

Topic 4

2D Arrays

"Throughout human history, we have been dependent on machines to survive. Fate, it seems, is not without a sense of irony. "

- *Morpheus (Laurence Fishburne)*
The Matrix



2D Arrays in Java

- ▶ Arrays with multiple dimensions may be declared and used

```
int[][] mat = new int[3][4];
```

- ▶ the number of pairs of square brackets indicates the dimension of the array.
- ▶ by convention, in a 2D array the first number indicates the row and the second the column
- ▶ Java multiple dimensional arrays are handles differently than in many other programming languages.

Two Dimensional Arrays

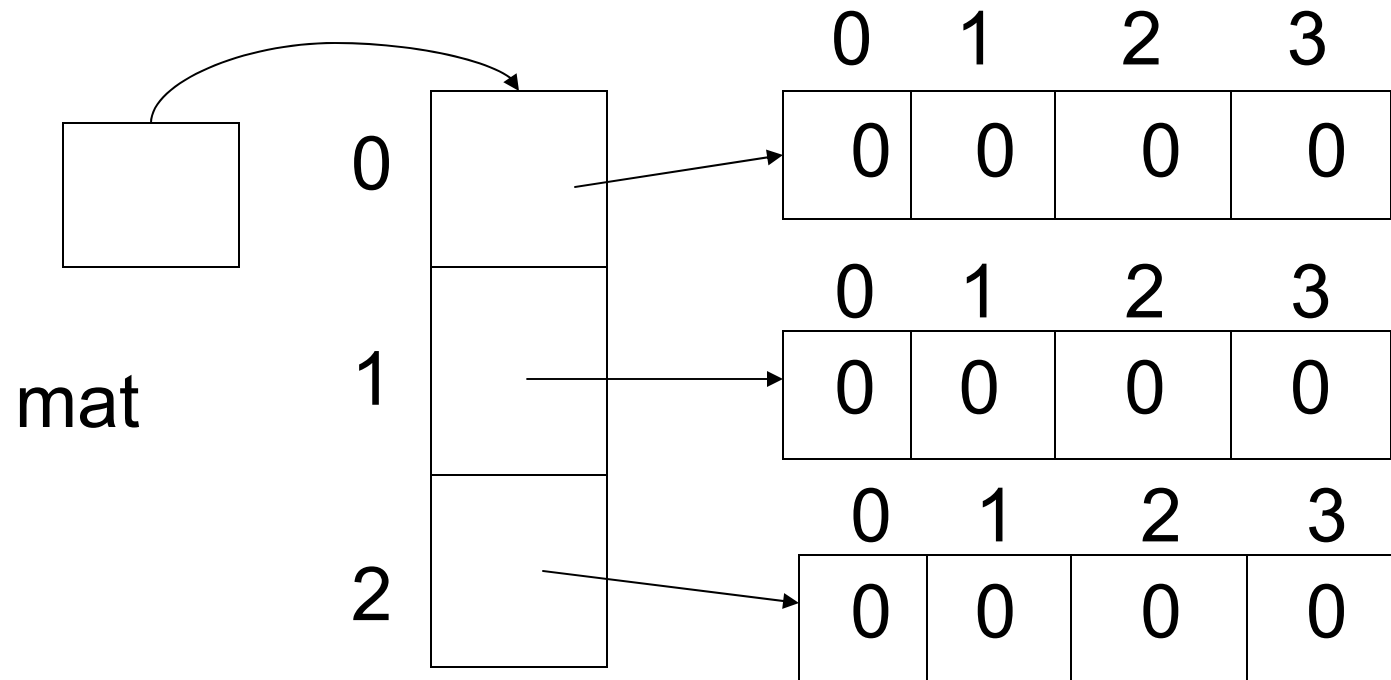
	0	1	2	3	column
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	

row

This is our abstract picture of the 2D array and treating it this way is fine.

```
mat[2][1] = 12;
```

The Real Picture



`mat` holds the memory address of an array with 3 elements. Each element holds the memory address of an array of 4 `ints`

Arrays of Multiple Dimension

- ▶ because multiple dimensional arrays are treated as arrays of arrays of arrays.....multiple dimensional arrays can be *ragged*
 - each row does not have to have the same number of columns

```
int[][] raggedMat = new int[5][];  
for(int i = 0; i < raggedMat.length; i++)  
    raggedMat[i] = new int[i + 1];
```

- each row array has its own length field

Ragged Arrays

- ▶ Ragged arrays are sometime useful, but normally we deal with *rectangular* matrices
 - each row has the same number of columns as every other row
 - use this a lot as precondition to methods that work on matrices
- ▶ working on matrices normally requires nested loops
 - why is this so hard?

Matrix Problems

- ▶ Write a method that filters an image (colors could simply be ints)
 - smooth / blur
 - sharpen
- ▶ Conway's Game of Life,
www.math.com/students/wonders/life/life.html
 - cells occupied or empty
 - next generation depends on the current and your 8 neighboring cells
 - occupied cell
 - 0 - 1 neighbors, die
 - 2 - 3 neighbors live
 - ≥ 4 neighbors die
 - unoccupied cell
 - 3 neighbors birth

The Filter Problem

- ▶ Given a 2D array of Color Objects
- ▶ The Color class: `java.awt.Color`
- ▶ Important methods:

```
Color(int r, int g, int b)
```

Creates an opaque sRGB color with the specified red, green, and blue values in the range (0 - 255).

```
int getBlue()
```

Returns the blue component in the range 0-255 in the default sRGB space.

```
int getRed()
```

```
int getGreen()
```

Complete the Following Method

```
public Color[][] smooth(Color[][] image)
/*
 *pre: image != null, image.length >= 1,
      image[0].length > 1, for all N such that 0
      <= N < image.length,
      image[N].length = image[0].length
 post: returns a smoothed image. Each cell
       in the returned matrix is the average of
       its neighbors
```