3);
224     cubs.entries[4].setFont(new Font("SansSerif", Font.PLAIN, 13));
225     cubs.p2b.add(cubs.lbls[5]);
226     cubs.p2b.add(cubs.entries[4]);
227     cubs.p2b.add(new Label("  "));
228     cubs.p2b.add(cubs.scramble2);
229     cubs.lbls[6] = new Label(instructions[6], Label.RIGHT);
230     cubs.lbls[6].setFont(new Font("SansSerif", Font.PLAIN, 13));
231     cubs.entries[5] = new TextField(3);
232     cubs.entries[5].setFont(new Font("SansSerif", Font.PLAIN, 13));
233     cubs.p2b.add(cubs.lbls[6]);
234     cubs.p2b.add(cubs.entries[5]);
235     cs[3].setSize(25, 5);
236     cubs.p2b.add(cs[3]);
237     cubs.p2b.add(cubs.forward);
238     cs[4].setSize(15, 5);
239     cubs.p2b.add(cs[4]);
240     cubs.p2b.add(cubs.reverse);

```

```

241     cs[5].setSize(24, 5);
242     cubs.p2b.add(cs[5]);
243     cs[6].setSize(3, 5);
244     cubs.p2c.add(cs[6]);
245     for (i = 6; i <= 8; i++)
246         cs[i].setSize(4, 5);
247     cubs.p2c.add(cubs.start);
248     cubs.p2c.add(cs[7]);
249     cubs.p2c.add(cubs.stop);
250     cubs.p2c.add(cs[8]);
251     cubs.p2c.add(cubs.move);
252     cubs.p2c.add(cs[9]);
253     cubs.p2c.add(cubs.help);
254     cs[10].setSize(7, 5);
255     cubs.p2c.add(cs[10]);
256     cubs.p2c.add(cubs.pause);
257     cubs.p2c.add(cs[11]);
258     cubs.p2c.add(cubs.discNums);
259     cubs.p2c.add(cs[12]);
260     cubs.p2c.add(cubs.round);
261     cubs.p2c.add(cs[13]);
262     cubs.p2c.add(cubs.log);
263     cubs.p2c.add(cs[14]);
264     cubs.p2c.add(cubs.copy);
265     cs[15].setSize(5, 5);
266     cubs.p2c.add(cs[15]);
267     cubs.p2d.add(cubs.resize);
268     for (i = 6; i <= 7; i++)
269     {
270         cubs.lbls[i+1] = new Label(instructions[i+1], Label.RIGHT);
271         cubs.lbls[i+1].setFont(new Font("SansSerif", Font.PLAIN, 13));
272         cubs.entries[i] = new TextField(defaults[i-1], 3);
273         cubs.entries[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
274         cubs.p2d.add(cubs.lbls[i+1]);
275         cubs.p2d.add(cubs.entries[i]);
276     }
277     cs[16].setSize(64, 5);
278     cubs.p2d.add(cs[16]);
279     cubs.p2d.add(cubs.colors);
280     for (i = 8; i <= 9; i++)
281     {
282         cubs.lbls[i+1] = new Label(instructions[i+1], Label.RIGHT);
283         cubs.lbls[i+1].setFont(new Font("SansSerif", Font.PLAIN, 13));
284         cubs.entries[i] = new TextField(defaults[i-1], 3);
285         cubs.entries[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
286         cubs.p2d.add(cubs.lbls[i+1]);
287         cubs.p2d.add(cubs.entries[i]);
288         cubs.p2d.add(cubs.shiftC[i-6]);
289         cubs.p2d.add(cubs.shiftC[i-8]);
290     }
291     cubs.p1.add(cubs.helpBox);
292     cubs.p3.add(cubs.p2);
293     cubs.p3.add(cubs.logBox);
294     cubs.p3.add(new Label("  "));
295     cubs.p0.add(cubs.p1, "Center");
296     cubs.p0.add(cubs.p3, "South");
297     cubs.pF.paintAll(cubs.pF.getGraphics());
298     cubs.entries[0].requestFocus();
299     System.gc();
300 }

```

```

301 }
302
303
304 class AppletImage
305 {
306     public Spire cubs2;
307     public Frame pF = new Frame("Towers of Chicago (1300 x 1000)");
308     public Graphics gx;
309     public Color[] c1, c2;
310     public Panel p0, p2, p2a, p2b, p2c, p2d, p3;
311     public MyPanel p1;
312     public Label[] lbls = new Label[11];
313     public TextField[] entries = new TextField[10];
314     public TextArea logBox = new TextArea(8, 55), helpBox;
315     public Button start, stop, move, help, copy, resize, colors;
316     public Button[] shiftC = new Button[4];
317     public Scrollbar speed = new Scrollbar(Scrollbar.HORIZONTAL, 50, 10, 1, 110);
318     public MyCanvas3 dial = new MyCanvas3();
319     public MyCanvas4 mc = new MyCanvas4();
320     public Checkbox scramble1, scramble2, forward, reverse, pause, discNums,
321         round, log;
322     public CheckboxGroup myCBG = new CheckboxGroup();
323     public MyKeyAdapter1 mka1;
324     public MyMouseAdapter1 mma1;
325     public MyKeyAdapter2 mka2;
326     public MyMouseAdapter2 mma2;
327     public MyKeyAdapter3 mka3;
328     public MyMouseAdapter3 mma3;
329     public MyKeyAdapter4 mka4;
330     public MyMouseAdapter4 mma4;
331     public MyKeyAdapter5 mka5;
332     public MyMouseAdapter5 mma5;
333     public MyKeyAdapter6 mka6;
334     public MyMouseAdapter6 mma6;
335     public MyKeyAdapter7 mka7;
336     public MyMouseAdapter7 mma7;
337     public MyKeyAdapter8 mka8;
338     public MyMouseAdapter8 mma8;
339     public MyKeyAdapter9 mka9;
340     public MyMouseAdapter9 mma9;
341     public MyKeyAdapter10 mka10;
342     public MyMouseAdapter10 mma10;
343     public MyKeyAdapter11 mka11;
344     public MyMouseAdapter11 mma11;
345     public MyItemListener1 mil1a, mil1b;
346     public MyItemListener2 mil2;
347     public MyAdjustmentListener mal;
348     public MyComponentAdapter mca;
349     public MyWindowAdapter mwa;
350     public GridBagLayout gbl0 = new GridBagLayout();
351     public GridBagConstraints gbc0 = new GridBagConstraints();
352     public int moveCountA, colorX, colorY, colorPos1, colorPos2,
353         oldColorX, oldColorY, oldDiscs, oldPegs, dn = 1;
354     public boolean running, scrambleK, scrambled;
355     public int[] valuesIndex;
356     public long[] powArray = new long[63];
357     public byte[][] moves;
358     public int[][] values, optimalsIndex, optOrderIndex;
359     public long[][] moveArray = new long[1000][100];
360     public boolean[][] doneYet = new boolean[1000][100];

```

```

361 public int[][][] optimals = new int[1000][100][1], optOrder;
362 public String moveCountB = "", logger = "", preLog = "", text = "", helpMsg =
363     "Robert Swartz\nwww.mathapplets.net\nTowers of Chicago\nCopyright 2024"+
364     "\n\nThis program solves the Towers of Chicago puzzle. In this problem,"+
365     " a set of discs are to be moved from one peg to another while observing"+
366     " the following 4 rules:\n(1) Only one disc can be moved at a time, "+
367     "(2) A larger disc can't be placed on top of a smaller one, (3) Auxiliary"
368     "+ pegs may be used for the temporary placement of the discs, and (4) The"
369     "+ task should be completed in a minimum number of moves.\n\nThis program"
370     "+ can display up to 50 discs by 10 pegs with a default window size of "
371     "+1300 x 1000. It can also print the moves for up to 1000 discs by "+
372     "+100 pegs. The applet can handle up to 20 rotating colors; the initial"+
373     " and final pegs are displayed in green and red, respectively. "+
374     "The discs are displayed either round or sharp. When the applet is "+
375     "paused, the move button can be used to step through the moves.\n\n"+
376     "The traditional version of the Towers of Chicago makes use of only 3 "+
377     "pegs: the initial peg, the final peg, and 1 auxiliary peg. However, "
378     "+the multipeg version makes use of any number of pegs: the initial peg,"
379     "+ the final peg, and 2 or more auxiliary pegs. As the number of pegs is"
380     "+ increased, while keeping the number of discs constant, the number of "+
381     "moves required usually decreases.\n\nThe optimal solution for the 3 peg "
382     "+problem is well known: recursively move n-1 discs from the initial peg"
383     "+ to the auxiliary peg, then move the remaining disc to the final peg, "+
384     "then move the first n-1 discs from the auxiliary peg to the final peg. "
385     "+This algorithm results in 2n-1 moves.\n\nThe presumed optimal "+
386     "Frame-Stewart algorithm that this program uses for n discs and p pegs is"
387     "+ as follows: recursively move k discs from the initial peg to an "+
388     "available auxiliary peg using the p peg subalgorithm, then move the "+
389     "remaining n-k discs from the initial peg to the final peg using the p-1"+
390     " peg subalgorithm, then move the first k discs from the auxiliary peg to"
391     "+ the final peg using the p peg subalgorithm. The program uses dynamic "
392     "+programming to find the k values. The k values are calculated to make "
393     "+the number of moves minimal.\n\nThis program can find different optimal"
394     "+ solutions for each combination of discs and pegs (where pegs are "+
395     "greater than 3). This is accomplished through 2 kinds of randomization,"
396     "+ Scramble k Values and Scramble Discs. Each of these uses a random "
397     "+number seed (an integer from 000 to 999). Scramble k Values uses a "
398     "+ different k value for each recursive instance of a particular "+
399     "subalgorithm; if this isn't selected, the Random k Seed sets constant k "
400     "+values for each subalgorithm. Scramble Discs simply shuffles the discs"
401     "+ around the auxiliary pegs.";
402
403 public Color[] colorScheme1(int discs, int num, int shift)
404 {
405     Color[] cx = new Color[discs];
406     for (int i = 0; i < discs; i++)
407         cx[i] = colorScheme3(i, num, shift, colorY);
408     return cx;
409 }
410
411 public Color[] colorScheme2(int pegs, int init, int fnl, int num, int shift)
412 {
413     Color[] cx = new Color[pegs];
414     int i, j;
415     for (i = 0; i < pegs; i++)
416     {
417         if (i == init-1)
418             cx[i] = new Color(0, 220, 0);
419         else if (i == fnl-1)
420             cx[i] = new Color(232, 0, 0);

```

