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

```java
package uk.ac.cam.jdw74.tick5;

import java.awt.Color;

public class AgingWorld extends WorldImpl {
    private int[][] world;

    public AgingWorld(int width, int height) {
        super(width,height);
        world = new int[height][width];
        for (int y = 0; y < getHeight(); ++y) {
            for (int x = 0; x < getWidth(); ++x)
                world[y][x] = 1000;
        }
    }

    private AgingWorld(AgingWorld w) {
        super(w);
        world = new int[w.getHeight()][w.getWidth()];
        for (int y = 0; y < getHeight(); ++y) {
            for (int x = 0; x < getWidth(); ++x)
                world[y][x] = w.world[y][x]+1;
        }
    }

    @Override
    public boolean getCell(int x, int y) {
        return getCellAge(x,y) == 0;
    }

    @Override
    protected WorldImpl nextGeneration() {
        WorldImpl nextWorld = new AgingWorld(this);
        for (int row = 0; row < getHeight(); ++row) {
            for (int col = 0; col < getWidth(); ++col) {
                boolean nextLive = computeCell(col, row);
                nextWorld.setCell(col, row, nextLive);
            }
        }
        return nextWorld;
    }

    @Override
    public void setCell(int x, int y, boolean live) {
        if (y<0 || y>=getHeight()) return;
        if (x<0 || x>=getWidth()) return;
        if (live)
            world[y][x] = 0;
    }

    public int getCellAge(int x, int y) {
        if (y<0 || y>=getHeight()) return Integer.MAX_VALUE;
        if (x<0 || x>=getWidth()) return Integer.MAX_VALUE;
        return world[y][x];
    }

    @Override
    protected String getCellAsString(int x, int y) {
        int age = getCellAge(x,y);
        if (age > 9) return "_";
        if (age == 0) return "#";
        return age+"";
    }
}
```

ArrayWorld.java

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

public class ArrayWorld extends WorldImpl {
    private boolean[][] cells;

    public ArrayWorld(int w, int h) {
        super(w, h);
        this.cells = new boolean[h][w];
    }

    protected ArrayWorld(ArrayWorld prev) {
        super(prev);
        this.cells = new boolean[prev.getHeight()][prev.getWidth()];
    }

    @Override
    public boolean getCell(int col, int row) {
        return 0 <= row && row < getHeight() &&
               0 <= col && col < getWidth() ?
            cells[row][col] : false;
    }

    @Override
    public void setCell(int col, int row, boolean alive) {
        if (0 <= row && row < getHeight() &&
            0 <= col && col < getWidth())
            cells[row][col] = alive;
    }

    @Override
    protected ArrayWorld nextGeneration() {
        ArrayWorld world = new ArrayWorld(this);
        for (int i = 0; i < getHeight(); i++)
            for (int j = 0; j < getWidth(); j++)
                world.setCell(j, i, computeCell(j, i));
        return world;
    }
}
```

PackedLong.java

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

public class PackedLong {

    /*
    * Unpack and return the nth bit from the packed number at index position;
    * position counts from zero (representing the least significant bit)
    * up to 63 (representing the most significant bit).
    */
    public static boolean get(long packed, int position) {
        // set "check" to equal 1 if the "position" bit in "packed" is set to 1
        long check = packed >> position & 1L;
        return (check == 1L);
    }

    /*
    * Set the nth bit in the packed number to the value given
    * and return the new packed number
    */
    public static long set(long packed, int position, boolean value) {
        if (value) {
            packed |= 1L << position;
            // update the value "packed" with the bit at "position" set to 1
        }
        else {
            packed &= ~(1L << position);
            // update the value "packed" with the bit a "position" set to 0
        }
        return packed;
    }
}
```

