

---

## 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

```
1  package uk.ac.cam.jdw74.tick3;
2
3  class ArrayLife {
4      public static void print(boolean[][] world) {
5          System.out.println("-");
6          for (int row = 0; row < world.length; row++) {
7              for (int col = 0; col < world[row].length; col++) {
8                  System.out.print(getCell(world, col, row) ? "#" : "_");
9              }
10             System.out.println();
11         }
12     }
13
14     public static boolean getCell(boolean[][] world, int col, int row) {
15         return 0 <= row && row < world.length &&
16             0 <= col && col < world[row].length ?
17             world[row][col] : false;
18     }
19
20     public static void setCell(boolean[][] world, int col, int row, boolean value) {
21         if (0 <= row && row < world.length &&
22             0 <= col && col < world[row].length)
23             world[row][col] = value;
24     }
25
26     public static int countNeighbours(boolean[][] world, int col, int row) {
27         return
28             (getCell(world, col - 1, row - 1) ? 1 : 0)
29             + (getCell(world, col, row - 1) ? 1 : 0)
30             + (getCell(world, col + 1, row - 1) ? 1 : 0)
31             + (getCell(world, col - 1, row) ? 1 : 0)
32             + (getCell(world, col + 1, row) ? 1 : 0)
33             + (getCell(world, col - 1, row + 1) ? 1 : 0)
34             + (getCell(world, col, row + 1) ? 1 : 0)
35             + (getCell(world, col + 1, row + 1) ? 1 : 0);
36     }
37
38     public static boolean computeCell(boolean[][] world, int col, int row) {
39         int count = countNeighbours(world, col, row);
40         return count == 3 || (getCell(world, col, row) && count == 2);
41     }
42
43     public static boolean[][] nextGeneration(boolean[][] world) {
44         boolean[][] nextWorld = new boolean[world.length][world[0].length];
45         for (int row = 0; row < world.length; row++)
46             for (int col = 0; col < world[row].length; col++)
47                 setCell(nextWorld, col, row,
48                     computeCell(world, col, row));
49         return nextWorld;
50     }
51
52     public static void play(boolean[][] world) throws Exception {
53         int userResponse = 0;
54         while (userResponse != 'q') {
55             print(world);
56             userResponse = System.in.read();
57             world = nextGeneration(world);
58         }
59     }
60
61     public static void main(String[] args) throws Exception {
62         int size = Integer.parseInt(args[0]);
63         long initial = Long.decode(args[1]);
64         boolean[][] world = new boolean[size][size];
65         for(int i = 0; i < 8; i++)
66             for(int j = 0; j < 8; j++)
67                 world[i + size / 2 - 4][j + size / 2 - 4] =
68                     PackedLong.get(initial, i * 8 + j);
69         play(world);
70     }
71 }
```

---

# FibonacciCache.java

```
0  package uk.ac.cam.jdw74.tick3;
1
2  public class FibonacciCache {
3      public static long[] fib = new long[20];
4
5      public static void store() {
6          if (fib.length == 0)
7              return;
8          fib[0] = 1L;
9          if (fib.length == 1)
10             return;
11          fib[1] = 1L;
12          for (int i = 2; i < fib.length; i++)
13              fib[i] = fib[i - 2] + fib[i - 1];
14      }
15
16      public static void reset() {
17          for (int i = 0; i < fib.length; i++)
18              fib[i] = 0L;
19      }
20
21      public static long get(int i) {
22          if (i < 0 || fib.length <= i)
23              return -1L;
24          else
25              return fib[i];
26      }
27  }
```

# PackedLong.java

```
0  package uk.ac.cam.jdw74.tick3;
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 }
```

---

# Pattern.java

```
0  package uk.ac.cam.jdw74.tick3;
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 }
```

---

# PatternLife.java

```
0  package uk.ac.cam.jdw74.tick3;
1
2  class PatternLife {
3      public static void print(boolean[][] world) {
4          System.out.println("-");
5          for (int row = 0; row < world.length; row++) {
6              for (int col = 0; col < world[row].length; col++) {
7                  System.out.print(getCell(world, col, row) ? "#" : "_");
8              }
9              System.out.println();
10         }
11     }
12
13     public static boolean getCell(boolean[][] world, int col, int row) {
14         return 0 <= row && row < world.length &&
15             0 <= col && col < world[row].length ?
16             world[row][col] : false;
17     }
18
19     public static void setCell(boolean[][] world, int col, int row, boolean value) {
20         if (0 <= row && row < world.length &&
21             0 <= col && col < world[row].length)
22             world[row][col] = value;
23     }
24
25     public static int countNeighbours(boolean[][] world, int col, int row) {
26         return
27             (getCell(world, col - 1, row - 1) ? 1 : 0)
28             + (getCell(world, col, row - 1) ? 1 : 0)
29             + (getCell(world, col + 1, row - 1) ? 1 : 0)
30             + (getCell(world, col - 1, row) ? 1 : 0)
31             + (getCell(world, col + 1, row) ? 1 : 0)
32             + (getCell(world, col - 1, row + 1) ? 1 : 0)
33             + (getCell(world, col, row + 1) ? 1 : 0)
34             + (getCell(world, col + 1, row + 1) ? 1 : 0);
35     }
36
37     public static boolean computeCell(boolean[][] world, int col, int row) {
38         int count = countNeighbours(world, col, row);
39         return count == 3 || (getCell(world, col, row) && count == 2);
40     }
41
42     public static boolean[][] nextGeneration(boolean[][] world) {
43         boolean[][] nextWorld = new boolean[world.length][world[0].length];
44         for (int row = 0; row < world.length; row++)
45             for (int col = 0; col < world[row].length; col++)
46                 setCell(nextWorld, col, row,
47                     computeCell(world, col, row));
48         return nextWorld;
49     }
50
51     public static void play(boolean[][] world) throws Exception {
52         int userResponse = 0;
53         while (userResponse != 'q') {
54             print(world);
55             userResponse = System.in.read();
56             world = nextGeneration(world);
57         }
58     }
59
60     public static void main(String[] args) throws Exception {
61         Pattern p = new Pattern(args[0]);
62         boolean[][] world = new boolean[p.getHeight()][p.getWidth()];
63         p.initialise(world);
64         play(world);
65     }
66 }
```