```

421     else
422     {
423         if (i < Math.min(init-1, fnl-1))
424             j = i;
425         else if (i < Math.max(init-1, fnl-1))
426             j = i-1;
427         else
428             j = i-2;
429         cx[i] = colorScheme3(j, num, shift, colorX);
430     }
431 }
432 return cx;
433 }
434
435 public Color colorScheme3(int pos, int num, int shift, int colorcount)
436 {
437     Color c;
438     switch ((pos%colorcount+shift)%num)
439     {
440         case 0 ->
441             c = new Color(48, 156, 172);
442         case 1 ->
443             c = new Color(176, 96, 80);
444         case 2 ->
445             c = new Color(150, 64, 192);
446         case 3 ->
447             c = new Color(96, 120, 144);
448         case 4 ->
449             c = new Color(216, 96, 96);
450         case 5 ->
451             c = new Color(128, 96, 48);
452         case 6 ->
453             c = new Color(64, 96, 176);
454         case 7 ->
455             c = new Color(160, 0, 128);
456         case 8 ->
457             c = new Color(96, 176, 208);
458         case 9 ->
459             c = new Color(108, 96, 176);
460         case 10 ->
461             c = new Color(128, 0, 0);
462         case 11 ->
463             c = new Color(64, 128, 128);
464         case 12 ->
465             c = new Color(172, 136, 64);
466         case 13 ->
467             c = new Color(0, 128, 192);
468         case 14 ->
469             c = new Color(208, 0, 208);
470         case 15 ->
471             c = new Color(128, 0, 208);
472         case 16 ->
473             c = new Color(128, 128, 0);
474         case 17 ->
475             c = new Color(208, 128, 128);
476         case 18 ->
477             c = new Color(128, 0, 128);
478         default ->
479             c = new Color(128, 64, 64);
480     }

```



```

481     return c;
482 }
483 }
484
485
486 class Spire extends Thread
487 {
488     public AppletImage cubs;
489     public int discs, pegs, init, fnl, seed1, seed2, time, count,
490         pointer = -1, direction = 1;
491     public boolean doSleep, error, sysError, tooMany;
492     public MyCanvas gameBoard;
493     public MyCanvas2 moveCanvas;
494     public Random rg;
495     public Vector nums = new Vector(100);
496     public MyExceptionA myE_A = new MyExceptionA();
497     public MyExceptionB myE_B = new MyExceptionB();
498
499     public Spire(AppletImage a, int n, int p, int i, int f, int s1, int s2)
500     {
501         cubs = a;
502         discs = n;
503         pegs = p;
504         init = i;
505         fnl = f;
506         seed1 = s1;
507         seed2 = s2;
508     }
509
510     public void run()
511     {
512         cubs.running = true;
513         cubs.p1.removeAll();
514         cubs.logBox.setText("\n\n\n    BUSY...");
515         cubs.text = " ";
516         cubs.mka3.cubs2 = this;
517         cubs.mma3.cubs2 = this;
518         timeage();
519         System.gc();
520         int i, j, a, r;
521         int[] aux = new int[pegs-2], aux2 = new int[pegs-2], order = new int[pegs
522             -2];
523         if ((discs != cubs.oldDiscs) || (cubs.colorY != cubs.oldColorY))
524         {
525             cubs.colorPos1 = cubs.colorY*1000000+7;
526             cubs.c1 = cubs.colorScheme1(discs, cubs.colorY, cubs.colorPos1);
527         }
528         if ((pegs != cubs.oldPegs) || (cubs.colorX != cubs.oldColorX))
529         {
530             cubs.colorPos2 = cubs.colorX*1000000;
531             cubs.c2 = cubs.colorScheme2(pegs, init, fnl, cubs.colorX,
532                 cubs.colorPos2);
533         }
534         cubs.oldDiscs = discs;
535         cubs.oldPegs = pegs;
536         cubs.oldColorY = cubs.colorY;
537         cubs.oldColorX = cubs.colorX;
538         if (pegs >= 4)
539         {
540             try

```

```

541     {
542         cubs.optOrder = new int[discs][pegs][1];
543         cubs.optOrderIndex = new int[discs][pegs];
544         cubs.optimalsIndex = new int[discs][pegs];
545         System.gc();
546         optimize();
547         nums.clear();
548         rg = new Random(seed1);
549         for (i = 3; i < discs; i++)
550             for (j = 3; j < pegs; j++)
551                 if (i >= j)
552                     {
553                         if (!cubs.running)
554                             throw myE_B;
555                         if (cubs.scrambleK)
556                             {
557                                 cubs.optOrder[i][j] = new int[cubs.optimals[i][j].
558                                     length];
559                                 for (a = 0; a < cubs.optimals[i][j].length; a++)
560                                     nums.add(a);
561                                 for (a = 0; a < cubs.optimals[i][j].length; a++)
562                                     {
563                                         r = myRandom(nums.size());
564                                         cubs.optOrder[i][j][a] = (int)nums.get(r);
565                                         nums.remove(r);
566                                     }
567                             }
568                         else
569                             cubs.optimalsIndex[i][j] = myRandom(cubs.optimals[i][j].
570                                 length);
571                     }
572     }
573     catch (MyExceptionB e1)
574     {
575         cubs.logBox.setText("");
576     }
577 }
578 if (cubs.running)
579 {
580     if (pegs == 3)
581     {
582         if (discs <= 30)
583         {
584             cubs.moveCountA = (int)cubs.powArray[discs-1];
585             cubs.moveCountB = ""+cubs.powArray[discs-1];
586         }
587         else if (discs <= 63)
588         {
589             cubs.moveCountA = Integer.MAX_VALUE-10;
590             cubs.moveCountB = ""+cubs.powArray[discs-1];
591         }
592         else
593         {
594             cubs.moveCountA = Integer.MAX_VALUE-10;
595             cubs.moveCountB = ""+Math.pow(2, discs);
596         }
597     }
598     else
599     {
600         if (cubs.moveArray[discs-1][pegs-1] <= Integer.MAX_VALUE-10)

```

```

601     {
602         cubs.moveCountA = (int)cubs.moveArray[discs-1][pegs-1];
603         cubs.moveCountB = ""+cubs.moveArray[discs-1][pegs-1];
604     }
605     else
606     {
607         cubs.moveCountA = Integer.MAX_VALUE-10;
608         cubs.moveCountB = ""+cubs.moveArray[discs-1][pegs-1];
609     }
610 }
611 cubs.logger = "    "+discs+" Discs x "+pegs+" Pegs\n    "
612             +cubs.moveCountB+" Total Moves\n";
613 cubs.preLog = cubs.logger;
614 for (i = 0; i < 10; i++)
615 {
616     sysError = false;
617     try
618     {
619         cubs.moves = new byte[2][cubs.moveCountA];
620     }
621     catch (java.lang.OutOfMemoryError e2)
622     {
623         sysError = true;
624     }
625     if (!sysError)
626         break;
627 }
628 if (!cubs.running)
629     cubs.logBox.setText("");
630 else
631 {
632     j = 0;
633     for (i = 1; i <= pegs; i++)
634         if (i != init && i != fnl)
635             {
636                 aux[j] = i;
637                 j++;
638             }
639     if (cubs.scrambled)
640     {
641         rg = new Random(seed2);
642         for (i = 0; i < aux.length; i++)
643             nums.add(i);
644         for (i = 0; i < aux.length; i++)
645             {
646                 r = myRandom(nums.size());
647                 order[i] = (int)nums.get(r);
648                 nums.remove(r);
649             }
650     }
651     else
652         for (i = 0; i < aux.length; i++)
653             order[i] = i;
654     for (i = 0; i < aux.length; i++)
655         aux2[i] = aux[order[i]];
656     cubs.valuesIndex = new int[pegs];
657     cubs.values = new int[discs][pegs];
658     for (i = 0; i < pegs; i++)
659         cubs.valuesIndex[i] = discs;
660     cubs.valuesIndex[init-1] = 0;

```

