tick5 submission from James Wood

Name	James Wood (jdw74)
College	ROBIN
Submission contents	uk/ac/cam/jdw74/tick5/AgingWorld.java uk/ac/cam/jdw74/tick5/ArrayWorld.java uk/ac/cam/jdw74/tick5/PackedLong.java uk/ac/cam/jdw74/tick5/PackedWorld.java uk/ac/cam/jdw74/tick5/Pattern.java uk/ac/cam/jdw74/tick5/PatternFormatException.java uk/ac/cam/jdw74/tick5/PatternLoader.java uk/ac/cam/jdw74/tick5/RefactorLife.java uk/ac/cam/jdw74/tick5/TestArrayWorld.java uk/ac/cam/jdw74/tick5/TestPackedWorld.java uk/ac/cam/jdw74/tick5/WorldImpl.java
Ticker	UNKNOWN
Ticker signature	

AgingWorld.java

```
 0  package uk.ac.cam.jdw74.tick5;
 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.tick5;
 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  }
```

PackedLong.java

```
 0  package uk.ac.cam.jdw74.tick5;
 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.tick5;
 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.tick5;
 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.tick5;
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.tick5;
 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  }
```

RefactorLife.java

```
 0  package uk.ac.cam.jdw74.tick5;
 1
 2  import java.util.List;
 3  import java.io.OutputStreamWriter;
 4  import java.io.Writer;
 5  import uk.ac.cam.acr31.life.World;
 6  import uk.ac.cam.acr31.life.WorldViewer;
 7
 8  class RefactorLife {
 9      public static void play(World world) throws Exception {
10          int userResponse = 0;
11          Writer w = new OutputStreamWriter(System.out);
12          WorldViewer viewer = new WorldViewer();
13          while (userResponse != 'q') {
14              world.print(w);
15              viewer.show(world);
16              userResponse = System.in.read();
17              world = world.nextGeneration(0);
18          }
19          viewer.close();
20      }
21
22      public static void main(String[] args) throws Exception {
23          if (args.length == 0) {
24              System.out.println("No argument given");
25              return;
26          }
27
28          boolean flagGiven = args[0].startsWith("--");
29          int flagOffset = flagGiven ? 1 : 0;
30          int mode;
31          if (flagGiven)
32              switch (args[0]) {
33              case "--array": mode = 0; break;
34              case  "--long": mode = 1; break;
35              case "--aging": mode = 2; break;
36              default:
37                  System.out.println("Invalid flag given");
38                  return;
39              }
40          else
41              mode = 0;
42
43          List<Pattern> ps;
44          String path = args[flagOffset + 0];
45          if (path.contains("://"))
46              ps = PatternLoader.loadFromURL(path);
47          else
48              ps = PatternLoader.loadFromDisk(path);
49
50          if (args.length == flagOffset + 1) {
51              int i = 0;
52              for (Pattern p : ps) {
53                  System.out.println(Integer.toString(i) + ") " + p.toString());
54                  i++;
55              }
56          }
57          else if (args.length == flagOffset + 2)
58              try {
59                  Pattern p = ps.get(Integer.parseInt(args[flagOffset + 1]));
60                  World world = null;
61                  switch (mode) {
62                  case 0: world = new ArrayWorld(p.getWidth(), p.getHeight());
63                      break;
64                  case 1: world = new PackedWorld();
65                      break;
66                  case 2: world = new AgingWorld(p.getWidth(), p.getHeight());
67                      break;
68                  }
69                  p.initialise(world);
70                  play(world);
71              }
```

```
72              catch (IndexOutOfBoundsException | NumberFormatException e) {
73                  System.out.println("Second argument is not a valid index");
74              }
75          else {
76              System.out.println("Too many arguments");
77              return;
78          }
79      }
80  }
```

