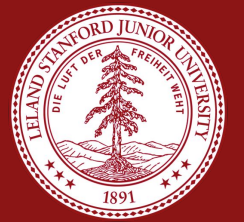
2D Arrays

Lecture 17

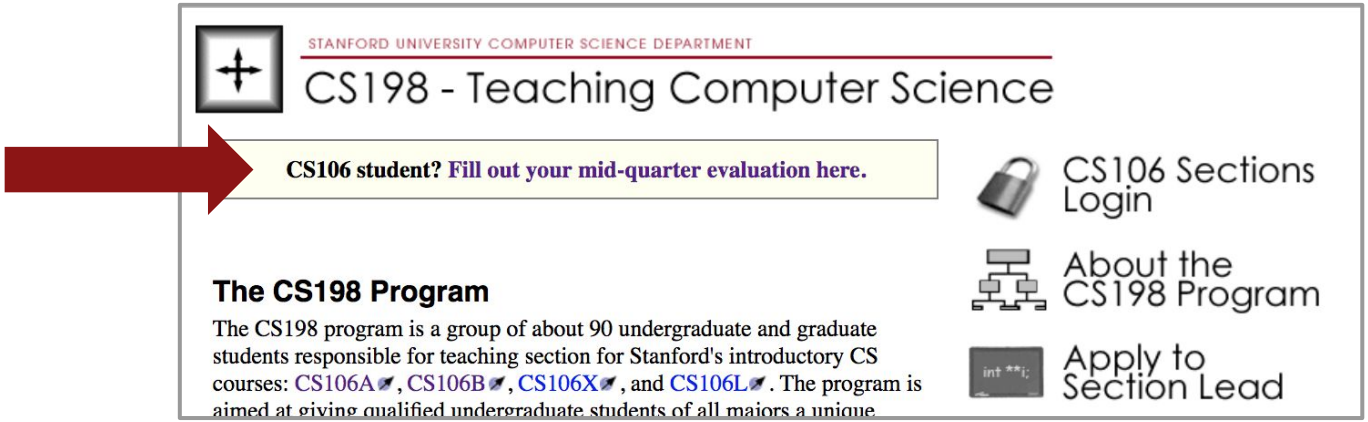
CS106A, Summer 2019
Sarai Gould && Laura Cruz-Albrecht

With inspiration from slides created by Keith Schwarz, Mehran Sahami, Eric Roberts, Stuart Reges, Chris Piech and others.



Announcements

- We're at the halfway point! Please fill out the Mid-Quarter Evaluation at cs198.stanford.edu.
  - Completely anonymous, you will receive a small amount of assignment extra credit if you complete it
  - Note: You have 30 minutes from opening the survey to complete it



Plan for Today

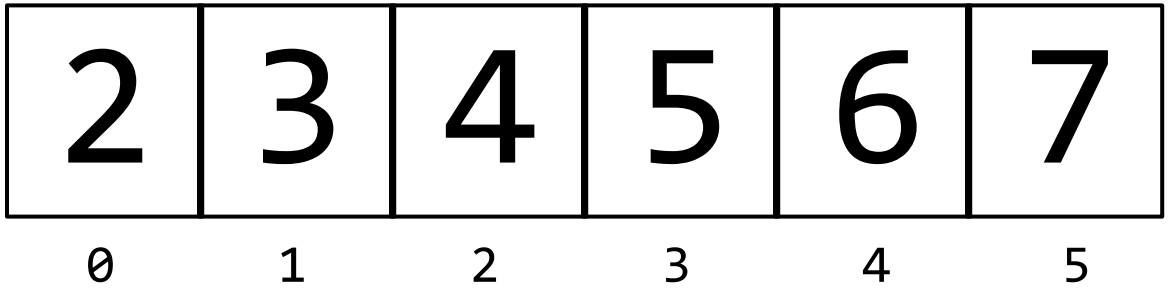
- Review: Arrays
- 2D Arrays
- Images as 2D Arrays
- Manipulating Images
- Practice: Grayscale

Plan for Today

- Review: Arrays
- 2D Arrays
- Images as 2D Arrays
- Manipulating Images
- Practice: Grayscale

Review: Arrays

- Each location is assigned an index, going from 0 to length-1.
- The type of data at each index depends on the type of array!




```
int arrayLen = myArray.length;       // 6
int last = myArray[arrayLen - 1];    // 7
```

Review: Arrays

```
int[] myArray = new int[5];
// OR
int[] myArray = {2, 3, 4, 5, 6, 7};

int arrayLen = myArray.length;

// Access elements with bracket notation
int first = myArray[0];
int last = myArray[arrayLen - 1];

// In arrays, we can change elements!
myArray[0] = 22;
```

Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```

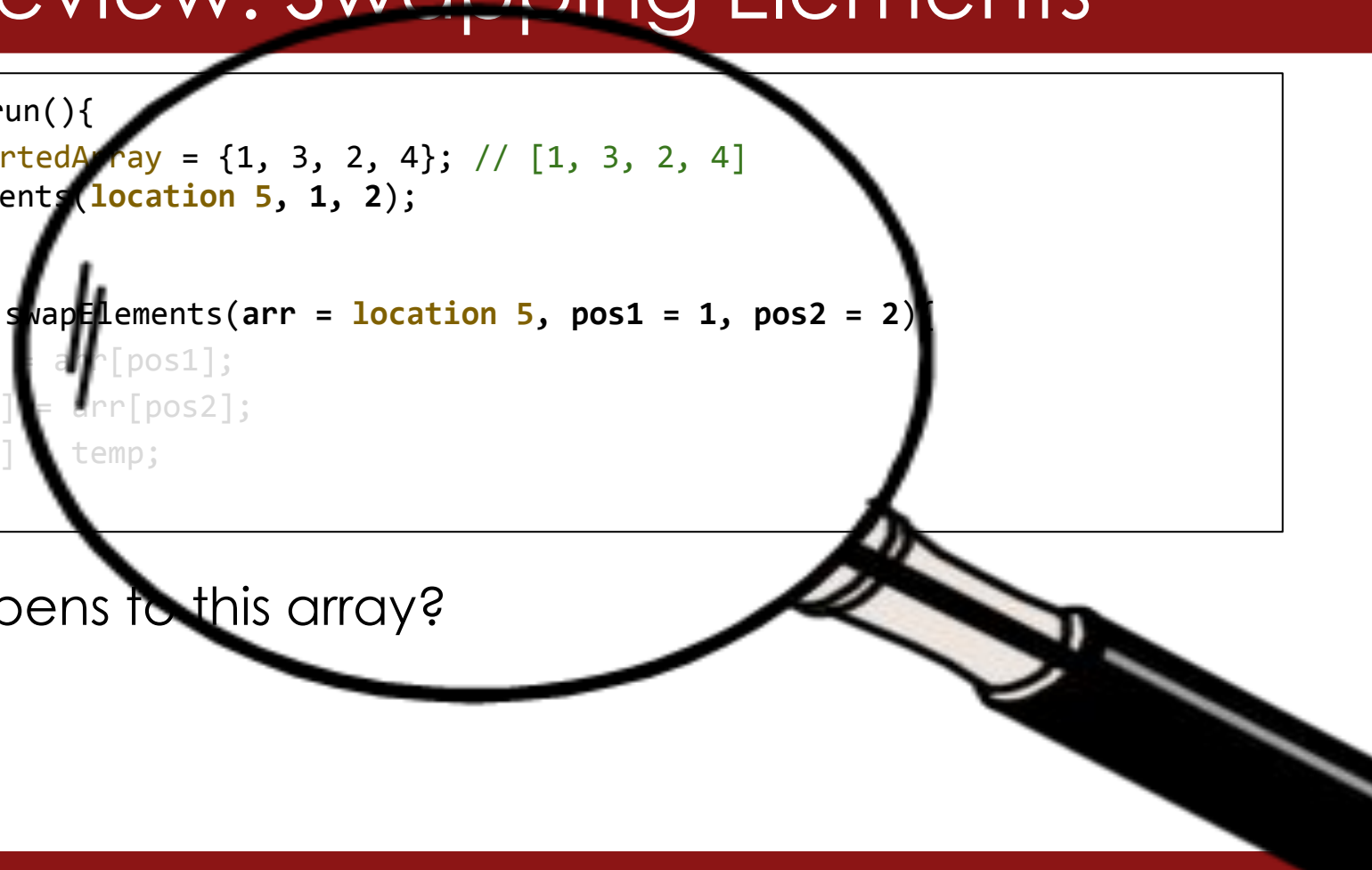
These are *indices*

What happens to this array?

Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(location 5, 1, 2);
}

private void swapElements(arr = location 5, pos1 = 1, pos2 = 2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```

What happens to this array?