```

661     for (i = 0; i < discs; i++)
662         cubs.values[i][init-1] = i+1;
663     drawBoard();
664     if (!tooMany)
665     {
666         cubs.gx = gameBoard.getGraphics();
667         gameBoard.paint0(cubs.gx);
668     }
669     else
670     {
671         cubs.gx = cubs.mc.getGraphics();
672         cubs.mc.paint(cubs.gx);
673     }
674     if (cubs.running)
675     {
676         cubs.pF.paintAll(cubs.pF.getGraphics());
677         cubs.logBox.setText(cubs.logger+"\n    BUSY...");
678         try
679         {
680             chicago(discs, init, fn1, aux2);
681         }
682         catch (MyExceptionA e4){}
683         catch (MyExceptionB e5)
684         {
685             cubs.logBox.setText(cubs.logger);
686             error = true;
687         }
688         if (!error)
689         {
690             try
691             {
692                 sleep(650);
693             }
694             catch (InterruptedException e6){}
695             cubs.logBox.setText(cubs.logger);
696             while (pointer < cubs.moveCountA-1)
697             {
698                 try
699                 {
700                     myMove2();
701                 }
702                 catch (MyExceptionB e7)
703                 {
704                     break;
705                 }
706                 catch (ArrayIndexOutOfBoundsException e8)
707                 {
708                     break;
709                 }
710             }
711         }
712     }
713 }
714 }
715 if (cubs.running && pointer >= 0)
716 {
717     try
718     {
719         sleep(Math.max(time, 650));
720     }

```

```

721         catch (InterruptedException e9){}
722         moveCanvas.setText3(discs, pegs, cubs.moveCountA);
723     }
724     if (cubs.running && (pointer == cubs.moveCountA-1 || pointer == -1))
725     {
726         cubs.logger+="\n    DONE";
727         cubs.preLog+="\n    DONE";
728         if (cubs.log.getState())
729             cubs.logBox.append("\n    DONE");
730         else
731             cubs.logBox.setText(cubs.preLog);
732     }
733     cubs.moves = null;
734     cubs.optOrder = null;
735     System.gc();
736     cubs.running = false;
737 }
738
739 public void drawBoard()
740 {
741     int i, w2, h2, w3, w = cubs.pF.getSize().width-30, h =
742         cubs.pF.getSize().height-250, h_mod = 1;
743     if (pegs == 3)
744         w2 = (int)(.6*w);
745     else if (pegs <= 5)
746         w2 = (int)(.7*w);
747     else if (pegs <= 7)
748         w2 = (int)(.78*w);
749     else if (pegs <= 9)
750         w2 = (int)(.85*w);
751     else if (pegs <= 12)
752         w2 = (int)(.91*w);
753     else
754         w2 = (int)(.88*w);
755     w = (w/pegs-w2/pegs)*(pegs-1)+w2;
756     if (discs <= 5)
757         h2 = (int)(.5*h);
758     else if (discs <= 10)
759         h2 = (int)(.6*h);
760     else if (discs <= 15)
761         h2 = (int)(.7*h);
762     else if (discs <= 20)
763         h2 = (int)(.75*h);
764     else if (discs <= 25)
765         h2 = (int)(.8*h);
766     else if (discs <= 32)
767         h2 = (int)(.88*h);
768     else if (discs <= 38)
769         h2 = (int)(.9*h);
770     else
771         h2 = (int)(.92*h);
772     w3 = (w2/pegs-(w2/pegs/2)%discs)/discs;
773     if (w3*discs >= w2/pegs-10)
774         w3 = (w2/pegs-10)/discs-1;
775     if (w3%2 == 1)
776         w3--;
777     cubs.p1.setLayout(cubs.gb10);
778     cubs.p1.myDraw(cubs.p1.getGraphics());
779     if (w3 == 0 || (h2-21)/discs < 3)
780     {

```

```

781     tooMany = true;
782     cubs.mc.setSize(w, h);
783     cubs.gbc0.anchor = GridBagConstraints.CENTER;
784     cubs.gbc0.ipadx = 0;
785     cubs.gbc0.gridy = 0;
786     cubs.gbl0.setConstraints(cubs.mc, cubs.gbc0);
787     cubs.p1.add(cubs.mc);
788     cubs.gbc0.anchor = GridBagConstraints.SOUTH;
789     cubs.gbc0.gridy = 10;
790     moveCanvas = new MyCanvas2(cubs, this);
791     moveCanvas.setSize(w-50, 21);
792     cubs.gbl0.setConstraints(moveCanvas, cubs.gbc0);
793     cubs.p1.add(moveCanvas);
794     cubs.mc.setText2("Not Enough Room to Display Discs and Pegs!!!");
795     moveCanvas.setText0();
796     moveCanvas.paint0(moveCanvas.getGraphics());
797 }
798 else
799 {
800     if (discs < 16)
801         h_mod = 3;
802     gameBoard = new MyCanvas(cubs, discs, pegs, w3, (h2-21)/discs-((h2-21)/
803         discs)%h_mod, w2/pegs+6, w/pegs-w2/pegs-7, (h-21-((h2-21)/discs)*
804         discs)/2, -(w/pegs-w2/pegs-7)/3);
805     gameBoard.setSize(w, h);
806     cubs.gbc0.anchor = GridBagConstraints.CENTER;
807     cubs.gbc0.ipadx = 0;
808     cubs.gbc0.gridy = 0;
809     cubs.gbl0.setConstraints(gameBoard, cubs.gbc0);
810     cubs.p1.add(gameBoard);
811     cubs.gbc0.anchor = GridBagConstraints.SOUTH;
812     cubs.gbc0.gridy = 10;
813     moveCanvas = new MyCanvas2(cubs, this);
814     moveCanvas.setSize(w-50, 21);
815     cubs.gbl0.setConstraints(moveCanvas, cubs.gbc0);
816     cubs.p1.add(moveCanvas);
817     gameBoard.myDraw0();
818     moveCanvas.setText0();
819     moveCanvas.paint0(moveCanvas.getGraphics());
820 }
821 }
822
823 public void optimize()throws MyExceptionB
824 {
825     int n, p, k, i;
826     long count2, count3;
827     for (n = 1; n <= discs; n++)
828         for (p = 4; p <= pegs; p++)
829             if (!cubs.doneYet[n-1][p-1])
830                 {
831                     cubs.doneYet[n-1][p-1] = true;
832                     nums.clear();
833                     count2 = Long.MAX_VALUE;
834                     k = 0;
835                     if (p == 4 && n > 61)
836                         k = n-61;
837                     for (; k < n; k++)
838                         {
839                             if (!cubs.running)
840                                 throw myE_B;

```

```

841         count3 = 0;
842         if (k > 0)
843         {
844             if (k >= p)
845                 count3 = 2*cubs.moveArray[k-1][p-1];
846             else
847                 count3 = 4*k-2;
848         }
849         if (p > 4)
850         {
851             if (n-k > p-2)
852                 count3+=cubs.moveArray[n-k-1][p-2];
853             else
854                 count3+=(n-k)*2-1;
855         }
856         else
857             count3+=cubs.powArray[n-k-1];
858         if (count3 < count2)
859         {
860             count2 = count3;
861             nums.clear();
862             nums.add(k);
863         }
864         else if (count3 == count2)
865             nums.add(k);
866     }
867     cubs.moveArray[n-1][p-1] = count2;
868     cubs.optimals[n-1][p-1] = new int[nums.size()];
869     for (i = 0; i < nums.size(); i++)
870         cubs.optimals[n-1][p-1][i] = (int)nums.get(i);
871     }
872 }
873
874 public void chicago(int discs2, int init2, int fnl2, int[] aux)throws
875     MyExceptionA, MyExceptionB
876 {
877     if (!cubs.running)
878         throw myE_B;
879     int k, i, r;
880     int[] AUX1, AUX2, order = new int[aux.length];
881     if (discs2 < aux.length+2)
882     {
883         if (cubs.scrambled)
884         {
885             for (i = 0; i < aux.length; i++)
886                 nums.add(i);
887             for (i = 0; i < aux.length; i++)
888             {
889                 r = myRandom(nums.size());
890                 order[i] = (int)nums.get(r);
891                 nums.remove(r);
892             }
893         }
894         else
895             for (i = 0; i < aux.length; i++)
896                 order[i] = i;
897         for (i = 0; i < discs2-1; i++)
898             myMove(init2, aux[order[i]]);
899         myMove(init2, fnl2);
900         for (i = discs2-2; i >= 0; i--)

```

```

901         myMove(aux[order[i]], fnl2);
902     }
903     else if (aux.length > 1)
904     {
905         if (cubs.scrambleK)
906         {
907             k = cubs.optimals[discs2-1][aux.length+1][cubs.optOrder[discs2
908                 -1][aux.length+1][cubs.optOrderIndex[discs2-1][aux.length+1]]];
909             cubs.optOrderIndex[discs2-1][aux.length+1]++;
910             cubs.optOrderIndex[discs2-1][aux.length+1]%=cubs.optimals[discs2
911                 -1][aux.length+1].length;
912         }
913         else
914             k = cubs.optimals[discs2-1][aux.length+1][cubs.optimalsIndex[discs2
915                 -1][aux.length+1]];
916         if (cubs.scrambleD)
917         {
918             for (i = 0; i < aux.length-1; i++)
919                 nums.add(i);
920             for (i = 0; i < aux.length-1; i++)
921             {
922                 r = myRandom(nums.size());
923                 order[i] = (int)nums.get(r);
924                 nums.remove(r);
925             }
926         }
927         else
928             for (i = 0; i < aux.length-1; i++)
929                 order[i] = i;
930         AUX2 = new int[aux.length];
931         for (i = 0; i < aux.length-1; i++)
932             AUX2[i] = aux[order[i]+1];
933         AUX2[aux.length-1] = fnl2;
934         chicago(k, init2, aux[0], AUX2);
935         if (cubs.scrambleD)
936         {
937             for (i = 0; i < aux.length-1; i++)
938                 nums.add(i);
939             for (i = 0; i < aux.length-1; i++)
940             {
941                 r = myRandom(nums.size());
942                 order[i] = (int)nums.get(r);
943                 nums.remove(r);
944             }
945         }
946         AUX1 = new int[aux.length-1];
947         for (i = 0; i < aux.length-1; i++)
948             AUX1[i] = aux[order[i]+1];
949         chicago(discs2-k, init2, fnl2, AUX1);
950         if (cubs.scrambleD)
951         {
952             for (i = 0; i < aux.length-1; i++)
953                 nums.add(i);
954             for (i = 0; i < aux.length-1; i++)
955             {
956                 r = myRandom(nums.size());
957                 order[i] = (int)nums.get(r);
958                 nums.remove(r);
959             }
960         }

```



