CS 106A, Lecture 17
2D Arrays and Images

suggested reading:

*Java Ch. 11.6-11.7*
Midterm!

Arrays

We are here

2D Arrays

Practice

Midterm!

The River of Java

HW5: ImageShop
Plan for Today

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images
• Practice: Brighten, Grayscale
Plan for Today

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images
• Practice: Brighten, Grayscale
A new variable type that is an **object** that represents an ordered, homogeneous list of data.

- Arrays have many *elements* that you can access using *indices*.

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>12</td>
<td>49</td>
<td>-2</td>
<td>26</td>
<td>5</td>
<td>17</td>
<td>-6</td>
<td>84</td>
<td>72</td>
<td>3</td>
</tr>
</tbody>
</table>

**length = 10**

- element 0
- element 4
- element 9
Creating an Array

`type[] name = new type[length];`

`int[] numbers = new int[5];`

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Creating an Array

Sometimes, we want to hardcode the elements of an array. Luckily, Java has a special syntax for initializing arrays to hardcoded numbers.

```java
// Java infers the array length
int[] numbers = {5, 32, 12, 2, 1, -1, 9};
```
name[index]    // get element at index

• Like Strings, indices go from 0 to the array's length - 1.

```java
for (int i = 0; i < 7; i++) {
    println(numbers[i]);
}
println(numbers[9]);    // exception
println(numbers[-1]);   // exception
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Putting Data In An Array

$name[index] = value; // set element at index$

- Like Strings, indices go from 0 to the array's length - 1.

```java
int[] numbers = new int[7];
for (int i = 0; i < 7; i++) {
    numbers[i] = i;
}
numbers[8] = 2; // exception
numbers[-1] = 5; // exception
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Similar to a String, you can get the length of an array by saying

```
myArray.length
```

Note that there are *no parentheses* at the end!
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations.

For instance, we can read in numbers from the user:

```java
int length = readInt("# of numbers? ");
int[] numbers = new int[length];
for (int i = 0; i < numbers.length; i++) {
    numbers[i] = readInt("Elem " + i + ": ");
}
```
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations.

For instance, we can sum up all of an array’s elements.

```java
int sum = 0;
for (int i = 0; i < numbers.length; i++) {
    sum += numbers[i];
}
println(sum);
```
Limitations of Arrays

• An array’s length is **fixed**. You cannot resize an existing array:

```java
int[] a = new int[4];
a.length = 10;  // error
```

• You cannot compare arrays with `==` or `equals`:

```java
int[] a1 = {42, -7, 1, 15};
int[] a2 = {42, -7, 1, 15};
if (a1 == a2) {  ...  }  // false!
if (a1.equals(a2)) {  ...  }  // false!
```

• An array does not know how to print itself:

```java
println(a1);  // [I@98f8c4]
```
# Arrays Methods To The Rescue!

- The class `Arrays` in package `java.util` has useful methods for manipulating arrays:

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Arrays.binarySearch(array, value)</code></td>
<td>returns the index of the given value in a <em>sorted</em> array (or &lt; 0 if not found)</td>
</tr>
<tr>
<td><code>Arrays.copyOf(array, length)</code></td>
<td>returns a new copy of array of given length</td>
</tr>
<tr>
<td><code>Arrays.equals(array1, array2)</code></td>
<td>returns true if the two arrays contain same elements in the same order</td>
</tr>
<tr>
<td><code>Arrays.fill(array, value);</code></td>
<td>sets every element to the given value</td>
</tr>
<tr>
<td><code>Arrays.sort(array);</code></td>
<td>arranges the elements into sorted order</td>
</tr>
<tr>
<td><code>Arrays.toString(array)</code></td>
<td>returns a string representing the array, such as &quot;[10, 30, -25, 17]&quot;</td>
</tr>
</tbody>
</table>
Plan for Today

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images
• Practice: Brighten, Grayscale, Shrink
The Matrix
2D Arrays ("Matrices")

\[
\begin{bmatrix}
  a_1 & b_1 & c_1 \\
  a_2 & b_2 & c_2 \\
  a_3 & b_3 & c_3
\end{bmatrix}
\]