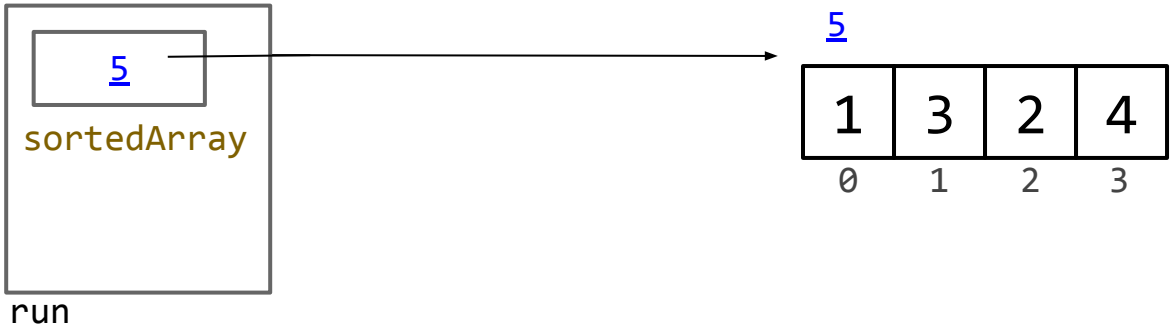


Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```

Review: Swapping Elements

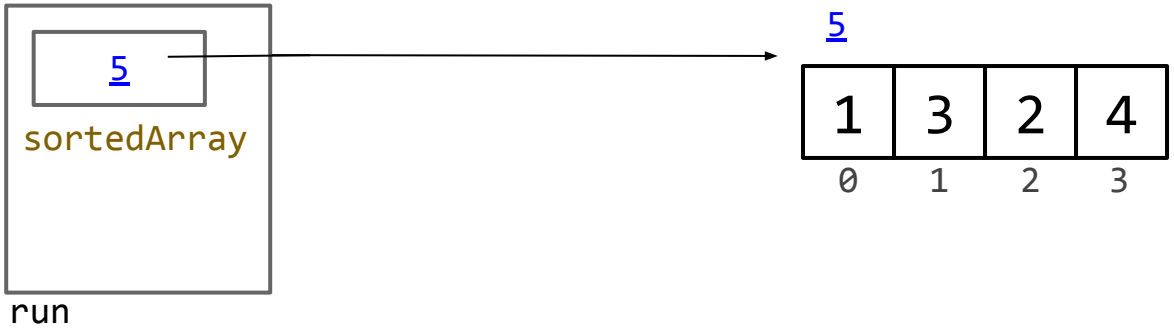
```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```

These are *indices*

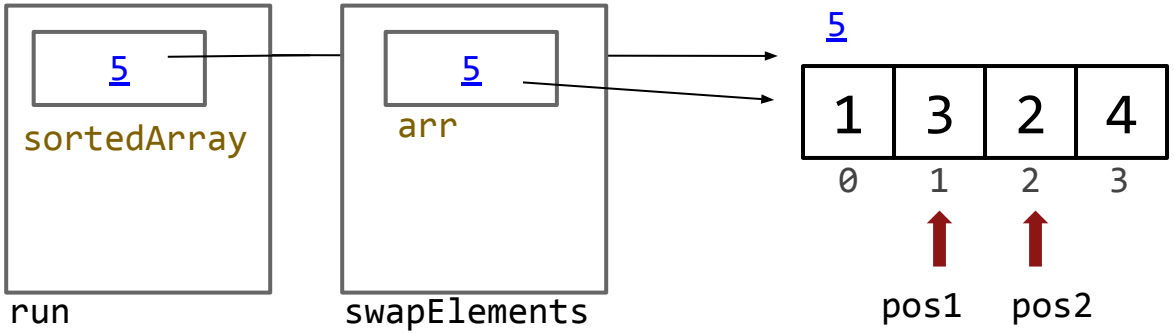


Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}
```

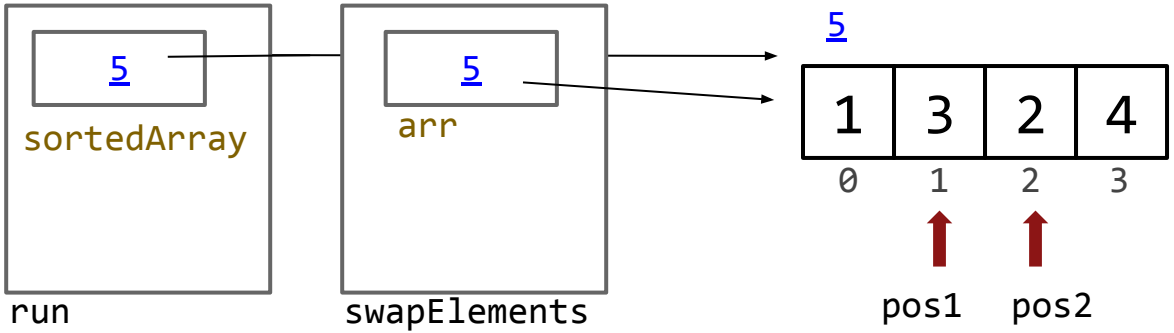
These are *indices*

```
private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



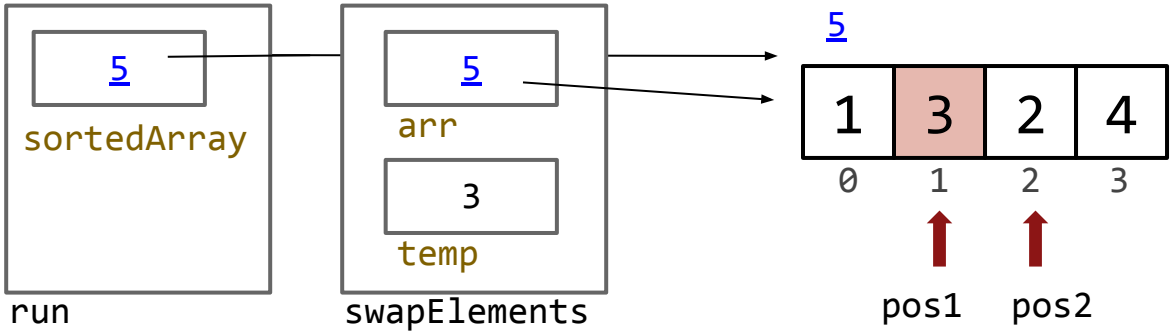
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



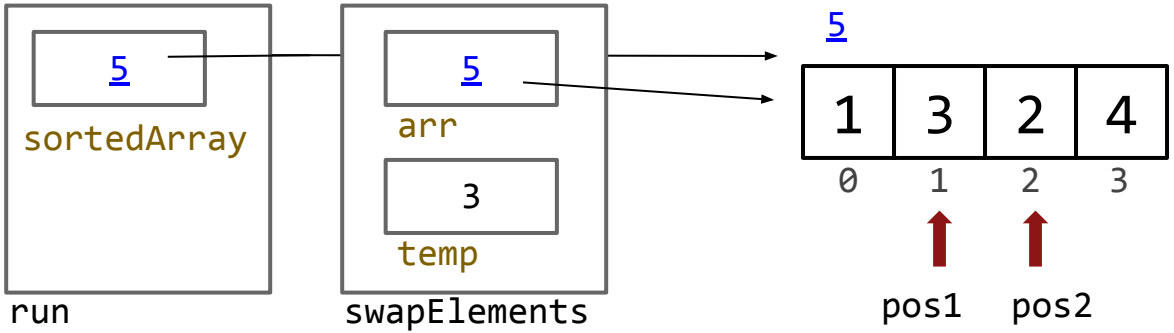
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



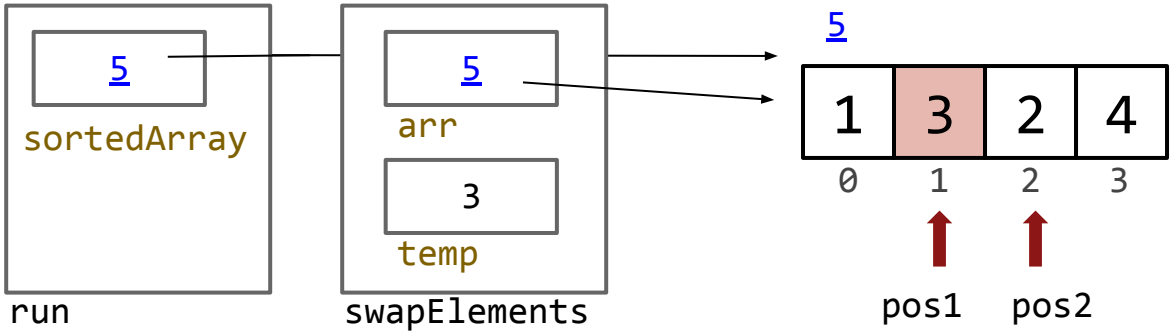
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



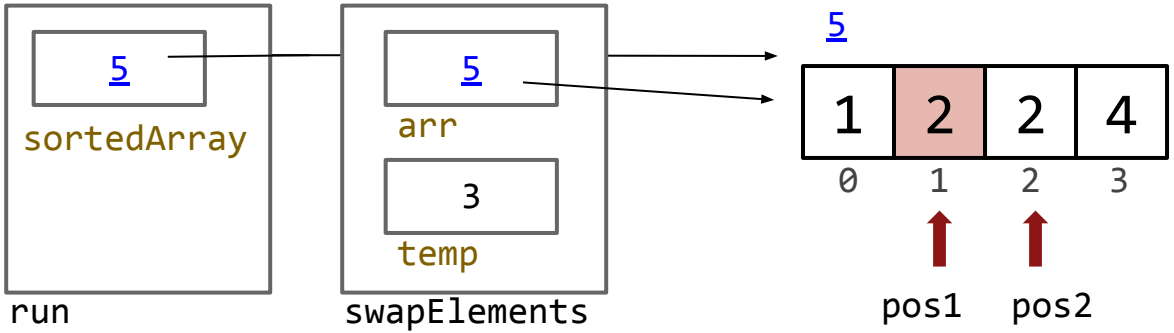
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



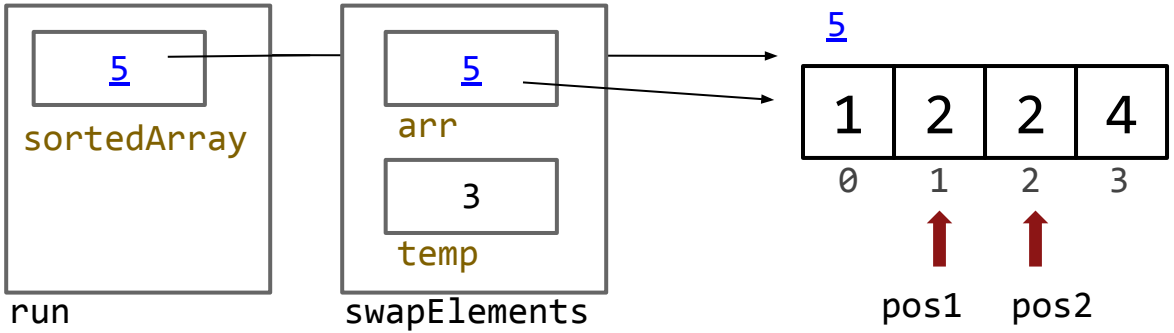
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



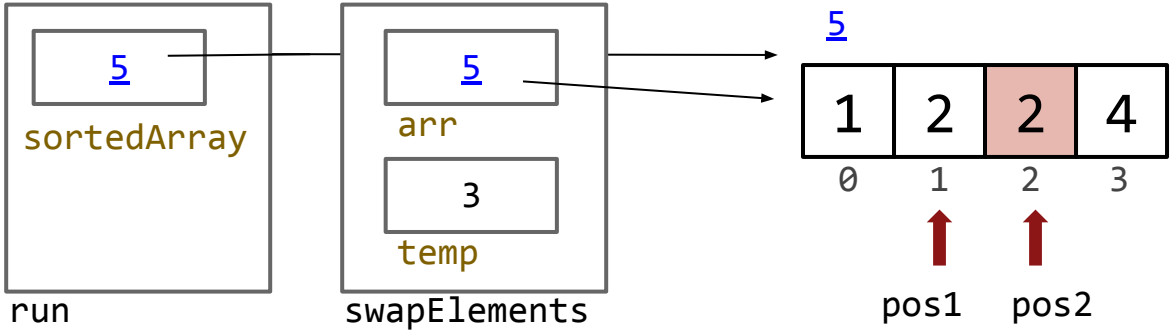
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



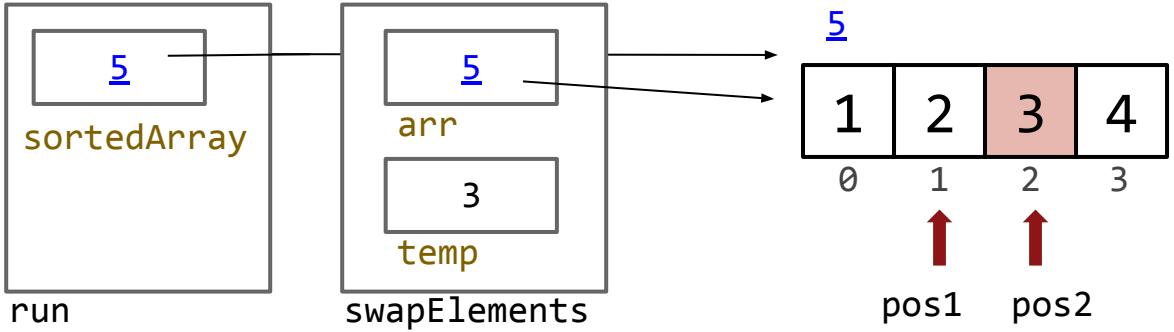
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



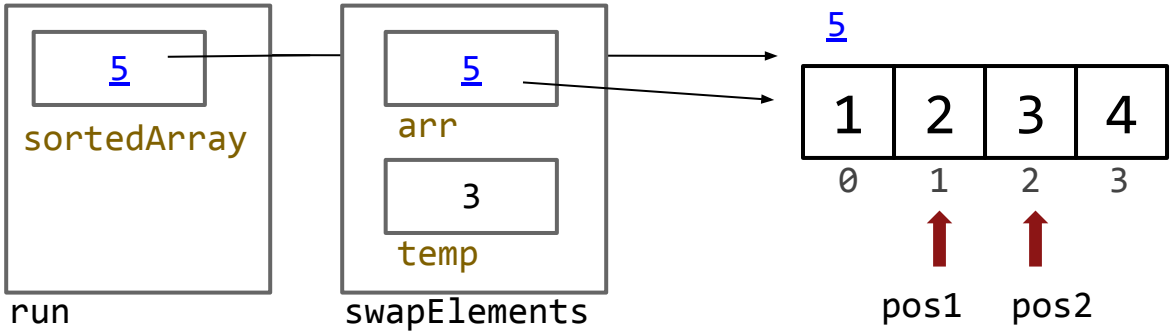
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



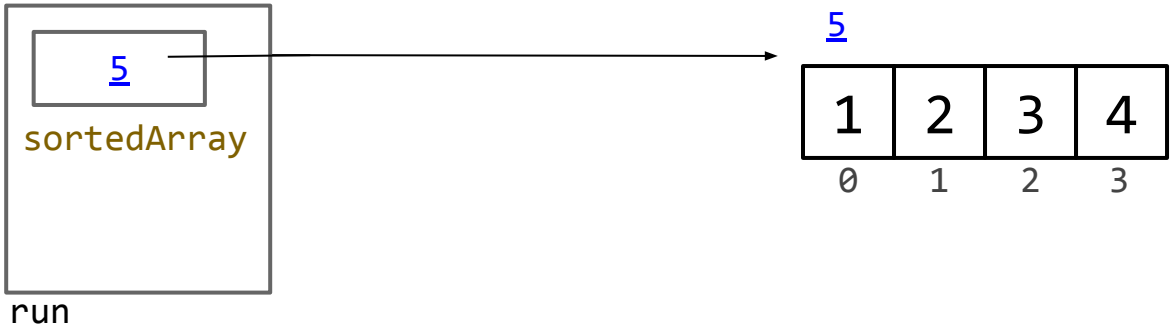
Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



Review: Swapping Elements

