tick3star submission from James Wood

Name	James Wood (jdw74)
College	ROBIN
Submission contents	uk/ac/cam/jdw74/tick3star/AnimatedLife.java uk/ac/cam/jdw74/tick3star/OutputAnimatedGif.java uk/ac/cam/jdw74/tick3star/Pattern.java uk/ac/cam/jdw74/tick3star/competition.txt uk/ac/cam/jdw74/tick3star/competition.gif generated.gif
Ticker	UNKNOWN
Ticker signature	

AnimatedLife.java

```
 0  package uk.ac.cam.jdw74.tick3star;
 1
 2  import java.io.IOException;
 3
 4  class AnimatedLife {
 5      public static boolean getCell(boolean[][] world, int col, int row) {
 6          return 0 <= row && row < world.length &&
 7                 0 <= col && col < world[row].length ?
 8              world[row][col] : false;
 9      }
10
11      public static void setCell(boolean[][] world, int col, int row,
12                                 boolean value) {
13          if (0 <= row && row < world.length &&
14              0 <= col && col < world[row].length)
15              world[row][col] = value;
16      }
17
18      public static int countNeighbours(boolean[][] world, int col, int row) {
19          return
20              (getCell(world, col - 1, row - 1) ? 1 : 0)
21            + (getCell(world, col    , row - 1) ? 1 : 0)
22            + (getCell(world, col + 1, row - 1) ? 1 : 0)
23            + (getCell(world, col - 1, row    ) ? 1 : 0)
24            + (getCell(world, col + 1, row    ) ? 1 : 0)
25            + (getCell(world, col - 1, row + 1) ? 1 : 0)
26            + (getCell(world, col    , row + 1) ? 1 : 0)
27            + (getCell(world, col + 1, row + 1) ? 1 : 0);
28      }
29
30      public static boolean computeCell(boolean[][] world, int col, int row) {
31          int count = countNeighbours(world, col, row);
32          return count == 3 || (getCell(world, col, row) && count == 2);
33      }
34
35      public static boolean[][] nextGeneration(boolean[][] world) {
36          boolean[][] nextWorld = new boolean[world.length][world[0].length];
37          for (int row = 0; row < world.length; row++)
38              for (int col = 0; col < world[row].length; col++)
39                  setCell(nextWorld, col, row,
40                          computeCell(world, col, row));
41          return nextWorld;
42      }
43
44      public static void play(boolean[][] world, int iterations, String filename)
45          throws IOException {
46          OutputAnimatedGif gif = new OutputAnimatedGif(filename);
47          for (int i = 0; i < iterations; i++) {
48              gif.addFrame(world);
49              world = nextGeneration(world);
50          }
51          gif.close();
52      }
53
54      public static void main(String[] args) throws Exception {
55           Pattern p = new Pattern(args[0]);
56           boolean[][] world = new boolean[p.getHeight()][p.getWidth()];
57           p.initialise(world);
58
59           play(world, Integer.parseInt(args[1]), args[2]);
60      }
61  }
```