WELCOME ..... TO THE MATRIX!!!!!!
2D Arrays

```java
type[][] name = new type[rows][columns];

int[][] a = new int[3][5];
```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a[0][0]</td>
<td>a[0][1]</td>
<td>a[0][2]</td>
<td>a[0][3]</td>
<td>a[0][4]</td>
</tr>
<tr>
<td>1</td>
<td>a[1][0]</td>
<td>a[1][1]</td>
<td>a[1][2]</td>
<td>a[1][3]</td>
<td>a[1][4]</td>
</tr>
</tbody>
</table>
Manipulating 2D Arrays

\[ \text{name[\text{row}][\text{col}]} \quad \text{// get element at row,col} \]

\[ \text{name[\text{row}][\text{col}]} = \text{value}; \quad \text{// set element at row,col} \]
2D arrays are arrays of arrays!
int[][] a = new int[3][4];
A 2D array is an array where every element is \textit{itself} an array.

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

```java
int[][] a = new int[3][4];
```

“array of” \(\text{int}\)

“array of” \(\text{int[]}\)
2D Arrays = Arrays of Arrays!

A 2D array is an array where every element is itself an array.

```java
int[][] a = new int[3][4];
...
int x = a[1][1];  // int at position (1, 1)

int[] firstRow = a[0];  // 1D array!

// NOTE: no way to get a single column 😞
```
How do we get the number of rows of a 2D array using the `length` property? How about the number of columns?
2D Array Dimensions

You can get the number of rows in a 2D array by saying:

```javascript
arr.length     // # rows in our 2D array
```

Since the length of each row is the same, you can get the number of columns in a 2D array by saying:

```javascript
arr[0].length   // # entries in row 0 ( = # cols)
```
We can use **double for-loops** to loop over each row, and then each column, in a 2D array.

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

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>75</td>
<td>61</td>
<td>83</td>
<td>71</td>
</tr>
<tr>
<td>1</td>
<td>94</td>
<td>89</td>
<td>98</td>
<td>91</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>54</td>
<td>51</td>
<td>49</td>
</tr>
</tbody>
</table>
```

“row-major” order
Practice: 2D Arrays

• Q: What is the array state after the code below?

```java
int[][] a = new int[4][3];
...
// fill with data at right
for (int r = 1; r < 4; r++) {
    for (int c = 0; c < 3; c++) {
        a[r][c] += a[r - 1][c];
    }
}
```

A.  
```
0      1      2    
0  1  3  5
1  1  3  5
2  1  3  5
3  1  3  5
```

B.  
```
0      1      2    
0  1  3  6
1  1  3  6
2  1  3  6
3  1  3  6
```

C.  
```
0      1      2    
0  1  2  3
1  2  4  6
2  2  4  6
3  2  4  6
```

D.  
```
0      1      2    
0  1  2  3
1  2  4  6
2  3  6  9
3  4  8 12
```
Limitations of 2D Arrays

• Unlike 1D arrays, you cannot compare 2D arrays with `Arrays.equals`. You must use `Arrays.deepEquals`.

```java
int[][] a1 = ...;
int[][] a2 = ...;
if (Arrays.deepEquals(a1, a2)) { ... }
```

• A 2D array does not know how to print itself:

```java
int[][] a = new int[rows][cols];
println(a); // [[I@8cf420
println(Arrays.toString(a)); // [[I@6b3f44,[I@32c2a8]...
println(Arrays.deepToString(a));
// [[0, 1, 2, 3, 4], [1, 2, ...
Summary: 2D Arrays

- Make a new 2D array
  \[
  \text{type}[][] \ \text{name} = \text{new type}[\text{rows}][\text{columns}];
  \]
- Get and set values using bracket notation
  \[
  \text{name}[\text{row}][\text{col}] \quad \text{// get elem at row,col}
  \]
  \[
  \text{name}[\text{row}][\text{col}] = \text{value}; \quad \text{// set elem at row,col}
  \]
- Get the number of rows and columns
  \[
  \text{arr.length} \quad \text{// # rows}
  \]
  \[
  \text{arr[0].length} \quad \text{// # columns}
  \]
- Iterate over a 2D array using a double for-loop
  \[
  \text{for (int row = 0; row < arr.length; row++)} \{
    \text{for (int col = 0; col < arr[0].length; col++)} \{
      \text{// do something with arr[row][col]};
    \}
  \}
  \]
Plan for Today

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images
• Practice: Brighten, Grayscale
Images are just grids (2D arrays!) of pixels! Pixels are just integer values from 0-255.
We can get a GIImage as a 2D array of pixels.