```
public void run(){
     int[] sortedArray = {1, 3, 2, 4}; // [1, 3, 2, 4]
     swapElements(sortedArray, 1, 2);
}

private void swapElements(int[] arr, int pos1, int pos2){
     int temp = arr[pos1];
     arr[pos1] = arr[pos2];
     arr[pos2] = temp;
}
```



Plan for Today

- Review: Arrays
- 2D Arrays
- Images as 2D Arrays
- Manipulating Images
- Practice: Grayscale

An Array

```
boolean[] b = new boolean[3];
```

An Array

```
boolean[] b = new boolean[3];
```

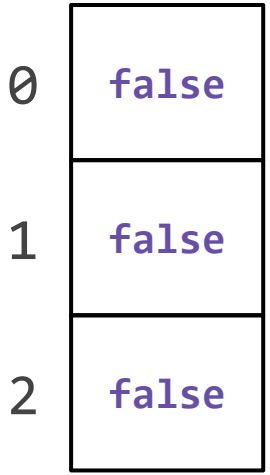
I want an array

An Array

```
boolean[] b = new boolean[3];
```

An Array

```
boolean[] b = new boolean[3];
```



An Array

```
int[] a = new int[3];
```

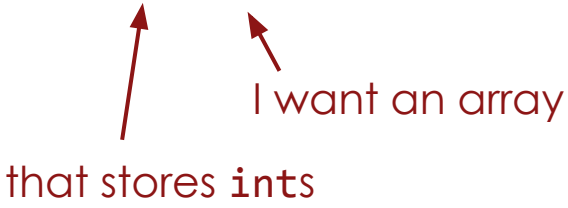
An Array

```
int[] a = new int[3];
```

I want an array

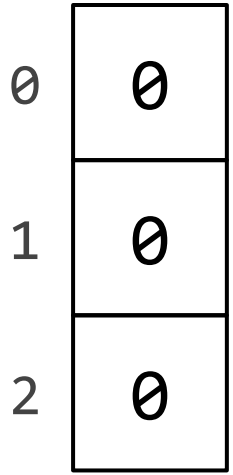
An Array

```
int[] a = new int[3];
```

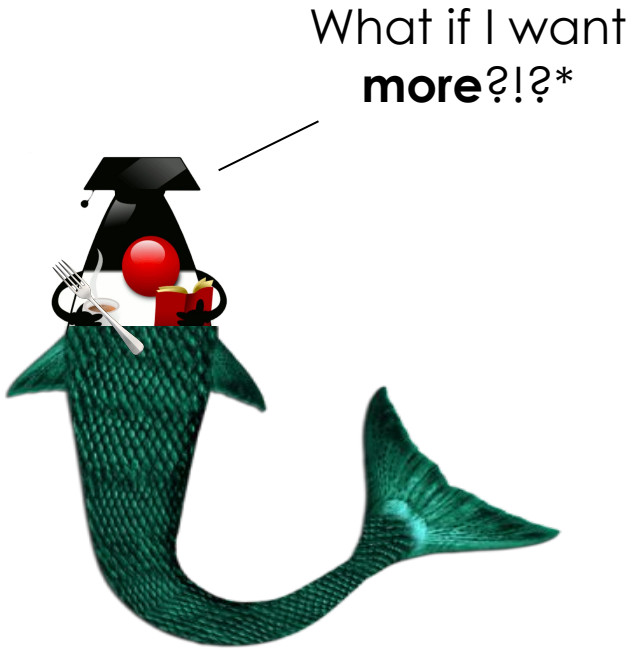


An Array