```

961     for (i = 0; i < aux.length-1; i++)
962         AUX2[i] = aux[order[i]+1];
963     AUX2[aux.length-1] = init2;
964     chicago(k, aux[0], fnl2, AUX2);
965 }
966 else
967 {
968     AUX1 = new int[1];
969     if (discs2 > 1)
970     {
971         AUX1[0] = fnl2;
972         chicago(discs2-1, init2, aux[0], AUX1);
973     }
974     myMove(init2, fnl2);
975     if (discs2 > 1)
976     {
977         AUX1[0] = init2;
978         chicago(discs2-1, aux[0], fnl2, AUX1);
979     }
980 }
981 }
982
983 public void myMove(int init2, int fnl2) throws MyExceptionA
984 {
985     if (count < cubs.moveCountA)
986     {
987         cubs.moves[0][count] = (byte)init2;
988         cubs.moves[1][count++] = (byte)fnl2;
989     }
990     else
991         throw myE_A;
992 }
993
994 public void myMove2() throws MyExceptionB
995 {
996     int num, init2, fnl2;
997     String logAppend, minus = "";
998     if (!cubs.pause.getState() && cubs.running)
999     {
1000         try
1001         {
1002             sleep(time);
1003         }
1004         catch (InterruptedException e1){}
1005     }
1006     if (!cubs.running)
1007         throw myE_B;
1008     doSleep = true;
1009     while (cubs.pause.getState() && doSleep)
1010     {
1011         if (!cubs.running)
1012             throw myE_B;
1013         try
1014         {
1015             sleep(35);
1016         }
1017         catch (InterruptedException e2){}
1018     }
1019     if (cubs.forward.getState())
1020     {

```

```

1021     if (direction == 1)
1022         pointer++;
1023         init2 = cubs.moves[0][pointer];
1024         fnl2 = cubs.moves[1][pointer];
1025         direction = 1;
1026     }
1027     else
1028     {
1029         if (direction == 2)
1030             pointer--;
1031             init2 = cubs.moves[1][pointer];
1032             fnl2 = cubs.moves[0][pointer];
1033             direction = 2;
1034     }
1035     num = cubs.values[cubs.valuesIndex[init2-1]][init2-1];
1036     if (!tooMany)
1037         gameBoard.myDraw(0, cubs.valuesIndex[init2-1], init2-1, cubs.gx);
1038     cubs.valuesIndex[init2-1]++;
1039     cubs.values[cubs.valuesIndex[fnl2-1]-1][fnl2-1] = num;
1040     if (!tooMany)
1041         gameBoard.myDraw(num, cubs.valuesIndex[fnl2-1]-1, fnl2-1, cubs.gx);
1042     if (direction == 2)
1043         minus = "-";
1044     moveCanvas.dn = direction;
1045     cubs.dn = direction;
1046     cubs.text = "move "+(pointer+1)+minus+" of "+cubs.moveCountB+
1047     " total moves          disc "+num+" was moved from peg "+init2+
1048     " to peg "+fnl2;
1049     moveCanvas.setText2(cubs.text, direction);
1050     moveCanvas.paint0(moveCanvas.getGraphics());
1051     cubs.valuesIndex[fnl2-1]--;
1052     if (direction == 2)
1053         minus = "-";
1054     logAppend = "\n    move "+(pointer+1)+minus+"\t    disc "+num+
1055     " was moved from peg "+init2+" to peg "+fnl2;
1056     if (cubs.log.getState())
1057         cubs.logBox.append(logAppend);
1058     cubs.logger+=logAppend;
1059 }
1060
1061 public void timeage()
1062 {
1063     int speedDial = cubs.speed.getValue();
1064     if (speedDial <= 30)
1065         time = 1025-10*speedDial;
1066     else if (speedDial <= 50)
1067         time = 1025-(int)(10*speedDial*.99);
1068     else if (speedDial <= 65)
1069         time = 1000-(int)(10*speedDial*.985);
1070     else if (speedDial <= 75)
1071         time = 975-(int)(10*speedDial*.98);
1072     else if (speedDial <= 82)
1073         time = 950-(int)(10*speedDial*.95);
1074     else if (speedDial <= 88)
1075         time = 900-(int)(10*speedDial*.85);
1076     else if (speedDial <= 93)
1077         time = 750-(int)(10*speedDial*.75);
1078     else if (speedDial <= 96)
1079         time = 675-(int)(10*speedDial*.65);
1080     else if (speedDial <= 98)

```

```

1081     time = 580-(int)(10*speedDial*.55);
1082     else
1083         time = 500-(int)(10*speedDial*.48);
1084     }
1085
1086     public int myRandom(int high)
1087     {
1088         long sum = 0, base = 1;
1089         byte[] b = new byte[64];
1090         rg.nextBytes(b);
1091         for (int i = 0; i <= 3; i++)
1092         {
1093             sum+=(b[i]+128)*base;
1094             base*=256;
1095         }
1096         return (int)(sum%high);
1097     }
1098 }
1099
1100
1101 class MyExceptionA extends Exception{}
1102
1103 class MyExceptionB extends Exception{}
1104
1105
1106 class MyKeyAdapter1 extends KeyAdapter
1107 {
1108     public AppletImage cubs;
1109     public int discs, pegs, init, fnl, seed1, seed2;
1110
1111     public MyKeyAdapter1(AppletImage ts)
1112     {
1113         cubs = ts;
1114     }
1115
1116     public void keyReleased(KeyEvent e)
1117     {
1118         boolean error1 = false, error2 = false, error3 = false;
1119         if ((e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1120             KeyEvent.VK_SPACE) && !cubs.running)
1121         {
1122             if (!cubs.entries[3].getText().toLowerCase().equals("max"))
1123             {
1124                 try
1125                 {
1126                     discs = Integer.parseInt(cubs.entries[0].getText());
1127                     pegs = Integer.parseInt(cubs.entries[1].getText());
1128                     init = Integer.parseInt(cubs.entries[2].getText());
1129                     fnl = Integer.parseInt(cubs.entries[3].getText());
1130                 }
1131                 catch (NumberFormatException e1)
1132                 {
1133                     error1 = true;
1134                 }
1135             }
1136             else
1137             {
1138                 try
1139                 {
1140                     discs = Integer.parseInt(cubs.entries[0].getText());

```

```

1141         pegs = Integer.parseInt(cubs.entries[1].getText());
1142         init = Integer.parseInt(cubs.entries[2].getText());
1143     }
1144     catch (NumberFormatException e2)
1145     {
1146         error1 = true;
1147     }
1148     if (!error1)
1149         fnl = pegs;
1150 }
1151 cubs.scrambleK = cubs.scramble1.getState();
1152 cubs.scrambleD = cubs.scramble2.getState();
1153 try
1154 {
1155     seed1 = Integer.parseInt(cubs.entries[4].getText());
1156 }
1157 catch (NumberFormatException e3)
1158 {
1159     error2 = true;
1160 }
1161 finally
1162 {
1163     if (seed1 < 0 || seed1 > 999)
1164         error2 = true;
1165 }
1166 try
1167 {
1168     seed2 = Integer.parseInt(cubs.entries[5].getText());
1169 }
1170 catch (NumberFormatException e4)
1171 {
1172     if (cubs.scrambled)
1173         error3 = true;
1174 }
1175 finally
1176 {
1177     if (cubs.scrambledD && (seed2 < 0 || seed2 > 999))
1178         error3 = true;
1179 }
1180 try
1181 {
1182     cubs.colorY = Integer.parseInt(cubs.entries[8].getText());
1183     cubs.colorX = Integer.parseInt(cubs.entries[9].getText());
1184 }
1185 catch (NumberFormatException e5)
1186 {
1187     error1 = true;
1188 }
1189 if (error1 || discs < 3 || pegs < 3 || discs > 1000 || pegs > 100 ||
1190     init < 1 || fnl < 1 || init == fnl || init > pegs || fnl > pegs ||
1191     cubs.colorX < 1 || cubs.colorX > 20 || cubs.colorY < 2 ||
1192     cubs.colorY > 20 || (pegs > 3 && (error2 || error3)))
1193     cubs.logBox.setText("\n\n\n    INVALID ENTRY!!!");
1194 else
1195 {
1196     cubs.cubs2 = new Spire(cubs, discs, pegs, init, fnl, seed1, seed2);
1197     cubs.cubs2.start();
1198 }
1199 }
1200 }

```

