```java
//Robert Swartz
//Towers of Chicago
//Copyright 2024
//Fusion Power Applications


import java.awt.*;
import java.awt.datatransfer.*;
import java.awt.event.*;
import java.util.*;


public class Towers
{
   public static void main(String[] args)
   {
      AppletImage cubs = new AppletImage();
      int i, w = 1300, h = 1000, h2 = 90;
      String[] instructions = {"# of Discs ", "# of Pegs ", "Initial Peg ",
         "Final Peg ", "Speed ", "Random k Seed ", "Random Disc Seed ",
         "Width ", "Height ", "Disc Colors ", "Peg Colors "},
         defaults = {"40", "8", "1", "max", "250", ""+w, ""+h, "20", "6"};
      cubs.powArray[0] = 1;
      for (i = 1; i <= 62; i++)
         cubs.powArray[i] = cubs.powArray[i-1]*2+1;
      cubs.pF.setLocation(125, 100);
      cubs.pF.setVisible(true);
      cubs.pF.setSize(w, h);
      cubs.pF.setMinimumSize(new Dimension(w, 3*h/5));
      cubs.mwa = new MyWindowAdapter(cubs);
      cubs.pF.addWindowListener(cubs.mwa);
      cubs.mca = new MyComponentAdapter(cubs);
      cubs.pF.addComponentListener(cubs.mca);
      cubs.p0 = new Panel(new BorderLayout());
      cubs.p0.setSize(w, h);
      cubs.pF.add(cubs.p0);
      cubs.start = new Button("    Start    ");
      cubs.stop = new Button("    Stop    ");
      cubs.move = new Button("    Move    ");
      cubs.help = new Button("    Help    ");
      cubs.copy = new Button(" Copy ");
      cubs.resize = new Button("  Resize  ");
      cubs.colors = new Button("  Colors  ");
      cubs.shiftC[0] = new Button(">");
      cubs.shiftC[1] = new Button(">");
      cubs.shiftC[2] = new Button("<");
      cubs.shiftC[3] = new Button("<");
      cubs.mka1 = new MyKeyAdapter1(cubs);
      cubs.mma1 = new MyMouseAdapter1(cubs);
      cubs.mka2 = new MyKeyAdapter2(cubs);
      cubs.mma2 = new MyMouseAdapter2(cubs);
      cubs.mka3 = new MyKeyAdapter3();
      cubs.mma3 = new MyMouseAdapter3();
      cubs.mka4 = new MyKeyAdapter4(cubs);
      cubs.mma4 = new MyMouseAdapter4(cubs);
      cubs.mka5 = new MyKeyAdapter5(cubs);
      cubs.mma5 = new MyMouseAdapter5(cubs);
      cubs.mka6 = new MyKeyAdapter6(cubs);
      cubs.mma6 = new MyMouseAdapter6(cubs);
      cubs.mka7 = new MyKeyAdapter7(cubs);
```

```
        cubs.mma7 = new MyMouseAdapter7(cubs);
        cubs.mka8 = new MyKeyAdapter8(cubs);
        cubs.mma8 = new MyMouseAdapter8(cubs);
        cubs.mka9 = new MyKeyAdapter9(cubs);
        cubs.mma9 = new MyMouseAdapter9(cubs);
        cubs.mka10 = new MyKeyAdapter10(cubs);
        cubs.mma10 = new MyMouseAdapter10(cubs);
        cubs.mka11 = new MyKeyAdapter11(cubs);
        cubs.mma11 = new MyMouseAdapter11(cubs);
        cubs.mil1a = new MyItemListener1(cubs);
        cubs.mil1b = new MyItemListener1(cubs);
        cubs.mil2 = new MyItemListener2(cubs);
        cubs.mal = new MyAdjustmentListener(cubs);
        cubs.start.setBackground(new Color(0, 220, 0));
        cubs.start.setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.start.addKeyListener(cubs.mka1);
        cubs.start.addMouseListener(cubs.mma1);
        cubs.stop.setBackground(new Color(232, 0, 0));
        cubs.stop.setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.stop.addKeyListener(cubs.mka2);
        cubs.stop.addMouseListener(cubs.mma2);
        cubs.move.setBackground(new Color(64, 128, 255));
        cubs.move.setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.move.addKeyListener(cubs.mka3);
        cubs.move.addMouseListener(cubs.mma3);
        cubs.help.setBackground(new Color(240, 128, 64));
        cubs.help.setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.help.addKeyListener(cubs.mka4);
        cubs.help.addMouseListener(cubs.mma4);
        cubs.copy.setBackground(new Color(170, 170, 234));
        cubs.copy.setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.copy.addKeyListener(cubs.mka5);
        cubs.copy.addMouseListener(cubs.mma5);
        cubs.resize.setBackground(new Color(160, 112, 160));
        cubs.resize.setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.resize.addKeyListener(cubs.mka6);
        cubs.resize.addMouseListener(cubs.mma6);
        cubs.colors.setBackground(new Color(180, 125, 110));
        cubs.colors.setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.colors.addKeyListener(cubs.mka7);
        cubs.colors.addMouseListener(cubs.mma7);
        cubs.shiftC[0].setBackground(new Color(55, 166, 174));
        cubs.shiftC[0].setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.shiftC[0].addKeyListener(cubs.mka8);
        cubs.shiftC[0].addMouseListener(cubs.mma8);
        cubs.shiftC[1].setBackground(new Color(55, 166, 174));
        cubs.shiftC[1].setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.shiftC[1].addKeyListener(cubs.mka9);
        cubs.shiftC[1].addMouseListener(cubs.mma9);
        cubs.shiftC[2].setBackground(new Color(55, 166, 174));
        cubs.shiftC[2].setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.shiftC[2].addKeyListener(cubs.mka10);
        cubs.shiftC[2].addMouseListener(cubs.mma10);
        cubs.shiftC[3].setBackground(new Color(55, 166, 174));
        cubs.shiftC[3].setFont(new Font("SansSerif", Font.BOLD, 13));
        cubs.shiftC[3].addKeyListener(cubs.mka11);
        cubs.shiftC[3].addMouseListener(cubs.mma11);
        cubs.scramble1 = new Checkbox("Scramble k Values");
        cubs.scramble1.setFont(new Font("SansSerif", Font.PLAIN, 13));
        cubs.scramble2 = new Checkbox("Scramble Discs");
```

```
      cubs.scramble2.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.forward = new Checkbox("Forward", cubs.myCBG, true);
      cubs.forward.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.reverse = new Checkbox("Reverse", cubs.myCBG, false);
      cubs.reverse.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.pause = new Checkbox("Pause");
      cubs.pause.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.discNums = new Checkbox("Numbers");
      cubs.discNums.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.discNums.addItemListener(cubs.mil1a);
      cubs.round = new Checkbox("Round");
      cubs.round.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.round.addItemListener(cubs.mil1b);
      cubs.log = new Checkbox("Log", true);
      cubs.log.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.log.addItemListener(cubs.mil2);
      cubs.speed.setSize(100, 20);
      cubs.speed.setBackground(new Color(64, 224, 224));
      cubs.speed.addAdjustmentListener(cubs.mal);
      cubs.dial.setSize(25, 15);
      cubs.logBox.setEditable(false);
      cubs.logBox.setSize((int)(.4*w), h2);
      cubs.logBox.setBackground(new Color(110, 255, 110));
      cubs.logBox.setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.helpBox = new TextArea(cubs.helpMsg, 100, 100,
         TextArea.SCROLLBARS_VERTICAL_ONLY);
      cubs.helpBox.setEditable(false);
      cubs.helpBox.setBackground(new Color(160, 224, 255));
      cubs.helpBox.setFont(new Font("SansSerif", Font.PLAIN, 17));
      cubs.p1 = new MyPanel(new BorderLayout());
      cubs.p1.setSize(w, h-h2);
      cubs.p2 = new Panel(new GridLayout(4, 1));
      cubs.p2.setSize((int)(.6*w), h2);
      cubs.p3 = new Panel(new FlowLayout(FlowLayout.CENTER, 6, 12));
      cubs.p3.setSize(w, h2);
      cubs.p3.setBackground(Color.lightGray);
      GridBagLayout gbl = new GridBagLayout();
      GridBagConstraints gbc = new GridBagConstraints();
      cubs.p2a = new Panel(gbl);
      cubs.p2a.setSize((int)(.6*w), h2/3);
      cubs.p2b = new Panel(new FlowLayout(FlowLayout.CENTER, 1, 7));
      cubs.p2b.setSize((int)(.6*w), h2/3);
      cubs.p2c = new Panel(new FlowLayout(FlowLayout.CENTER, 10, 7));
      cubs.p2c.setSize((int)(.6*w), h2/3);
      cubs.p2d = new Panel(new FlowLayout(FlowLayout.CENTER, 1, 7));
      cubs.p2d.setSize((int)(.6*w), h2/3);
      cubs.p2.add(cubs.p2a);