```
int[] a = new int[3];
```



A Thought



* possibly a little mermaid reference 🐟

A Thought

* possibly a little mermaid reference 🐟

An Array… of Arrays

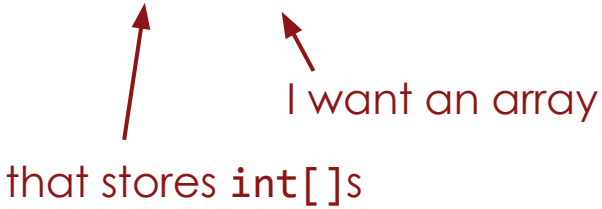
```
int[][] matrix = new int[3][4];
```

An Array… of Arrays

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

An Array… of Arrays

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



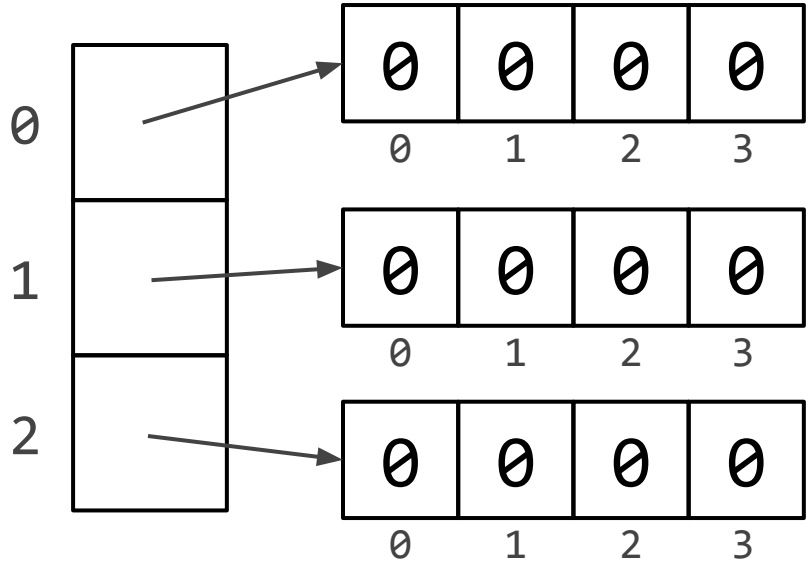
An Array… of Arrays

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

that stores `int[]`s
aka, that stores arrays!

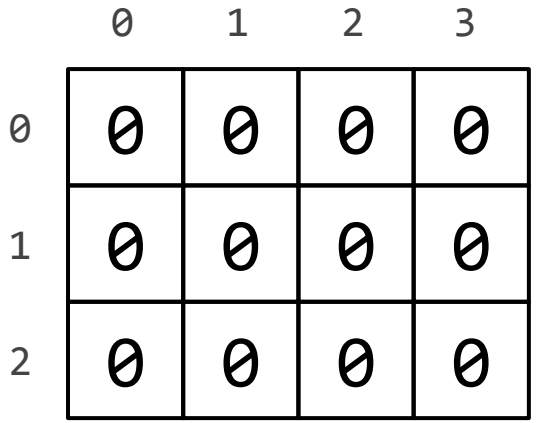
An Array... of Arrays

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



A 2D Array

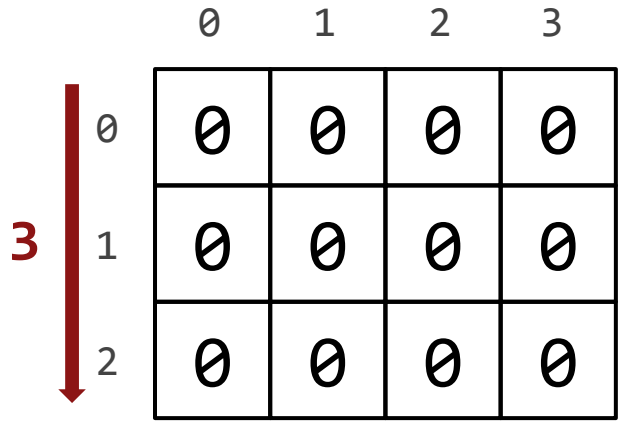
```
int[][] matrix = new int[3][4];
```



A 2D Array

Number of rows

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

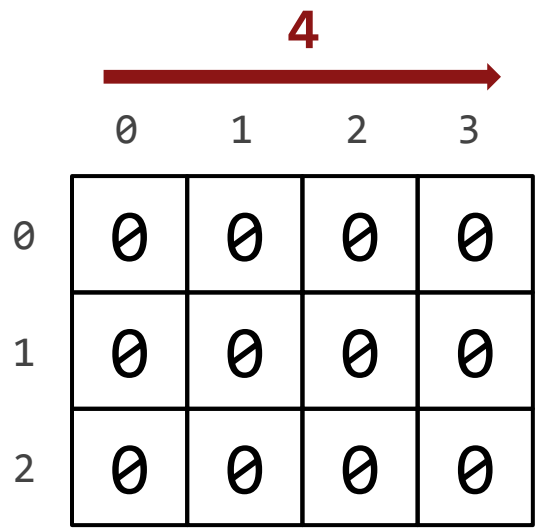


A 2D Array

Number of rows

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

Number of columns



Reasoning about 2D Arrays

There are two main ways to reason about a 2D array:

1. **As an array of arrays**
   - A 2D array is an array whose elements are themselves arrays

2. **As a unit**
   - A 2D array represents a grid / matrix

Manipulating 2D Arrays

```
type[][] name = new type[numRows][numCols];

name[row][col];              // get element at row, col

name[row][col] = value;      // set element at row, col
```

Row, then Col

```
matrix[    row    ][    col    ]
```

Row, then Col

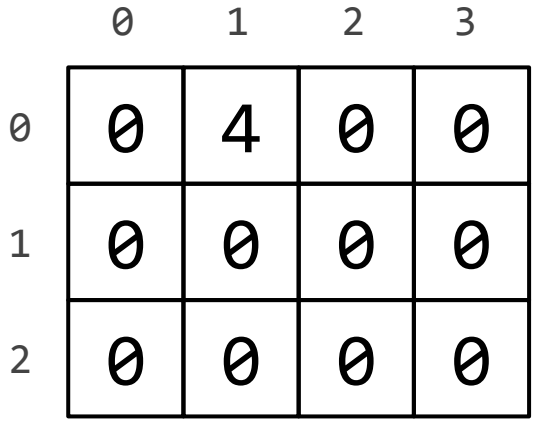
R is for StanfoRd

C is for Cal

```
matrix[ ][ ]
```

Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
...
```



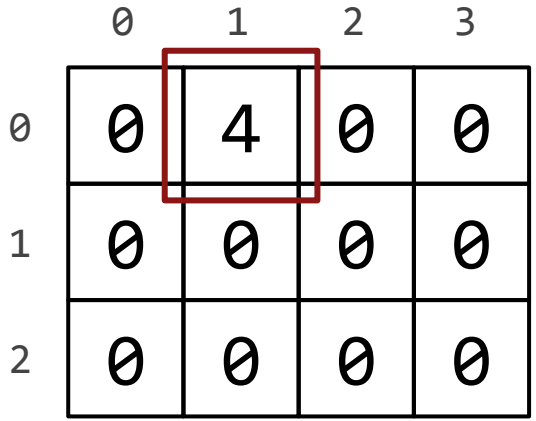
Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
…

int val = matrix[0][1];
```

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

Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
...

int val = matrix[0][1];    // 4
```



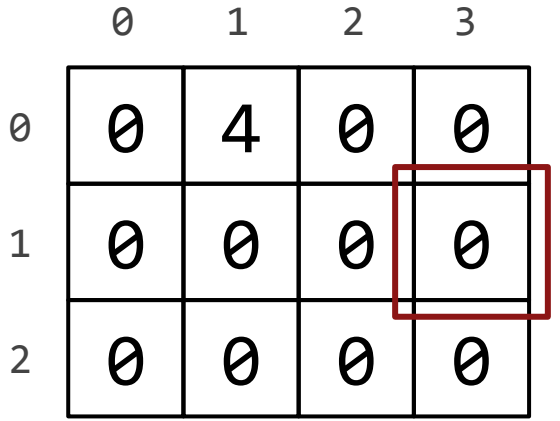
Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
…

int val = matrix[0][1];    // 4

matrix[1][3] = 8;
```

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

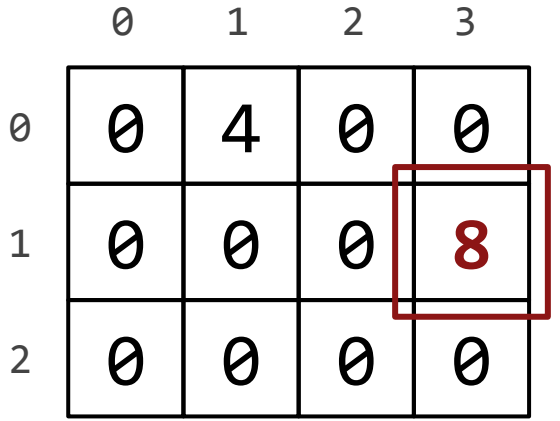
Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
…

int val = matrix[0][1];    // 4

matrix[1][3] = 8;
```



Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
…

int val = matrix[0][1];    // 4

matrix[1][3] = 8;
```



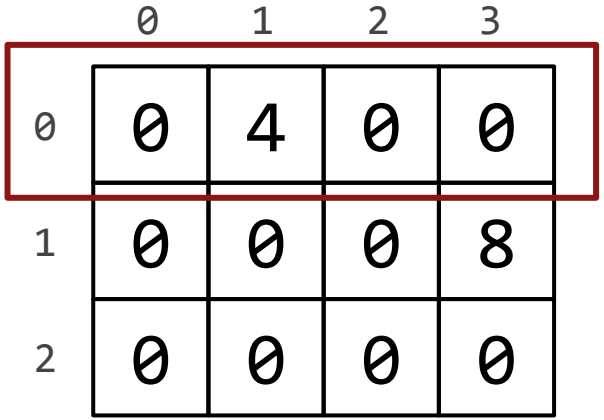
Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
…

int val = matrix[0][1];    // 4

matrix[1][3] = 8;

int[] firstRow = matrix[0];
```