OutputAnimatedGif.java

```
 0   package uk.ac.cam.jdw74.tick3star;
 1
 2   import java.awt.*;
 3   import java.awt.image.*;
 4   import java.io.*;
 5   import javax.imageio.*;
 6   import javax.imageio.stream.*;
 7   import javax.imageio.metadata.*;
 8
 9   public class OutputAnimatedGif {
10       private FileImageOutputStream output;
11       private ImageWriter writer;
12
13       private int cellWidth = 6;
14       private int cellHeight = 6;
15
16       public OutputAnimatedGif(String file) throws IOException {
17           this.output = new FileImageOutputStream(new File(file));
18           this.writer = ImageIO.getImageWritersByMIMEType("image/gif").next();
19           this.writer.setOutput(output);
20           this.writer.prepareWriteSequence(null);
21       }
22
23       private BufferedImage makeFrame(boolean[][] world) {
24           //TODO: complete this method
25           BufferedImage image = new BufferedImage(cellWidth * world[0].length,
26                                                   cellHeight * world.length,
27                                                   BufferedImage.TYPE_INT_RGB);
28           Graphics g = image.getGraphics();
29           int worldHeight = world.length;
30           int worldWidth = world[0].length;
31
32           g.setColor(Color.black);
33           g.fillRect(0, 0, worldWidth * cellWidth, worldHeight * cellHeight);
34           g.setColor(Color.white);
35           for (int i = 0; i < worldHeight; i++)
36               for (int j = 0; j < worldWidth; j++)
37                   if (world[i][j])
38                       g.fillRect(j * cellWidth, i * cellHeight,
39                                  cellWidth, cellHeight);
40
41           g.dispose();
42           return image;
43       }
44
45       public void addFrame(boolean[][] world) throws IOException {
46           BufferedImage image = makeFrame(world);
47           try {
48               IIOMetadataNode node = new IIOMetadataNode("javax_imageio_gif_image_1.0");
49               IIOMetadataNode extension = new IIOMetadataNode("GraphicControlExtension");
50               extension.setAttribute("disposalMethod", "none");
51               extension.setAttribute("userInputFlag", "FALSE");
52               extension.setAttribute("transparentColorFlag", "FALSE");
53               extension.setAttribute("delayTime", "1");
54               extension.setAttribute("transparentColorIndex", "255");
55               node.appendChild(extension);
56               IIOMetadataNode appExtensions = new IIOMetadataNode("ApplicationExtensions");
57               IIOMetadataNode appExtension = new IIOMetadataNode("ApplicationExtension");
58               appExtension.setAttribute("applicationID", "NETSCAPE");
59               appExtension.setAttribute("authenticationCode", "2.0");
60               byte[] b = "\u0021\u00ff
\u000bNETSCAPE2.0\u0003\u0001\u0000\u0000\u0000".getBytes();
61               appExtension.setUserObject(b);
62               appExtensions.appendChild(appExtension);
63               node.appendChild(appExtensions);
64
65               IIOMetadata metadata;
66               metadata = writer.getDefaultImageMetadata(new ImageTypeSpecifier(image), null);
67               metadata.mergeTree("javax_imageio_gif_image_1.0", node);
68
69               IIOImage t = new IIOImage(image, null, metadata);
70               writer.writeToSequence(t, null);
71           }
```

```
72          catch (IIOInvalidTreeException e) {
73              throw new IOException(e);
74          }
75      }
76
77      public void close() throws IOException {
78          writer.endWriteSequence();
79      }
80  }
```

