Assignment 4

• Your basic shape drawing should be
  – centered on (0, 0)
  – with respect to some size field
• Constructor (and functions) should take parameters
• Use `mousePressed()` function not the global variable
  – call your constructor in there to create objects
• Comment your class fields, function parameters and variables

Review

• Recursion
• Call Stack

Coding Examples

• recursive sum
• recursive `findMax`

Creating a maze, recursively

1. Start with a rectangular region defined by its upper left and lower right corners
2. Divide the region at a random location through its more narrow dimension
3. Add an opening at a random location
4. Repeat on two rectangular subregions

Example

• `recursiveMaze` with 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];`

```
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.

```java
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.

Interesting Patterns – Conway’s Game of Life

An example of "Emergence"


2D Array of Booleans

```java
int N = 5;
boolean[][] cell = new boolean[N][N];
cell[1][2] = true;
```

```java
// 3-Dimensional Array
int N = 50;
boolean[][][] cell = new boolean[N][N][2];
cell[1][2][0] = true;
```

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
float[][] vals;
void setup() {
   vals = new float[20][300];
   for (int i=0; i<20; i++) {
      println(vals[i].length);
   }
}

What are we printing?

2D Array as an array of arrays
- Each element of a 2D array is a 1D array
- Thus each element of a 2D array has a length
- Declaration can be tiered:
  - float[][] vals;
  - float[20][] vals;
  - float[20][300] vals;
- Each element array does not have to be the same length

Ragged Arrays
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},
};

Example
- ragged

Challenge
- Recall the graySquares example
- Modify to plot black squares whenever both the row and column indices of a cell are even and white otherwise.