---

# ReferenceTest.java

```
0  package uk.ac.cam.jdw74.tick3;
1
2  public class ReferenceTest {
3      public static void main(String[] args) {
4          int[][] i = new int[2][2];
5          int[][] j = {i[1],{1,2,3},{4,5,6,7}};
6          int[][][] k = {i,j};
7
8          System.out.println(k[0][1][0]++); // 0; i = {{0,0},{1,0}}
9          System.out.println(++k[1][0][0]); // 2; i = {{0,0},{2,0}}
10         System.out.println(i[1][0]);      // 2
11         System.out.println(--j[0][0]);     // 1; i = {{0,0},{1,0}}
12     }
13 }
```

---

# StringArrayLife.java

```
1  package uk.ac.cam.jdw74.tick3;
2
3  class StringArrayLife {
4      public static void print(boolean[][] world) {
5          System.out.println("-");
6          for (int row = 0; row < world.length; row++) {
7              for (int col = 0; col < world[row].length; col++) {
8                  System.out.print(getCell(world, col, row) ? "#" : "_");
9              }
10             System.out.println();
11         }
12     }
13
14     public static boolean getCell(boolean[][] world, int col, int row) {
15         return 0 <= row && row < world.length &&
16             0 <= col && col < world[row].length ?
17             world[row][col] : false;
18     }
19
20     public static void setCell(boolean[][] world, int col, int row, boolean value) {
21         if (0 <= row && row < world.length &&
22             0 <= col && col < world[row].length)
23             world[row][col] = value;
24     }
25
26     public static int countNeighbours(boolean[][] world, int col, int row) {
27         return
28             (getCell(world, col - 1, row - 1) ? 1 : 0)
29             + (getCell(world, col, row - 1) ? 1 : 0)
30             + (getCell(world, col + 1, row - 1) ? 1 : 0)
31             + (getCell(world, col - 1, row) ? 1 : 0)
32             + (getCell(world, col + 1, row) ? 1 : 0)
33             + (getCell(world, col - 1, row + 1) ? 1 : 0)
34             + (getCell(world, col, row + 1) ? 1 : 0)
35             + (getCell(world, col + 1, row + 1) ? 1 : 0);
36     }
37
38     public static boolean computeCell(boolean[][] world, int col, int row) {
39         int count = countNeighbours(world, col, row);
40         return count == 3 || (getCell(world, col, row) && count == 2);
41     }
42
43     public static boolean[][] nextGeneration(boolean[][] world) {
44         boolean[][] nextWorld = new boolean[world.length][world[0].length];
45         for (int row = 0; row < world.length; row++)
46             for (int col = 0; col < world[row].length; col++)
47                 setCell(nextWorld, col, row,
48                     computeCell(world, col, row));
49         return nextWorld;
50     }
51
52     public static void play(boolean[][] world) throws Exception {
53         int userResponse = 0;
54         while (userResponse != 'q') {
55             print(world);
56             userResponse = System.in.read();
57             world = nextGeneration(world);
58         }
59     }
60
61     public static void main(String[] args) throws Exception {
62         String formatString = args[0];
63         String[] parts = formatString.split(":");
64         if (parts.length != 7) {
65             System.out.println("Error: incorrect format");
66             return;
67         }
68
69         String name = parts[0];
70         String author = parts[1];
71         int width = Integer.parseInt(parts[2]);
```

---

```
72     int height = Integer.parseInt(parts[3]);
73     int x = Integer.parseInt(parts[4]);
74     int y = Integer.parseInt(parts[5]);
75     String pattern = parts[6];
76
77     String[] rows = pattern.split(" ");
78     int patternHeight = rows.length;
79     int patternWidth = rows[0].length();
80     char[][] cells = new char[patternHeight][patternWidth];
81     for (int i = 0; i < cells.length; i++)
82         cells[i] = rows[i].toCharArray();
83
84     boolean[][] world = new boolean[height][width];
85
86     for (int i = 0; i < patternHeight; i++)
87         for (int j = 0; j < patternWidth; j++)
88             setCell(world, x + j, y + i, cells[i][j] == '1');
89
90     play(world);
91 }
92 }
```