

LEC 06

CSE 122

2D Arrays

BEFORE WE START

*Talk to your neighbors:  
What is your go-to study snack?*

---

**Instructors** Tristan Huber & Hunter Schafer**TAs**Ambika  
Andrew  
Audrey  
Autumn  
Ayush  
Ben  
Colton  
Di  
Eesha  
ElizabethEvelyn  
Jacob  
Jaylyn  
Jin  
Joe  
Kevin  
Leon  
Megana  
Melissa  
MiaPoojitha  
Rishi  
Rucha  
Shivani  
Shreya  
Steven  
Suhani  
Yijia  
Ziao


Questions during Class?

Raise hand or send here

sli.do #cse122




# Lecture Outline

- **Announcements** 
- 2D Arrays Review
- Images
- Images with 2D Arrays!

# Announcements

- First round of Quiz 0 Retakes happened yesterday!
  - Maximum one retake per quiz
  - Quiz 0 retakes still possible 4/25 and 5/2 (same day as Q1)
  - Grades for Quiz 0 Retakes will be released after 5/2
- Programming Assignment 1 is due tomorrow
- Creative Project 1 released this Friday (image editing)
- Quiz 1 scheduled for 5/2

# Lecture Outline

- Announcements
- **2D Arrays Review** 
- Images
- Images with 2D Arrays!

# (PCM) Arrays

- The type of an array is *ElementType*[]  
- *ElementType* can be any type!
- Can store multiple elements *of the same type*
- Need to specify length of array and type of elements it will store at creation

```
int[] arr = new int[4];
```

type                      name                      array creation code

int[]

double[]

String[]

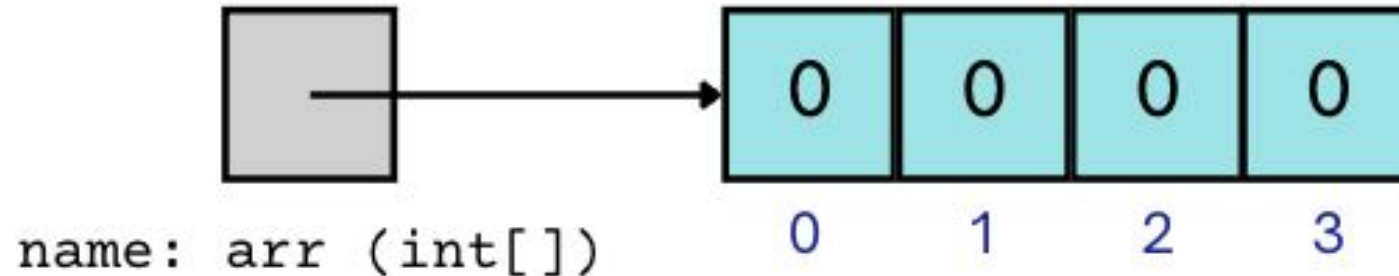
boolean[]

char[]

# (PCM) Arrays

- The type of an array is *ElementType*[]  
- *ElementType* can be any type!
- Can store multiple elements *of the same type*
- Need to specify length of array and type of elements it will store at creation

```
int[] arr = new int[4];
```



# (PCM) 2D Arrays

*An array of arrays!*

- The *ElementType* of the array is another array itself!
  - Your first example of “nested data structures”
  - There will be more!

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

type            name            array creation code

int[][]

double[][]

String[][]

boolean[][]

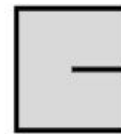
char[][]

# (PCM) 2D Arrays

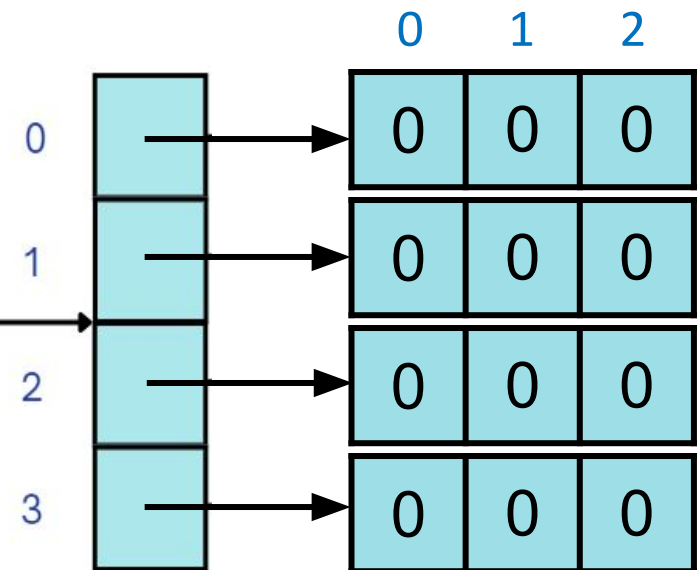
An array whose elements are arrays\*.