      cubs.p2.add(cubs.p2b);
      cubs.p2.add(cubs.p2c);
      cubs.p2.add(cubs.p2d);
      Canvas[] cs = new Canvas[17];
      for (i = 0; i < 17; i++)
      {
         cs[i] = new Canvas();
         cs[i].setSize(1, 5);
      }
      gbc.gridx = 0;
      gbc.gridy = 0;
      cs[0].setSize(13, 5);
      gbl.setConstraints(cs[0], gbc);
      cubs.p2a.add(cs[0]);
      gbc.weightx = 2;
      for (i = 0; i < 4; i++)
      {
         gbc.gridx = 2*i+1;
         gbc.anchor = GridBagConstraints.EAST;
         cubs.lbls[i] = new Label(instructions[i], Label.RIGHT);
         cubs.lbls[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
         gbl.setConstraints(cubs.lbls[i], gbc);
         cubs.p2a.add(cubs.lbls[i]);
         gbc.gridx = 2*i+2;
         gbc.anchor = GridBagConstraints.WEST;
         cubs.entries[i] = new TextField(defaults[i], 3);
         cubs.entries[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
         gbl.setConstraints(cubs.entries[i], gbc);
         cubs.p2a.add(cubs.entries[i]);
      }
      cubs.lbls[4] = new Label(instructions[4], Label.RIGHT);
      cubs.lbls[4].setFont(new Font("SansSerif", Font.PLAIN, 13));
      gbc.gridx = 9;
      gbc.anchor = GridBagConstraints.EAST;
      gbl.setConstraints(cubs.lbls[4], gbc);
      cubs.p2a.add(cubs.lbls[4]);
      gbc.gridx = 10;
      gbc.weightx = 60;
      gbc.anchor = GridBagConstraints.CENTER;
      gbc.fill = GridBagConstraints.HORIZONTAL;
      gbl.setConstraints(cubs.speed, gbc);
      cubs.p2a.add(cubs.speed);
      gbc.gridx = 11;
      gbc.weightx = 3;
      gbl.setConstraints(cubs.dial, gbc);
      cubs.p2a.add(cubs.dial);
      gbc.gridx = 12;
      cs[1].setSize(7, 5);
      gbl.setConstraints(cs[1], gbc);
      cubs.p2a.add(cs[1]);
      cs[2].setSize(30, 5);
      cubs.p2b.add(cs[2]);
      cubs.p2b.add(cubs.scramble1);
      cubs.lbls[5] = new Label(instructions[5], Label.RIGHT);
      cubs.lbls[5].setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.entries[4] = new TextField(defaults[4], 3);
      cubs.entries[4].setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.p2b.add(cubs.lbls[5]);
      cubs.p2b.add(cubs.entries[4]);
      cubs.p2b.add(new Label("    "));
      cubs.p2b.add(cubs.scramble2);
      cubs.lbls[6] = new Label(instructions[6], Label.RIGHT);
      cubs.lbls[6].setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.entries[5] = new TextField(3);
      cubs.entries[5].setFont(new Font("SansSerif", Font.PLAIN, 13));
      cubs.p2b.add(cubs.lbls[6]);
      cubs.p2b.add(cubs.entries[5]);
      cs[3].setSize(25, 5);
      cubs.p2b.add(cs[3]);
      cubs.p2b.add(cubs.forward);
      cs[4].setSize(15, 5);
      cubs.p2b.add(cs[4]);
      cubs.p2b.add(cubs.reverse);
      cs[5].setSize(24, 5);
      cubs.p2b.add(cs[5]);
      cs[6].setSize(3, 5);
      cubs.p2c.add(cs[6]);
      for (i = 6; i <= 8; i++)
         cs[i].setSize(4, 5);
      cubs.p2c.add(cubs.start);
      cubs.p2c.add(cs[7]);
      cubs.p2c.add(cubs.stop);
      cubs.p2c.add(cs[8]);
      cubs.p2c.add(cubs.move);
      cubs.p2c.add(cs[9]);
      cubs.p2c.add(cubs.help);
      cs[10].setSize(7, 5);
      cubs.p2c.add(cs[10]);
      cubs.p2c.add(cubs.pause);
      cubs.p2c.add(cs[11]);
      cubs.p2c.add(cubs.discNums);
      cubs.p2c.add(cs[12]);
      cubs.p2c.add(cubs.round);
      cubs.p2c.add(cs[13]);
      cubs.p2c.add(cubs.log);
      cubs.p2c.add(cs[14]);
      cubs.p2c.add(cubs.copy);
      cs[15].setSize(5, 5);
      cubs.p2c.add(cs[15]);
      cubs.p2d.add(cubs.resize);
      for (i = 6; i <= 7; i++)
      {
         cubs.lbls[i+1] = new Label(instructions[i+1], Label.RIGHT);
         cubs.lbls[i+1].setFont(new Font("SansSerif", Font.PLAIN, 13));
         cubs.entries[i] = new TextField(defaults[i-1], 3);
         cubs.entries[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
         cubs.p2d.add(cubs.lbls[i+1]);
         cubs.p2d.add(cubs.entries[i]);
      }
      cs[16].setSize(64, 5);
      cubs.p2d.add(cs[16]);
      cubs.p2d.add(cubs.colors);
      for (i = 8; i <= 9; i++)
      {
         cubs.lbls[i+1] = new Label(instructions[i+1], Label.RIGHT);
         cubs.lbls[i+1].setFont(new Font("SansSerif", Font.PLAIN, 13));
         cubs.entries[i] = new TextField(defaults[i-1], 3);
         cubs.entries[i].setFont(new Font("SansSerif", Font.PLAIN, 13));
         cubs.p2d.add(cubs.lbls[i+1]);
         cubs.p2d.add(cubs.entries[i]);
         cubs.p2d.add(cubs.shiftC[i-6]);
         cubs.p2d.add(cubs.shiftC[i-8]);
      }
      cubs.p1.add(cubs.helpBox);
      cubs.p3.add(cubs.p2);
      cubs.p3.add(cubs.logBox);
      cubs.p3.add(new Label("   "));
      cubs.p0.add(cubs.p1, "Center");
      cubs.p0.add(cubs.p3, "South");
      cubs.pF.paintAll(cubs.pF.getGraphics());
      cubs.entries[0].requestFocus();
      System.gc();
   }
```

```java
}


class AppletImage
{
   public Spire cubs2;
   public Frame pF = new Frame("Towers of Chicago  (1300  x  1000)");
   public Graphics gx;
   public Color[] c1, c2;
   public Panel p0, p2, p2a, p2b, p2c, p2d, p3;
   public MyPanel p1;
   public Label[] lbls = new Label[11];
   public TextField[] entries = new TextField[10];
   public TextArea logBox = new TextArea(8, 55), helpBox;
   public Button start, stop, move, help, copy, resize, colors;
   public Button[] shiftC = new Button[4];
   public Scrollbar speed = new Scrollbar(Scrollbar.HORIZONTAL, 50, 10, 1, 110);
   public MyCanvas3 dial = new MyCanvas3();
   public MyCanvas4 mc = new MyCanvas4();
   public Checkbox scramble1, scramble2, forward, reverse, pause, discNums,
      round, log;
   public CheckboxGroup myCBG = new CheckboxGroup();
   public MyKeyAdapter1 mka1;
   public MyMouseAdapter1 mma1;
   public MyKeyAdapter2 mka2;
   public MyMouseAdapter2 mma2;
   public MyKeyAdapter3 mka3;
   public MyMouseAdapter3 mma3;
   public MyKeyAdapter4 mka4;
   public MyMouseAdapter4 mma4;
   public MyKeyAdapter5 mka5;
   public MyMouseAdapter5 mma5;
   public MyKeyAdapter6 mka6;
   public MyMouseAdapter6 mma6;
   public MyKeyAdapter7 mka7;
   public MyMouseAdapter7 mma7;
   public MyKeyAdapter8 mka8;
   public MyMouseAdapter8 mma8;
   public MyKeyAdapter9 mka9;
   public MyMouseAdapter9 mma9;
   public MyKeyAdapter10 mka10;
   public MyMouseAdapter10 mma10;
   public MyKeyAdapter11 mka11;
   public MyMouseAdapter11 mma11;
   public MyItemListener1 mil1a, mil1b;
   public MyItemListener2 mil2;
   public MyAdjustmentListener mal;
   public MyComponentAdapter mca;
   public MyWindowAdapter mwa;
   public GridBagLayout gbl0 = new GridBagLayout();
   public GridBagConstraints gbc0 = new GridBagConstraints();
   public int moveCountA, colorX, colorY, colorPos1, colorPos2,
      oldColorX, oldColorY, oldDiscs, oldPegs, dn = 1;
   public boolean running, scrambleK, scrambleD;
   public int[] valuesIndex;
   public long[] powArray = new long[63];
   public byte[][] moves;
   public int[][] values, optimalsIndex, optOrderIndex;
   public long[][] moveArray = new long[1000][100];
   public boolean[][] doneYet = new boolean[1000][100];
   public int[][][] optimals = new int[1000][100][1], optOrder;
   public String moveCountB = "", logger = "", preLog = "", text = "", helpMsg =
      "Robert Swartz\nwww.mathapplets.net\nTowers of Chicago\nCopyright 2024"+
      "\n\nThis program solves the Towers of Chicago puzzle.  In this problem,"+
      " a set of discs are to be moved from one peg to another while observing"+
      " the following 4 rules:\n(1) Only one disc can be moved at a time, "+
      "(2) A larger disc can't be placed on top of a smaller one, (3) Auxiliary"
      +" pegs may be used for the temporary placement of the discs, and (4) The"
      +" task should be completed in a minimum number of moves.\n\nThis program"
      +" can display up to 50 discs by 10 pegs with a default window size of "
