tick6 submission from James Wood

Name	James Wood (jdw74)
College	ROBIN
Submission contents	uk/ac/cam/jdw74/tick6/AgingWorld.java uk/ac/cam/jdw74/tick6/ArrayWorld.java uk/ac/cam/jdw74/tick6/ControlPanel.java uk/ac/cam/jdw74/tick6/GamePanel.java uk/ac/cam/jdw74/tick6/GuiLife.java uk/ac/cam/jdw74/tick6/HelloSwingWorld.java uk/ac/cam/jdw74/tick6/PackedLong.java uk/ac/cam/jdw74/tick6/PackedWorld.java uk/ac/cam/jdw74/tick6/Pattern.java uk/ac/cam/jdw74/tick6/PatternFormatException.java uk/ac/cam/jdw74/tick6/PatternLoader.java uk/ac/cam/jdw74/tick6/PatternPanel.java uk/ac/cam/jdw74/tick6/SourcePanel.java uk/ac/cam/jdw74/tick6/Strings.java uk/ac/cam/jdw74/tick6/WorldImpl.java screenshot.png
Ticker	UNKNOWN
Ticker signature	

AgingWorld.java

```
 0  package uk.ac.cam.jdw74.tick6;
 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.tick6;
 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.tick6;
 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
11  public class ControlPanel extends JPanel {
12
13      private JSlider zoomSlider;
14      private JSlider stepSlider;
15      private JSlider speedSlider;
16      private JRadioButton longButton;
17      private JRadioButton arrayButton;
18      private JRadioButton agingButton;
19
20      private JSlider createNewSlider(int min, int max, int start, String s) {
21          Box panel = Box.createHorizontalBox();
22          add(panel);
23          panel.add(new JLabel(s));
24          JSlider slider = new JSlider(min,max,start);
25          panel.add(slider);
26          return slider;
27      }
28
29      private JRadioButton createNewButton(String s, ButtonGroup g, Box b) {
30          JRadioButton r = new JRadioButton(s, true);
31          g.add(r);
32          b.add(r);
33          return r;
34      }
35
36      public ControlPanel() {
37          super();
38          setLayout(new BoxLayout(this,BoxLayout.Y_AXIS));
39
40          zoomSlider = createNewSlider(1,20,1,Strings.CONTROL_ZOOM);
41          add(Box.createVerticalStrut(10)); //add 10px of extra space
42          stepSlider = createNewSlider(0,10,0,Strings.CONTROL_STEP);
43          add(Box.createVerticalStrut(10)); //add 10px of extra space
44          speedSlider = createNewSlider(0,100,0,Strings.CONTROL_SPEED);
45          add(Box.createVerticalStrut(10)); //add 10px of extra space
46
47          Box worldPanel = Box.createHorizontalBox();
48          add(worldPanel);
49          worldPanel.add(new JLabel(Strings.STORAGE_WORLD_TYPE));
50          ButtonGroup group = new ButtonGroup();
51          longButton = createNewButton(Strings.STORAGE_LONG,group,worldPanel);
52          arrayButton = createNewButton(Strings.STORAGE_ARRAY,group,worldPanel);
53          agingButton = createNewButton(Strings.STORAGE_AGING,group,worldPanel);
54          arrayButton.setSelected(true);
55          add(Box.createVerticalStrut(10)); //add 10px of extra space
56      }
57
58      public World initialiseWorld(Pattern p) throws PatternFormatException {
59          World result = null;
60          if (longButton.isSelected()) {
61              result = new PackedWorld();
62          }
63          else if (arrayButton.isSelected()) {
64              result = new ArrayWorld(p.getWidth(),p.getHeight());
65          }
66          else if (agingButton.isSelected()) {
67              result = new AgingWorld(p.getWidth(),p.getHeight());
68          }
69          if (result != null) p.initialise(result);
70          return result;
71      }

72  }
```

GamePanel.java

```
 0  package uk.ac.cam.jdw74.tick6;
 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 Dimension getPreferredSize() {
15          return new Dimension(width, height);
16      }
17
18      protected void paintComponent(Graphics g) {
19          if (current == null) return;
20          g.setColor(java.awt.Color.WHITE);
21          g.fillRect(0, 0, width, height);
22          current.draw(g, width, height);
23          if (zoom > 4) {
24              g.setColor(java.awt.Color.LIGHT_GRAY);
25              for (int i = 1; i < current.getWidth(); i++) {
26                  g.drawLine(i * zoom, 0, i * zoom, height);
27              }
28              for (int i = 1; i < current.getHeight(); i++) {
29                  g.drawLine(0, i * zoom, width, i * zoom);
30              }
31          }
32      }
33
34      private void computeSize() {
35          if (current == null)  return;
36          int newWidth = current.getWidth() * zoom;
37          int newHeight = current.getHeight() * zoom;
38          if (newWidth != width || newHeight != height) {
39              width = newWidth;
40              height = newHeight;
41              revalidate(); //trigger the GamePanel to re-layout its components
42          }
43      }
44
45      public void display(World w) {
46          current = w;
47          computeSize();
48          repaint();
49      }
50  }
```

