tick3 submission from James Wood

Name	James Wood (jdw74)
College	ROBIN
Submission contents	uk/ac/cam/jdw74/tick3/ArrayLife.java uk/ac/cam/jdw74/tick3/FibonacciCache.java uk/ac/cam/jdw74/tick3/PackedLong.java uk/ac/cam/jdw74/tick3/Pattern.java uk/ac/cam/jdw74/tick3/PatternLife.java uk/ac/cam/jdw74/tick3/ReferenceTest.java uk/ac/cam/jdw74/tick3/StringArrayLife.java
Ticker	UNKNOWN
Ticker signature	

ArrayLife.java

```java
package uk.ac.cam.jdw74.tick3;

class ArrayLife {
    public static void print(boolean[][] world) {
        System.out.println("-");
        for (int row = 0; row < world.length; row++) {
            for (int col = 0; col < world[row].length; col++) {
                System.out.print(getCell(world, col, row) ? "#" : "_");
            }
            System.out.println();
        }
    }

    public static boolean getCell(boolean[][] world, int col, int row) {
        return 0 <= row && row < world.length &&
               0 <= col && col < world[row].length ?
            world[row][col] : false;
    }

    public static void setCell(boolean[][] world, int col, int row, boolean value) {
        if (0 <= row && row < world.length &&
            0 <= col && col < world[row].length)
            world[row][col] = value;
    }

    public static int countNeighbours(boolean[][] world, int col, int row) {
        return
            (getCell(world, col - 1, row - 1) ? 1 : 0)
          + (getCell(world, col    , row - 1) ? 1 : 0)
          + (getCell(world, col + 1, row - 1) ? 1 : 0)
          + (getCell(world, col - 1, row    ) ? 1 : 0)
          + (getCell(world, col + 1, row    ) ? 1 : 0)
          + (getCell(world, col - 1, row + 1) ? 1 : 0)
          + (getCell(world, col    , row + 1) ? 1 : 0)
          + (getCell(world, col + 1, row + 1) ? 1 : 0);
    }

    public static boolean computeCell(boolean[][] world, int col, int row) {
        int count = countNeighbours(world, col, row);
        return count == 3 || (getCell(world, col, row) && count == 2);
    }

    public static boolean[][] nextGeneration(boolean[][] world) {
        boolean[][] nextWorld = new boolean[world.length][world[0].length];
        for (int row = 0; row < world.length; row++)
            for (int col = 0; col < world[row].length; col++)
                setCell(nextWorld, col, row,
                        computeCell(world, col, row));
        return nextWorld;
    }

    public static void play(boolean[][] world) throws Exception {
        int userResponse = 0;
        while (userResponse != 'q') {
            print(world);
            userResponse = System.in.read();
            world = nextGeneration(world);
        }
    }

    public static void main(String[] args) throws Exception {
        int size = Integer.parseInt(args[0]);
        long initial = Long.decode(args[1]);
        boolean[][] world = new boolean[size][size];
        for(int i = 0; i < 8; i++)
            for(int j = 0; j < 8; j++)
                world[i + size / 2 - 4][j + size / 2 - 4] =
                    PackedLong.get(initial, i * 8 + j);
        play(world);
    }
}
```

FibonacciCache.java

```java
package uk.ac.cam.jdw74.tick3;

public class FibonacciCache {
    public static long[] fib = new long[20];

    public static void store() {
        if (fib.length == 0)
            return;
        fib[0] = 1L;
        if (fib.length == 1)
            return;
        fib[1] = 1L;
        for (int i = 2; i < fib.length; i++)
            fib[i] = fib[i - 2] + fib[i - 1];
    }

    public static void reset() {
        for (int i = 0; i < fib.length; i++)
            fib[i] = 0L;
    }

    public static long get(int i) {
        if (i < 0 || fib.length <= i)
            return -1L;
        else
            return fib[i];
    }
}
```

PackedLong.java

```java
package uk.ac.cam.jdw74.tick3;

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;
    }
}
```

Pattern.java

```java
package uk.ac.cam.jdw74.tick3;

import java.text.ParseException;

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 ParseException {
        String[] parts = format.split(":");
        if (parts.length != 7)
            throw new ParseException("Incorrect pattern format", 0);

        name = parts[0];
        author = parts[1];
        width = Integer.parseInt(parts[2]);
        height = Integer.parseInt(parts[3]);
        startCol = Integer.parseInt(parts[4]);
        startRow = Integer.parseInt(parts[5]);
        cells = parts[6];
    }

    public void initialise(boolean[][] world) {
        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++)
                world[startRow + i][startCol + j] = values[i][j] == '1';
    }
}
```

