Review
• Recursion
• Call Stack

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];

Processing 2D Arrays
• Need two indices, one for the rows and one for the columns.
• int[][] grays = {{0, 20, 40},
{60, 80, 100},
{120, 140, 160},
{180, 200, 220}};
• 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);

Cellular Automata
A regular grid of Cells
Two States
1. On
2. Off
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"
Interesting Patterns – Conway’s Game of Life

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

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

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

```java
int N = 5;
boolean[][][] cell = new boolean[N][N][2];
```

---

![Diagram of Conway's Game of Life patterns and top-level procedure](http://en.wikipedia.org/wiki/Conway%27s_game_of_life)
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.

```java
float[][] vals;
void setup() {
  vals = new float[20][300];
  // Add your code here
}
// Closing brace for setup()
```

Ragged Arrays

```java
int[][] numbers = {
  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
  {1, 3, 5, 7, 9},
  {0, 2, 4, 6, 8, 10},
  {2, 3, 5, 7},
  {0},
};
```

Float 2D array and draw it to the sketch as grayscale levels.

```java
float[][] grays = new float[100][100];
int cellSize = 5;
void setup() {
  size(500, 500);
  for (int i=0; i<grays.length; i++) {
    for (int j=0; j<grays[i].length; j++) {
      grays[i][j] = int(random(255));
    }
  }
  for (int i=0; i<grays.length; i++) {
    for (int j=0; j<grays[i].length; j++) {
      fill(grays[i][j]);
      pushMatrix();
      translate(j*cellSize, i*cellSize);
      rect(0, 0, cellSize, cellSize);
      popMatrix();
    }
  }
}
```

Challenge

- Modify the previous example to plot black squares whenever both the row and column of a cell are even.