GIImage img = new GIImage("res/daisy.jpg");
int[][][] pixels = img.getPixelArray();
int pixel = pixels[0][0];  // top-left pixel
Example: Pointillism

Pointillism is an art style where many small dots of color are combined to make a larger image.

A Sunday on La Grande Jatte, Georges Seurat
Example: Pointillism

Pointillism is an art style where many small dots of color are combined to make a larger image.
Example: Pointillism

Pointillism is an art style where many small dots of color are combined to make a larger image.
Example: Pointillism

Pointillism is an art style where many small dots of color are combined to make a larger image.

Repeat many times:
1. Pick a random pixel from an image
2. Find the pixel’s color
3. “Paint” a large brush stroke of that color in the corresponding location
Example: Pointillism
Example: Pointillism

\[ c = 36 \]

\[ r = 46 \]
Example: Pointillism

\[ r = 46 \]

\[ c = 36 \]
Example: Pointillism
Example: Pointillism

```java
GImage daisy = new GImage("res/daisy.jpg");
int[][][] pixels = daisy.getPixelArray();
int rows = pixels.length;
int cols = pixels[0].length;

for (int i = 0; i < NUM_SAMPLES; i++) {
    int c = RandomGenerator.getInstance().nextInt(cols);
    int r = RandomGenerator.getInstance().nextInt(rows);
    int pixel = pixels[r][c];
    Color color = new Color(pixel);
    addColoredCircle(r, c, color);
}
```
Example: Pointillism

GImage daisy = new GImage("res/daisy.jpg");
int[][][] pixels = daisy.getPixelArray();
int rows = pixels.length;
int cols = pixels[0].length;

for (int i = 0; i < NUM_SAMPLES; i++) {
    int c = RandomGenerator.getInstance().nextInt(cols);
    int r = RandomGenerator.getInstance().nextInt(rows);
    int pixel = pixels[r][c];
    Color color = new Color(pixel);
    addColoredCircle(r, c, color);
}
Example: Pointillism

```java
GImage daisy = new GImage("res/daisy.jpg");
int[][][] pixels = daisy.getPixelArray();
int rows = pixels.length;
int cols = pixels[0].length;

for (int i = 0; i < NUM_SAMPLES; i++) {
    int c = RandomGenerator.getInstance().nextInt(cols);
    int r = RandomGenerator.getInstance().nextInt(rows);
    int pixel = pixels[r][c];
    Color color = new Color(pixel);
    addColoredCircle(r, c, color);
}
```
Example: Pointillism

GImage daisy = new GImage(“res/daisy.jpg”);
int[][] pixels = daisy.getPixelArray();
int rows = pixels.length;
int cols = pixels[0].length;

for (int i = 0; I < NUM_SAMPLES; i++) {
    int c = RandomGenerator.getInstance().nextInt(cols);
    int r = RandomGenerator.getInstance().nextInt(rows);
    int pixel = pixels[r][c];
    Color color = new Color(pixel);
    addColoredCircle(r, c, color);
}
GImage daisy = new GImage("res/daisy.jpg");
int[][] pixels = daisy.getPixelArray();
int rows = pixels.length;
int cols = pixels[0].length;

for (int i = 0; i < NUM_SAMPLES; i++) {
    int c = RandomGenerator.getInstance().nextInt(cols);
    int r = RandomGenerator.getInstance().nextInt(rows);
    int pixel = pixels[r][c];
    Color color = new Color(pixel);
    addColoredCircle(r, c, color);
}
GImage daisy = new GImage(“res/daisy.jpg”);
int[][][] pixels = daisy.getPixelArray();
int rows = pixels.length;
int cols = pixels[0].length;

for (int i = 0; i < NUM_SAMPLES; i++) {
    int c = RandomGenerator.getInstance().nextInt(cols);
    int r = RandomGenerator.getInstance().nextInt(rows);
    int pixel = pixels[r][c];
    Color color = new Color(pixel);
    addColoredCircle(r, c, color);
}
Example: Pointillism

