tick7 submission from James Wood

Name	James Wood (jdw74)
College	ROBIN
Submission contents	uk/ac/cam/jdw74/tick7/AgingWorld.java uk/ac/cam/jdw74/tick7/ArrayWorld.java uk/ac/cam/jdw74/tick7/ControlPanel.java uk/ac/cam/jdw74/tick7/GamePanel.java uk/ac/cam/jdw74/tick7/GuiLife.java uk/ac/cam/jdw74/tick7/HelloActionWorld.java uk/ac/cam/jdw74/tick7/HelloActionWorld2.java uk/ac/cam/jdw74/tick7/PackedLong.java uk/ac/cam/jdw74/tick7/PackedWorld.java uk/ac/cam/jdw74/tick7/Pattern.java uk/ac/cam/jdw74/tick7/PatternFormatException.java uk/ac/cam/jdw74/tick7/PatternLoader.java uk/ac/cam/jdw74/tick7/PatternPanel.java uk/ac/cam/jdw74/tick7/SourcePanel.java uk/ac/cam/jdw74/tick7/Strings.java uk/ac/cam/jdw74/tick7/WorldImpl.java screenshot.png
Ticker	UNKNOWN
Ticker signature	

AgingWorld.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import java.awt.Color;
 3
 4  public class AgingWorld extends WorldImpl {
 5      private int[][] world;
 6
 7      public AgingWorld(int width, int height) {
 8          super(width,height);
 9          world = new int[height][width];
10          for (int y = 0; y < getHeight(); ++y) {
11              for (int x = 0; x < getWidth(); ++x)
12                  world[y][x] = 1000;
13          }
14      }
15
16      private AgingWorld(AgingWorld w) {
17          super(w);
18          world = new int[w.getHeight()][w.getWidth()];
19          for (int y = 0; y < getHeight(); ++y) {
20              for (int x = 0; x < getWidth(); ++x)
21                  world[y][x] = w.world[y][x]+1;
22          }
23      }
24
25      @Override
26      public boolean getCell(int x, int y) {
27          return getCellAge(x,y) == 0;
28      }
29
30      @Override
31      protected WorldImpl nextGeneration() {
32          WorldImpl nextWorld = new AgingWorld(this);
33          for (int row = 0; row < getHeight(); ++row) {
34              for (int col = 0; col < getWidth(); ++col) {
35                  boolean nextLive = computeCell(col, row);
36                  nextWorld.setCell(col, row, nextLive);
37              }
38          }
39          return nextWorld;
40      }
41
42      @Override
43      public void setCell(int x, int y, boolean live) {
44          if (y<0 || y>=getHeight()) return;
45          if (x<0 || x>=getWidth()) return;
46          if (live)
47              world[y][x] = 0;
48      }
49
50      public int getCellAge(int x, int y) {
51          if (y<0 || y>=getHeight()) return Integer.MAX_VALUE;
52          if (x<0 || x>=getWidth()) return Integer.MAX_VALUE;
53          return world[y][x];
54      }
55
56      @Override
57      protected String getCellAsString(int x, int y) {
58          int age = getCellAge(x,y);
59          if (age > 9) return "_";
60          if (age == 0) return "#";
61          return age+"";
62      }
63  }
```

ArrayWorld.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  public class ArrayWorld extends WorldImpl {
 3      private boolean[][] cells;
 4
 5      public ArrayWorld(int w, int h) {
 6          super(w, h);
 7          this.cells = new boolean[h][w];
 8      }
 9
10      protected ArrayWorld(ArrayWorld prev) {
11          super(prev);
12          this.cells = new boolean[prev.getHeight()][prev.getWidth()];
13      }
14
15      @Override
16      public boolean getCell(int col, int row) {
17          return 0 <= row && row < getHeight() &&
18                 0 <= col && col < getWidth() ?
19              cells[row][col] : false;
20      }
21
22      @Override
23      public void setCell(int col, int row, boolean alive) {
24          if (0 <= row && row < getHeight() &&
25              0 <= col && col < getWidth())
26              cells[row][col] = alive;
27      }
28
29      @Override
30      protected ArrayWorld nextGeneration() {
31          ArrayWorld world = new ArrayWorld(this);
32          for (int i = 0; i < getHeight(); i++)
33              for (int j = 0; j < getWidth(); j++)
34                  world.setCell(j, i, computeCell(j, i));
35          return world;
36      }
37  }
```