      +"1300  x  1000.  It can also print the moves for up to 1000 discs by "+
      "100 pegs.  The applet can handle up to 20 rotating colors; the initial"+
      " and final pegs are displayed in green and red, respectively.  "+
      "The discs are displayed either round or sharp.  When the applet is "+
      "paused, the move button can be used to step through the moves.\n\n"+
      "The traditional version of the Towers of Chicago makes use of only 3 "+
      "pegs:  the initial peg, the final peg, and 1 auxiliary peg.  However, "
      +"the multipeg version makes use of any number of pegs:  the initial peg,"
      +" the final peg, and 2 or more auxiliary pegs.  As the number of pegs is"
      +" increased, while keeping the number of discs constant, the number of "+
      "moves required usually decreases.\n\nThe optimal solution for the 3 peg "
      +"problem is well known:  recursively move n-1 discs from the initial peg"
      +" to the auxiliary peg, then move the remaining disc to the final peg, "+
      "then move the first n-1 discs from the auxiliary peg to the final peg.  "
      +"This algorithm results in 2^n-1 moves.\n\nThe presumed optimal "+
      "Frame-Stewart algorithm that this program uses for n discs and p pegs is"
      +" as follows:  recursively move k discs from the initial peg to an "+
      "available auxiliary peg using the p peg subalgorithm, then move the "+
      "remaining n-k discs from the initial peg to the final peg using the p-1"+
      " peg subalgorithm, then move the first k discs from the auxiliary peg to"
      +" the final peg using the p peg subalgorithm.  The program uses dynamic "
      +"programming to find the k values.  The k values are calculated to make "
      +"the number of moves minimal.\n\nThis program can find different optimal"
      +" solutions for each combination of discs and pegs (where pegs are "+
      "greater than 3).  This is accomplished through 2 kinds of randomization,"
      +" Scramble k Values and Scramble Discs.  Each of these uses a random "
      +"number seed (an integer from 000 to 999).  Scramble k Values uses a"
      +" different k value for each recursive instance of a particular "+
      "subalgorithm; if this isn't selected, the Random k Seed sets constant k "
      +"values for each subalgorithm.  Scramble Discs simply shuffles the discs"
      +" around the auxiliary pegs.";

   public Color[] colorScheme1(int discs, int num, int shift)
   {
      Color[] cx = new Color[discs];
      for (int i = 0; i < discs; i++)
         cx[i] = colorScheme3(i, num, shift, colorY);
      return cx;
   }

   public Color[] colorScheme2(int pegs, int init, int fnl, int num, int shift)
   {
      Color[] cx = new Color[pegs];
      int i, j;
      for (i = 0; i < pegs; i++)
      {
         if (i == init-1)
            cx[i] = new Color(0, 220, 0);
         else if (i == fnl-1)
            cx[i] = new Color(232, 0, 0);
```

```java
            else
            {
               if (i < Math.min(init-1, fnl-1))
                  j = i;
               else if (i < Math.max(init-1, fnl-1))
                  j = i-1;
               else
                  j = i-2;
               cx[i] = colorScheme3(j, num, shift, colorX);
            }
         }
         return cx;
      }

      public Color colorScheme3(int pos, int num, int shift, int colorcount)
      {
         Color c;
         switch ((pos%colorcount+shift)%num)
         {
            case 0 ->
               c = new Color(48, 156, 172);
            case 1 ->
               c = new Color(176, 96, 80);
            case 2 ->
               c = new Color(150, 64, 192);
            case 3 ->
               c = new Color(96, 120, 144);
            case 4 ->
               c = new Color(216, 96, 96);
            case 5 ->
               c = new Color(128, 96, 48);
            case 6 ->
               c = new Color(64, 96, 176);
            case 7 ->
               c = new Color(160, 0, 128);
            case 8 ->
               c = new Color(96, 176, 208);
            case 9 ->
               c = new Color(108, 96, 176);
            case 10 ->
               c = new Color(128, 0, 0);
            case 11 ->
               c = new Color(64, 128, 128);
            case 12 ->
               c = new Color(172, 136, 64);
            case 13 ->
               c = new Color(0, 128, 192);
            case 14 ->
               c = new Color(208, 0, 208);
            case 15 ->
               c = new Color(128, 0, 208);
            case 16 ->
               c = new Color(128, 128, 0);
            case 17 ->
               c = new Color(208, 128, 128);
            case 18 ->
               c = new Color(128, 0, 128);
            default ->
               c = new Color(128, 64, 64);
         }
```

```java
        return c;
    }
}


class Spire extends Thread
{
    public AppletImage cubs;
    public int discs, pegs, init, fnl, seed1, seed2, time, count,
        pointer = -1, direction = 1;
    public boolean doSleep, error, sysError, tooMany;
    public MyCanvas gameBoard;
    public MyCanvas2 moveCanvas;
    public Random rg;
    public Vector nums = new Vector(100);
    public MyExceptionA myE_A = new MyExceptionA();
    public MyExceptionB myE_B = new MyExceptionB();

    public Spire(AppletImage a, int n, int p, int i, int f, int s1, int s2)
    {
        cubs = a;
        discs = n;
        pegs = p;
        init = i;
        fnl = f;
        seed1 = s1;
        seed2 = s2;
    }

    public void run()
    {
        cubs.running = true;
        cubs.p1.removeAll();
        cubs.logBox.setText("\n\n\n    BUSY...");
        cubs.text = " ";
        cubs.mka3.cubs2 = this;
        cubs.mma3.cubs2 = this;
        timeage();
        System.gc();
        int i, j, a, r;
        int[] aux = new int[pegs-2], aux2 = new int[pegs-2], order = new int[pegs
            -2];
        if ((discs != cubs.oldDiscs) || (cubs.colorY != cubs.oldColorY))
        {
            cubs.colorPos1 = cubs.colorY*1000000+7;
            cubs.c1 = cubs.colorScheme1(discs, cubs.colorY, cubs.colorPos1);
        }
        if ((pegs != cubs.oldPegs) || (cubs.colorX != cubs.oldColorX))
        {
            cubs.colorPos2 = cubs.colorX*1000000;
            cubs.c2 = cubs.colorScheme2(pegs, init, fnl, cubs.colorX,
                cubs.colorPos2);
        }
        cubs.oldDiscs = discs;
        cubs.oldPegs = pegs;
        cubs.oldColorY = cubs.colorY;
        cubs.oldColorX = cubs.colorX;
        if (pegs >= 4)
        {
            try
            {
                cubs.optOrder = new int[discs][pegs][1];
                cubs.optOrderIndex = new int[discs][pegs];
                cubs.optimalsIndex = new int[discs][pegs];
                System.gc();
                optimize();
                nums.clear();
                rg = new Random(seed1);
                for (i = 3; i < discs; i++)
                   for (j = 3; j < pegs; j++)
                      if (i >= j)
                      {
                         if (!cubs.running)
                            throw myE_B;
                         if (cubs.scrambleK)
                         {
                            cubs.optOrder[i][j] = new int[cubs.optimals[i][j].
                               length];
                            for (a = 0; a < cubs.optimals[i][j].length; a++)
                               nums.add(a);
                            for (a = 0; a < cubs.optimals[i][j].length; a++)
                            {
                               r = myRandom(nums.size());
                               cubs.optOrder[i][j][a] = (int)nums.get(r);
                               nums.remove(r);
                            }
                         }
                         else
                            cubs.optimalsIndex[i][j] = myRandom(cubs.optimals[i][j].