PackedWorld.java

```java
package uk.ac.cam.jdw74.tick5;

public class PackedWorld extends WorldImpl {
    private int generation;
    private long cells;

    public PackedWorld() {
        super(8, 8);
        cells = 0;
    }

    protected PackedWorld(PackedWorld prev) {
        super(prev);
        cells = 0;
    }

    @Override
    public boolean getCell(int col, int row) {
        return 0 <= row && row < 8 && 0 <= col && col < 8 ?
            PackedLong.get(cells, row * 8 + col) : false;
    }

    @Override
    public void setCell(int col, int row, boolean alive) {
        if (0 <= row && row < 8 && 0 <= col && col < 8)
            cells = PackedLong.set(cells, row * 8 + col, alive);
    }

    @Override
    protected PackedWorld nextGeneration() {
        PackedWorld world = new PackedWorld(this);
        for (int i = 0; i < 8; i++)
            for (int j = 0; j < 8; j++)
                world.setCell(j, i, computeCell(j, i));
        return world;
    }
}
```

Pattern.java

```java
package uk.ac.cam.jdw74.tick5;

import java.text.ParseException;
import uk.ac.cam.acr31.life.World;

public class Pattern {

    private String name;
    private String author;
    private int width;
    private int height;
    private int startCol;
    private int startRow;
    private String cells;

    public String getName() { return name; }
    public void setName(String x) { name = x; }

    public String getAuthor() { return author; }
    public void setAuthor(String x) { author = x; }

    public int getWidth() { return width; }
    public void setWidth(int x) { width = x; }

    public int getHeight() { return height; }
    public void setHeight(int x) { height = x; }

    public int getStartCol() { return startCol; }
    public void setStartCol(int x) { startCol = x; }

    public int getStartRow() { return startRow; }
    public void setStartRow(int x) { startRow = x; }

    public String getCells() { return cells; }
    public void setCells(String x) { cells = x; }

    public Pattern(String format) throws PatternFormatException {
        String[] parts = format.split(":");
        if (parts.length < 7)
            throw new PatternFormatException("Too few arguments");
        if (parts.length > 7)
            throw new PatternFormatException("Too many arguments");

        name = parts[0];
        author = parts[1];
        try {
            width = Integer.parseInt(parts[2]);
            if (width <= 0) throw new NumberFormatException();
        }
        catch (NumberFormatException e) {
            throw new PatternFormatException(
                "Width argument not a positive integer");
        }
        try {
            height = Integer.parseInt(parts[3]);
            if (height <= 0) throw new NumberFormatException();
        }
        catch (NumberFormatException e) {
            throw new PatternFormatException(
                "Height argument not a positive integer");
        }
        try {
            startCol = Integer.parseInt(parts[4]);
        }
        catch (NumberFormatException e) {
            throw new PatternFormatException(
                "x coördinate not an integer");
        }
        try {
            startRow = Integer.parseInt(parts[5]);
        }
        catch (NumberFormatException e) {
            throw new PatternFormatException(
                "y coördinate not an integer");
        }
        cells = parts[6];
    }

    public void initialise(World world) throws PatternFormatException {
        String[] rows = cells.split(" ");
        char[][] values = new char[rows.length][];
        for (int i = 0; i < rows.length; i++)
            values[i] = rows[i].toCharArray();

        for (int i = 0; i < values.length; i++)
            for (int j = 0; j < values[i].length; j++)
                if (" 01".indexOf(values[i][j]) == -1)
                    throw new PatternFormatException(
                    "Pattern contains characters other than ‘0’, ‘1’ and ‘ ’");
                else
                    world.setCell(startCol + j, startRow + i,
                                  values[i][j] == '1');
    }

    @Override
    public String toString() {
        return name + ":" + author + ":" + width + ":" + height + ":" +
            startCol + ":" + startRow + ":" + cells;
    }
}
```

PatternFormatException.java

```java
package uk.ac.cam.jdw74.tick5;

public class PatternFormatException extends Exception {
    public PatternFormatException(String message) {
        super(message);
    }
}
```