ControlPanel.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import javax.swing.JPanel;
 3  import javax.swing.JSlider;
 4  import javax.swing.JRadioButton;
 5  import javax.swing.JLabel;
 6  import javax.swing.Box;
 7  import javax.swing.ButtonGroup;
 8  import javax.swing.BoxLayout;
 9  import uk.ac.cam.acr31.life.World;
10  import uk.ac.cam.acr31.life.hash.HashWorld;
11  import javax.swing.event.ChangeListener;
12  import javax.swing.event.ChangeEvent;
13
14  public abstract class ControlPanel extends JPanel {
15
16      private JSlider zoomSlider;
17      private JSlider stepSlider;
18      private JSlider speedSlider;
19      private JRadioButton longButton;
20      private JRadioButton arrayButton;
21      private JRadioButton agingButton;
22      private JRadioButton hashButton;
23
24      protected abstract void onSpeedChange(int value);
25      protected abstract void onStepChange(int value);
26      protected abstract void onZoomChange(int value);
27
28      private JSlider createNewSlider(int min, int max, int start, String s) {
29          Box panel = Box.createHorizontalBox();
30          add(panel);
31          panel.add(new JLabel(s));
32          JSlider slider = new JSlider(min,max,start);
33          panel.add(slider);
34          return slider;
35      }
36
37      private JRadioButton createNewButton(String s, ButtonGroup g, Box b) {
38          JRadioButton r = new JRadioButton(s, true);
39          g.add(r);
40          b.add(r);
41          return r;
42      }
43
44      public ControlPanel() {
45          super();
46          setLayout(new BoxLayout(this,BoxLayout.Y_AXIS));
47
48          zoomSlider = createNewSlider(1,20,10,Strings.CONTROL_ZOOM);
49          add(Box.createVerticalStrut(10)); //add 10px of extra space
50          stepSlider = createNewSlider(0,10,0,Strings.CONTROL_STEP);
51          add(Box.createVerticalStrut(10)); //add 10px of extra space
52          speedSlider = createNewSlider(0,100,0,Strings.CONTROL_SPEED);
53          add(Box.createVerticalStrut(10)); //add 10px of extra space
54
55          speedSlider.addChangeListener(new ChangeListener() {
56                  public void stateChanged(ChangeEvent e) {
57                      if (!speedSlider.getValueIsAdjusting())
58                          onSpeedChange(speedSlider.getValue());
59                  }
60              });
61          stepSlider.addChangeListener(new ChangeListener() {
62                  public void stateChanged(ChangeEvent e) {
63                      if (!stepSlider.getValueIsAdjusting())
64                          onStepChange(stepSlider.getValue());
65                  }
66              });
67          zoomSlider.addChangeListener(new ChangeListener() {
68                  public void stateChanged(ChangeEvent e) {
69                      if (!zoomSlider.getValueIsAdjusting())
70                          onZoomChange(zoomSlider.getValue());
71                  }
```

```
72              });
73
74          Box worldPanel = Box.createHorizontalBox();
75          add(worldPanel);
76          worldPanel.add(new JLabel(Strings.STORAGE_WORLD_TYPE));
77          ButtonGroup group = new ButtonGroup();
78          longButton = createNewButton(Strings.STORAGE_LONG,group,worldPanel);
79          arrayButton = createNewButton(Strings.STORAGE_ARRAY,group,worldPanel);
80          agingButton = createNewButton(Strings.STORAGE_AGING,group,worldPanel);
81          hashButton = createNewButton(Strings.STORAGE_HASH,group,worldPanel);
82          arrayButton.setSelected(true);
83          add(Box.createVerticalStrut(10)); //add 10px of extra space
84      }
85
86      public World initialiseWorld(Pattern p) throws PatternFormatException {
87          World result = null;
88          if (longButton.isSelected()) {
89              result = new PackedWorld();
90          }
91          else if (arrayButton.isSelected()) {
92              result = new ArrayWorld(p.getWidth(),p.getHeight());
93          }
94          else if (agingButton.isSelected()) {
95              result = new AgingWorld(p.getWidth(),p.getHeight());
96          }
97          else if (hashButton.isSelected()) {
98              result = new HashWorld(p.getWidth(),p.getHeight());
99          }
100         if (result != null) p.initialise(result);
101         return result;
102     }
103 }
```

GamePanel.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import javax.swing.JPanel;
 3  import uk.ac.cam.acr31.life.World;
 4  import java.awt.Dimension;
 5  import java.awt.Graphics;
 6
 7  public class GamePanel extends JPanel {
 8
 9      private int zoom = 10; //Number of pixels used to represent a cell
10      private int width = 1; //Width of game board in pixels
11      private int height = 1;//Height of game board in pixels
12      private World current = null;
13
14      public void setZoom(int value) {
15          zoom = value;
16      }
17
18      public Dimension getPreferredSize() {
19          return new Dimension(width, height);
20      }
21
22      protected void paintComponent(Graphics g) {
23          if (current == null) return;
24          g.setColor(java.awt.Color.WHITE);
25          g.fillRect(0, 0, width, height);
26          current.draw(g, width, height);
27          if (zoom > 4) {
28              g.setColor(java.awt.Color.LIGHT_GRAY);
29              for (int i = 1; i < current.getWidth(); i++) {
30                  g.drawLine(i * zoom, 0, i * zoom, height);
31              }
32              for (int i = 1; i < current.getHeight(); i++) {
33                  g.drawLine(0, i * zoom, width, i * zoom);
34              }
35          }
36      }
37
38      private void computeSize() {
39          if (current == null)  return;
40          int newWidth = current.getWidth() * zoom;
41          int newHeight = current.getHeight() * zoom;
42          if (newWidth != width || newHeight != height) {
43              width = newWidth;
44              height = newHeight;
45              revalidate(); //trigger the GamePanel to re-layout its components
46          }
47      }
48
49      public void display(World w) {
50          current = w;
51          computeSize();
52          repaint();
53      }
54  }
```