*reference semantics!*

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



name: a (int[][])



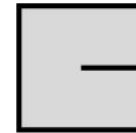


# (PCM) 2D Arrays

*An array of arrays!*

... a slightly more condensed view...

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



name: a (int[][])

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

**a[2][1]**

# (PCM) 2D Array Traversals

```
for (int i = 0; i < list.length; i++) {  
    for (int j = 0; j < list[i].length; j++) {  
        // do something with list[i][j]  
    }  
}
```

# Arrays Utility Class

Method	Description
<code>Arrays.toString(array);</code>	Returns a <code>String</code> representing the array, such as "[10, 30, -25, 17]"
<code>Arrays.fill(array, value);</code>	Sets every element to the given value
<code>Arrays.equals(array1, array2);</code>	Returns <code>true</code> if the two arrays contain the same elements in the same order
<code>Arrays.deepToString(array);</code>	Returns a <code>String</code> representing the array; if the array contains other arrays as elements, the <code>String</code> represents their contents, and so on. For example, "[[99, 151], [30, 5]]"
<code>Arrays.deepEquals(array1, array2);</code>	Returns <code>true</code> if the two arrays contain the same elements in the same order; if the array(s) contain other arrays as elements, their contents are tested for equality, and so on.

# Applications of 2D Arrays

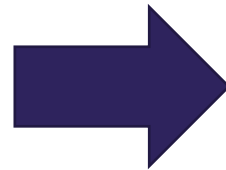
- Matrices
  - Useful in various applications requiring complex math!
- Board games
  - (e.g., chess/checkerboard, tic tac toe, sudoku)
- Representing information in a grid or table
  - (e.g., scorekeeping, gradebook)
- Image processing

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

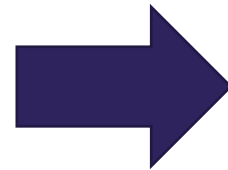



# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

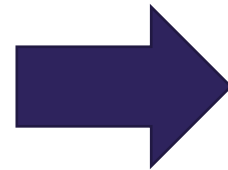



# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



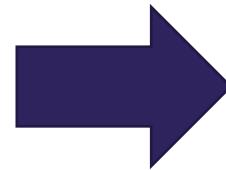
93				

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



93	169			

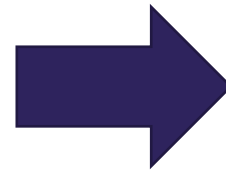


# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



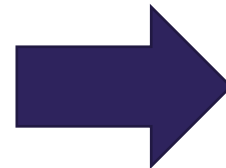
93	169	84		

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



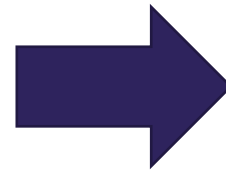
93	169	84	83	

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



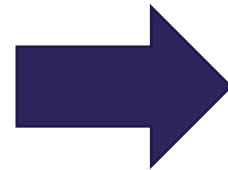
93	169	84	83	103

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



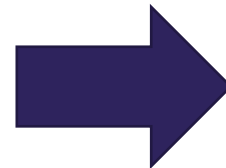
93	169	84	83	103
136				

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



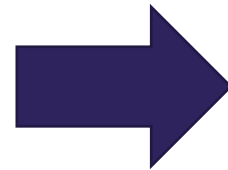
93	169	84	83	103
136	115			

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



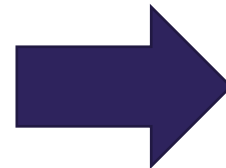
93	169	84	83	103
136	115	91		

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



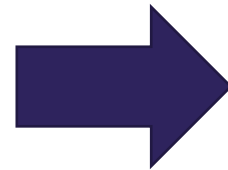
93	169	84	83	103
136	115	91	143	

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



93	169	84	83	103
136	115	91	143	81

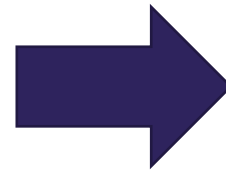


# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



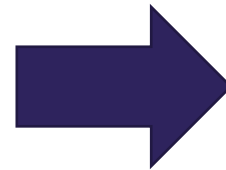
93	169	84	83	103
136	115	91	143	81
119				

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



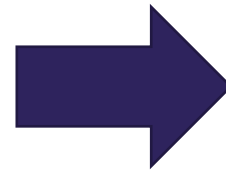
93	169	84	83	103
136	115	91	143	81
119	77			