	0	1	2	3
0	0	4	0	0
1	0	0	0	8
2	0	0	0	0

Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
…

int val = matrix[0][1];    // 4

matrix[1][3] = 8;

int[] firstRow = matrix[0];
```



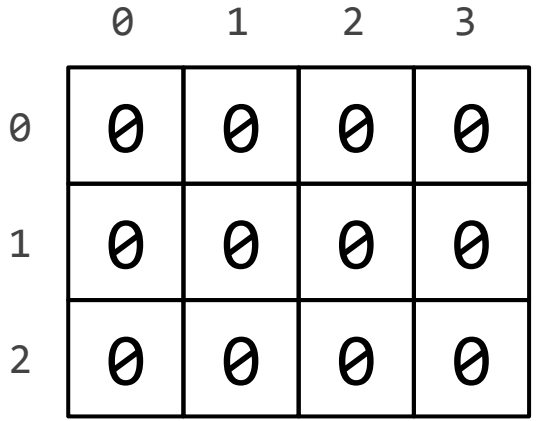
Manipulating 2D Arrays

```
int[][] matrix = new int[3][4];
…

int val = matrix[0][1];   // 4

matrix[1][3] = 8;

int[] firstRow = matrix[0];

// note: no way to get single col
```

	0	1	2	3
0	0	4	0	0
1	0	0	0	8
2	0	0	0	0

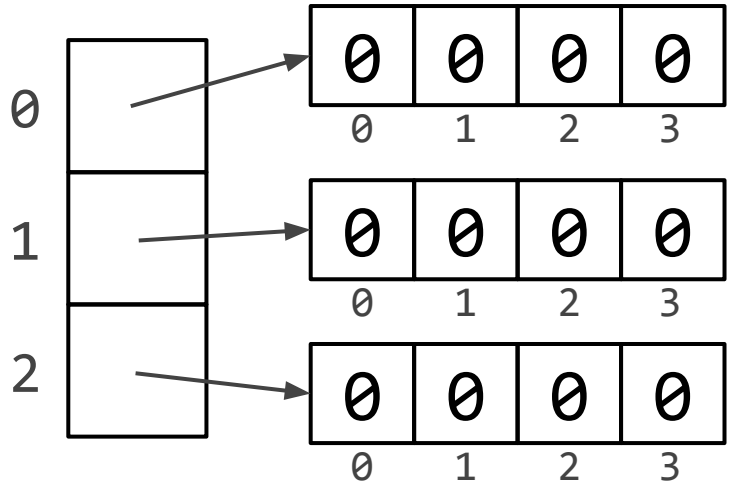
2D Array Dimensions

- How can we get the number of rows using `.length`? How about the number of columns?



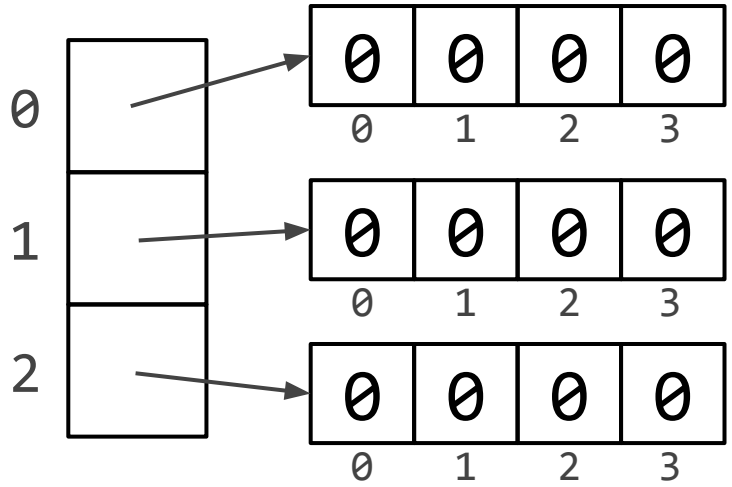
2D Array Dimensions

- How can we get the number of rows using `.length`? How about the number of columns?



2D Array Dimensions

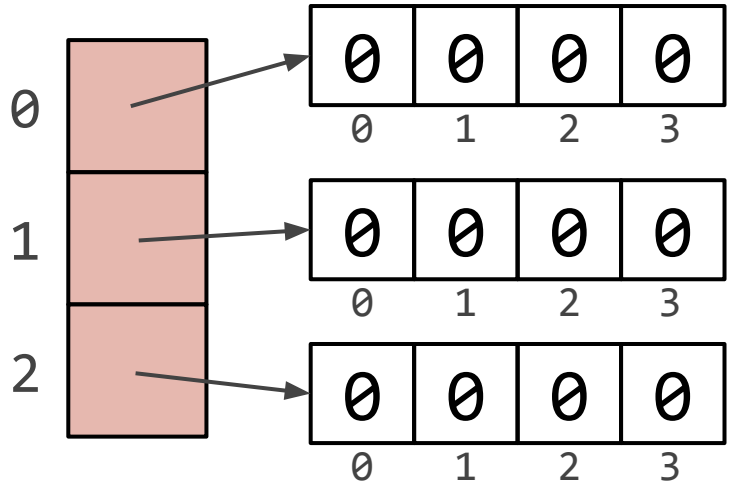
- How can we get the number of rows using `.length`? How about the number of columns?




```
int numRows = ??
```

2D Array Dimensions

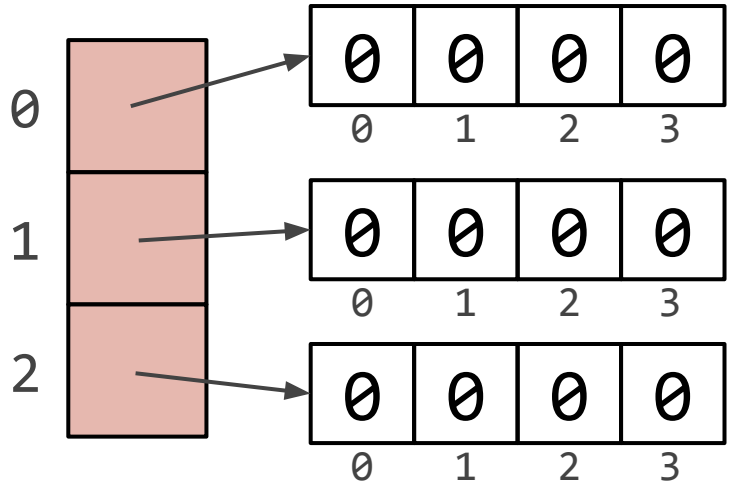
- How can we get the number of rows using `.length`? How about the number of columns?




```
int numRows = ??
```

2D Array Dimensions

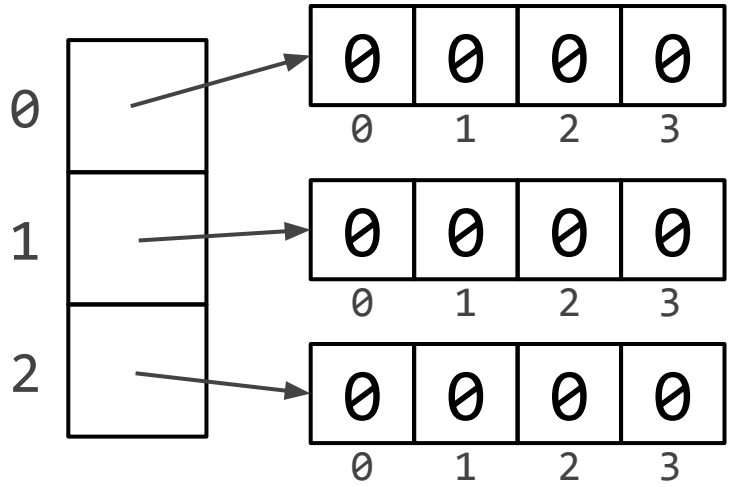
- How can we get the number of rows using `.length`? How about the number of columns?




```
int numRows = arr.length;
```

2D Array Dimensions

- How can we get the number of rows using `.length`? How about the number of columns?

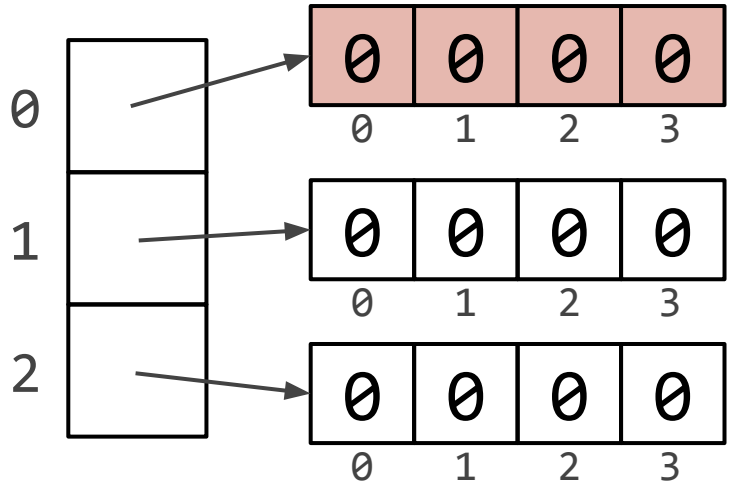



```
int numRows = arr.length;
```

```
int numCols = ??
```

2D Array Dimensions

- How can we get the number of rows using `.length`? How about the number of columns?

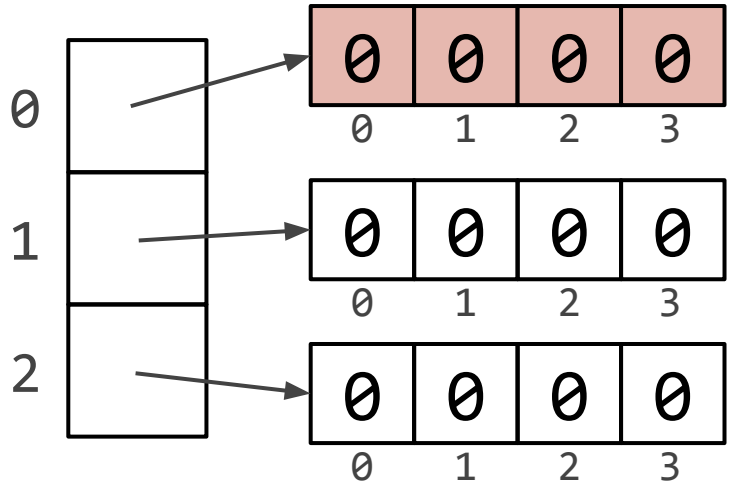



```
int numRows = arr.length;
```

```
int numCols = ??
```

2D Array Dimensions

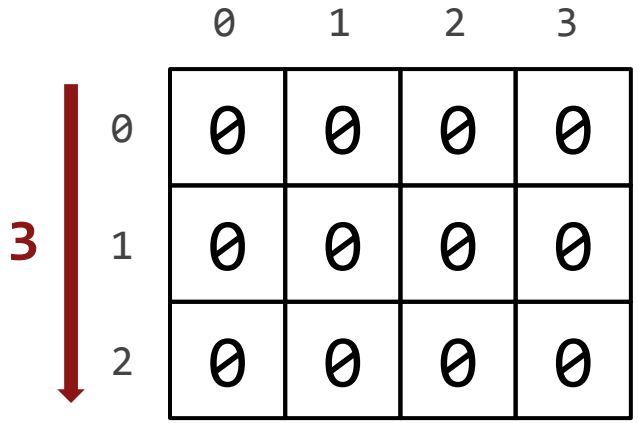
- How can we get the number of rows using `.length`? How about the number of columns?