GuiLife.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import java.awt.BorderLayout;
 3  import javax.swing.border.Border;
 4  import javax.swing.BorderFactory;
 5  import javax.swing.Box;
 6  import javax.swing.JComponent;
 7  import javax.swing.JFrame;
 8  import javax.swing.JPanel;
 9  import javax.swing.JScrollPane;
10  import javax.swing.border.EtchedBorder;
11  import java.util.List;
12  import java.io.IOException;
13  import uk.ac.cam.acr31.life.World;
14  import java.awt.event.ActionListener;
15  import java.awt.event.ActionEvent;
16  import javax.swing.Timer;
17  import javax.swing.event.ChangeListener;
18  import javax.swing.event.ChangeEvent;
19  import javax.swing.JOptionPane;
20  import javax.swing.JFileChooser;
21  import java.io.File;
22  import java.io.FileReader;
23
24  public class GuiLife extends JFrame {
25      private PatternPanel patternPanel;
26      private ControlPanel controlPanel;
27      private GamePanel gamePanel;
28
29      private World world;
30      private int timeDelay = 500;
31      private int timeStep = 0;
32
33      private Timer playTimer = new Timer(timeDelay, new ActionListener() {
34              public void actionPerformed(ActionEvent e) {
35                  doTimeStep();
36              }
37          });
38
39      void doTimeStep() {
40          if (world != null) {
41              world = world.nextGeneration(timeStep);
42              gamePanel.display(world);
43          }
44      }
45
46      public GuiLife() {
47          super("GuiLife");
48          setSize(640, 480);
49          setDefaultCloseOperation(EXIT_ON_CLOSE);
50          setLayout(new BorderLayout());
51          JComponent optionsPanel = createOptionsPanel();
52          add(optionsPanel, BorderLayout.WEST);
53          JComponent gamePanel = createGamePanel();
54          add(gamePanel, BorderLayout.CENTER);
55      }
56
57      private JComponent createOptionsPanel() {
58          Box result = Box.createVerticalBox();
59          result.add(createSourcePanel());
60          result.add(createPatternPanel());
61          result.add(createControlPanel());
62          return result;
63      }
64
65      private void addBorder(JComponent component, String title) {
66          Border etch = BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);
67          Border tb = BorderFactory.createTitledBorder(etch,title);
68          component.setBorder(tb);
69      }
70
71      private JComponent createGamePanel() {
```