                               length);
                      }
            }
            catch (MyExceptionB e1)
            {
                cubs.logBox.setText("");
            }
        }
        if (cubs.running)
        {
            if (pegs == 3)
            {
                if (discs <= 30)
                {
                   cubs.moveCountA = (int)cubs.powArray[discs-1];
                   cubs.moveCountB = ""+cubs.powArray[discs-1];
                }
                else if (discs <= 63)
                {
                   cubs.moveCountA = Integer.MAX_VALUE-10;
                   cubs.moveCountB = ""+cubs.powArray[discs-1];
                }
                else
                {
                   cubs.moveCountA = Integer.MAX_VALUE-10;
                   cubs.moveCountB = ""+Math.pow(2, discs);
                }
            }
            else
            {
                if (cubs.moveArray[discs-1][pegs-1] <= Integer.MAX_VALUE-10)
```

```
            {
               cubs.moveCountA = (int)cubs.moveArray[discs-1][pegs-1];
               cubs.moveCountB = ""+cubs.moveArray[discs-1][pegs-1];
            }
            else
            {
               cubs.moveCountA = Integer.MAX_VALUE-10;
               cubs.moveCountB = ""+cubs.moveArray[discs-1][pegs-1];
            }
         }
         cubs.logger = "     "+discs+" Discs  x  "+pegs+" Pegs\n    "
            +cubs.moveCountB+" Total Moves\n";
         cubs.preLog = cubs.logger;
         for (i = 0; i < 10; i++)
         {
            sysError = false;
            try
            {
               cubs.moves = new byte[2][cubs.moveCountA];
            }
            catch (java.lang.OutOfMemoryError e2)
            {
               sysError = true;
            }
            if (!sysError)
               break;
         }
         if (!cubs.running)
            cubs.logBox.setText("");
         else
         {
            j = 0;
            for (i = 1; i <= pegs; i++)
               if (i != init && i != fnl)
               {
                  aux[j] = i;
                  j++;
               }
            if (cubs.scrambleD)
            {
               rg = new Random(seed2);
               for (i = 0; i < aux.length; i++)
                  nums.add(i);
               for (i = 0; i < aux.length; i++)
               {
                  r = myRandom(nums.size());
                  order[i] = (int)nums.get(r);
                  nums.remove(r);
               }
            }
            else
               for (i = 0; i < aux.length; i++)
                  order[i] = i;
            for (i = 0; i < aux.length; i++)
               aux2[i] = aux[order[i]];
            cubs.valuesIndex = new int[pegs];
            cubs.values = new int[discs][pegs];
            for (i = 0; i < pegs; i++)
               cubs.valuesIndex[i] = discs;
            cubs.valuesIndex[init-1] = 0;
```

```java
      for (i = 0; i < discs; i++)
         cubs.values[i][init-1] = i+1;
      drawBoard();
      if (!tooMany)
      {
         cubs.gx = gameBoard.getGraphics();
         gameBoard.paint0(cubs.gx);
      }
      else
      {
         cubs.gx = cubs.mc.getGraphics();
         cubs.mc.paint(cubs.gx);
      }
      if (cubs.running)
      {
         cubs.pF.paintAll(cubs.pF.getGraphics());
         cubs.logBox.setText(cubs.logger+"\n    BUSY...");
         try
         {
            chicago(discs, init, fnl, aux2);
         }
         catch (MyExceptionA e4){}
         catch (MyExceptionB e5)
         {
            cubs.logBox.setText(cubs.logger);
            error = true;
         }
         if (!error)
         {
            try
            {
               sleep(650);
            }
            catch (InterruptedException e6){}
            cubs.logBox.setText(cubs.logger);
            while (pointer < cubs.moveCountA-1)
            {
               try
               {
                  myMove2();
               }
               catch (MyExceptionB e7)
               {
                  break;
               }
               catch (ArrayIndexOutOfBoundsException e8)
               {
                  break;
               }
            }
         }
      }
   }
}
if (cubs.running && pointer >= 0)
{
   try
   {
      sleep(Math.max(time, 650));
   }
```

```
         catch (InterruptedException e9){}
         moveCanvas.setText3(discs, pegs, cubs.moveCountA);
      }
      if (cubs.running && (pointer == cubs.moveCountA-1 || pointer == -1))
      {
         cubs.logger+="\n    DONE";
         cubs.preLog+="\n    DONE";
         if (cubs.log.getState())
            cubs.logBox.append("\n    DONE");
         else
            cubs.logBox.setText(cubs.preLog);
      }
      cubs.moves = null;
      cubs.optOrder = null;
      System.gc();
      cubs.running = false;
   }

   public void drawBoard()
   {
      int i, w2, h2, w3, w = cubs.pF.getSize().width-30, h =
         cubs.pF.getSize().height-250, h_mod = 1;
      if (pegs == 3)
         w2 = (int)(.6*w);
      else if (pegs <= 5)
         w2 = (int)(.7*w);
      else if (pegs <= 7)
         w2 = (int)(.78*w);
      else if (pegs <= 9)
         w2 = (int)(.85*w);
      else if (pegs <= 12)
         w2 = (int)(.91*w);
      else
         w2 = (int)(.88*w);
      w = (w/pegs-w2/pegs)*(pegs-1)+w2;
      if (discs <= 5)
         h2 = (int)(.5*h);
      else if (discs <= 10)
         h2 = (int)(.6*h);
      else if (discs <= 15)
         h2 = (int)(.7*h);
      else if (discs <= 20)
         h2 = (int)(.75*h);
      else if (discs <= 25)
         h2 = (int)(.8*h);
      else if (discs <= 32)
         h2 = (int)(.88*h);
      else if (discs <= 38)
         h2 = (int)(.9*h);
      else
         h2 = (int)(.92*h);
      w3 = (w2/pegs-(w2/pegs/2)%discs)/discs;
      if (w3*discs >= w2/pegs-10)
         w3 = (w2/pegs-10)/discs-1;
      if (w3%2 == 1)
         w3--;
      cubs.p1.setLayout(cubs.gbl0);
      cubs.p1.myDraw(cubs.p1.getGraphics());
      if (w3 == 0 || (h2-21)/discs < 3)
      {
         tooMany = true;
         cubs.mc.setSize(w, h);
         cubs.gbc0.anchor = GridBagConstraints.CENTER;
         cubs.gbc0.ipadx = 0;
         cubs.gbc0.gridy = 0;
         cubs.gbl0.setConstraints(cubs.mc, cubs.gbc0);
         cubs.p1.add(cubs.mc);
         cubs.gbc0.anchor = GridBagConstraints.SOUTH;
         cubs.gbc0.gridy = 10;
         moveCanvas = new MyCanvas2(cubs, this);
         moveCanvas.setSize(w-50, 21);
         cubs.gbl0.setConstraints(moveCanvas, cubs.gbc0);
         cubs.p1.add(moveCanvas);
         cubs.mc.setText2("Not Enough Room to Display Discs and Pegs!!!");
         moveCanvas.setText0();
         moveCanvas.paint0(moveCanvas.getGraphics());
      }
      else
      {
         if (discs < 16)
            h_mod = 3;
         gameBoard = new MyCanvas(cubs, discs, pegs, w3, (h2-21)/discs-((h2-21)/
            discs)%h_mod, w2/pegs+6, w/pegs-w2/pegs-7, (h-21-((h2-21)/discs)*
            discs)/2, -(w/pegs-w2/pegs-7)/3);
         gameBoard.setSize(w, h);
         cubs.gbc0.anchor = GridBagConstraints.CENTER;
         cubs.gbc0.ipadx = 0;
         cubs.gbc0.gridy = 0;
         cubs.gbl0.setConstraints(gameBoard, cubs.gbc0);
         cubs.p1.add(gameBoard);
         cubs.gbc0.anchor = GridBagConstraints.SOUTH;
         cubs.gbc0.gridy = 10;
         moveCanvas = new MyCanvas2(cubs, this);
         moveCanvas.setSize(w-50, 21);
         cubs.gbl0.setConstraints(moveCanvas, cubs.gbc0);
         cubs.p1.add(moveCanvas);
         gameBoard.myDraw0();
         moveCanvas.setText0();
         moveCanvas.paint0(moveCanvas.getGraphics());
      }
   }

   public void optimize()throws MyExceptionB
   {
      int n, p, k, i;
      long count2, count3;
      for (n = 1; n <= discs; n++)
         for (p = 4; p <= pegs; p++)
            if (!cubs.doneYet[n-1][p-1])
            {
               cubs.doneYet[n-1][p-1] = true;
               nums.clear();
               count2 = Long.MAX_VALUE;
               k = 0;
               if (p == 4 && n > 61)
                  k = n-61;
               for (; k < n; k++)
               {
                  if (!cubs.running)
                     throw myE_B;
```

```
               count3 = 0;
               if (k > 0)
               {
                  if (k >= p)
                     count3 = 2*cubs.moveArray[k-1][p-1];
                  else
                     count3 = 4*k-2;
               }
               if (p > 4)
               {
                  if (n-k > p-2)
                     count3+=cubs.moveArray[n-k-1][p-2];
                  else
                     count3+=(n-k)*2-1;
               }
               else
                  count3+=cubs.powArray[n-k-1];
               if (count3 < count2)
               {
                  count2 = count3;
                  nums.clear();
                  nums.add(k);
               }
               else if (count3 == count2)
                  nums.add(k);
            }
            cubs.moveArray[n-1][p-1] = count2;
            cubs.optimals[n-1][p-1] = new int[nums.size()];
            for (i = 0; i < nums.size(); i++)
               cubs.optimals[n-1][p-1][i] = (int)nums.get(i);
         }
}

public void chicago(int discs2, int init2, int fnl2, int[] aux)throws
   MyExceptionA, MyExceptionB
{
   if (!cubs.running)
      throw myE_B;
   int k, i, r;
   int[] AUX1, AUX2, order = new int[aux.length];
   if (discs2 < aux.length+2)
   {
      if (cubs.scrambleD)
      {
         for (i = 0; i < aux.length; i++)
            nums.add(i);
         for (i = 0; i < aux.length; i++)
         {
            r = myRandom(nums.size());
            order[i] = (int)nums.get(r);
            nums.remove(r);
         }
      }
      else
         for (i = 0; i < aux.length; i++)
            order[i] = i;
      for (i = 0; i < discs2-1; i++)
         myMove(init2, aux[order[i]]);
      myMove(init2, fnl2);
      for (i = discs2-2; i >= 0; i--)
```

```
            myMove(aux[order[i]], fnl2);
      }
      else if (aux.length > 1)
      {
         if (cubs.scrambleK)
         {
            k = cubs.optimals[discs2-1][aux.length+1][cubs.optOrder[discs2
               -1][aux.length+1][cubs.optOrderIndex[discs2-1][aux.length+1]]];
            cubs.optOrderIndex[discs2-1][aux.length+1]++;
            cubs.optOrderIndex[discs2-1][aux.length+1]%=cubs.optimals[discs2
               -1][aux.length+1].length;
         }
         else
            k = cubs.optimals[discs2-1][aux.length+1][cubs.optimalsIndex[discs2
               -1][aux.length+1]];
         if (cubs.scrambleD)
         {
            for (i = 0; i < aux.length-1; i++)
               nums.add(i);
            for (i = 0; i < aux.length-1; i++)