PatternLife.java

```java
package uk.ac.cam.jdw74.tick3;

class PatternLife {
    public static void print(boolean[][] world) {
        System.out.println("-");
        for (int row = 0; row < world.length; row++) {
            for (int col = 0; col < world[row].length; col++) {
                System.out.print(getCell(world, col, row) ? "#" : "_");
            }
            System.out.println();
        }
    }

    public static boolean getCell(boolean[][] world, int col, int row) {
        return 0 <= row && row < world.length &&
               0 <= col && col < world[row].length ?
            world[row][col] : false;
    }

    public static void setCell(boolean[][] world, int col, int row, boolean value) {
        if (0 <= row && row < world.length &&
            0 <= col && col < world[row].length)
            world[row][col] = value;
    }

    public static int countNeighbours(boolean[][] world, int col, int row) {
        return
            (getCell(world, col - 1, row - 1) ? 1 : 0)
          + (getCell(world, col    , row - 1) ? 1 : 0)
          + (getCell(world, col + 1, row - 1) ? 1 : 0)
          + (getCell(world, col - 1, row    ) ? 1 : 0)
          + (getCell(world, col + 1, row    ) ? 1 : 0)
          + (getCell(world, col - 1, row + 1) ? 1 : 0)
          + (getCell(world, col    , row + 1) ? 1 : 0)
          + (getCell(world, col + 1, row + 1) ? 1 : 0);
    }

    public static boolean computeCell(boolean[][] world, int col, int row) {
        int count = countNeighbours(world, col, row);
        return count == 3 || (getCell(world, col, row) && count == 2);
    }

    public static boolean[][] nextGeneration(boolean[][] world) {
        boolean[][] nextWorld = new boolean[world.length][world[0].length];
        for (int row = 0; row < world.length; row++)
            for (int col = 0; col < world[row].length; col++)
                setCell(nextWorld, col, row,
                        computeCell(world, col, row));
        return nextWorld;
    }

    public static void play(boolean[][] world) throws Exception {
        int userResponse = 0;
        while (userResponse != 'q') {
            print(world);
            userResponse = System.in.read();
            world = nextGeneration(world);
        }
    }

    public static void main(String[] args) throws Exception {
         Pattern p = new Pattern(args[0]);
         boolean[][] world = new boolean[p.getHeight()][p.getWidth()];
         p.initialise(world);
         play(world);
    }
}
```

ReferenceTest.java

```java
package uk.ac.cam.jdw74.tick3;

public class ReferenceTest {
    public static void main(String[] args) {
        int[][] i = new int[2][2];
        int[][] j = {i[1],{1,2,3},{4,5,6,7}};
        int[][][] k = {i,j};

        System.out.println(k[0][1][0]++); // 0; i = {{0,0},{1,0}}
        System.out.println(++k[1][0][0]); // 2; i = {{0,0},{2,0}}
        System.out.println(i[1][0]);      // 2
        System.out.println(--j[0][0]);    // 1; i = {{0,0},{1,0}}
    }
}
```

StringArrayLife.java

```java
package uk.ac.cam.jdw74.tick3;

class StringArrayLife {
    public static void print(boolean[][] world) {
        System.out.println("-");
        for (int row = 0; row < world.length; row++) {
            for (int col = 0; col < world[row].length; col++) {
                System.out.print(getCell(world, col, row) ? "#" : "_");
            }
            System.out.println();
        }
    }

    public static boolean getCell(boolean[][] world, int col, int row) {
        return 0 <= row && row < world.length &&
               0 <= col && col < world[row].length ?
            world[row][col] : false;
    }

    public static void setCell(boolean[][] world, int col, int row, boolean value) {
        if (0 <= row && row < world.length &&
            0 <= col && col < world[row].length)
            world[row][col] = value;
    }

    public static int countNeighbours(boolean[][] world, int col, int row) {
        return
            (getCell(world, col - 1, row - 1) ? 1 : 0)
          + (getCell(world, col    , row - 1) ? 1 : 0)
          + (getCell(world, col + 1, row - 1) ? 1 : 0)
          + (getCell(world, col - 1, row    ) ? 1 : 0)
          + (getCell(world, col + 1, row    ) ? 1 : 0)
          + (getCell(world, col - 1, row + 1) ? 1 : 0)
          + (getCell(world, col    , row + 1) ? 1 : 0)
          + (getCell(world, col + 1, row + 1) ? 1 : 0);
    }

    public static boolean computeCell(boolean[][] world, int col, int row) {
        int count = countNeighbours(world, col, row);
        return count == 3 || (getCell(world, col, row) && count == 2);
    }

    public static boolean[][] nextGeneration(boolean[][] world) {
        boolean[][] nextWorld = new boolean[world.length][world[0].length];
        for (int row = 0; row < world.length; row++)
            for (int col = 0; col < world[row].length; col++)
                setCell(nextWorld, col, row,
                        computeCell(world, col, row));
        return nextWorld;
    }

    public static void play(boolean[][] world) throws Exception {
        int userResponse = 0;
        while (userResponse != 'q') {
            print(world);
            userResponse = System.in.read();
            world = nextGeneration(world);
        }
    }

    public static void main(String[] args) throws Exception {

        String formatString = args[0];
        String[] parts = formatString.split(":");
        if (parts.length != 7) {
            System.out.println("Error: incorrect format");
            return;
        }

        String name = parts[0];
        String author = parts[1];
        int width = Integer.parseInt(parts[2]);
        int height = Integer.parseInt(parts[3]);
        int x = Integer.parseInt(parts[4]);
        int y = Integer.parseInt(parts[5]);
        String pattern = parts[6];

        String[] rows = pattern.split(" ");
        int patternHeight = rows.length;
        int patternWidth = rows[0].length();
        char[][] cells = new char[patternHeight][patternWidth];
        for (int i = 0; i < cells.length; i++)
            cells[i] = rows[i].toCharArray();

        boolean[][] world = new boolean[height][width];

        for (int i = 0; i < patternHeight; i++)
            for (int j = 0; j < patternWidth; j++)
                setCell(world, x + j, y + i, cells[i][j] == '1');

        play(world);
    }
}
```