```
72          JPanel holder = new JPanel();
73          addBorder(holder,Strings.PANEL_GAMEVIEW);
74          gamePanel = new GamePanel();
75          holder.add(gamePanel);
76          return new JScrollPane(holder);
77      }
78
79      private JComponent createSourcePanel() {
80          JPanel result = new SourcePanel(){
81                  @Override
82                  protected boolean setSourceFile() {
83                      JFileChooser chooser = new JFileChooser();
84                      int returnVal = chooser.showOpenDialog(this);
85                      if (returnVal == JFileChooser.APPROVE_OPTION) {
86                          File f = chooser.getSelectedFile();
87                          try {
88                              List<Pattern> list =
89                                  PatternLoader.load(new FileReader(f));
90                              patternPanel.setPatterns(list);
91                              resetWorld();
92                              return true;
93                          }
94                          catch (IOException ioe) {}
95                      }
96                      return false;
97                  }
98
99                  @Override
100                 protected boolean setSourceNone() {
101                     world = null;
102                     patternPanel.setPatterns(null);
103                     resetWorld();
104                     return true;
105                 }
106
107                 @Override
108                 protected boolean setSourceLibrary() {
109                     String u =
110                         "http://www.cl.cam.ac.uk/teaching/current/ProgJava/" +
111                         "nextlife.txt";
112                     return setSourceWeb(u);
113                 }
114
115                 @Override
116                 protected boolean setSourceFourStar() {
117                     String u =
118                         "http://www.cl.cam.ac.uk/teaching/current/ProgJava/" +
119                         "competition.txt";
120                     return setSourceWeb(u);
121                 }
122
123                 private boolean setSourceWeb(String url) {
124                     try {
125                         List<Pattern> list = PatternLoader.loadFromURL(url);
126                         patternPanel.setPatterns(list);
127                         resetWorld();
128                         return true;
129                     }
130                     catch (IOException ioe) {}
131                     return false;
132                 }
133             };
134         addBorder(result, Strings.PANEL_SOURCE);
135         return result;
136     }
137
138     private JComponent createPatternPanel() {
139         PatternPanel result = new PatternPanel() {
140                 @Override
141                 protected void onPatternChange() {
142                     resetWorld();
143                 }
```

```
144             };
145         addBorder(result, Strings.PANEL_PATTERN);
146         patternPanel = result;
147         return result;
148     }
149 
150     private JComponent createControlPanel() {
151         controlPanel = new ControlPanel() {
152                 @Override
153                 protected void onSpeedChange(int value) {
154                     playTimer.setDelay(1 + (100 - value) * 10);
155                 }
156 
157                 @Override
158                 protected void onStepChange(int value) {
159                     timeStep = value;
160                 }
161 
162                 @Override
163                 protected void onZoomChange(int value) {
164                     gamePanel.setZoom(value);
165                 }
166             };
167         addBorder(controlPanel, Strings.PANEL_CONTROL);
168         return controlPanel;
169     }
170 
171     private void resetWorld() {
172         Pattern current = patternPanel.getCurrentPattern();
173         world = null;
174         if (current != null) {
175             try {
176                 world = controlPanel.initialiseWorld(current);
177             }
178             catch (PatternFormatException e) {
179                 JOptionPane.showMessageDialog(
180                     this, "Error initialising world",
181                     "An error occurred when initialising the world. " +
182                     e.getMessage(), JOptionPane.ERROR_MESSAGE);
183             }
184         }
185         gamePanel.display(world);
186         repaint();
187     }
188 
189     public static void main(String[] args) {
190         GuiLife gui = new GuiLife();
191         gui.playTimer.start();
192         gui.resetWorld();
193         gui.setVisible(true);
194     }
195 }
```

HelloActionWorld.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import javax.swing.JFrame;
 3  import javax.swing.JLabel;
 4  import javax.swing.JButton;
 5  import java.awt.event.ActionListener;
 6  import java.awt.event.ActionEvent;
 7  import javax.swing.BoxLayout;
 8
 9  public class HelloActionWorld extends JFrame {
10
11      private JLabel label;
12
13      //an "inner" class which handles events of type "ActionEvent"
14      private class ButtonAction implements ActionListener {
15          private int count = 0;
16
17          public void actionPerformed(ActionEvent e) {
18              count++;
19              label.setText(count == 1 ?
20                            "Button pressed 1 time" :
21                            "Button pressed " + Integer.toString(count) +
22                            " times");
23          }
24      }
25
26      HelloActionWorld() {
27          super("Hello Action");                  //create window & set title text
28          setDefaultCloseOperation(EXIT_ON_CLOSE); //close button on window quits app.
29
30          //configure the layout of the pane associated with this window as a "BoxLayout"
31          setLayout(new BoxLayout(getContentPane(),BoxLayout.Y_AXIS));
32
33          label = new JLabel("Button unpressed");  //create graphical text label
34          add(label);                              //associate "label" with window
35          JButton button = new JButton("Press me");//create graphical button
36          add(button);                             //associated "button" with window
37
38          //add a new instance of "ButtonAction" as an event handler for "button"
39          button.addActionListener(new ButtonAction());
40
41          setSize(320,240);                        //set size of window
42      }
43
44      public static void main(String[] args) {
45          HelloActionWorld hello = new HelloActionWorld(); //create instance
46          hello.setVisible(true);                          //display window to user
47      }
48  }
```