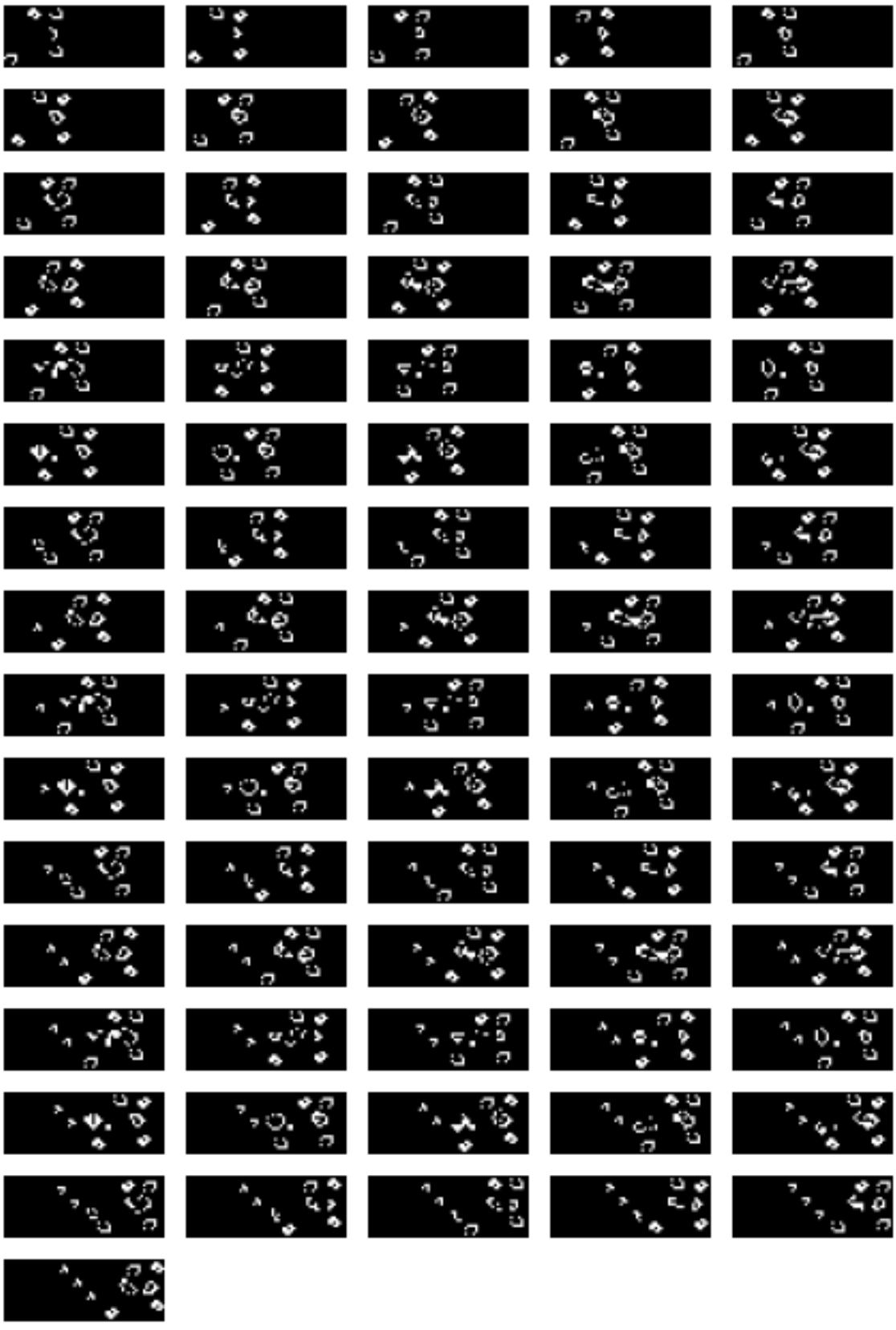
Pattern.java

```
 0  package uk.ac.cam.jdw74.tick3star;
 1
 2  import java.text.ParseException;
 3
 4  public class Pattern {
 5
 6      private String name;
 7      private String author;
 8      private int width;
 9      private int height;
10      private int startCol;
11      private int startRow;
12      private String cells;
13
14      public String getName() { return name; }
15      public void setName(String x) { name = x; }
16
17      public String getAuthor() { return author; }
18      public void setAuthor(String x) { author = x; }
19
20      public int getWidth() { return width; }
21      public void setWidth(int x) { width = x; }
22
23      public int getHeight() { return height; }
24      public void setHeight(int x) { height = x; }
25
26      public int getStartCol() { return startCol; }
27      public void setStartCol(int x) { startCol = x; }
28
29      public int getStartRow() { return startRow; }
30      public void setStartRow(int x) { startRow = x; }
31
32      public String getCells() { return cells; }
33      public void setCells(String x) { cells = x; }
34
35      public Pattern(String format) throws ParseException {
36          String[] parts = format.split(":");
37          if (parts.length != 7)
38              throw new ParseException("Incorrect pattern format", 0);
39
40          name = parts[0];
41          author = parts[1];
42          width = Integer.parseInt(parts[2]);
43          height = Integer.parseInt(parts[3]);
44          startCol = Integer.parseInt(parts[4]);
45          startRow = Integer.parseInt(parts[5]);
46          cells = parts[6];
47      }
48
49      public void initialise(boolean[][] world) {
50          String[] rows = cells.split(" ");
51          char[][] values = new char[rows.length][];
52          for (int i = 0; i < rows.length; i++)
53              values[i] = rows[i].toCharArray();
54
55          for (int i = 0; i < values.length; i++)
56              for (int j = 0; j < values[i].length; j++)
57                  world[startRow + i][startCol + j] = values[i][j] == '1';
58      }
59  }
```

competition.txt

```
 0   "Space rake:Bill Gosper (1971):60:23:0:1:00000000011000000010 00000000011111000000001
000000000011011000100001 000000000000110000001111 000000000000000000000000 000000000000000000000000
000000000000000000000000 000000000000000000010000 000000000000000000001100 000000000000000000000100
000000000000000000000100 000000000000000000001000 000000000000000000000000 000000000000000000000000
000000000000000000000010 000000000000000000000001 000000000000000000010001 011110000000000001111
100010000000000000000000 000010000000000000000000 000100000000000000000000" 76 competition.gif
```

competition.gif



generated.gif