GImage daisy = new GImage("res/daisy.jpg");
int[][] pixels = daisy.getPixelArray();
int rows = pixels.length;
int cols = pixels[0].length;

for (int i = 0; i < NUM_SAMPLES; i++) {
    int c = RandomGenerator.getInstance().nextInt(cols);
    int r = RandomGenerator.getInstance().nextInt(rows);
    int pixel = pixels[r][c];
    Color color = new Color(pixel);
    addColoredCircle(r, c, color);
}
// Assume canvas is exactly image size
private void addColoredCircle(int r, int col, Color c) {
    double size = CIRCLE_RADIUS * 2;
    GOval circle = new GOval(size, size);
    circle.setFilled(true);
    circle.setColor(c);
    add(oval, col - CIRCLE_RADIUS, r - CIRCLE_RADIUS);
}
Plan for Today

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images
• Practice: Brighten, Grayscale
• Out of curiosity, do you also happen to like broccoli?
• Can I see more Daisy?
• A slingshot may help you with candy delivery (your accuracy tends to drop off after ~10 rows). Make sure its a low-powered slingshot, though, as I don't think students would appreciate being pelted with candy at high speeds.
Plan for Today

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images
• Practice: Brighten, Grayscale
Images *encode* the R, G, and B values of a pixel into a single integer between 0 and 255. You can convert between this **pixel value** and the individual **RGB values**.

```java
int[][][] pixels = image.getPixelArray();
int px = pixels[0][0];
int red = GImage.getRed(px);
int green = GImage.getGreen(px);
int blue = GImage.getBlue(px);
```
Images *encode* the R, G, and B values of a pixel into a single integer between 0 and 255. You can convert between this *pixel value* and the individual *RGB values*.

You can also create pixels with your own RGB values.

```java
int r = ...
int g = ...
int b = ...
int pixel = GImage.createRGBPixel(r, g, b);
```
Images as 2D Arrays

We can get a GImage as a 2D array of pixels, and modify it any way we want. Then, we can create a new GImage with the modified pixels.

```java
GImage img = new GImage("res/daisy.jpg");
int[][][] pixels = img.getPixelArray();
... (modify pixels)
img.setPixelArray(pixels);  // update image

// or make a new GImage
GImage newImg = new GImage(pixels);
```
Modifying Image Pixels

• There are many cool image algorithms based around modifying individual pixels in an image: grayscale, brighten, normalize, remove red-eye...

grayscale

zoom
GImage img = new GImage("res/daisy.jpg");

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>img</strong>.getPixelArray()</td>
<td>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</td>
</tr>
<tr>
<td><strong>img</strong>.setPixelArray(\textit{array});</td>
<td>updates pixels using the given 2D array of ints</td>
</tr>
<tr>
<td>\texttt{GImage.createRGBPixel(r, g, b)}</td>
<td>returns an int that merges the given amounts of red, green and blue (each 0-255)</td>
</tr>
<tr>
<td>\texttt{GImage.getRed(px)}</td>
<td>returns the redness, greenness, or blueness of the given pixel as an integer from 0-255</td>
</tr>
<tr>
<td>\texttt{GImage.getGreen(px)}</td>
<td></td>
</tr>
<tr>
<td>\texttt{GImage.getBlue(px)}</td>
<td></td>
</tr>
</tbody>
</table>
Recap: Modifying Pixels

• **Extract** pixel RGB colors with `GImage.getRed/Blue/Green`.

  ```java
  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.

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

• **Combine** the RGB back together into a single `int`.

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

• **Update** the image with your modified pixels when finished.

  ```java
  image.setPixelArray(pixels);
  ```
Changing Image Size

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

Example: **Half the size** of an image.

```java
int[][][] pixels = img.getPixelArray();
int[][][] smaller = new int[pixels.length / 2][pixels[0].length / 2];
...
// set to be the pixels of ‘smaller’
img.setPixelArray(smaller);
```
Plan for Today

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images

• Practice: Brighten, Grayscale
Recap

• Recap: Arrays
• 2D Arrays
• Images as 2D Arrays
• Announcements
• Modifying Images
• Practice: Brighten, Grayscale

Next time: more practice with arrays