HelloActionWorld2.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1  import javax.swing.JFrame;       import java.awt.event.ActionListener;
 2  import javax.swing.JLabel;       import java.awt.event.ActionEvent;
 3  import javax.swing.JButton;      import javax.swing.BoxLayout;
 4
 5  public class HelloActionWorld2 extends JFrame {
 6
 7      HelloActionWorld2() {
 8          //create window & set title text
 9          super("Hello Action");
10          //close button on window quits app.
11          setDefaultCloseOperation(EXIT_ON_CLOSE);
12          //configure the layout of the pane associated with this window as a
13          //  "BoxLayout"
14          setLayout(new BoxLayout(getContentPane(),BoxLayout.Y_AXIS));
15          //create graphical text label
16          JLabel label = new JLabel("Button unpressed");
17          add(label);  //associate "label" with window
18          JButton button = new JButton("Press me"); //create graphical button
19          add(button); //associated "button" with window
20          //create an instance of an anonymous inner class to hand the event
21          button.addActionListener(new ActionListener(){
22                  private int count = 0;
23
24                  public void actionPerformed(ActionEvent e) {
25                      count++;
26                      label.setText(count == 1 ?
27                                    "Button pressed 1 time" :
28                                    "Button pressed " + Integer.toString(count) +
29                                    " times");
30                  }
31              });
32          setSize(320,240);                       //set size of window
33      }
34
35      public static void main(String[] args) {
36          HelloActionWorld2 hello = new HelloActionWorld2(); //create instance
37          hello.setVisible(true);                          //display window to user
38      }
39  }
```

PackedLong.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  public class PackedLong {
 3
 4      /*
 5      * Unpack and return the nth bit from the packed number at index position;
 6      * position counts from zero (representing the least significant bit)
 7      * up to 63 (representing the most significant bit).
 8      */
 9      public static boolean get(long packed, int position) {
10          // set "check" to equal 1 if the "position" bit in "packed" is set to 1
11          long check = packed >> position & 1L;
12          return (check == 1L);
13      }
14
15      /*
16      * Set the nth bit in the packed number to the value given
17      * and return the new packed number
18      */
19      public static long set(long packed, int position, boolean value) {
20          if (value) {
21              packed |= 1L << position;
22              // update the value "packed" with the bit at "position" set to 1
23          }
24          else {
25              packed &= ~(1L << position);
26              // update the value "packed" with the bit a "position" set to 0
27          }
28          return packed;
29      }
30  }
```

