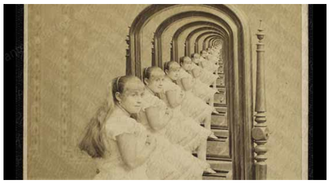
Feedback from Quiz 4

- Array of classes vs. class holding arrays
- when is the keyword new used?
- Array index vs. array value: a[i] = v;
- How do we visit or revisit each element of an array?
- Functions: arguments, return values

Review

- Recursion
- Call Stack



Coding Examples

- recursive findMax

Two-dimensional Arrays

- Visualized as a grid
- `int[][] grays = {{0, 20, 40},`
- `                  {60, 80, 100},`
- `                  {120, 140, 160},`
- `                  {180, 200, 220}};`
- `int[][] grays = new int[4][3];`

	0	1	2
0	0	20	40
1	60	80	100
2	120	140	160
3	180	200	220

Indexing 2D Arrays

- Need two indices, one for the rows and one for the columns.
- `grays[2][1] = 255;`
- `grays[2][3] = 0;`

Lengths of 2D Arrays

- `int[][] grays = new int[80][100];`
- `println(grays.length);`
- `println(grays[0].length);`

Exercise

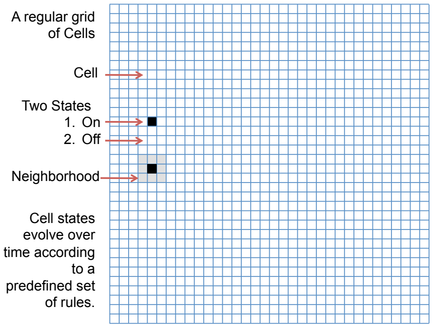
Add the necessary lines of code within `setup()` to fill the `vals` array with random numbers of your choosing. Your implementation must use `for` loops.

```
float[][] vals;
void setup() {
         vals = new float[20][300];

         // Add your code here

}
```

Examples

- graySquares

Cellular Automata



A regular grid of Cells

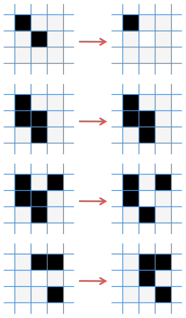
Cell

Two States
1. On
2. Off

Neighborhood

Cell states evolve over time according to a predefined set of rules.

Sample Set of Rules – Conway's Game of Life

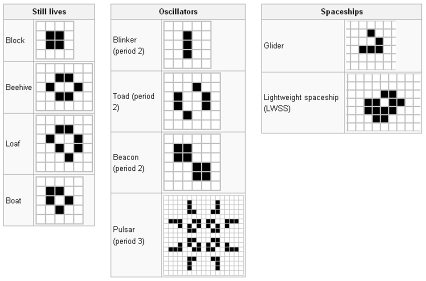


1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

An example of "Emergence"

http://en.wikipedia.org/wiki/Conway%27s_game_of_life

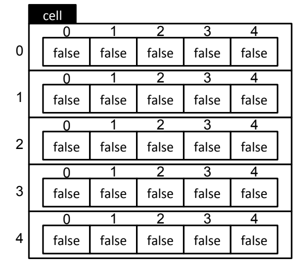
Interesting Patterns – Conway's Game of Life



http://en.wikipedia.org/wiki/Conway%27s_game_of_life

2D Array of Booleans

```
int N = 5;
boolean[][] cell = new boolean[N][N];
```

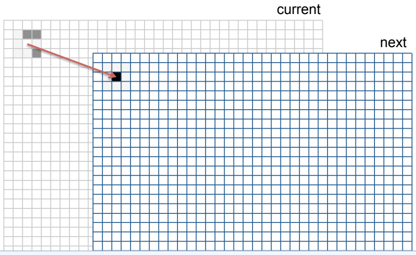


cell

	0	1	2	3	4
0	false	false	false	false	false
1	false	false	false	false	false
2	false	false	false	false	false
3	false	false	false	false	false
4	false	false	false	false	false

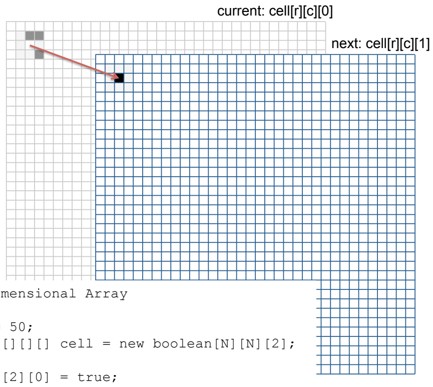
```
int N = 5;
boolean[][] cell = new boolean[N][N];

cell[1][2] = true;
```

cell	0	1	2	3	4
0	false	false	false	false	false
1	false	false	**true**	false	false
2	false	false	false	false	false
3	false	false	false	false	false
4	false	false	false	false	false

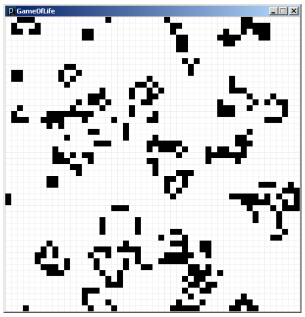


Top-level procedure

1. Draw the current grid
2. Advance game by applying rules to all cells of current and filling next
3. Swap current and next grid



```
// 3-Dimensional Array

int N = 50;
boolean[][][] cell = new boolean[N][N][2];

cell[1][2][0] = true;
```



What are we printing?

```
float[][] vals;

void setup() {
  vals = new float[20][300];

  for (int i=0; i<20; i++) {
    println(vals[i].length);
  }
}
```