PatternLoader.java

```java
package uk.ac.cam.jdw74.tick5;

import java.io.Reader;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.FileReader;
import java.util.List;
import java.util.LinkedList;
import java.net.URL;
import java.net.URLConnection;

public class PatternLoader {

    public static List<Pattern> load(Reader r) throws IOException {
        BufferedReader buff = new BufferedReader(r);
        List<Pattern> results = new LinkedList<>();

        String line;
        while ((line = buff.readLine()) != null)
            try {
                results.add(new Pattern(line));
            }
            catch (PatternFormatException e) {}

        return results;
    }

    public static List<Pattern> loadFromURL(String url) throws IOException {
        URL destination = new URL(url);
        URLConnection conn = destination.openConnection();
        return load(new InputStreamReader(conn.getInputStream()));
    }

    public static List<Pattern> loadFromDisk(String filename)
        throws IOException {
        return load(new FileReader(filename));
    }
}
```

RefactorLife.java

```java
package uk.ac.cam.jdw74.tick5;

import java.util.List;
import java.io.OutputStreamWriter;
import java.io.Writer;
import uk.ac.cam.acr31.life.World;
import uk.ac.cam.acr31.life.WorldViewer;

class RefactorLife {
    public static void play(World world) throws Exception {
        int userResponse = 0;
        Writer w = new OutputStreamWriter(System.out);
        WorldViewer viewer = new WorldViewer();
        while (userResponse != 'q') {
            world.print(w);
            viewer.show(world);
            userResponse = System.in.read();
            world = world.nextGeneration(0);
        }
        viewer.close();
    }

    public static void main(String[] args) throws Exception {
        if (args.length == 0) {
            System.out.println("No argument given");
            return;
        }

        boolean flagGiven = args[0].startsWith("--");
        int flagOffset = flagGiven ? 1 : 0;
        int mode;
        if (flagGiven)
            switch (args[0]) {
            case "--array": mode = 0; break;
            case  "--long": mode = 1; break;
            case "--aging": mode = 2; break;
            default:
                System.out.println("Invalid flag given");
                return;
            }
        else
            mode = 0;

        List<Pattern> ps;
        String path = args[flagOffset + 0];
        if (path.contains("://"))
            ps = PatternLoader.loadFromURL(path);
        else
            ps = PatternLoader.loadFromDisk(path);

        if (args.length == flagOffset + 1) {
            int i = 0;
            for (Pattern p : ps) {
                System.out.println(Integer.toString(i) + ") " + p.toString());
                i++;
            }
        }
        else if (args.length == flagOffset + 2)
            try {
                Pattern p = ps.get(Integer.parseInt(args[flagOffset + 1]));
                World world = null;
                switch (mode) {
                case 0: world = new ArrayWorld(p.getWidth(), p.getHeight());
                    break;
                case 1: world = new PackedWorld();
                    break;
                case 2: world = new AgingWorld(p.getWidth(), p.getHeight());
                    break;
                }
                p.initialise(world);
                play(world);
            }
            catch (IndexOutOfBoundsException | NumberFormatException e) {
                System.out.println("Second argument is not a valid index");
            }
        else {
            System.out.println("Too many arguments");
            return;
        }
    }
}
```