PackedWorld.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  public class PackedWorld extends WorldImpl {
 3      private int generation;
 4      private long cells;
 5
 6      public PackedWorld() {
 7          super(8, 8);
 8          cells = 0;
 9      }
10
11      protected PackedWorld(PackedWorld prev) {
12          super(prev);
13          cells = 0;
14      }
15
16      @Override
17      public boolean getCell(int col, int row) {
18          return 0 <= row && row < 8 && 0 <= col && col < 8 ?
19              PackedLong.get(cells, row * 8 + col) : false;
20      }
21
22      @Override
23      public void setCell(int col, int row, boolean alive) {
24          if (0 <= row && row < 8 && 0 <= col && col < 8)
25              cells = PackedLong.set(cells, row * 8 + col, alive);
26      }
27
28      @Override
29      protected PackedWorld nextGeneration() {
30          PackedWorld world = new PackedWorld(this);
31          for (int i = 0; i < 8; i++)
32              for (int j = 0; j < 8; j++)
33                  world.setCell(j, i, computeCell(j, i));
34          return world;
35      }
36  }
```

Pattern.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import java.text.ParseException;
 3  import uk.ac.cam.acr31.life.World;
 4
 5  public class Pattern {
 6
 7      private String name;
 8      private String author;
 9      private int width;
10      private int height;
11      private int startCol;
12      private int startRow;
13      private String cells;
14
15      public String getName() { return name; }
16      public void setName(String x) { name = x; }
17
18      public String getAuthor() { return author; }
19      public void setAuthor(String x) { author = x; }
20
21      public int getWidth() { return width; }
22      public void setWidth(int x) { width = x; }
23
24      public int getHeight() { return height; }
25      public void setHeight(int x) { height = x; }
26
27      public int getStartCol() { return startCol; }
28      public void setStartCol(int x) { startCol = x; }
29
30      public int getStartRow() { return startRow; }
31      public void setStartRow(int x) { startRow = x; }
32
33      public String getCells() { return cells; }
34      public void setCells(String x) { cells = x; }
35
36      public Pattern(String format) throws PatternFormatException {
37          String[] parts = format.split(":");
38          if (parts.length < 7)
39              throw new PatternFormatException("Too few arguments");
40          if (parts.length > 7)
41              throw new PatternFormatException("Too many arguments");
42
43          name = parts[0];
44          author = parts[1];
45          try {
46              width = Integer.parseInt(parts[2]);
47              if (width <= 0) throw new NumberFormatException();
48          }
49          catch (NumberFormatException e) {
50              throw new PatternFormatException(
51                  "Width argument not a positive integer");
52          }
53          try {
54              height = Integer.parseInt(parts[3]);
55              if (height <= 0) throw new NumberFormatException();
56          }
57          catch (NumberFormatException e) {
58              throw new PatternFormatException(
59                  "Height argument not a positive integer");
60          }
61          try {
62              startCol = Integer.parseInt(parts[4]);
63          }
64          catch (NumberFormatException e) {
65              throw new PatternFormatException(
66                  "x coördinate not an integer");
67          }
68          try {
69              startRow = Integer.parseInt(parts[5]);
70          }
71          catch (NumberFormatException e) {
```

```
72              throw new PatternFormatException(
73                  "y coördinate not an integer");
74          }
75          cells = parts[6];
76      }
77
78      public void initialise(World world) throws PatternFormatException {
79          String[] rows = cells.split(" ");
80          char[][] values = new char[rows.length][];
81          for (int i = 0; i < rows.length; i++)
82              values[i] = rows[i].toCharArray();
83
84          for (int i = 0; i < values.length; i++)
85              for (int j = 0; j < values[i].length; j++)
86                  if (" 01".indexOf(values[i][j]) == -1)
87                      throw new PatternFormatException(
88                      "Pattern contains characters other than ‘0’, ‘1’ and ‘ ’");
89                  else
90                      world.setCell(startCol + j, startRow + i,
91                                    values[i][j] == '1');
92      }
93
94      @Override
95      public String toString() {
96          return name + ":" + author + ":" + width + ":" + height + ":" +
97              startCol + ":" + startRow + ":" + cells;
98      }
99  }
```

PatternFormatException.java

```
0   package uk.ac.cam.jdw74.tick7;
1
2   public class PatternFormatException extends Exception {
3       public PatternFormatException(String message) {
4           super(message);
5       }
6   }
```

PatternLoader.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import java.io.Reader;
 3  import java.io.BufferedReader;
 4  import java.io.IOException;
 5  import java.io.InputStreamReader;
 6  import java.io.FileReader;
 7  import java.util.List;
 8  import java.util.LinkedList;
 9  import java.net.URL;
10  import java.net.URLConnection;
11
12  public class PatternLoader {
13
14      public static List<Pattern> load(Reader r) throws IOException {
15          BufferedReader buff = new BufferedReader(r);
16          List<Pattern> results = new LinkedList<>();
17
18          String line;
19          while ((line = buff.readLine()) != null)
20              try {
21                  results.add(new Pattern(line));
22              }
23              catch (PatternFormatException e) {}
24
25          return results;
26      }
27
28      public static List<Pattern> loadFromURL(String url) throws IOException {
29          URL destination = new URL(url);
30          URLConnection conn = destination.openConnection();
31          return load(new InputStreamReader(conn.getInputStream()));
32      }
33
34      public static List<Pattern> loadFromDisk(String filename)
35          throws IOException {
36          return load(new FileReader(filename));
37      }
38  }
```

PatternPanel.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import javax.swing.JPanel;
 3  import javax.swing.JList;
 4  import java.awt.BorderLayout;
 5  import javax.swing.JScrollPane;
 6  import java.util.List;
 7  import java.util.ArrayList;
 8  import javax.swing.event.ListSelectionListener;
 9  import javax.swing.event.ListSelectionEvent;
10
11  public abstract class PatternPanel extends JPanel {
12      private JList guiList;
13      private Pattern currentPattern;
14      private List<Pattern> patternList;
15
16      protected abstract void onPatternChange();
17
18      public PatternPanel() {
19          super();
20          setLayout(new BorderLayout());
21          guiList = new JList();
22          add(new JScrollPane(guiList));
23          currentPattern = null;
24          patternList = null;
25
26          guiList.addListSelectionListener(new ListSelectionListener() {
27                  public void valueChanged(ListSelectionEvent e) {
28                      if (!e.getValueIsAdjusting() && (patternList != null)) {
29                          int sel = guiList.getSelectedIndex();
30                          if (sel != -1) {
31                              currentPattern = patternList.get(sel);
32                              onPatternChange();
33                          }
34                      }
35                  }
36              });
37      }
38
39      public void setPatterns(List<Pattern> list) {
40          patternList = list;
41          if (list == null) {
42              currentPattern = null; //if list is null, then no valid pattern
43              guiList.setListData(new String[]{}); //no list item to select
44              return;
45          }
46
47          ArrayList<String> names = new ArrayList<String>();
48
49          for (Pattern p : list) {
50              names.add(p.getName() + " (" + p.getAuthor() + ")");
51          }
52
53          guiList.setListData(names.toArray());
54          currentPattern = list.get(0); //select first element in list
55          guiList.setSelectedIndex(0);  //select first element in guiList
56      }
57
58      public Pattern getCurrentPattern() {
59          return currentPattern;
60      }
61  }
```