GuiLife.java

```
 0  package uk.ac.cam.jdw74.tick6;
 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
15  public class GuiLife extends JFrame {
16      private PatternPanel patternPanel;
17      private ControlPanel controlPanel;
18      private GamePanel gamePanel;
19
20      public GuiLife() {
21          super("GuiLife");
22          setSize(640, 480);
23          setDefaultCloseOperation(EXIT_ON_CLOSE);
24          setLayout(new BorderLayout());
25          JComponent optionsPanel = createOptionsPanel();
26          add(optionsPanel, BorderLayout.WEST);
27          JComponent gamePanel = createGamePanel();
28          add(gamePanel, BorderLayout.CENTER);
29      }
30
31      private JComponent createOptionsPanel() {
32          Box result = Box.createVerticalBox();
33          result.add(createSourcePanel());
34          result.add(createPatternPanel());
35          result.add(createControlPanel());
36          return result;
37      }
38
39      private void addBorder(JComponent component, String title) {
40          Border etch = BorderFactory.createEtchedBorder(EtchedBorder.LOWERED);
41          Border tb = BorderFactory.createTitledBorder(etch,title);
42          component.setBorder(tb);
43      }
44
45      private JComponent createGamePanel() {
46          JPanel holder = new JPanel();
47          addBorder(holder,Strings.PANEL_GAMEVIEW);
48          GamePanel result = new GamePanel();
49          holder.add(result);
50          gamePanel = result;
51          return new JScrollPane(holder);
52      }
53
54      private JComponent createSourcePanel() {
55          SourcePanel result = new SourcePanel();
56          addBorder(result, Strings.PANEL_SOURCE);
57          return result;
58      }
59
60      private JComponent createPatternPanel() {
61          PatternPanel result = new PatternPanel();
62          addBorder(result, Strings.PANEL_PATTERN);
63          patternPanel = result;
64          return result;
65      }
66
67      private JComponent createControlPanel() {
68          ControlPanel result = new ControlPanel();
69          addBorder(result, Strings.PANEL_CONTROL);
70          controlPanel = result;
71          return result;
```

```
72      }
73
74      public static void main(String[] args) {
75          GuiLife gui = new GuiLife();
76          try {
77              String url =
78                  "http://www.cl.cam.ac.uk/teaching/current/ProgJava/life.txt";
79              List<Pattern> list = PatternLoader.loadFromURL(url);
80              gui.patternPanel.setPatterns(list);
81              World w = gui.controlPanel.initialiseWorld(list.get(1));
82              gui.gamePanel.display(w);
83          }
84          catch (IOException ioe) {}
85          catch (PatternFormatException e) {
86              System.out.println("Error: malformed pattern");
87          }
88          gui.setVisible(true);
89      }
90  }
```

HelloSwingWorld.java

```
 0  package uk.ac.cam.jdw74.tick6;
 1
 2  import javax.swing.JFrame;
 3  import javax.swing.JLabel;
 4
 5  public class HelloSwingWorld extends JFrame {
 6      HelloSwingWorld() {
 7          super("Hello Swing");
 8          setDefaultCloseOperation(EXIT_ON_CLOSE);
 9          JLabel text = new JLabel("Hello Swing");
10          add(text);
11          setSize(320,240);
12      }
13
14      public static void main(String[] args) {
15          HelloSwingWorld hello = new HelloSwingWorld();
16          hello.setVisible(true);
17      }
18  }
```

PackedLong.java

```
 0  package uk.ac.cam.jdw74.tick6;
 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.tick6;
 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.tick6;
 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.tick6;
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.tick6;
 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.tick6;
 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
 9  public class PatternPanel extends JPanel {
10
11      private JList guiList;
12
13      public PatternPanel() {
14          super();
15          setLayout(new BorderLayout());
16          guiList = new JList();
17          add(new JScrollPane(guiList));
18      }
19
20      public void setPatterns(List<Pattern> list) {
21          ArrayList<String> names = new ArrayList<String>();
22
23          for (Pattern p : list) {
24              names.add(p.getName() + " (" + p.getAuthor() + ")");
25          }
26
27          guiList.setListData(names.toArray());
28      }
29  }
```

SourcePanel.java

```
 0  package uk.ac.cam.jdw74.tick6;
 1
 2  import javax.swing.BoxLayout;
 3  import javax.swing.JPanel;
 4  import javax.swing.JRadioButton;
 5  import javax.swing.ButtonGroup;
 6
 7  public class SourcePanel extends JPanel {
 8
 9      public SourcePanel() {
10          super();
11          setLayout(new BoxLayout(this,BoxLayout.X_AXIS));
12          JRadioButton none = new JRadioButton(Strings.BUTTON_SOURCE_NONE, true);
13          JRadioButton file = new JRadioButton(Strings.BUTTON_SOURCE_FILE, true);
14          JRadioButton library =
15              new JRadioButton(Strings.BUTTON_SOURCE_LIBRARY, true);
16          JRadioButton fourStar =
17              new JRadioButton(Strings.BUTTON_SOURCE_FOURSTAR, true);
18          //add RadioButtons to this JPanel
19          add(none);
20          add(file);
21          add(library);
22          add(fourStar);
23          //create a ButtonGroup containing all four buttons
24          //Only one Button in a ButtonGroup can be selected at once
25          ButtonGroup group = new ButtonGroup();
26          group.add(none);
27          group.add(file);
28          group.add(library);
29          group.add(fourStar);
30      }
31  }
```

Strings.java

```
 0  package uk.ac.cam.jdw74.tick6;
 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  }
```

WorldImpl.java

```
 0  package uk.ac.cam.jdw74.tick6;
 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