```
int numRows = arr.length;

// # entries in row 0 = # cols
int numCols = arr[0].length;
```

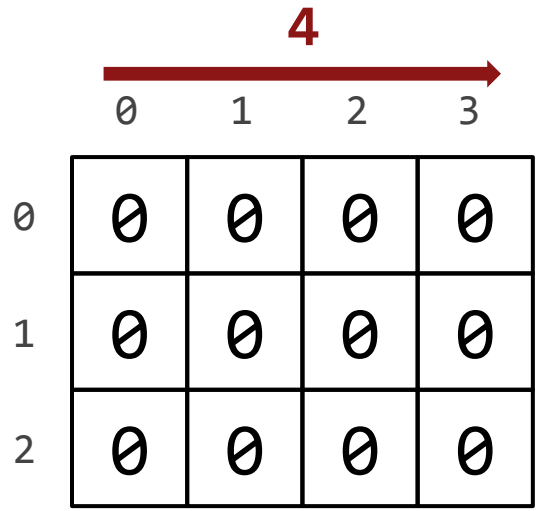
Get Number of Rows

```
private int numRows(int[][] matrix) {
    return matrix.length;
}
```



Get Number of Columns

```
private int numCols(int[][] matrix) {
    return matrix[0].length;
}
```



2D Arrays ❤️ For Loops

- The canonical way to loop over a 2D array is with a double `for` loop

```
type[][] arr = ...
for (int row = 0; row < arr.length; row++) {
    for (int col = 0; col < arr[0].length; col++) {
        // do something with arr[row][col] ...
    }
}
```

2D Arrays ❤️ For Loops

- The canonical way to loop over a 2D array is with a double `for` loop

```
type[][] arr = ...
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        // do something with arr[row][col] ...
    }
}
```

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

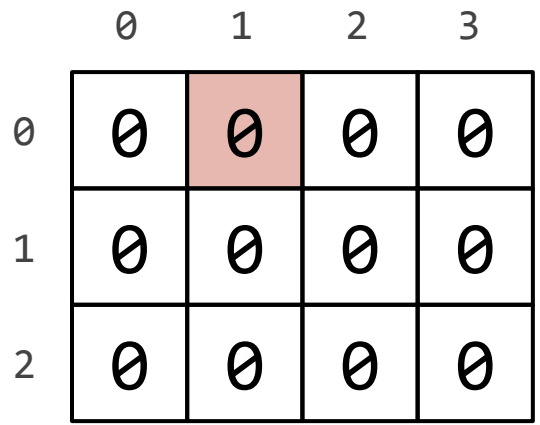
2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```



2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

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

2D Arrays ❤️ For Loops

```
int[][] arr = new int[3][4];
for (int row = 0; row < numRows(arr); row++) {
    for (int col = 0; col < numCols(arr); col++) {
        arr[row][col] = row + col;
    }
}
```

	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5

2D Arrays on 1 Slide

1. **Make a 2D array**

```
double[][] grid = new double[nRows][nCols];
```

2D Arrays on 1 Slide

1. **Make a 2D array**

```
double[][] grid = new double[nRows][nCols];
```

2. **Set and get values from a 2D array using bracket notation**

```
grid[2][1] = 2.2;
println("Upper left val is: " + grid[0][0]);
```

2D Arrays on 1 Slide

1. **Make a 2D array**

```
double[][] grid = new double[nRows][nCols];
```

2. **Set and get values from a 2D array using bracket notation**

```
grid[2][1] = 2.2;
println("Upper left val is: " + grid[0][0]);
```

3. **Get the number of rows and columns of a 2D array (tip: define method)**

```
int numRows = grid.length;
int numCols = grid[0].length;
```

2D Arrays on 1 Slide

1. **Make a 2D array**

```
double[][] grid = new double[nRows][nCols];
```

2. **Set and get values from a 2D array using bracket notation**

```
grid[2][1] = 2.2;
println("Upper left val is: " + grid[0][0]);
```

3. **Get the number of rows and columns of a 2D array (tip: define method)**

```
int numRows = grid.length;
int numCols = grid[0].length;
```

4. **Use a double for loop to iterate over the entire 2D array**

```
for (int r = 0; r < grid.length; r++) {
     for (int c = 0; c < grid[0].length; c++) {
          // something with grid[r][c]
     }
}
```

Plan for Today

- Review: Arrays
- 2D Arrays
- Images as 2D Arrays
- Manipulating Images
- Practice: Grayscale


Images are
2D arrays!

Images

- Images are 2D arrays of **pixels**
- Pixels are simply integers with red, green, and blue components (each between 0 and 255)

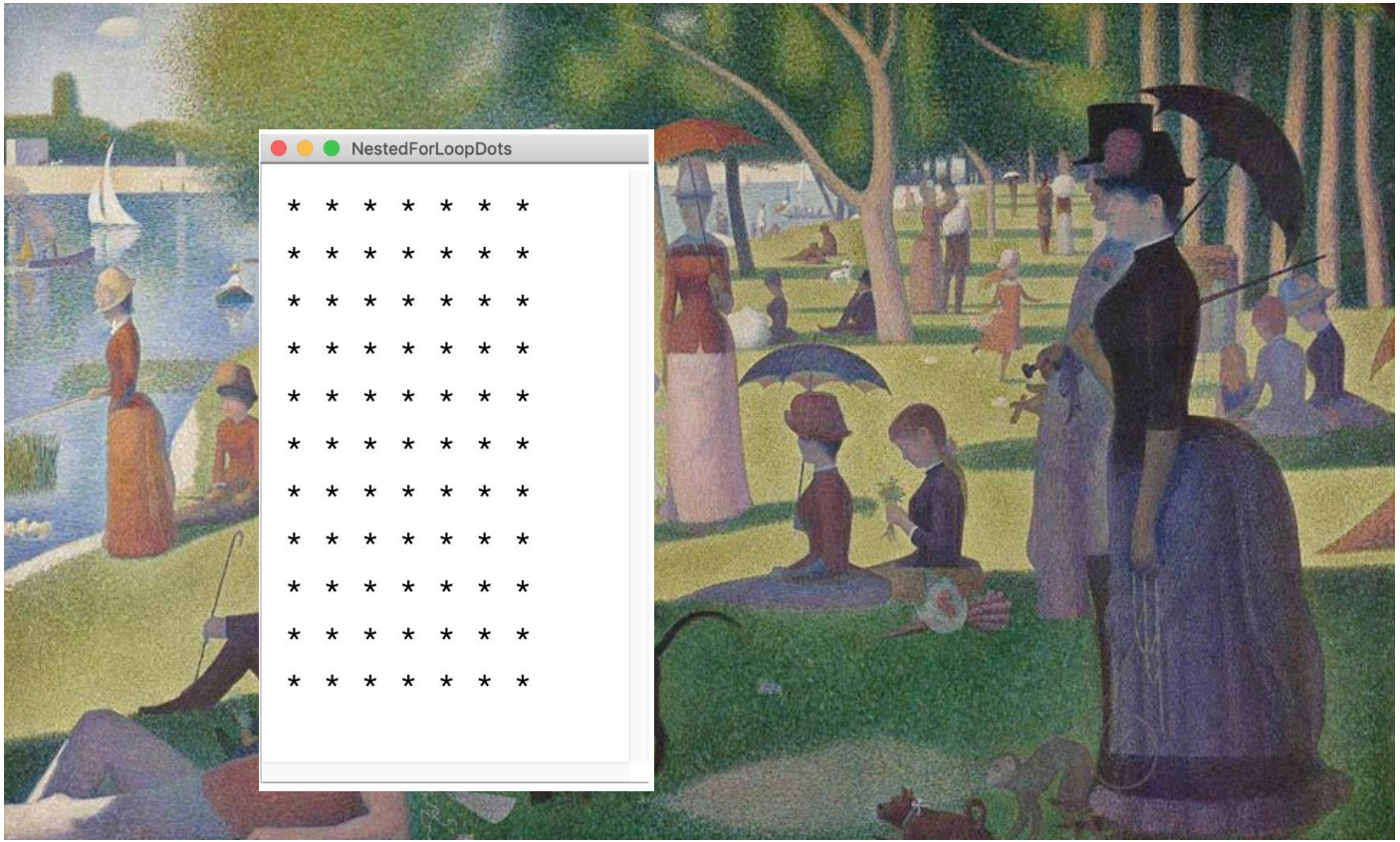
Images as 2D Arrays

We can get a GImage as a 2D array of pixels.