SourcePanel.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import javax.swing.BoxLayout;
 3  import javax.swing.JPanel;
 4  import javax.swing.JRadioButton;
 5  import javax.swing.ButtonGroup;
 6  import java.awt.event.ActionListener;
 7  import java.awt.event.ActionEvent;
 8
 9  public abstract class SourcePanel extends JPanel {
10      private JRadioButton current;
11
12      protected abstract boolean setSourceNone();
13      protected abstract boolean setSourceFile();
14      protected abstract boolean setSourceLibrary();
15      protected abstract boolean setSourceFourStar();
16
17      public SourcePanel() {
18          super();
19          setLayout(new BoxLayout(this,BoxLayout.X_AXIS));
20          JRadioButton none = new JRadioButton(Strings.BUTTON_SOURCE_NONE, true);
21          JRadioButton file = new JRadioButton(Strings.BUTTON_SOURCE_FILE, true);
22          JRadioButton library =
23              new JRadioButton(Strings.BUTTON_SOURCE_LIBRARY, true);
24          JRadioButton fourStar =
25              new JRadioButton(Strings.BUTTON_SOURCE_FOURSTAR, true);
26          //add RadioButtons to this JPanel
27          add(none);
28          add(file);
29          add(library);
30          add(fourStar);
31          //create a ButtonGroup containing all four buttons
32          //Only one Button in a ButtonGroup can be selected at once
33          ButtonGroup group = new ButtonGroup();
34          group.add(none);
35          group.add(file);
36          group.add(library);
37          group.add(fourStar);
38          current = none;
39
40          none.addActionListener(new ActionListener() {
41                  public void actionPerformed(ActionEvent e) {
42                      if (setSourceNone())
43                          //successful: none found and patterns loaded
44                          current = none;
45                      else
46                          //unsuccessful: re-enable previous source choice
47                          current.setSelected(true);
48                  }
49              });
50          file.addActionListener(new ActionListener() {
51                  public void actionPerformed(ActionEvent e) {
52                      if (setSourceFile())
53                          //successful: file found and patterns loaded
54                          current = file;
55                      else
56                          //unsuccessful: re-enable previous source choice
57                          current.setSelected(true);
58                  }
59              });
60          library.addActionListener(new ActionListener() {
61                  public void actionPerformed(ActionEvent e) {
62                      if (setSourceLibrary())
63                          //successful: library found and patterns loaded
64                          current = library;
65                      else
66                          //unsuccessful: re-enable previous source choice
67                          current.setSelected(true);
68                  }
69              });
70          fourStar.addActionListener(new ActionListener() {
71                  public void actionPerformed(ActionEvent e) {
```

```
72                      if (setSourceFourStar())
73                          //successful: fourStar found and patterns loaded
74                          current = fourStar;
75                      else
76                          //unsuccessful: re-enable previous source choice
77                          current.setSelected(true);
78                  }
79              });
80      }
81  }
```

Strings.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  public class Strings {
 3      public static final String PANEL_SOURCE = "Source";
 4      public static final String PANEL_PATTERN = "Starting pattern";
 5      public static final String PANEL_CONTROL = "Control";
 6      public static final String PANEL_GAMEVIEW = "Game State";
 7      public static final String BUTTON_SOURCE_NONE = "None";
 8      public static final String BUTTON_SOURCE_FILE = "File";
 9      public static final String BUTTON_SOURCE_LIBRARY = "Library";
10      public static final String BUTTON_SOURCE_FOURSTAR = "4* Submissions";
11      public static final String CONTROL_ZOOM = "Zoom";
12      public static final String CONTROL_STEP = "Step";
13      public static final String CONTROL_SPEED = "Speed";
14      public static final String STORAGE_WORLD_TYPE = "World type";
15      public static final String STORAGE_LONG = "Long";
16      public static final String STORAGE_ARRAY = "Array";
17      public static final String STORAGE_AGING = "Aging";
18      public static final String STORAGE_HASH = "Hash";
19  }
```