```

1201 }
1202
1203
1204 class MyMouseAdapter1 extends MouseAdapter
1205 {
1206     public AppletImage cubs;
1207     public int discs, pegs, init, fnl, seed1, seed2;
1208
1209     public MyMouseAdapter1(AppletImage ts)
1210     {
1211         cubs = ts;
1212     }
1213
1214     public void mousePressed(MouseEvent e)
1215     {
1216         boolean error1 = false, error2 = false, error3 = false;
1217         if (e.getButton() == MouseEvent.BUTTON1 && !cubs.running)
1218         {
1219             if (!cubs.entries[3].getText().toLowerCase().equals("max"))
1220             {
1221                 try
1222                 {
1223                     discs = Integer.parseInt(cubs.entries[0].getText());
1224                     pegs = Integer.parseInt(cubs.entries[1].getText());
1225                     init = Integer.parseInt(cubs.entries[2].getText());
1226                     fnl = Integer.parseInt(cubs.entries[3].getText());
1227                 }
1228                 catch (NumberFormatException e1)
1229                 {
1230                     error1 = true;
1231                 }
1232             }
1233             else
1234             {
1235                 try
1236                 {
1237                     discs = Integer.parseInt(cubs.entries[0].getText());
1238                     pegs = Integer.parseInt(cubs.entries[1].getText());
1239                     init = Integer.parseInt(cubs.entries[2].getText());
1240                 }
1241                 catch (NumberFormatException e2)
1242                 {
1243                     error1 = true;
1244                 }
1245                 if (!error1)
1246                     fnl = pegs;
1247             }
1248             cubs.scrambleK = cubs.scramble1.getState();
1249             cubs.scrambleD = cubs.scramble2.getState();
1250             try
1251             {
1252                 seed1 = Integer.parseInt(cubs.entries[4].getText());
1253             }
1254             catch (NumberFormatException e3)
1255             {
1256                 error2 = true;
1257             }
1258             finally
1259             {
1260                 if (seed1 < 0 || seed1 > 999)

```

```

1261         error2 = true;
1262     }
1263     try
1264     {
1265         seed2 = Integer.parseInt(cubs.entries[5].getText());
1266     }
1267     catch (NumberFormatException e4)
1268     {
1269         if (cubs.scrambled)
1270             error3 = true;
1271     }
1272     finally
1273     {
1274         if (cubs.scrambled && (seed2 < 0 || seed2 > 999))
1275             error3 = true;
1276     }
1277     try
1278     {
1279         cubs.colorY = Integer.parseInt(cubs.entries[8].getText());
1280         cubs.colorX = Integer.parseInt(cubs.entries[9].getText());
1281     }
1282     catch (NumberFormatException e5)
1283     {
1284         error1 = true;
1285     }
1286     if (error1 || discs < 3 || pegs < 3 || discs > 1000 || pegs > 100 ||
1287         init < 1 || fnl < 1 || init == fnl || init > pegs || fnl > pegs ||
1288         cubs.colorX < 1 || cubs.colorX > 20 || cubs.colorY < 2 ||
1289         cubs.colorY > 20 || (pegs > 3 && (error2 || error3)))
1290         cubs.logBox.setText("\n\n\n    INVALID ENTRY!!!");
1291     else
1292     {
1293         cubs.cubs2 = new Spire(cubs, discs, pegs, init, fnl, seed1, seed2);
1294         cubs.cubs2.start();
1295     }
1296 }
1297 }
1298 }
1299
1300
1301 class MyKeyAdapter2 extends KeyAdapter
1302 {
1303     public AppletImage cubs;
1304
1305     public MyKeyAdapter2(AppletImage ts)
1306     {
1307         cubs = ts;
1308     }
1309
1310     public void keyReleased(KeyEvent e)
1311     {
1312         if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1313             KeyEvent.VK_SPACE)
1314         {
1315             cubs.running = false;
1316             cubs.moves = null;
1317             cubs.optOrder = null;
1318             cubs.p1.removeAll();
1319             cubs.cubs2 = null;
1320             System.gc();

```

```

1321     }
1322   }
1323 }
1324
1325
1326 class MyMouseAdapter2 extends MouseAdapter
1327 {
1328     public AppletImage cubs;
1329
1330     public MyMouseAdapter2(AppletImage ts)
1331     {
1332         cubs = ts;
1333     }
1334
1335     public void mousePressed(MouseEvent e)
1336     {
1337         if (e.getButton() == MouseEvent.BUTTON1)
1338         {
1339             cubs.running = false;
1340             cubs.moves = null;
1341             cubs.optOrder = null;
1342             cubs.p1.removeAll();
1343             cubs.cubs2 = null;
1344             System.gc();
1345         }
1346     }
1347 }
1348
1349
1350 class MyKeyAdapter3 extends KeyAdapter
1351 {
1352     public Spire cubs2;
1353
1354     public void keyPressed(KeyEvent e)
1355     {
1356         if ((e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1357             KeyEvent.VK_SPACE) && cubs2 != null)
1358             cubs2.doSleep = false;
1359     }
1360 }
1361
1362
1363 class MyMouseAdapter3 extends MouseAdapter
1364 {
1365     public Spire cubs2;
1366
1367     public void mousePressed(MouseEvent e)
1368     {
1369         if (e.getButton() == MouseEvent.BUTTON1 && cubs2 != null)
1370             cubs2.doSleep = false;
1371     }
1372 }
1373
1374
1375 class MyKeyAdapter4 extends KeyAdapter
1376 {
1377     public AppletImage cubs;
1378
1379     public MyKeyAdapter4(AppletImage ts)
1380     {

```

```

1381     cubs = ts;
1382 }
1383
1384 public void keyReleased(KeyEvent e)
1385 {
1386     if ((e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1387         KeyEvent.VK_SPACE) && !cubs.running)
1388     {
1389         cubs.p1.removeAll();
1390         cubs.p1.setLayout(new BorderLayout());
1391         cubs.p1.add(cubs.helpBox);
1392         cubs.pF.paintAll(cubs.pF.getGraphics());
1393     }
1394 }
1395 }
1396
1397
1398 class MyMouseAdapter4 extends MouseAdapter
1399 {
1400     public AppletImage cubs;
1401
1402     public MyMouseAdapter4(AppletImage ts)
1403     {
1404         cubs = ts;
1405     }
1406
1407     public void mousePressed(MouseEvent e)
1408     {
1409         if (e.getButton() == MouseEvent.BUTTON1 && !cubs.running)
1410         {
1411             cubs.p1.removeAll();
1412             cubs.p1.setLayout(new BorderLayout());
1413             cubs.p1.add(cubs.helpBox);
1414             cubs.pF.paintAll(cubs.pF.getGraphics());
1415         }
1416     }
1417 }
1418
1419
1420 class MyKeyAdapter5 extends KeyAdapter
1421 {
1422     public AppletImage cubs;
1423
1424     public MyKeyAdapter5(AppletImage ts)
1425     {
1426         cubs = ts;
1427     }
1428
1429     public void keyReleased(KeyEvent e)
1430     {
1431         if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1432             KeyEvent.VK_SPACE)
1433             Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new
1434                 StringSelection(cubs.logger), new StringSelection(cubs.logger));
1435     }
1436 }
1437
1438
1439 class MyMouseAdapter5 extends MouseAdapter
1440 {

```



```

1441 public AppletImage cubs;
1442
1443 public MyMouseAdapter5(AppletImage ts)
1444 {
1445     cubs = ts;
1446 }
1447
1448 public void mousePressed(MouseEvent e)
1449 {
1450     if (e.getButton() == MouseEvent.BUTTON1)
1451         Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new
1452             StringSelection(cubs.logger), new StringSelection(cubs.logger));
1453 }
1454 }
1455
1456
1457 class MyKeyAdapter6 extends KeyAdapter
1458 {
1459     public AppletImage cubs;
1460     public int X, Y;
1461
1462     public MyKeyAdapter6(AppletImage ts)
1463     {
1464         cubs = ts;
1465     }
1466
1467     public void keyTyped(KeyEvent e)
1468     {
1469         if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1470             KeyEvent.VK_SPACE)
1471         {
1472             try
1473             {
1474                 X = Integer.parseInt(cubs.entries[6].getText());
1475                 Y = Integer.parseInt(cubs.entries[7].getText());
1476             }
1477             catch (NumberFormatException e2) {}
1478             finally
1479             {
1480                 cubs.pF.setSize(X, Y);
1481                 if (cubs.cubs2 != null)
1482                 {
1483                     cubs.p1.removeAll();
1484                     cubs.cubs2.drawBoard();
1485                     if (!cubs.cubs2.tooMany)
1486                         cubs.gx = cubs.cubs2.gameBoard.getGraphics();
1487                     cubs.pF.paintAll(cubs.pF.getGraphics());
1488                 }
1489             }
1490         }
1491     }
1492 }
1493
1494
1495 class MyMouseAdapter6 extends MouseAdapter
1496 {
1497     public AppletImage cubs;
1498     public int X, Y;
1499
1500     public MyMouseAdapter6(AppletImage ts)

```

```

1501     {
1502         cubs = ts;
1503     }
1504
1505     public void mousePressed(MouseEvent e)
1506     {
1507         if (e.getButton() == MouseEvent.BUTTON1)
1508         {
1509             try
1510             {
1511                 X = Integer.parseInt(cubs.entries[6].getText());
1512                 Y = Integer.parseInt(cubs.entries[7].getText());
1513             }
1514             catch (NumberFormatException e2) {}
1515             finally
1516             {
1517                 cubs.pF.setSize(X, Y);
1518                 if (cubs.cubs2 != null)
1519                 {
1520                     cubs.p1.removeAll();
1521                     cubs.cubs2.drawBoard();
1522                     if (!cubs.cubs2.tooMany)
1523                         cubs.gx = cubs.cubs2.gameBoard.getGraphics();
1524                     cubs.pF.paintAll(cubs.pF.getGraphics());
1525                 }
1526             }
1527         }
1528     }
1529 }
1530
1531
1532 class MyKeyAdapter7 extends KeyAdapter
1533 {
1534     public AppletImage cubs;
1535     public int Y, X;
1536
1537     public MyKeyAdapter7(AppletImage ts)
1538     {
1539         cubs = ts;
1540     }
1541
1542     public void keyTyped(KeyEvent e)
1543     {
1544         if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1545             KeyEvent.VK_SPACE)
1546         {
1547             try
1548             {
1549                 Y = Integer.parseInt(cubs.entries[8].getText());
1550                 X = Integer.parseInt(cubs.entries[9].getText());
1551             }
1552             catch (NumberFormatException e2) {}
1553             finally
1554             {
1555                 if (cubs.cubs2 != null)
1556                     if (!cubs.cubs2.tooMany)
1557                     {
1558                         if (Y >= 2 && Y <= 20 && X >= 1 && X <= 20)
1559                         {
1560                             cubs.colorY = Y;

```