```
GImage img = new GImage("res/passionflower.jpg");
int[][] pixels = img.getPixelArray();
int topLeftPixel = pixels[0][0];
```

Time for Art

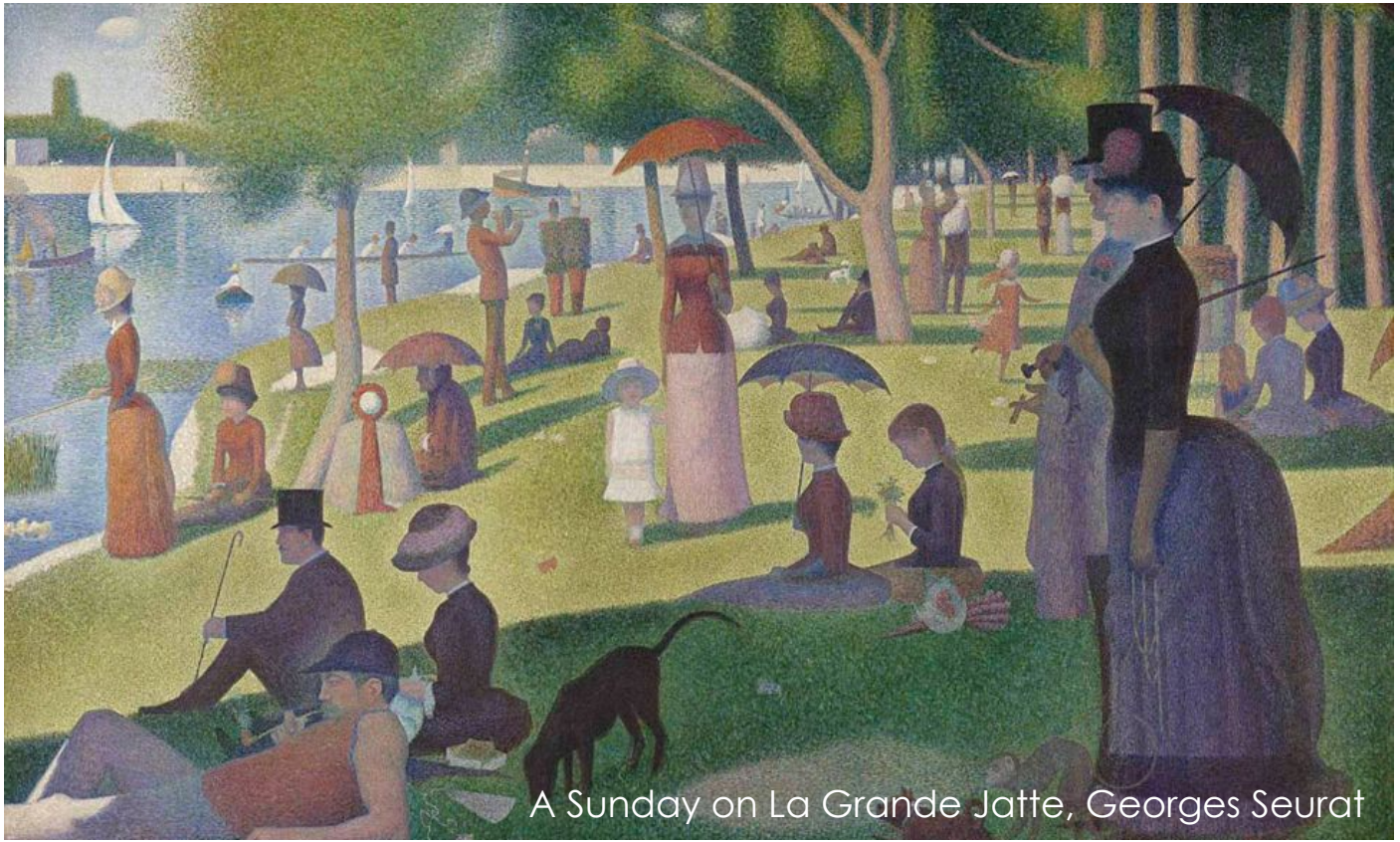
Isn't This Art???



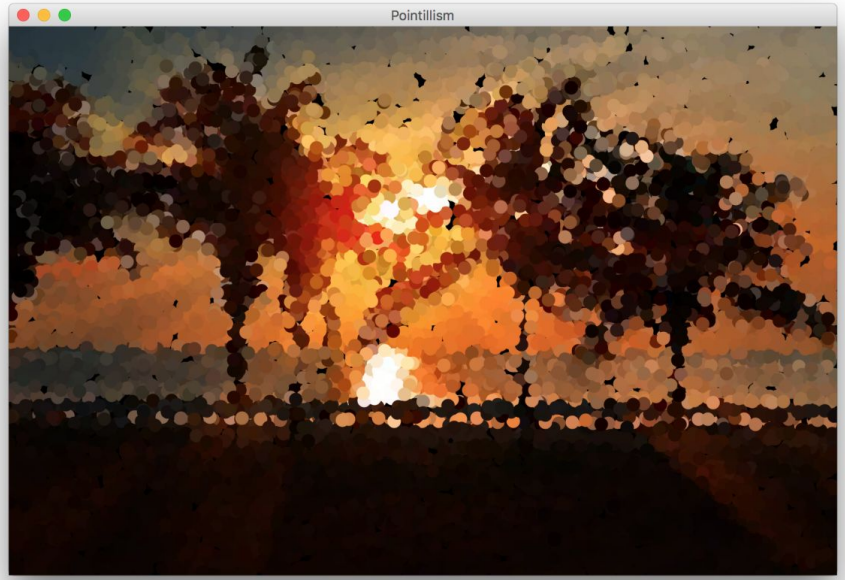
Not really the same, but okay...

Let's do Pointillism!

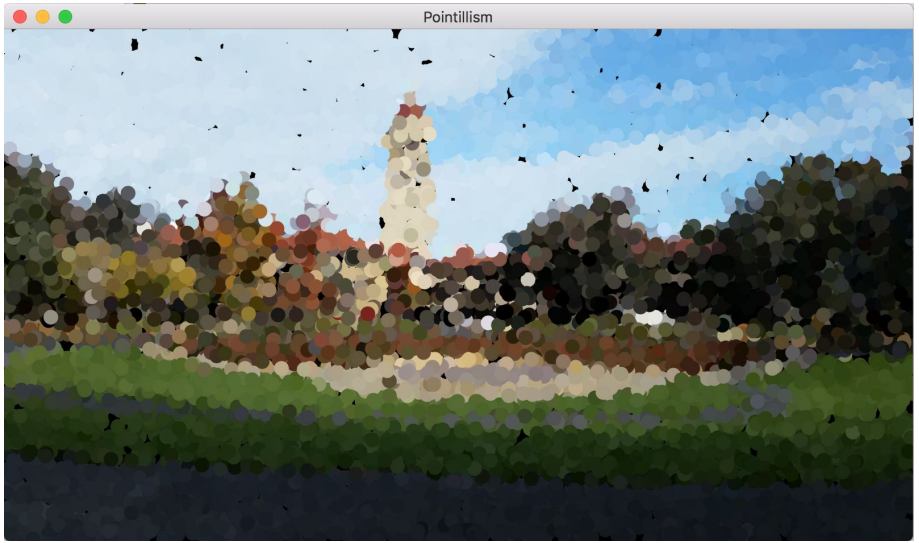
Let's make *this*



Example: Pointillism



Example: Pointillism



Example: Pointillism

Repeat many times:

1. Pick a random pixel from an image.
2. Find the pixel's color
3. "Paint" a rather large brush stroke at a corresponding location, with the color

Example: Pointillism

Example: Pointillism

c = 220

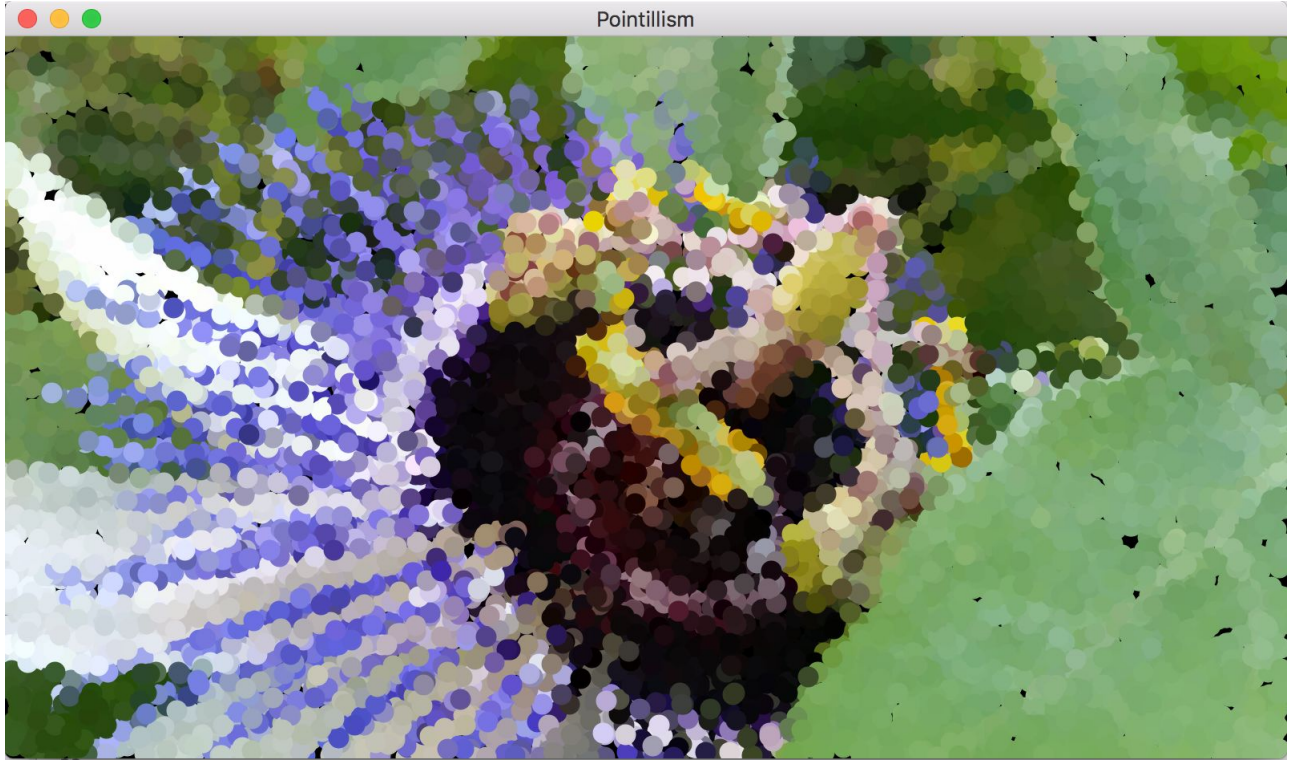
r = 340

Example: Pointillism

c = 220

r = 340

Example: Pointillism



Let’s Code It!

Plan for Today

- Review: Arrays
- 2D Arrays
- Images as 2D Arrays
- Manipulating Images
- Practice: Grayscale

Working with Image Arrays