            {
               r = myRandom(nums.size());
               order[i] = (int)nums.get(r);
               nums.remove(r);
            }
         }
         else
            for (i = 0; i < aux.length-1; i++)
               order[i] = i;
         AUX2 = new int[aux.length];
         for (i = 0; i < aux.length-1; i++)
            AUX2[i] = aux[order[i]+1];
         AUX2[aux.length-1] = fnl2;
         chicago(k, init2, aux[0], AUX2);
         if (cubs.scrambleD)
         {
            for (i = 0; i < aux.length-1; i++)
               nums.add(i);
            for (i = 0; i < aux.length-1; i++)
            {
               r = myRandom(nums.size());
               order[i] = (int)nums.get(r);
               nums.remove(r);
            }
         }
         AUX1 = new int[aux.length-1];
         for (i = 0; i < aux.length-1; i++)
            AUX1[i] = aux[order[i]+1];
         chicago(discs2-k, init2, fnl2, AUX1);
         if (cubs.scrambleD)
         {
            for (i = 0; i < aux.length-1; i++)
               nums.add(i);
            for (i = 0; i < aux.length-1; i++)
            {
               r = myRandom(nums.size());
               order[i] = (int)nums.get(r);
               nums.remove(r);
            }
         }
         for (i = 0; i < aux.length-1; i++)
            AUX2[i] = aux[order[i]+1];
         AUX2[aux.length-1] = init2;
         chicago(k, aux[0], fnl2, AUX2);
      }
      else
      {
         AUX1 = new int[1];
         if (discs2 > 1)
         {
            AUX1[0] = fnl2;
            chicago(discs2-1, init2, aux[0], AUX1);
         }
         myMove(init2, fnl2);
         if (discs2 > 1)
         {
            AUX1[0] = init2;
            chicago(discs2-1, aux[0], fnl2, AUX1);
         }
      }
   }

   public void myMove(int init2, int fnl2)throws MyExceptionA
   {
      if (count < cubs.moveCountA)
      {
         cubs.moves[0][count] = (byte)init2;
         cubs.moves[1][count++] = (byte)fnl2;
      }
      else
         throw myE_A;
   }

   public void myMove2()throws MyExceptionB
   {
      int num, init2, fnl2;
      String logAppend, minus = "";
      if (!cubs.pause.getState() && cubs.running)
      {
         try
         {
            sleep(time);
         }
         catch (InterruptedException e1){}
      }
      if (!cubs.running)
         throw myE_B;
      doSleep = true;
      while (cubs.pause.getState() && doSleep)
      {
         if (!cubs.running)
            throw myE_B;
         try
         {
            sleep(35);
         }
         catch (InterruptedException e2){}
      }
      if (cubs.forward.getState())
      {
         if (direction == 1)
            pointer++;
         init2 = cubs.moves[0][pointer];
         fnl2 = cubs.moves[1][pointer];
         direction = 1;
      }
      else
      {
         if (direction == 2)
            pointer--;
         init2 = cubs.moves[1][pointer];
         fnl2 = cubs.moves[0][pointer];
         direction = 2;
      }
      num = cubs.values[cubs.valuesIndex[init2-1]][init2-1];
      if (!tooMany)
         gameBoard.myDraw(0, cubs.valuesIndex[init2-1], init2-1, cubs.gx);
      cubs.valuesIndex[init2-1]++;
      cubs.values[cubs.valuesIndex[fnl2-1]-1][fnl2-1] = num;
      if (!tooMany)
         gameBoard.myDraw(num, cubs.valuesIndex[fnl2-1]-1, fnl2-1, cubs.gx);
      if (direction == 2)
         minus = "-";
      moveCanvas.dn = direction;
      cubs.dn = direction;
      cubs.text = "move "+(pointer+1)+minus+" of "+cubs.moveCountB+
      " total moves               disc "+num+" was moved from peg "+init2+
      " to peg "+fnl2;
      moveCanvas.setText2(cubs.text, direction);
      moveCanvas.paint0(moveCanvas.getGraphics());
      cubs.valuesIndex[fnl2-1]--;
      if (direction == 2)
         minus = "-";
      logAppend = "\n     move "+(pointer+1)+minus+"\t   disc "+num+
         " was moved from peg "+init2+" to peg "+fnl2;
      if (cubs.log.getState())
         cubs.logBox.append(logAppend);
      cubs.logger+=logAppend;
   }

   public void timeage()
   {
      int speedDial = cubs.speed.getValue();
      if (speedDial <= 30)
         time = 1025-10*speedDial;
      else if (speedDial <= 50)
         time = 1025-(int)(10*speedDial*.99);
      else if (speedDial <= 65)
         time = 1000-(int)(10*speedDial*.985);
      else if (speedDial <= 75)
         time = 975-(int)(10*speedDial*.98);
      else if (speedDial <= 82)
         time = 950-(int)(10*speedDial*.95);
      else if (speedDial <= 88)
         time = 900-(int)(10*speedDial*.85);
      else if (speedDial <= 93)
         time = 750-(int)(10*speedDial*.75);
      else if (speedDial <= 96)
         time = 675-(int)(10*speedDial*.65);
      else if (speedDial <= 98)
```

```java
         time = 580-(int)(10*speedDial*.55);
      else
         time = 500-(int)(10*speedDial*.48);
   }

   public int myRandom(int high)
   {
      long sum = 0, base = 1;
      byte[] b = new byte[64];
      rg.nextBytes(b);
      for (int i = 0; i <= 3; i++)
      {
         sum+=(b[i]+128)*base;
         base*=256;
      }
      return (int)(sum%high);
   }
}


class MyExceptionA extends Exception{}

class MyExceptionB extends Exception{}


class MyKeyAdapter1 extends KeyAdapter
{
   public AppletImage cubs;
   public int discs, pegs, init, fnl, seed1, seed2;

   public MyKeyAdapter1(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      boolean error1 = false, error2 = false, error3 = false;
      if ((e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE) && !cubs.running)
      {
         if (!cubs.entries[3].getText().toLowerCase().equals("max"))
         {
            try
            {
               discs = Integer.parseInt(cubs.entries[0].getText());
               pegs = Integer.parseInt(cubs.entries[1].getText());
               init = Integer.parseInt(cubs.entries[2].getText());
               fnl = Integer.parseInt(cubs.entries[3].getText());
            }
            catch (NumberFormatException e1)
            {
               error1 = true;
            }
         }
         else
         {
            try
            {
               discs = Integer.parseInt(cubs.entries[0].getText());
```

```
                  pegs = Integer.parseInt(cubs.entries[1].getText());
                  init = Integer.parseInt(cubs.entries[2].getText());
               }
               catch (NumberFormatException e2)
               {
                  error1 = true;
               }
               if (!error1)
                  fnl = pegs;
            }
            cubs.scrambleK = cubs.scramble1.getState();
            cubs.scrambleD = cubs.scramble2.getState();
            try
            {
               seed1 = Integer.parseInt(cubs.entries[4].getText());
            }
            catch (NumberFormatException e3)
            {
               error2 = true;
            }
            finally
            {
               if (seed1 < 0 || seed1 > 999)
                  error2 = true;
            }
            try
            {
               seed2 = Integer.parseInt(cubs.entries[5].getText());
            }
            catch (NumberFormatException e4)
            {
               if (cubs.scrambleD)
                  error3 = true;
            }
            finally
            {
               if (cubs.scrambleD && (seed2 < 0 || seed2 > 999))
                  error3 = true;
            }
            try
            {
               cubs.colorY = Integer.parseInt(cubs.entries[8].getText());
               cubs.colorX = Integer.parseInt(cubs.entries[9].getText());
            }
            catch (NumberFormatException e5)
            {
               error1 = true;
            }
            if (error1 || discs < 3 || pegs < 3 || discs > 1000 || pegs > 100 ||
               init < 1 || fnl < 1 || init == fnl || init > pegs || fnl > pegs ||
               cubs.colorX < 1 || cubs.colorX > 20 || cubs.colorY < 2 ||
               cubs.colorY > 20 || (pegs > 3 && (error2 || error3)))
               cubs.logBox.setText("\n\n\n    INVALID ENTRY!!!");
            else
            {
               cubs.cubs2 = new Spire(cubs, discs, pegs, init, fnl, seed1, seed2);
               cubs.cubs2.start();
            }
         }
      }
```

```java
}


class MyMouseAdapter1 extends MouseAdapter
{
   public AppletImage cubs;
   public int discs, pegs, init, fnl, seed1, seed2;

   public MyMouseAdapter1(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      boolean error1 = false, error2 = false, error3 = false;
      if (e.getButton() == MouseEvent.BUTTON1 && !cubs.running)
      {
         if (!cubs.entries[3].getText().toLowerCase().equals("max"))
         {
            try
            {
               discs = Integer.parseInt(cubs.entries[0].getText());
               pegs = Integer.parseInt(cubs.entries[1].getText());
               init = Integer.parseInt(cubs.entries[2].getText());
               fnl = Integer.parseInt(cubs.entries[3].getText());
            }
            catch (NumberFormatException e1)
            {
               error1 = true;
            }
         }
         else
         {
            try
            {
               discs = Integer.parseInt(cubs.entries[0].getText());
               pegs = Integer.parseInt(cubs.entries[1].getText());
               init = Integer.parseInt(cubs.entries[2].getText());
            }
            catch (NumberFormatException e2)
            {
               error1 = true;
            }
            if (!error1)
               fnl = pegs;
         }
         cubs.scrambleK = cubs.scramble1.getState();
         cubs.scrambleD = cubs.scramble2.getState();
         try
         {
            seed1 = Integer.parseInt(cubs.entries[4].getText());
         }
         catch (NumberFormatException e3)
         {
            error2 = true;
         }
         finally
         {
            if (seed1 < 0 || seed1 > 999)
```

```java
                 error2 = true;
          }
          try
          {
             seed2 = Integer.parseInt(cubs.entries[5].getText());
          }
          catch (NumberFormatException e4)
          {
             if (cubs.scrambleD)
                error3 = true;
          }
          finally
          {
             if (cubs.scrambleD && (seed2 < 0 || seed2 > 999))
                error3 = true;
          }
          try
          {
             cubs.colorY = Integer.parseInt(cubs.entries[8].getText());
             cubs.colorX = Integer.parseInt(cubs.entries[9].getText());
          }
          catch (NumberFormatException e5)
          {
             error1 = true;
          }
          if (error1 || discs < 3 || pegs < 3 || discs > 1000 || pegs > 100 ||