TestArrayWorld.java

```
 0  package uk.ac.cam.jdw74.tick5;
 1
 2  import java.io.Writer;
 3  import java.awt.Graphics;
 4  import java.io.PrintWriter;
 5
 6  public class TestArrayWorld implements World {
 7
 8      private int generation;
 9      private int width;
10      private int height;
11      private boolean[][] cells;
12
13      public TestArrayWorld(int w, int h) {
14          width = w;
15          height = h;
16          generation = 0;
17          cells = new boolean[h][w];
18      }
19
20      protected TestArrayWorld(TestArrayWorld prev) {
21          width = prev.width;
22          height = prev.height;
23          generation = prev.generation;
24          cells = new boolean[prev.height][prev.width];
25      }
26
27      public boolean getCell(int col, int row) {
28          return 0 <= row && row < cells.length &&
29                 0 <= col && col < cells[row].length ?
30              cells[row][col] : false;
31      }
32      public void setCell(int col, int row, boolean alive) {
33          if (0 <= row && row < cells.length &&
34              0 <= col && col < cells[row].length)
35              cells[row][col] = alive;
36      }
37      public int getWidth() { return width; }
38      public int getHeight() { return height; }
39      public int getGeneration() { return generation; }
40      public int getPopulation() {
41          int n = 0;
42          for (int i = 0; i < height; i++)
43              for (int j = 0; j < width; j++)
44                  if (getCell(j, i)) n++;
45          return n;
46      }
47      public void print(Writer w) {
48          PrintWriter pw = new PrintWriter(w);
49
50          pw.println("-");
51          for (int row = 0; row < height; row++) {
52              for (int col = 0; col < width; col++)
53                  pw.print(getCell(col, row) ? "#" : "_");
54              pw.println();
55          }
56
57          pw.flush();
58      }
59      public void draw(Graphics g, int width, int height) { /*Leave empty*/ }
60
61      private int countNeighbours(int col, int row) {
62          return
63              (getCell(col - 1, row - 1) ? 1 : 0)
64            + (getCell(col    , row - 1) ? 1 : 0)
65            + (getCell(col + 1, row - 1) ? 1 : 0)
66            + (getCell(col - 1, row    ) ? 1 : 0)
67            + (getCell(col + 1, row    ) ? 1 : 0)
68            + (getCell(col - 1, row + 1) ? 1 : 0)
69            + (getCell(col    , row + 1) ? 1 : 0)
70            + (getCell(col + 1, row + 1) ? 1 : 0);
71      }
```

```
72
73      private boolean computeCell(int col, int row) {
74          int count = countNeighbours(col, row);
75          return count == 3 || (getCell(col, row) && count == 2);
76      }
77
78      private TestArrayWorld nextGeneration() {
79          //Construct a new TestArrayWorld object to hold the next generation:
80          TestArrayWorld world = new TestArrayWorld(this);
81          for (int i = 0; i < height; i++)
82              for (int j = 0; j < width; j++)
83                  world.setCell(j, i, computeCell(j, i));
84          return world;
85      }
86
87      public World nextGeneration(int log2StepSize) {
88          TestArrayWorld world = this;
89          for (int i = 0; i < 1 << log2StepSize; i++)
90              world = world.nextGeneration();
91          return world;
92      }
93  }
```

TestPackedWorld.java

```
 0  package uk.ac.cam.jdw74.tick5;
 1
 2  import java.io.Writer;
 3  import java.awt.Graphics;
 4  import java.io.PrintWriter;
 5
 6  public class TestPackedWorld implements World {
 7
 8      private int generation;
 9      private long cells;
10
11      public TestPackedWorld() {
12          generation = 0;
13          cells = 0;
14      }
15
16      protected TestPackedWorld(TestPackedWorld prev) {
17          generation = prev.generation;
18          cells = 0;
19      }
20
21      public boolean getCell(int col, int row) {
22          return 0 <= row && row < 8 && 0 <= col && col < 8 ?
23              PackedLong.get(cells, row * 8 + col) : false;
24      }
25      public void setCell(int col, int row, boolean alive) {
26          if (0 <= row && row < 8 && 0 <= col && col < 8)
27              cells = PackedLong.set(cells, row * 8 + col, alive);
28      }
29      public int getWidth() { return 8; }
30      public int getHeight() { return 8; }
31      public int getGeneration() { return generation; }
32      public int getPopulation() {
33          int n = 0;
34          for (int i = 0; i < 8; i++)
35              for (int j = 0; j < 8; j++)
36                  if (getCell(j, i)) n++;
37          return n;
38      }
39      public void print(Writer w) {
40          PrintWriter pw = new PrintWriter(w);
41
42          pw.println("-");
43          for (int row = 0; row < 8; row++) {
44              for (int col = 0; col < 8; col++)
45                  pw.print(getCell(col, row) ? "#" : "_");
46              pw.println();
47          }
48
49          pw.flush();
50      }
51      public void draw(Graphics g, int width, int height) { /*Leave empty*/ }
52
53      private int countNeighbours(int col, int row) {
54          return
55              (getCell(col - 1, row - 1) ? 1 : 0)
56            + (getCell(col    , row - 1) ? 1 : 0)
57            + (getCell(col + 1, row - 1) ? 1 : 0)
58            + (getCell(col - 1, row    ) ? 1 : 0)
59            + (getCell(col + 1, row    ) ? 1 : 0)
60            + (getCell(col - 1, row + 1) ? 1 : 0)
61            + (getCell(col    , row + 1) ? 1 : 0)
62            + (getCell(col + 1, row + 1) ? 1 : 0);
63      }
64
65      private boolean computeCell(int col, int row) {
66          int count = countNeighbours(col, row);
67          return count == 3 || (getCell(col, row) && count == 2);
68      }
69
70      private TestPackedWorld nextGeneration() {
71          //Construct a new TestPackedWorld object to hold the next generation:
```

```
72          TestPackedWorld world = new TestPackedWorld(this);
73          for (int i = 0; i < 8; i++)
74              for (int j = 0; j < 8; j++)
75                  world.setCell(j, i, computeCell(j, i));
76          return world;
77      }
78
79      public World nextGeneration(int log2StepSize) {
80          TestPackedWorld world = this;
81          for (int i = 0; i < 1 << log2StepSize; i++)
82              world = world.nextGeneration();
83          return world;
84      }
85  }
```

WorldImpl.java

```
 0  package uk.ac.cam.jdw74.tick5;
 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 }
```