```

1561         cubs.colorX = X;
1562         cubs.oldColorY = Y;
1563         cubs.oldColorX = X;
1564         cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
1565             colorPos1);
1566         cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
1567             init, cubs.cubs2.fn1, X, cubs.colorPos2);
1568         cubs.cubs2.gameBoard.update(cubs.gx);
1569     }
1570 }
1571 }
1572 }
1573 }
1574 }
1575
1576
1577 class MyMouseAdapter7 extends MouseAdapter
1578 {
1579     public AppletImage cubs;
1580     public int Y, X;
1581
1582     public MyMouseAdapter7(AppletImage ts)
1583     {
1584         cubs = ts;
1585     }
1586
1587     public void mousePressed(MouseEvent e)
1588     {
1589         if (e.getButton() == MouseEvent.BUTTON1)
1590         {
1591             try
1592             {
1593                 Y = Integer.parseInt(cubs.entries[8].getText());
1594                 X = Integer.parseInt(cubs.entries[9].getText());
1595             }
1596             catch (NumberFormatException e2) {}
1597             finally
1598             {
1599                 if (cubs.cubs2 != null)
1600                     if (!cubs.cubs2.tooMany)
1601                     {
1602                         if (Y >= 2 && Y <= 20 && X >= 1 && X <= 20)
1603                         {
1604                             cubs.colorY = Y;
1605                             cubs.colorX = X;
1606                             cubs.oldColorY = Y;
1607                             cubs.oldColorX = X;
1608                             cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
1609                                 colorPos1);
1610                             cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
1611                                 init, cubs.cubs2.fn1, X, cubs.colorPos2);
1612                             cubs.cubs2.gameBoard.update(cubs.gx);
1613                         }
1614                     }
1615                 }
1616             }
1617         }
1618     }
1619
1620

```

```

1621 class MyKeyAdapter8 extends KeyAdapter
1622 {
1623     public AppletImage cubs;
1624     public int Y;
1625
1626     public MyKeyAdapter8(AppletImage ts)
1627     {
1628         cubs = ts;
1629     }
1630
1631     public void keyReleased(KeyEvent e)
1632     {
1633         if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1634             KeyEvent.VK_SPACE)
1635         {
1636             try
1637             {
1638                 Y = Integer.parseInt(cubs.entries[8].getText());
1639             }
1640             catch (NumberFormatException e2) {}
1641             finally
1642             {
1643                 if (cubs.cubs2 != null)
1644                     if (!cubs.cubs2.tooMany)
1645                     {
1646                         if (Y >= 2 && Y <= 20)
1647                         {
1648                             cubs.colorY = Y;
1649                             cubs.colorPos1++;
1650                             cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
1651                                 colorPos1);
1652                             cubs.cubs2.gameBoard.paint0(cubs.gx);
1653                         }
1654                     }
1655             }
1656         }
1657     }
1658 }
1659
1660
1661 class MyMouseAdapter8 extends MouseAdapter
1662 {
1663     public AppletImage cubs;
1664     public int Y;
1665
1666     public MyMouseAdapter8(AppletImage ts)
1667     {
1668         cubs = ts;
1669     }
1670
1671     public void mousePressed(MouseEvent e)
1672     {
1673         if (e.getButton() == MouseEvent.BUTTON1)
1674         {
1675             try
1676             {
1677                 Y = Integer.parseInt(cubs.entries[8].getText());
1678             }
1679             catch (NumberFormatException e2) {}
1680             finally

```

```

1681     {
1682         if (cubs.cubs2 != null)
1683             if (!cubs.cubs2.tooMany)
1684                 {
1685                     if (Y >= 2 && Y <= 20)
1686                         {
1687                             cubs.colorY = Y;
1688                             cubs.colorPos1++;
1689                             cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
1690                                 colorPos1);
1691                             cubs.cubs2.gameBoard.paint0(cubs.gx);
1692                         }
1693                 }
1694             }
1695         }
1696     }
1697 }
1698
1699
1700 class MyKeyAdapter9 extends KeyAdapter
1701 {
1702     public AppletImage cubs;
1703     public int X;
1704
1705     public MyKeyAdapter9(AppletImage ts)
1706     {
1707         cubs = ts;
1708     }
1709
1710     public void keyReleased(KeyEvent e)
1711     {
1712         if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1713             KeyEvent.VK_SPACE)
1714             {
1715                 try
1716                 {
1717                     X = Integer.parseInt(cubs.entries[9].getText());
1718                 }
1719                 catch (NumberFormatException e2) {}
1720                 finally
1721                 {
1722                     if (cubs.cubs2 != null)
1723                         if (!cubs.cubs2.tooMany)
1724                             {
1725                                 if (X >= 1 && X <= 20)
1726                                     {
1727                                         cubs.colorX = X;
1728                                         cubs.colorPos2--;
1729                                         cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
1730                                             init, cubs.cubs2.fn1, X, cubs.colorPos2);
1731                                         cubs.cubs2.gameBoard.paint0(cubs.gx);
1732                                     }
1733                             }
1734                         }
1735                 }
1736             }
1737 }
1738
1739
1740 class MyMouseAdapter9 extends MouseAdapter

```

```

1741 {
1742     public AppletImage cubs;
1743     public int X;
1744
1745     public MyMouseAdapter9(AppletImage ts)
1746     {
1747         cubs = ts;
1748     }
1749
1750     public void mousePressed(MouseEvent e)
1751     {
1752         if (e.getButton() == MouseEvent.BUTTON1)
1753         {
1754             try
1755             {
1756                 X = Integer.parseInt(cubs.entries[9].getText());
1757             }
1758             catch (NumberFormatException e2) {}
1759             finally
1760             {
1761                 if (cubs.cubs2 != null)
1762                 {
1763                     if (!cubs.cubs2.tooMany)
1764                     {
1765                         if (X >= 1 && X <= 20)
1766                         {
1767                             cubs.colorX = X;
1768                             cubs.colorPos2--;
1769                             cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
1770                                 init, cubs.cubs2.fn1, X, cubs.colorPos2);
1771                             cubs.cubs2.gameBoard.paint0(cubs.gx);
1772                         }
1773                     }
1774                 }
1775             }
1776         }
1777
1778
1779     class MyKeyAdapter10 extends KeyAdapter
1780     {
1781         public AppletImage cubs;
1782         public int Y;
1783
1784         public MyKeyAdapter10(AppletImage ts)
1785         {
1786             cubs = ts;
1787         }
1788
1789         public void keyReleased(KeyEvent e)
1790         {
1791             if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1792                 KeyEvent.VK_SPACE)
1793             {
1794                 try
1795                 {
1796                     Y = Integer.parseInt(cubs.entries[8].getText());
1797                 }
1798                 catch (NumberFormatException e2) {}
1799                 finally
1800                 {

```

```

1801         if (cubs.cubs2 != null)
1802             if (!cubs.cubs2.tooMany)
1803                 {
1804                     if (Y >= 2 && Y <= 20)
1805                         {
1806                             cubs.colorY = Y;
1807                             cubs.colorPos1--;
1808                             cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
1809                                 colorPos1);
1810                             cubs.cubs2.gameBoard.paint0(cubs.gx);
1811                         }
1812                 }
1813             }
1814         }
1815     }
1816 }
1817
1818
1819 class MyMouseListener10 extends MouseAdapter
1820 {
1821     public AppletImage cubs;
1822     public int Y;
1823
1824     public MyMouseListener10(AppletImage ts)
1825     {
1826         cubs = ts;
1827     }
1828
1829     public void mousePressed(MouseEvent e)
1830     {
1831         if (e.getButton() == MouseEvent.BUTTON1)
1832             {
1833                 try
1834                 {
1835                     Y = Integer.parseInt(cubs.entries[8].getText());
1836                 }
1837                 catch (NumberFormatException e2) {}
1838                 finally
1839                 {
1840                     if (cubs.cubs2 != null)
1841                         if (!cubs.cubs2.tooMany)
1842                             {
1843                                 if (Y >= 2 && Y <= 20)
1844                                     {
1845                                         cubs.colorY = Y;
1846                                         cubs.colorPos1--;
1847                                         cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
1848                                             colorPos1);
1849                                         cubs.cubs2.gameBoard.paint0(cubs.gx);
1850                                     }
1851                             }
1852                         }
1853                 }
1854             }
1855 }
1856
1857
1858 class MyKeyListener11 extends KeyAdapter
1859 {
1860     public AppletImage cubs;

```