             init < 1 || fnl < 1 || init == fnl || init > pegs || fnl > pegs ||
             cubs.colorX < 1 || cubs.colorX > 20 || cubs.colorY < 2 ||
             cubs.colorY > 20 || (pegs > 3 && (error2 || error3)))
             cubs.logBox.setText("\n\n\n    INVALID ENTRY!!!");
          else
          {
             cubs.cubs2 = new Spire(cubs, discs, pegs, init, fnl, seed1, seed2);
             cubs.cubs2.start();
          }
       }
    }
}


class MyKeyAdapter2 extends KeyAdapter
{
   public AppletImage cubs;

   public MyKeyAdapter2(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE)
      {
         cubs.running = false;
         cubs.moves = null;
         cubs.optOrder = null;
         cubs.p1.removeAll();
         cubs.cubs2 = null;
         System.gc();
      }
   }
}


class MyMouseAdapter2 extends MouseAdapter
{
   public AppletImage cubs;

   public MyMouseAdapter2(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1)
      {
         cubs.running = false;
         cubs.moves = null;
         cubs.optOrder = null;
         cubs.p1.removeAll();
         cubs.cubs2 = null;
         System.gc();
      }
   }
}


class MyKeyAdapter3 extends KeyAdapter
{
   public Spire cubs2;

   public void keyPressed(KeyEvent e)
   {
      if ((e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE) && cubs2 != null)
         cubs2.doSleep = false;
   }
}


class MyMouseAdapter3 extends MouseAdapter
{
   public Spire cubs2;

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1 && cubs2 != null)
         cubs2.doSleep = false;
   }
}


class MyKeyAdapter4 extends KeyAdapter
{
   public AppletImage cubs;

   public MyKeyAdapter4(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      if ((e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE) && !cubs.running)
      {
         cubs.p1.removeAll();
         cubs.p1.setLayout(new BorderLayout());
         cubs.p1.add(cubs.helpBox);
         cubs.pF.paintAll(cubs.pF.getGraphics());
      }
   }
}


class MyMouseAdapter4 extends MouseAdapter
{
   public AppletImage cubs;

   public MyMouseAdapter4(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1 && !cubs.running)
      {
         cubs.p1.removeAll();
         cubs.p1.setLayout(new BorderLayout());
         cubs.p1.add(cubs.helpBox);
         cubs.pF.paintAll(cubs.pF.getGraphics());
      }
   }
}


class MyKeyAdapter5 extends KeyAdapter
{
   public AppletImage cubs;

   public MyKeyAdapter5(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE)
         Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new
            StringSelection(cubs.logger), new StringSelection(cubs.logger));
   }
}


class MyMouseAdapter5 extends MouseAdapter
{
   public AppletImage cubs;

   public MyMouseAdapter5(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1)
         Toolkit.getDefaultToolkit().getSystemClipboard().setContents(new
            StringSelection(cubs.logger), new StringSelection(cubs.logger));
   }
}


class MyKeyAdapter6 extends KeyAdapter
{
   public AppletImage cubs;
   public int X, Y;

   public MyKeyAdapter6(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyTyped(KeyEvent e)
   {
      if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE)
      {
         try
         {
            X = Integer.parseInt(cubs.entries[6].getText());
            Y = Integer.parseInt(cubs.entries[7].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            cubs.pF.setSize(X, Y);
            if (cubs.cubs2 != null)
            {
                  cubs.p1.removeAll();
                  cubs.cubs2.drawBoard();
                  if (!cubs.cubs2.tooMany)
                  cubs.gx = cubs.cubs2.gameBoard.getGraphics();
                  cubs.pF.paintAll(cubs.pF.getGraphics());
            }
         }
      }
   }
}


class MyMouseAdapter6 extends MouseAdapter
{
   public AppletImage cubs;
   public int X, Y;

   public MyMouseAdapter6(AppletImage ts)
```

```
    {
       cubs = ts;
    }

    public void mousePressed(MouseEvent e)
    {
       if (e.getButton() == MouseEvent.BUTTON1)
       {
          try
          {
             X = Integer.parseInt(cubs.entries[6].getText());
             Y = Integer.parseInt(cubs.entries[7].getText());
          }
          catch (NumberFormatException e2) {}
          finally
          {
             cubs.pF.setSize(X, Y);
             if (cubs.cubs2 != null)
             {
                   cubs.p1.removeAll();
                   cubs.cubs2.drawBoard();
                   if (!cubs.cubs2.tooMany)
                   cubs.gx = cubs.cubs2.gameBoard.getGraphics();
                   cubs.pF.paintAll(cubs.pF.getGraphics());
             }
          }
       }
    }
}


class MyKeyAdapter7 extends KeyAdapter
{
    public AppletImage cubs;
    public int Y, X;

    public MyKeyAdapter7(AppletImage ts)
    {
       cubs = ts;
    }

    public void keyTyped(KeyEvent e)
    {
       if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
          KeyEvent.VK_SPACE)
       {
          try
          {
             Y = Integer.parseInt(cubs.entries[8].getText());
             X = Integer.parseInt(cubs.entries[9].getText());
          }
          catch (NumberFormatException e2) {}
          finally
          {
             if (cubs.cubs2 != null)
                if (!cubs.cubs2.tooMany)
                {
                   if (Y >= 2 && Y <= 20 && X >= 1 && X <= 20)
                   {
                      cubs.colorY = Y;
```

```java
                     cubs.colorX = X;
                     cubs.oldColorY = Y;
                     cubs.oldColorX = X;
                     cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
                        colorPos1);
                     cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
                        init, cubs.cubs2.fnl, X, cubs.colorPos2);
                     cubs.cubs2.gameBoard.update(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyMouseAdapter7 extends MouseAdapter
{
   public AppletImage cubs;
   public int Y, X;

   public MyMouseAdapter7(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1)
      {
         try
         {
            Y = Integer.parseInt(cubs.entries[8].getText());
            X = Integer.parseInt(cubs.entries[9].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (Y >= 2 && Y <= 20 && X >= 1 && X <= 20)
                  {
                     cubs.colorY = Y;
                     cubs.colorX = X;
                     cubs.oldColorY = Y;
                     cubs.oldColorX = X;
                     cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
                        colorPos1);
                     cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
                        init, cubs.cubs2.fnl, X, cubs.colorPos2);
                     cubs.cubs2.gameBoard.update(cubs.gx);
                  }
               }
         }
      }
   }
}


```

```java
class MyKeyAdapter8 extends KeyAdapter
{
   public AppletImage cubs;
   public int Y;

   public MyKeyAdapter8(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE)
      {
         try
         {
            Y = Integer.parseInt(cubs.entries[8].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (Y >= 2 && Y <= 20)
                  {
                     cubs.colorY = Y;
                     cubs.colorPos1++;
                     cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
                        colorPos1);
                     cubs.cubs2.gameBoard.paint0(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyMouseAdapter8 extends MouseAdapter
{
   public AppletImage cubs;
   public int Y;

   public MyMouseAdapter8(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1)
      {
         try
         {
            Y = Integer.parseInt(cubs.entries[8].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (Y >= 2 && Y <= 20)
                  {
                     cubs.colorY = Y;
                     cubs.colorPos1++;
                     cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
                        colorPos1);
                     cubs.cubs2.gameBoard.paint0(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyKeyAdapter9 extends KeyAdapter
{
   public AppletImage cubs;
   public int X;

   public MyKeyAdapter9(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE)
      {
         try
         {
            X = Integer.parseInt(cubs.entries[9].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (X >= 1 && X <= 20)
                  {
                     cubs.colorX = X;
                     cubs.colorPos2--;
                     cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
                        init, cubs.cubs2.fnl, X, cubs.colorPos2);
                     cubs.cubs2.gameBoard.paint0(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyMouseAdapter9 extends MouseAdapter
```

```java
{
   public AppletImage cubs;
   public int X;

   public MyMouseAdapter9(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1)
      {
         try
         {
            X = Integer.parseInt(cubs.entries[9].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (X >= 1 && X <= 20)
                  {
                     cubs.colorX = X;
                     cubs.colorPos2--;
                     cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
                        init, cubs.cubs2.fnl, X, cubs.colorPos2);
                     cubs.cubs2.gameBoard.paint0(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyKeyAdapter10 extends KeyAdapter
{
   public AppletImage cubs;
   public int Y;

   public MyKeyAdapter10(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE)
      {
         try
         {
            Y = Integer.parseInt(cubs.entries[8].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (Y >= 2 && Y <= 20)
                  {
                     cubs.colorY = Y;
                     cubs.colorPos1--;
                     cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
                        colorPos1);
                     cubs.cubs2.gameBoard.paint0(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyMouseAdapter10 extends MouseAdapter
{
   public AppletImage cubs;
   public int Y;

   public MyMouseAdapter10(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1)
      {
         try
         {
            Y = Integer.parseInt(cubs.entries[8].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (Y >= 2 && Y <= 20)
                  {
                     cubs.colorY = Y;
                     cubs.colorPos1--;
                     cubs.c1 = cubs.colorScheme1(cubs.cubs2.discs, Y, cubs.