WorldImpl.java

```
 0  package uk.ac.cam.jdw74.tick7;
 1
 2  import java.awt.Color;
 3  import java.awt.Graphics;
 4  import java.io.Writer;
 5  import java.io.PrintWriter;
 6  import uk.ac.cam.acr31.life.World;
 7
 8  public abstract class WorldImpl implements World {
 9
10      private int width;
11      private int height;
12      private int generation;
13
14      protected WorldImpl(int width, int height) {
15          this.width = width;
16          this.height = height;
17          this.generation = 0;
18      }
19
20      protected WorldImpl(WorldImpl prev) {
21          this.width = prev.width;
22          this.height = prev.height;
23          this.generation = prev.generation + 1;
24      }
25
26      public int getWidth() { return this.width; }
27
28      public int getHeight() { return this.height; }
29
30      public int getGeneration() { return this.generation; }
31
32      public int getPopulation() { return 0; }
33
34      protected String getCellAsString(int col,int row) {
35          return getCell(col,row) ? "#" : "_";
36      }
37
38      protected Color getCellAsColour(int col,int row) {
39          return getCell(col,row) ? Color.BLACK : Color.WHITE;
40      }
41      public void draw(Graphics g,int width, int height) {
42          int worldWidth = getWidth();
43          int worldHeight = getHeight();
44
45          double colScale = (double)width/(double)worldWidth;
46          double rowScale = (double)height/(double)worldHeight;
47
48          for(int col=0; col<worldWidth; ++col) {
49              for(int row=0; row<worldHeight; ++row) {
50                  int colPos = (int)(col*colScale);
51                  int rowPos = (int)(row*rowScale);
52                  int nextCol = (int)((col+1)*colScale);
53                  int nextRow = (int)((row+1)*rowScale);
54
55                  if (g.hitClip(colPos,rowPos,nextCol-colPos,nextRow-rowPos)) {
56                      g.setColor(getCellAsColour(col, row));
57                      g.fillRect(colPos,rowPos,nextCol-colPos,nextRow-rowPos);
58                  }
59              }
60          }
61      }
62
63      public World nextGeneration(int log2StepSize) {
64          WorldImpl world = this;
65          for (int i = 0; i < 1 << log2StepSize; i++)
66              world = world.nextGeneration();
67          return world;
68      }
69
70      public void print(Writer w) {
71          PrintWriter pw = new PrintWriter(w);
```

```
72
73          pw.println("-");
74          for (int row = 0; row < this.height; row++) {
75              for (int col = 0; col < this.width; col++)
76                  pw.print(getCellAsString(col, row));
77              pw.println();
78          }
79
80          pw.flush();
81      }
82
83      protected int countNeighbours(int col, int row) {
84          return
85              (getCell(col - 1, row - 1) ? 1 : 0)
86            + (getCell(col    , row - 1) ? 1 : 0)
87            + (getCell(col + 1, row - 1) ? 1 : 0)
88            + (getCell(col - 1, row    ) ? 1 : 0)
89            + (getCell(col + 1, row    ) ? 1 : 0)
90            + (getCell(col - 1, row + 1) ? 1 : 0)
91            + (getCell(col    , row + 1) ? 1 : 0)
92            + (getCell(col + 1, row + 1) ? 1 : 0);
93      }
94
95      protected boolean computeCell(int col, int row) {
96          int count = countNeighbours(col, row);
97          return count == 3 || (getCell(col, row) && count == 2);
98      }
99
100     // Will be implemented by child class.
101     // Return true if cell (col,row) is alive.
102     public abstract boolean getCell(int col,int row);
103
104     // Will be implemented by child class. Set a cell to be live or dead.
105     public abstract void setCell(int col, int row, boolean alive);
106
107     // Will be implemented by child class. Step forward one generation.
108     protected abstract WorldImpl nextGeneration();
109 }
```

screenshot.png