```

1861     public int X;
1862
1863     public MyKeyAdapter11(AppletImage ts)
1864     {
1865         cubs = ts;
1866     }
1867
1868     public void keyReleased(KeyEvent e)
1869     {
1870         if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
1871             KeyEvent.VK_SPACE)
1872         {
1873             try
1874             {
1875                 X = Integer.parseInt(cubs.entries[9].getText());
1876             }
1877             catch (NumberFormatException e2) {}
1878             finally
1879             {
1880                 if (cubs.cubs2 != null)
1881                     if (!cubs.cubs2.tooMany)
1882                     {
1883                         if (X >= 1 && X <= 20)
1884                         {
1885                             cubs.colorX = X;
1886                             cubs.colorPos2++;
1887                             cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
1888                                 init, cubs.cubs2.fn1, X, cubs.colorPos2);
1889                             cubs.cubs2.gameBoard.paint0(cubs.gx);
1890                         }
1891                     }
1892             }
1893         }
1894     }
1895 }
1896
1897
1898 class MyMouseAdapter11 extends MouseAdapter
1899 {
1900     public AppletImage cubs;
1901     public int X;
1902
1903     public MyMouseAdapter11(AppletImage ts)
1904     {
1905         cubs = ts;
1906     }
1907
1908     public void mousePressed(MouseEvent e)
1909     {
1910         if (e.getButton() == MouseEvent.BUTTON1)
1911         {
1912             try
1913             {
1914                 X = Integer.parseInt(cubs.entries[9].getText());
1915             }
1916             catch (NumberFormatException e2) {}
1917             finally
1918             {
1919                 if (cubs.cubs2 != null)
1920                     if (!cubs.cubs2.tooMany)

```



```

1921         {
1922             if (X >= 1 && X <= 20)
1923             {
1924                 cubs.colorX = X;
1925                 cubs.colorPos2++;
1926                 cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegS, cubs.cubs2.
1927                     init, cubs.cubs2.fn1, X, cubs.colorPos2);
1928                 cubs.cubs2.gameBoard.paint0(cubs.gx);
1929             }
1930         }
1931     }
1932 }
1933 }
1934 }
1935
1936
1937 class MyPanel extends Panel
1938 {
1939     public MyPanel(BorderLayout bl)
1940     {
1941         super(bl);
1942     }
1943
1944     public void myDraw(Graphics g)
1945     {
1946         if (g != null)
1947         {
1948             g.setColor(Color.blue);
1949             g.fillRect(0, 0, getSize().width, 4);
1950             g.fillRect(0, 0, 4, getSize().height);
1951             g.fillRect(0, getSize().height-4, getSize().width, 4);
1952             g.fillRect(getSize().width-4, 0, 4, getSize().height);
1953         }
1954     }
1955
1956     public void paint(Graphics g)
1957     {
1958         myDraw(g);
1959     }
1960 }
1961
1962
1963 class MyItemListener1 implements ItemListener
1964 {
1965     public AppletImage cubs;
1966
1967     public MyItemListener1(AppletImage ts)
1968     {
1969         cubs = ts;
1970     }
1971
1972     public void itemStateChanged(ItemEvent e)
1973     {
1974         if (cubs.cubs2 != null)
1975             if (!cubs.cubs2.tooMany)
1976                 cubs.cubs2.gameBoard.update(cubs.gx);
1977     }
1978 }
1979
1980

```

```

1981 class MyItemListener2 implements ItemListener
1982 {
1983     public AppletImage cubs;
1984
1985     public MyItemListener2(AppletImage ts)
1986     {
1987         cubs = ts;
1988     }
1989
1990     public void itemStateChanged(ItemEvent e)
1991     {
1992         if (cubs.log.getState())
1993             cubs.logBox.setText(cubs.logger);
1994         else
1995             cubs.logBox.setText(cubs.preLog);
1996     }
1997 }
1998
1999
2000 class MyAdjustmentListener implements AdjustmentListener
2001 {
2002     public AppletImage cubs;
2003
2004     public MyAdjustmentListener(AppletImage ts)
2005     {
2006         cubs = ts;
2007     }
2008
2009     public void adjustmentValueChanged(AdjustmentEvent e)
2010     {
2011         if (cubs.running)
2012             cubs.cubs2.timeage();
2013         cubs.dial.setText2(""+cubs.speed.getValue());
2014     }
2015 }
2016
2017
2018 class MyCanvas extends Canvas
2019 {
2020     public AppletImage cubs;
2021     public int discs, pegs, w, h, labelW, hgap, vgap, x;
2022
2023     public MyCanvas(AppletImage ts, int n, int p, int w0, int h0, int lW,
2024         int x2, int y2, int cor)
2025     {
2026         cubs = ts;
2027         discs = n;
2028         pegs = p;
2029         w = w0;
2030         h = h0;
2031         labelW = lW;
2032         hgap = x2;
2033         vgap = y2;
2034         x = cor;
2035     }
2036
2037     public void myDraw0()
2038     {
2039         Graphics g = getGraphics();
2040         int n = 0;

```



```

2101     {
2102     g.fillRoundRect(pPos*(labelW+hgap)+hgap+(labelW-w*n)/2-8+x, 2*h*
2103     nPos/3+h*discs/3+vgap, w*n+16,
2104     2*h/3-(h*(discs+nPos))%3, 2*roundness(), roundness());
2105     g.setColor(new Color(0, 0, 128));
2106     g.drawRoundRect(pPos*(labelW+hgap)+hgap+(labelW-w*n)/2-8+x, 2*h*
2107     nPos/3+h*discs/3+vgap, w*n+16,
2108     2*h/3-(h*(discs+nPos))%3, 2*roundness(), roundness());
2109     }
2110     if (cubs.discNums.getState() && Math.min(w+7, h-5) > 7)
2111     {
2112     g.setColor(new Color(0, 24, 160));
2113     if (discs >= 16)
2114     {
2115     if (Math.min(w+7, h-3) == 8)
2116     {
2117     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+7, h-3))
2118     );
2119     if (n <= 9)
2120     g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2121     Math.min(w+6, h+1)/4+x,
2122     nPos*h+(h+Math.min(w+6, h-3))/2+vgap);
2123     else
2124     g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2125     Math.min(w+6, h+1)/2-2+x,
2126     nPos*h+(h+Math.min(w+6, h-3))/2+vgap);
2127     }
2128     else if (Math.min(w+7, h-3) == 9 || Math.min(w+7, h-3) == 10)
2129     {
2130     if (h == 12)
2131     {
2132     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+9, h
2133     -4)));
2134     x0 = 3;
2135     }
2136     else
2137     {
2138     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+9, h
2139     -3)));
2140     x0 = 1;
2141     }
2142     if (n <= 9)
2143     g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2144     Math.min(w+6, h+1)/4+x,
2145     nPos*h+(int)((h+Math.min(w+6, h-x0))/1.85)+vgap);
2146     else
2147     g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2148     Math.min(w+6, h+1)/2-2+x,
2149     nPos*h+(int)((h+Math.min(w+6, h-x0))/1.85)+vgap);
2150     }
2151     else if (Math.min(w+6, h-3) > 10 && Math.min(w+6, h-3) < 15)
2152     {
2153     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+6, h-2))
2154     );
2155     if (n <= 9)
2156     g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2157     Math.min(w+7, h-2)/4+x,
2158     nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+vgap);
2159     else
2160     g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-

```

```

2161         (int)(Math.min(w+7, h-2)
2162         /1.85)+x, nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+
2163         vgap);
2164     }
2165     else if (Math.min(w+7, h-2) == 15 || Math.min(w+7, h-2) == 16)
2166     {
2167         g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+7, h-2))
2168         );
2169         if (n <= 9)
2170             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2171             Math.min(w+7, h-2)/4-1+x,
2172             nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+vgap);
2173         else
2174             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2175             (int)(Math.min(w+7, h-2)
2176             /1.52)+x, nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+
2177             vgap);
2178     }
2179     else if (Math.min(w+7, h-2) > 16)
2180     {
2181         g.setFont(new Font("Tahoma", Font.PLAIN, Math.min(w+8, h-2)
2182         ));
2183         if (n <= 9)
2184             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2185             Math.min(w+6, h-3)/4+x,
2186             nPos*h+(int)((h+Math.min(w+8, h-3))/2.15)+vgap);
2187         else
2188             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2189             (int)(Math.min(w+6, h-3)
2190             /1.65)+x, nPos*h+(int)((h+Math.min(w+8, h-3))/2.15)+
2191             vgap);
2192     }
2193     else
2194     {
2195         g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+9, h-1))
2196         );
2197         if (n <= 9)
2198             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2199             Math.min(w+8, h-2)/4+x,
2200             nPos*h+(int)((h+Math.min(w+9, h-2))/2.15)+vgap);
2201         else
2202             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2203             (int)(Math.min(w+6, h-3)
2204             /1.65)+x, nPos*h+(int)((h+Math.min(w+9, h-2))/2.15)+
2205             vgap);
2206     }
2207 }
2208 else
2209 {
2210     if (Math.min(w+7, 2*h/3-2) > 7 && Math.min(w+7, 2*h/3-2) < 16)
2211     {
2212         g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+3,
2213         2*h/3)));
2214         if (n <= 9)
2215             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2216             Math.min(w+3, 2*h/3-4)/4+x,
2217             2*h*nPos/3+h*discs/3+Math.min(w+5, 2*h/3-3)+vgap);
2218         else
2219             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2220             Math.min(w+3, 2*h/3-4)/2+x,

```