TestArrayWorld.java

```java
package uk.ac.cam.jdw74.tick5;

import java.io.Writer;
import java.awt.Graphics;
import java.io.PrintWriter;

public class TestArrayWorld implements World {

    private int generation;
    private int width;
    private int height;
    private boolean[][] cells;

    public TestArrayWorld(int w, int h) {
        width = w;
        height = h;
        generation = 0;
        cells = new boolean[h][w];
    }

    protected TestArrayWorld(TestArrayWorld prev) {
        width = prev.width;
        height = prev.height;
        generation = prev.generation;
        cells = new boolean[prev.height][prev.width];
    }

    public boolean getCell(int col, int row) {
        return 0 <= row && row < cells.length &&
               0 <= col && col < cells[row].length ?
            cells[row][col] : false;
    }
    public void setCell(int col, int row, boolean alive) {
        if (0 <= row && row < cells.length &&
            0 <= col && col < cells[row].length)
            cells[row][col] = alive;
    }
    public int getWidth() { return width; }
    public int getHeight() { return height; }
    public int getGeneration() { return generation; }
    public int getPopulation() {
        int n = 0;
        for (int i = 0; i < height; i++)
            for (int j = 0; j < width; j++)
                if (getCell(j, i)) n++;
        return n;
    }
    public void print(Writer w) {
        PrintWriter pw = new PrintWriter(w);

        pw.println("-");
        for (int row = 0; row < height; row++) {
            for (int col = 0; col < width; col++)
                pw.print(getCell(col, row) ? "#" : "_");
            pw.println();
        }

        pw.flush();
    }
    public void draw(Graphics g, int width, int height) { /*Leave empty*/ }

    private int countNeighbours(int col, int row) {
        return
            (getCell(col - 1, row - 1) ? 1 : 0)
          + (getCell(col    , row - 1) ? 1 : 0)
          + (getCell(col + 1, row - 1) ? 1 : 0)
          + (getCell(col - 1, row    ) ? 1 : 0)
          + (getCell(col + 1, row    ) ? 1 : 0)
          + (getCell(col - 1, row + 1) ? 1 : 0)
          + (getCell(col    , row + 1) ? 1 : 0)
          + (getCell(col + 1, row + 1) ? 1 : 0);
    }

    private boolean computeCell(int col, int row) {
        int count = countNeighbours(col, row);
        return count == 3 || (getCell(col, row) && count == 2);
    }

    private TestArrayWorld nextGeneration() {
        //Construct a new TestArrayWorld object to hold the next generation:
        TestArrayWorld world = new TestArrayWorld(this);
        for (int i = 0; i < height; i++)
            for (int j = 0; j < width; j++)
                world.setCell(j, i, computeCell(j, i));
        return world;
    }

    public World nextGeneration(int log2StepSize) {
        TestArrayWorld world = this;
        for (int i = 0; i < 1 << log2StepSize; i++)
            world = world.nextGeneration();
        return world;
    }
}
```

TestPackedWorld.java

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

import java.io.Writer;
import java.awt.Graphics;
import java.io.PrintWriter;

public class TestPackedWorld implements World {

    private int generation;
    private long cells;

    public TestPackedWorld() {
        generation = 0;
        cells = 0;
    }

    protected TestPackedWorld(TestPackedWorld prev) {
        generation = prev.generation;
        cells = 0;
    }

    public boolean getCell(int col, int row) {
        return 0 <= row && row < 8 && 0 <= col && col < 8 ?
            PackedLong.get(cells, row * 8 + col) : false;
    }
    public void setCell(int col, int row, boolean alive) {
        if (0 <= row && row < 8 && 0 <= col && col < 8)
            cells = PackedLong.set(cells, row * 8 + col, alive);
    }
    public int getWidth() { return 8; }
    public int getHeight() { return 8; }
    public int getGeneration() { return generation; }
    public int getPopulation() {
        int n = 0;
        for (int i = 0; i < 8; i++)
            for (int j = 0; j < 8; j++)
                if (getCell(j, i)) n++;
        return n;
    }
    public void print(Writer w) {
        PrintWriter pw = new PrintWriter(w);

        pw.println("-");
        for (int row = 0; row < 8; row++) {
            for (int col = 0; col < 8; col++)
                pw.print(getCell(col, row) ? "#" : "_");
            pw.println();
        }

        pw.flush();
    }
    public void draw(Graphics g, int width, int height) { /*Leave empty*/ }

    private int countNeighbours(int col, int row) {
        return
            (getCell(col - 1, row - 1) ? 1 : 0)
          + (getCell(col    , row - 1) ? 1 : 0)
          + (getCell(col + 1, row - 1) ? 1 : 0)
          + (getCell(col - 1, row    ) ? 1 : 0)
          + (getCell(col + 1, row    ) ? 1 : 0)
          + (getCell(col - 1, row + 1) ? 1 : 0)
          + (getCell(col    , row + 1) ? 1 : 0)
          + (getCell(col + 1, row + 1) ? 1 : 0);
    }

    private boolean computeCell(int col, int row) {
        int count = countNeighbours(col, row);
        return count == 3 || (getCell(col, row) && count == 2);
    }

    private TestPackedWorld nextGeneration() {
        //Construct a new TestPackedWorld object to hold the next generation:
```