                        colorPos1);
                     cubs.cubs2.gameBoard.paint0(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyKeyAdapter11 extends KeyAdapter
{
   public AppletImage cubs;
   public int X;

   public MyKeyAdapter11(AppletImage ts)
   {
      cubs = ts;
   }

   public void keyReleased(KeyEvent e)
   {
      if (e.getKeyCode() == KeyEvent.VK_ENTER || e.getKeyCode() ==
         KeyEvent.VK_SPACE)
      {
         try
         {
            X = Integer.parseInt(cubs.entries[9].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
               {
                  if (X >= 1 && X <= 20)
                  {
                     cubs.colorX = X;
                     cubs.colorPos2++;
                     cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
                        init, cubs.cubs2.fnl, X, cubs.colorPos2);
                     cubs.cubs2.gameBoard.paint0(cubs.gx);
                  }
               }
         }
      }
   }
}


class MyMouseAdapter11 extends MouseAdapter
{
   public AppletImage cubs;
   public int X;

   public MyMouseAdapter11(AppletImage ts)
   {
      cubs = ts;
   }

   public void mousePressed(MouseEvent e)
   {
      if (e.getButton() == MouseEvent.BUTTON1)
      {
         try
         {
            X = Integer.parseInt(cubs.entries[9].getText());
         }
         catch (NumberFormatException e2) {}
         finally
         {
            if (cubs.cubs2 != null)
               if (!cubs.cubs2.tooMany)
```

```java
                    {
                        if (X >= 1 && X <= 20)
                        {
                            cubs.colorX = X;
                            cubs.colorPos2++;
                            cubs.c2 = cubs.colorScheme2(cubs.cubs2.pegs, cubs.cubs2.
                                init, cubs.cubs2.fnl, X, cubs.colorPos2);
                            cubs.cubs2.gameBoard.paint0(cubs.gx);
                        }
                    }
                }
            }
        }
}


class MyPanel extends Panel
{
    public MyPanel(BorderLayout bl)
    {
        super(bl);
    }

    public void myDraw(Graphics g)
    {
        if (g != null)
        {
            g.setColor(Color.blue);
            g.fillRect(0, 0, getSize().width, 4);
            g.fillRect(0, 0, 4, getSize().height);
            g.fillRect(0, getSize().height-4, getSize().width, 4);
            g.fillRect(getSize().width-4, 0, 4, getSize().height);
        }
    }

    public void paint(Graphics g)
    {
        myDraw(g);
    }
}


class MyItemListener1 implements ItemListener
{
    public AppletImage cubs;

    public MyItemListener1(AppletImage ts)
    {
        cubs = ts;
    }

    public void itemStateChanged(ItemEvent e)
    {
        if (cubs.cubs2 != null)
            if (!cubs.cubs2.tooMany)
                cubs.cubs2.gameBoard.update(cubs.gx);
    }
}


```

```java
class MyItemListener2 implements ItemListener
{
   public AppletImage cubs;

   public MyItemListener2(AppletImage ts)
   {
      cubs = ts;
   }

   public void itemStateChanged(ItemEvent e)
   {
      if (cubs.log.getState())
         cubs.logBox.setText(cubs.logger);
      else
         cubs.logBox.setText(cubs.preLog);
   }
}


class MyAdjustmentListener implements AdjustmentListener
{
   public AppletImage cubs;

   public MyAdjustmentListener(AppletImage ts)
   {
      cubs = ts;
   }

   public void adjustmentValueChanged(AdjustmentEvent e)
   {
      if (cubs.running)
         cubs.cubs2.timeage();
      cubs.dial.setText2(""+cubs.speed.getValue());
   }
}


class MyCanvas extends Canvas
{
   public AppletImage cubs;
   public int discs, pegs, w, h, labelW, hgap, vgap, x;

   public MyCanvas(AppletImage ts, int n, int p, int w0, int h0, int lW,
      int x2, int y2, int cor)
   {
      cubs = ts;
      discs = n;
      pegs = p;
      w = w0;
      h = h0;
      labelW = lW;
      hgap = x2;
      vgap = y2;
      x = cor;
   }

   public void myDraw0()
   {
      Graphics g = getGraphics();
      int n = 0;
```

```
        if (g != null)
            for (int i = 0; i < pegs; i++)
            {
                g.setColor(cubs.c2[i]);
                g.fillRect(i*(labelW+hgap)+hgap+labelW/2-5+x, vgap, 11, h*discs);
                g.fillRect(i*(labelW+hgap)+hgap+x, h*discs+vgap, labelW, 21);
                g.setColor(Color.black);
                g.setFont(new Font("Verdana", Font.BOLD, 15));
                if (i < 9)
                    g.drawString(""+(i+1), i*(labelW+hgap)+hgap+labelW/2-5+x, h*discs
                        +16+vgap);
                else
                    g.drawString(""+(i+1), i*(labelW+hgap)+hgap+labelW/2-10+x,
                        h*discs+16+vgap);
                for (int j = cubs.valuesIndex[i]; j < discs; j++)
                {
                    n = cubs.values[j][i];
                    g.setColor(cubs.c1[n-1]);
                    myDraw(n, j, i, cubs.gx);
                }
            }
    }

    public void myDraw(int n, int nPos, int pPos, Graphics g)
    {
        int x0 = 0;
        if (g != null)
        {
            if (n == 0)
            {
                if (discs >= 16)
                {
                    g.clearRect(pPos*(labelW+hgap)+hgap+x, nPos*h+vgap, labelW, h);
                    g.setColor(cubs.c2[pPos]);
                    g.fillRect(pPos*(labelW+hgap)+hgap+labelW/2-5+x, nPos*h+vgap, 11,
                    h);
                }
                else
                {
                    g.clearRect(pPos*(labelW+hgap)+hgap+x, 2*h*nPos/3+h*discs/3+vgap,
                    labelW, 2*h/3-(h*(discs+nPos))%3);
                    g.setColor(cubs.c2[pPos]);
                    g.fillRect(pPos*(labelW+hgap)+hgap+labelW/2-5+x, 2*h*nPos/3+h*
                    discs/3+vgap, 11, 2*h/3-(h*(discs+nPos))%3);
                }
                if (nPos == (discs-1))
                    g.fillRect(pPos*(labelW+hgap)+hgap+x, h*discs+vgap, labelW, 2);
            }
            else
            {
                g.setColor(cubs.c1[n-1]);
                if (discs >= 16)
                {
                    g.fillRoundRect(pPos*(labelW+hgap)+hgap+(labelW-w*n)/2-8+x, nPos*
                        h+vgap, w*n+16, h, 2*roundness(), roundness());
                    g.setColor(new Color(0, 0, 128));
                    g.drawRoundRect(pPos*(labelW+hgap)+hgap+(labelW-w*n)/2-8+x, nPos*
                        h+vgap, w*n+16, h, 2*roundness(), roundness());
                }
                else
            {
               g.fillRoundRect(pPos*(labelW+hgap)+hgap+(labelW-w*n)/2-8+x, 2*h*
                  nPos/3+h*discs/3+vgap, w*n+16,
                  2*h/3-(h*(discs+nPos))%3, 2*roundness(), roundness());
               g.setColor(new Color(0, 0, 128));
               g.drawRoundRect(pPos*(labelW+hgap)+hgap+(labelW-w*n)/2-8+x, 2*h*
                  nPos/3+h*discs/3+vgap, w*n+16,
                  2*h/3-(h*(discs+nPos))%3, 2*roundness(), roundness());
            }
            if (cubs.discNums.getState() && Math.min(w+7, h-5) > 7)
            {
               g.setColor(new Color(0, 24, 160));
               if (discs >= 16)
               {
                  if (Math.min(w+7, h-3) == 8)
                  {
                     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+7, h-3))
                        );
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+6, h+1)/4+x,
                           nPos*h+(h+Math.min(w+6, h-3))/2+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+6, h+1)/2-2+x,
                           nPos*h+(h+Math.min(w+6, h-3))/2+vgap);
                  }
                  else if (Math.min(w+7, h-3) == 9 || Math.min(w+7, h-3) == 10)
                  {
                     if (h == 12)
                     {
                        g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+9, h
                           -4)));
                        x0 = 3;
                     }
                     else
                     {
                        g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+9, h
                           -3)));
                        x0 = 1;
                     }
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+6, h+1)/4+x,
                           nPos*h+(int)((h+Math.min(w+6, h-x0))/1.85)+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+6, h+1)/2-2+x,
                           nPos*h+(int)((h+Math.min(w+6, h-x0))/1.85)+vgap);
                  }
                  else if (Math.min(w+6, h-3) > 10 && Math.min(w+6, h-3) < 15)
                  {
                     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+6, h-2))
                        );
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+7, h-2)/4+x,
                           nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           (int)(Math.min(w+7, h-2)
                           /1.85)+x, nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+
                           vgap);
                  }
                  else if (Math.min(w+7, h-2) == 15 || Math.min(w+7, h-2) == 16)
                  {
                     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+7, h-2))
                        );
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+7, h-2)/4-1+x,
                           nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           (int)(Math.min(w+7, h-2)
                           /1.52)+x, nPos*h+(int)((h+Math.min(w+7, h-2))/2.15)+