```

2221         2*h*nPos/3+h*discs/3+Math.min(w+5, 2*h/3-3)+vgap);
2222     }
2223     else if (Math.min(w+7, 2*h/3-5) >= 16 && Math.min(w+7, 2*h/3
2224         -5) <= 50)
2225     {
2226         if (Math.min(w+7, 2*h/3-2) <= 18)
2227             g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+7,
2228                 2*h/3-5)));
2229         else
2230             g.setFont(new Font("Tahoma", Font.PLAIN, Math.min(w+7,
2231                 2*h/3-5)));
2232         if (n <= 9)
2233             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2234                 Math.min(w+7, 2*h/3-2)/4+x,
2235                 2*h*nPos/3+h*discs/3+(int)((2*h/3+Math.min(w+7,
2236                 2*h/3-7))/2.15)+vgap);
2237         else
2238             g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
2239                 Math.min(w+7, 2*h/3-2)/2-1+x,
2240                 2*h*nPos/3+h*discs/3+(int)((2*h/3+Math.min(w+7,
2241                 2*h/3-7))/2.15)+vgap);
2242     }
2243 }
2244 }
2245 }
2246 }
2247 }
2248
2249 public int roundness()
2250 {
2251     int arc = 0;
2252     if (cubs.round.getState())
2253     {
2254         if (w >= 140 && h <= 30)
2255         {
2256             if (h > 20)
2257                 arc = 15;
2258             else if (h > 15 && h <= 20)
2259                 arc = 12;
2260             else if (h >= 13 && h <= 15)
2261                 arc = 10;
2262         }
2263         else if (w >= 140 && h > 30)
2264         {
2265             if (h >= 60)
2266                 arc = 17;
2267             if (h >= 40 && h < 60)
2268                 arc = 15;
2269             else
2270                 arc = 13;
2271         }
2272         else if (w < 140 && h <= 30)
2273         {
2274             if (h > 20)
2275                 arc = 13;
2276             else if (h > 15 && h <= 20)
2277                 arc = 11;
2278             else if (h >= 13 && h <= 15)
2279                 arc = 10;
2280         }

```

```

2281         else if (w < 140 && h > 30)
2282             arc = 13;
2283     }
2284     return arc;
2285 }
2286
2287 public void paint0(Graphics g)
2288 {
2289     if (g != null)
2290         for (int i = 0; i < pegs; i++)
2291         {
2292             g.clearRect(i*(labelW+hgap)+hgap+x, vgap, labelW, h*discs);
2293             g.setColor(cubs.c2[i]);
2294             g.fillRect(i*(labelW+hgap)+hgap+labelW/2-5+x, vgap, 11, h*discs);
2295             g.fillRect(i*(labelW+hgap)+hgap+x, h*discs+vgap, labelW, 21);
2296             g.setColor(Color.black);
2297             g.setFont(new Font("Verdana", Font.BOLD, 15));
2298             if (i < 9)
2299                 g.drawString(""+(i+1), i*(labelW+hgap)+hgap+labelW/2-5+x, h*discs
2300                     +16+vgap);
2301             else
2302                 g.drawString(""+(i+1), i*(labelW+hgap)+hgap+labelW/2-10+x,
2303                     h*discs+16+vgap);
2304             for (int j = cubs.valuesIndex[i]; j < discs; j++)
2305             {
2306                 g.setColor(cubs.c1[j]);
2307                 myDraw(cubs.values[j][i], j, i, g);
2308             }
2309         }
2310 }
2311
2312 public void paint(Graphics g)
2313 {
2314     paint0(g);
2315 }
2316 }
2317
2318
2319 class MyCanvas2 extends Canvas
2320 {
2321     public AppletImage cubs;
2322     public Spire cubs2;
2323     public String text = " ", altText = " ", altText0 = " ";
2324     public int dn, discs, pegs, moves, altPointer, altDn;
2325
2326     public MyCanvas2(AppletImage c, Spire c2)
2327     {
2328         cubs = c;
2329         cubs2 = c2;
2330     }
2331
2332     public void setText0()
2333     {
2334         if (cubs.running || (!cubs.running && cubs2.pointer <= 1))
2335             setText2(cubs.text, cubs.dn);
2336         else if (cubs2.pointer >= cubs.moveCountA-1)
2337             setText3(cubs.cubs2.discs, cubs.cubs2.pegs, cubs.moveCountA);
2338     }
2339
2340     public void setText2(String s, int dn2)

```

```

2341     {
2342         Graphics g = getGraphics();
2343         text = s;
2344         dn = dn2;
2345         if (g != null)
2346         {
2347             g.setFont(new Font("Tahoma", Font.BOLD, 13));
2348             g.setColor(Color.white);
2349             if (altDn == 1)
2350                 g.drawString(altText, getSize().width/2-205-(""+altPointer)
2351                     .length()*5, 15);
2352             else
2353                 g.drawString(altText, getSize().width/2-205-(""+altPointer)
2354                     .length()+1)*5, 15);
2355             if (dn == 1)
2356             {
2357                 g.setColor(new Color(0, 0, 224));
2358                 g.drawString(text, getSize().width/2-205-(""+(cubs2.pointer+1))
2359                     .length()*5, 15);
2360             }
2361             else
2362             {
2363                 g.setColor(new Color(128, 0, 224));
2364                 g.drawString(text, getSize().width/2-205-(""+(cubs2.pointer+1))
2365                     .length()+1)*5, 15);
2366             }
2367         }
2368         altText = text;
2369         altPointer = cubs2.pointer+1;
2370         altDn = dn;
2371     }
2372
2373     public void setText3(int discs2, int pegs2, int moves2)
2374     {
2375         Graphics g = getGraphics();
2376         if (g != null)
2377         {
2378             discs = discs2;
2379             pegs = pegs2;
2380             moves = moves2;
2381             g.clearRect(0, 0, getSize().width, getSize().height);
2382             g.setColor(new Color(0, 0, 224));
2383             g.setFont(new Font("Tahoma", Font.BOLD, 13));
2384             g.drawString(discs+" Disc", getSize().width/2-150-(""+discs).
2385                 length()-2)*3, 15);
2386             g.drawString("x", getSize().width/2-107+(""+discs).length()*5, 15);
2387             g.drawString(pegs+" Peg Problem Done In "+moves+" Moves", getSize().
2388                 width/2-92+(""+discs).length()*5, 15);
2389         }
2390     }
2391
2392     public void paint0(Graphics g)
2393     {
2394         if (g != null)
2395         {
2396             g.setFont(new Font("Tahoma", Font.BOLD, 13));
2397             if (cubs.running || (!cubs.running && cubs2.pointer <= 1))
2398             {
2399                 if (dn == 1)
2400                 {

```



```

2401         g.setColor(new Color(0, 0, 224));
2402         g.drawString(text, getSize().width/2-205-(""+(cubs2.pointer+1))
2403             .length()*5, 15);
2404     }
2405     else
2406     {
2407         g.setColor(new Color(128, 0, 224));
2408         g.drawString(text, getSize().width/2-205-(""+(cubs2.pointer+1))
2409             .length()+1)*5, 15);
2410     }
2411 }
2412 else if (cubs2.pointer >= cubs.moveCountA-1)
2413     setText3(cubs.cubs2.discs, cubs.cubs2.pegs, cubs.moveCountA);
2414 }
2415 }
2416
2417 public void paint(Graphics g)
2418 {
2419     paint0(g);
2420 }
2421 }
2422
2423
2424 class MyCanvas3 extends Canvas
2425 {
2426     public String text = "50", altText = "50";
2427
2428     public void setText2(String s)
2429     {
2430         text = s;
2431         Graphics g = getGraphics();
2432         if (g != null)
2433         {
2434             g.setFont(new Font("Verdana", Font.BOLD, 12));
2435             g.setColor(Color.lightGray);
2436             g.drawString(altText, 6, 12);
2437             update(g);
2438             g.setColor(new Color(48, 0, 192));
2439             g.drawString(text, 6, 12);
2440         }
2441         altText = text;
2442     }
2443
2444     public void paint(Graphics g)
2445     {
2446         if (g != null)
2447         {
2448             g.setFont(new Font("Verdana", Font.BOLD, 12));
2449             g.setColor(new Color(48, 0, 192));
2450             g.drawString(text, 6, 12);
2451         }
2452     }
2453 }
2454
2455
2456 class MyCanvas4 extends Canvas
2457 {
2458     public String text = " ";
2459
2460     public void setText2(String s)

```

```

2461     {
2462         text = s;
2463         Graphics g = getGraphics();
2464         if (g != null)
2465         {
2466             g.setFont(new Font("Times", Font.PLAIN, 24));
2467             g.drawString(text, getSize().width/2-225, getSize().height/2);
2468         }
2469     }
2470
2471     public void paint(Graphics g)
2472     {
2473         if (g != null)
2474         {
2475             g.setFont(new Font("Times", Font.PLAIN, 24));
2476             g.drawString(text, getSize().width/2-225, getSize().height/2);
2477         }
2478     }
2479 }
2480
2481
2482 class MyComponentAdapter extends ComponentAdapter
2483 {
2484     public AppletImage cubs;
2485
2486     public MyComponentAdapter(AppletImage ts)
2487     {
2488         cubs = ts;
2489     }
2490
2491     public void componentResized(ComponentEvent e)
2492     {
2493         cubs.pF.setTitle("Towers of Chicago (" + cubs.pF.getSize().width + " x "
2494             + cubs.pF.getSize().height + ")");
2495     }
2496 }
2497
2498
2499 class MyWindowAdapter extends WindowAdapter
2500 {
2501     public AppletImage cubs;
2502
2503     public MyWindowAdapter(AppletImage ts)
2504     {
2505         cubs = ts;
2506     }
2507
2508     public void windowClosing(WindowEvent e)
2509     {
2510         cubs.running = false;
2511         cubs.pF.dispose();
2512     }
2513 }

```