```
        TestPackedWorld world = new TestPackedWorld(this);
        for (int i = 0; i < 8; i++)
            for (int j = 0; j < 8; j++)
                world.setCell(j, i, computeCell(j, i));
        return world;
    }

    public World nextGeneration(int log2StepSize) {
        TestPackedWorld world = this;
        for (int i = 0; i < 1 << log2StepSize; i++)
            world = world.nextGeneration();
        return world;
    }
}
```

WorldImpl.java

```java
package uk.ac.cam.jdw74.tick5;

import java.awt.Color;
import java.awt.Graphics;
import java.io.Writer;
import java.io.PrintWriter;
import uk.ac.cam.acr31.life.World;

public abstract class WorldImpl implements World {

    private int width;
    private int height;
    private int generation;

    protected WorldImpl(int width, int height) {
        this.width = width;
        this.height = height;
        this.generation = 0;
    }

    protected WorldImpl(WorldImpl prev) {
        this.width = prev.width;
        this.height = prev.height;
        this.generation = prev.generation + 1;
    }

    public int getWidth() { return this.width; }

    public int getHeight() { return this.height; }

    public int getGeneration() { return this.generation; }

    public int getPopulation() { return 0; }

    protected String getCellAsString(int col,int row) {
        return getCell(col,row) ? "#" : "_";
    }

    protected Color getCellAsColour(int col,int row) {
        return getCell(col,row) ? Color.BLACK : Color.WHITE;
    }
    public void draw(Graphics g,int width, int height) {
        int worldWidth = getWidth();
        int worldHeight = getHeight();

        double colScale = (double)width/(double)worldWidth;
        double rowScale = (double)height/(double)worldHeight;

        for(int col=0; col<worldWidth; ++col) {
            for(int row=0; row<worldHeight; ++row) {
                int colPos = (int)(col*colScale);
                int rowPos = (int)(row*rowScale);
                int nextCol = (int)((col+1)*colScale);
                int nextRow = (int)((row+1)*rowScale);

                if (g.hitClip(colPos,rowPos,nextCol-colPos,nextRow-rowPos)) {
                    g.setColor(getCellAsColour(col, row));
                    g.fillRect(colPos,rowPos,nextCol-colPos,nextRow-rowPos);
                }
            }
        }
    }

    public World nextGeneration(int log2StepSize) {
        WorldImpl world = this;
        for (int i = 0; i < 1 << log2StepSize; i++)
            world = world.nextGeneration();
        return world;
    }

    public void print(Writer w) {
        PrintWriter pw = new PrintWriter(w);

        pw.println("-");
        for (int row = 0; row < this.height; row++) {
            for (int col = 0; col < this.width; col++)
                pw.print(getCellAsString(col, row));
            pw.println();
        }

        pw.flush();
    }

    protected int countNeighbours(int col, int row) {
        return
            (getCell(col - 1, row - 1) ? 1 : 0)
          + (getCell(col    , row - 1) ? 1 : 0)
          + (getCell(col + 1, row - 1) ? 1 : 0)
          + (getCell(col - 1, row    ) ? 1 : 0)
          + (getCell(col + 1, row    ) ? 1 : 0)
          + (getCell(col - 1, row + 1) ? 1 : 0)
          + (getCell(col    , row + 1) ? 1 : 0)
          + (getCell(col + 1, row + 1) ? 1 : 0);
    }

    protected boolean computeCell(int col, int row) {
        int count = countNeighbours(col, row);
        return count == 3 || (getCell(col, row) && count == 2);
    }

    // Will be implemented by child class.
    // Return true if cell (col,row) is alive.
    public abstract boolean getCell(int col,int row);

    // Will be implemented by child class. Set a cell to be live or dead.
    public abstract void setCell(int col, int row, boolean alive);

    // Will be implemented by child class. Step forward one generation.
    protected abstract WorldImpl nextGeneration();
}
```