                           vgap);
                  }
                  else if (Math.min(w+7, h-2) > 16)
                  {
                     g.setFont(new Font("Tahoma", Font.PLAIN, Math.min(w+8, h-2)
                        ));
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+6, h-3)/4+x,
                           nPos*h+(int)((h+Math.min(w+8, h-3))/2.15)+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           (int)(Math.min(w+6, h-3)
                           /1.65)+x, nPos*h+(int)((h+Math.min(w+8, h-3))/2.15)+
                           vgap);
                  }
                  else
                  {
                     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+9, h-1))
                        );
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+8, h-2)/4+x,
                           nPos*h+(int)((h+Math.min(w+9, h-2))/2.15)+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           (int)(Math.min(w+6, h-3)
                           /1.65)+x, nPos*h+(int)((h+Math.min(w+9, h-2))/2.15)+
                           vgap);
                  }
               }
               else
               {
                  if (Math.min(w+7, 2*h/3-2) > 7 && Math.min(w+7, 2*h/3-2) < 16)
                  {
                     g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+3,
                        2*h/3)));
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+3, 2*h/3-4)/4+x,
                           2*h*nPos/3+h*discs/3+Math.min(w+5, 2*h/3-3)+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+3, 2*h/3-4)/2+x,
                           2*h*nPos/3+h*discs/3+Math.min(w+5, 2*h/3-3)+vgap);
                  }
                  else if (Math.min(w+7, 2*h/3-5) >= 16 && Math.min(w+7, 2*h/3
                     -5) <= 50)
                  {
                     if (Math.min(w+7, 2*h/3-2) <= 18)
                        g.setFont(new Font("Tahoma", Font.BOLD, Math.min(w+7,
                           2*h/3-5)));
                     else
                        g.setFont(new Font("Tahoma", Font.PLAIN, Math.min(w+7,
                           2*h/3-5)));
                     if (n <= 9)
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+7, 2*h/3-2)/4+x,
                           2*h*nPos/3+h*discs/3+(int)((2*h/3+Math.min(w+7,
                           2*h/3-7))/2.15)+vgap);
                     else
                        g.drawString(""+n, pPos*(labelW+hgap)+hgap+labelW/2-
                           Math.min(w+7, 2*h/3-2)/2-1+x,
                           2*h*nPos/3+h*discs/3+(int)((2*h/3+Math.min(w+7,
                           2*h/3-7))/2.15)+vgap);
                  }
               }
            }
         }
      }
   }

   public int roundness()
   {
      int arc = 0;
      if (cubs.round.getState())
      {
         if (w >= 140 && h <= 30)
         {
            if (h > 20)
               arc = 15;
            else if (h > 15 && h <= 20)
               arc = 12;
            else if (h >= 13 && h <= 15)
               arc = 10;
         }
         else if (w >= 140 && h > 30)
         {
            if (h >= 60)
               arc = 17;
            if (h >= 40 && h < 60)
               arc = 15;
            else
               arc = 13;
         }
         else if (w < 140 && h <= 30)
         {
            if (h > 20)
               arc = 13;
            else if (h > 15 && h <= 20)
               arc = 11;
            else if (h >= 13 && h <= 15)
               arc = 10;
         }
```

```java
        else if (w < 140 && h > 30)
           arc = 13;
     }
     return arc;
  }

  public void paint0(Graphics g)
  {
     if (g != null)
        for (int i = 0; i < pegs; i++)
        {
           g.clearRect(i*(labelW+hgap)+hgap+x, vgap, labelW, h*discs);
           g.setColor(cubs.c2[i]);
           g.fillRect(i*(labelW+hgap)+hgap+labelW/2-5+x, vgap, 11, h*discs);
           g.fillRect(i*(labelW+hgap)+hgap+x, h*discs+vgap, labelW, 21);
           g.setColor(Color.black);
           g.setFont(new Font("Verdana", Font.BOLD, 15));
           if (i < 9)
              g.drawString(""+(i+1), i*(labelW+hgap)+hgap+labelW/2-5+x, h*discs
                 +16+vgap);
           else
              g.drawString(""+(i+1), i*(labelW+hgap)+hgap+labelW/2-10+x,
                 h*discs+16+vgap);
           for (int j = cubs.valuesIndex[i]; j < discs; j++)
           {
              g.setColor(cubs.c1[j]);
              myDraw(cubs.values[j][i], j, i, g);
           }
        }
  }

  public void paint(Graphics g)
  {
     paint0(g);
  }
}


class MyCanvas2 extends Canvas
{
  public AppletImage cubs;
  public Spire cubs2;
  public String text = " ", altText = " ", altText0 = " ";
  public int dn, discs, pegs, moves, altPointer, altDn;

  public MyCanvas2(AppletImage c, Spire c2)
  {
     cubs = c;
     cubs2 = c2;
  }

  public void setText0()
  {
        if (cubs.running || (!cubs.running && cubs2.pointer <= 1))
           setText2(cubs.text, cubs.dn);
        else if (cubs2.pointer >= cubs.moveCountA-1)
           setText3(cubs.cubs2.discs, cubs.cubs2.pegs, cubs.moveCountA);
  }

  public void setText2(String s, int dn2)
```

```
   {
      Graphics g = getGraphics();
      text = s;
      dn = dn2;
      if (g != null)
      {
         g.setFont(new Font("Tahoma", Font.BOLD, 13));
         g.setColor(Color.white);
         if (altDn == 1)
            g.drawString(altText, getSize().width/2-205-(""+altPointer)
               .length()*5, 15);
         else
            g.drawString(altText, getSize().width/2-205-((""+altPointer)
               .length()+1)*5, 15);
         if (dn == 1)
         {
            g.setColor(new Color(0, 0, 224));
            g.drawString(text, getSize().width/2-205-(""+(cubs2.pointer+1))
               .length()*5, 15);
         }
         else
         {
            g.setColor(new Color(128, 0, 224));
            g.drawString(text, getSize().width/2-205-((""+(cubs2.pointer+1))
               .length()+1)*5, 15);
         }
      }
      altText = text;
      altPointer = cubs2.pointer+1;
      altDn = dn;
   }

   public void setText3(int discs2, int pegs2, int moves2)
   {
      Graphics g = getGraphics();
      if (g != null)
      {
         discs = discs2;
         pegs = pegs2;
         moves = moves2;
         g.clearRect(0, 0, getSize().width, getSize().height);
         g.setColor(new Color(0, 0, 224));
         g.setFont(new Font("Tahoma", Font.BOLD, 13));
         g.drawString(discs+" Disc", getSize().width/2-150-((""+discs).
            length()-2)*3, 15);
         g.drawString("x", getSize().width/2-107+(""+discs).length()*5, 15);
         g.drawString(pegs+" Peg Problem Done In "+moves+" Moves", getSize().
            width/2-92+(""+discs).length()*5, 15);
      }
   }

   public void paint0(Graphics g)
   {
      if (g != null)
      {
         g.setFont(new Font("Tahoma", Font.BOLD, 13));
         if (cubs.running || (!cubs.running && cubs2.pointer <= 1))
         {
            if (dn == 1)
            {
               g.setColor(new Color(0, 0, 224));
               g.drawString(text, getSize().width/2-205-(""+(cubs2.pointer+1))
                  .length()*5, 15);
            }
            else
            {
               g.setColor(new Color(128, 0, 224));
               g.drawString(text, getSize().width/2-205-((""+(cubs2.pointer+1))
                  .length()+1)*5, 15);
            }
         }
         else if (cubs2.pointer >= cubs.moveCountA-1)
            setText3(cubs.cubs2.discs, cubs.cubs2.pegs, cubs.moveCountA);
      }
   }

   public void paint(Graphics g)
   {
      paint0(g);
   }
}


class MyCanvas3 extends Canvas
{
   public String text = "50", altText = "50";

   public void setText2(String s)
   {
      text = s;
      Graphics g = getGraphics();
      if (g != null)
      {
         g.setFont(new Font("Verdana", Font.BOLD, 12));
         g.setColor(Color.lightGray);
         g.drawString(altText, 6, 12);
         update(g);
         g.setColor(new Color(48, 0, 192));
         g.drawString(text, 6, 12);
      }
      altText = text;
   }

   public void paint(Graphics g)
   {
      if (g != null)
      {
         g.setFont(new Font("Verdana", Font.BOLD, 12));
         g.setColor(new Color(48, 0, 192));
         g.drawString(text, 6, 12);
      }
   }
}


class MyCanvas4 extends Canvas
{
   public String text = " ";

   public void setText2(String s)
   {
      text = s;
      Graphics g = getGraphics();
      if (g != null)
      {
         g.setFont(new Font("Times", Font.PLAIN, 24));
         g.drawString(text, getSize().width/2-225, getSize().height/2);
      }
   }

   public void paint(Graphics g)
   {
      if (g != null)
      {
         g.setFont(new Font("Times", Font.PLAIN, 24));
         g.drawString(text, getSize().width/2-225, getSize().height/2);
      }
   }
}


class MyComponentAdapter extends ComponentAdapter
{
   public AppletImage cubs;

   public MyComponentAdapter(AppletImage ts)
   {
      cubs = ts;
   }

   public void componentResized(ComponentEvent e)
   {
      cubs.pF.setTitle("Towers of Chicago  ("+cubs.pF.getSize().width+"  x  "
         +cubs.pF.getSize().height+")");
   }
}


class MyWindowAdapter extends WindowAdapter
{
   public AppletImage cubs;

   public MyWindowAdapter(AppletImage ts)
   {
      cubs = ts;
   }

   public void windowClosing(WindowEvent e)
   {
      cubs.running = false;
      cubs.pF.dispose();
   }
}
```