```
GImage img = new GImage("res/passionflower.jpg");
```

Working with Image Arrays

```
GImage img = new GImage("res/passionflower.jpg");
```

1. Get a GImage as a 2D array of pixels

```
int[][] pixels = img.getPixelArray();
```

Working with Image Arrays

```
GImage img = new GImage("res/passionflower.jpg");
```

1. Get a GImage as a 2D array of pixels

```
int[][] pixels = img.getPixelArray();
```

2. Modify the pixel array or create a new pixel array

```
// access individual pixels with: pixels[r][c]
```

Working with Image Arrays

```
GImage img = new GImage("res/passionflower.jpg");
```

1. Get a GImage as a 2D array of pixels

   ```
   int[][] pixels = img.getPixelArray();
   ```

2. Modify the pixel array or create a new pixel array

   ```
   // access individual pixels with: pixels[r][c]
   ```

3. Update or create a new GImage with the modified pixels

   ```
   img.setPixelArray(pixels);          // update image
   ```

Working with Image Arrays

```
GImage img = new GImage("res/passionflower.jpg");
```

1. Get a GImage as a 2D array of pixels

```
int[][] pixels = img.getPixelArray();
```

2. Modify the pixel array or create a new pixel array

```
// access individual pixels with: pixels[r][c]
```

3. Update or create a new GImage with the modified pixels

```
img.setPixelArray(pixels);              // update image
// OR
GImage newImg = new GImage(pixels);  // make new GImage
```

Working with Image Arrays

```
GImage img = new GImage("res/passionflower.jpg");
```

1. Get a GImage as a 2D array of pixels

```
int[][] pixels = img.getPixelArray();
```

2. Modify the pixel array or create a new pixel array

```
// access individual pixels with: pixels[r][c]
```

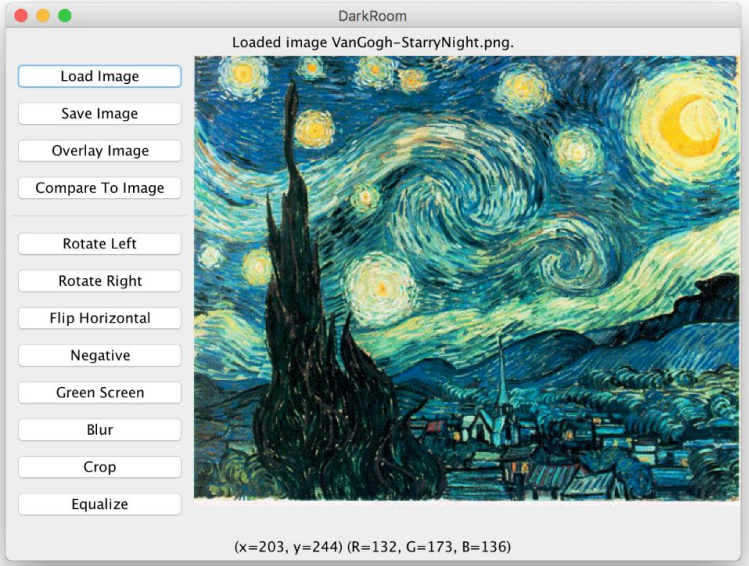
3. Update or create a new GImage with the modified pixels

```
img.setPixelArray(pixels);              // update image
// OR
GImage newImg = new GImage(pixels);  // make new GImage
```

Modifying Image Pixels

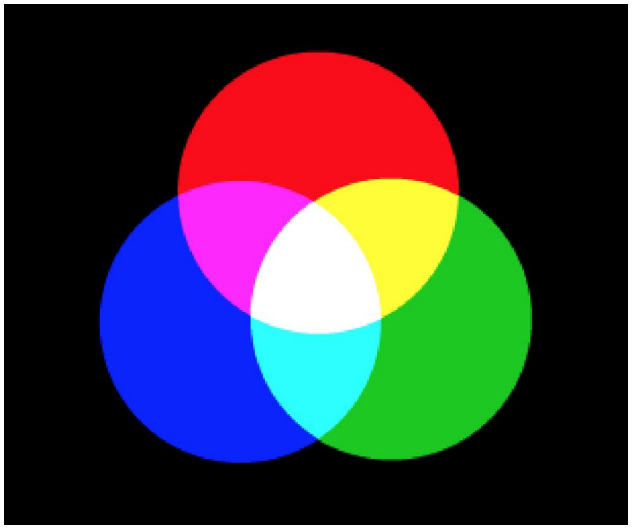
There are many neat image algorithms based on modifying individual pixels in an image:

- grayscale
- brighten
- normalize
- remove red-eye
- …



Pixels and RGB

- Pixels encode the R, G, and B values (0-255) of a pixel into a single integer.
- You can convert between this pixel value and the individual RGB values.



Pixels and RGB

You can get individual red, green, and blue values from a pixel.

```
int[][] pixels = image.getPixelArray();
int pixel = pixels[0][0];

int red = GImage.getRed(pixel);       // 0-255
int green = GImage.getGreen(pixel);   // 0-255
int blue = GImage.getBlue(pixel);     // 0-255
```

Pixels and RGB

You can also *create* a pixel from individual R, G, B values.

```
int red = ...
int green = ...
int blue = ...
int pixel = GImage.createRGBPixel(red, green, blue);
```

Modifying Pixels

- Extract pixel RGB colors with GImage.getRed/Blue/Green.

```
int red = GImage.getRed(pixels[0][0]);      // 0-255
int green = GImage.getGreen(pixels[0][0]); // 0-255
int blue = GImage.getBlue(pixels[0][0]);    // 0-255
```

Modifying Pixels

- Extract pixel RGB colors with GImage.getRed/Blue/Green.

```
int red = GImage.getRed(pixels[0][0]);      // 0-255
int green = GImage.getGreen(pixels[0][0]); // 0-255
int blue = GImage.getBlue(pixels[0][0]);    // 0-255
```

- Modify the color components for a given pixel.

```
red = 0; // remove redness
```

Modifying Pixels

- Extract pixel RGB colors with GImage.getRed/Blue/Green.

```
int red = GImage.getRed(pixels[0][0]);      // 0-255
int green = GImage.getGreen(pixels[0][0]); // 0-255
int blue = GImage.getBlue(pixels[0][0]);    // 0-255
```

- Modify the color components for a given pixel.

```
red = 0; // remove redness
```

- Combine the RGB values back together into a single int.

```
pixels[0][0] = GImage.createRGBPixel(red, green, blue);
```

Modifying Pixels

- Extract pixel RGB colors with GImage.getRed/Blue/Green.

```
int red = GImage.getRed(pixels[0][0]);     // 0-255
int green = GImage.getGreen(pixels[0][0]); // 0-255
int blue = GImage.getBlue(pixels[0][0]);   // 0-255
```

- Modify the color components for a given pixel.

```
red = 0; // remove redness
```

- Combine the RGB values back together into a single int.

```
pixels[0][0] = GImage.createRGBPixel(red, green, blue);
```

- Update image with your modified pixels when finished.

```
image.setPixelArray(pixels);
```

GImage Pixel Methods

Method name	Description
`img.getPixelArray()`	returns pixels as 2D array of ints, where each int in the array contains all 3 of Red, Green, and Blue merged into a single integer
`img.setPixelArray(array);`	updates pixels using the given 2D array of ints
`GImage.createRGBPixel(r, g, b)`	returns an int that merges the given amounts of red, green and blue (each 0-255)

`GImage.getRed(px)`
`GImage.getGreen(px)`
`GImage.getBlue(px)`
	returns the redness, greenness, or blueness of the given pixel as an integer from 0-255

A Note on Image Size

- Destination image is same size → often modify array in place
- Destination image is different size → need a new array

A Note on Image Size

- Destination image is same size → often modify array in place
- Destination image is different size → need a new array

Example: Halve the size of an image.

```
int[][] pixels = img.getPixelArray();
int[][] smaller = new int[numRows(pixels) / 2][numCols(pixels)  / 2];
...
// set to be the pixels of 'smaller'
img.setPixelArray(smaller);
```

Plan for Today

- Review: Arrays
- 2D Arrays
- Images as 2D Arrays
- Manipulating Images
- Practice: Grayscale

Let’s Code It!

Plan for Today

- Review: Arrays
- 2D Arrays
- Images as 2D Arrays
- Manipulating Images
- Practice: Grayscale

Extra practice: Brighten

Reminder: MQE Feedback form

Next Time: ArrayLists!

[Extra] Brighten

- Practice: write a program that brightens an image each time you click on it!
- You can brighten a pixel by adding a small value (say, 5) to each of the red, green, and blue values.
- See lecture code for solution.