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



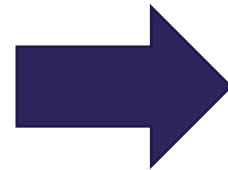
93	169	84	83	103
136	115	91	143	81
119	77	98		

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



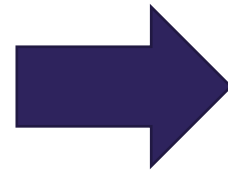
93	169	84	83	103
136	115	91	143	81
119	77	98	105	

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



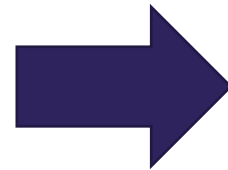
93	169	84	83	103
136	115	91	143	81
119	77	98	105	13

# matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

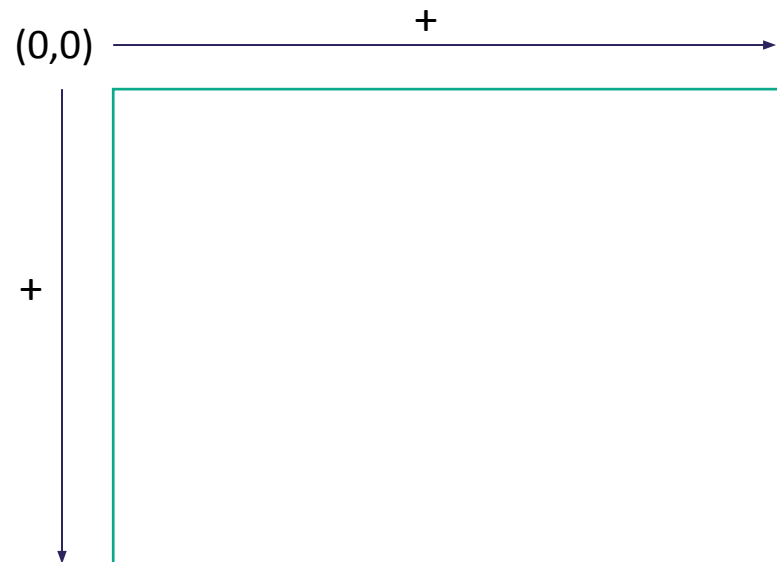
# Lecture Outline

- Announcements
- 2D Arrays Review
- **Images** 
- Images with 2D Arrays!

# Images

From the computer's perspective, images are just a big grid of values called **pixels**.

Each pixel shows a different color based on a specified value.





# Images

If images are just grids of pixels, and we can think of 2D arrays as grids,

We can represent images as 2D arrays of pixels!

Further, since each pixel is shown as a specific color,

We can represent images as 2D arrays of colors!

# Images in Java

- `Picture.java`

- Represents the idea of a picture in your program

- `Color.java`

- Represents colors in your program!

- Uses the RGB color scheme where each color is made up of some amount (0-255) of **red**, **green**, and **blue**

# Images in Java: `Picture.java`

```
Picture pic = new Picture("gumball.png");
```

Methods	Descriptions
<code>pic.getPixels();</code>	Returns a <code>Color[][]</code> representing the colors in the grid of pixels.
<code>pic.setPixels(colorArray);</code>	Sets the grid of pixels in the picture based on the given <code>colorArray</code> .
<code>pic.save(fileName);</code>	Saves the current picture to a file with the given <code>fileName</code> .
<code>pic.show();</code>	Shows the current picture in a window on the screen.*


\* This functionality doesn't work perfectly on Ed, it's probably easier to use the `save()` method!

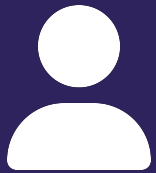
# Images in Java: Color.java

```
Color color = new Color(redVal, greenVal, blueVal);
```

Methods	Descriptions
<code>color.getRed();</code>	Returns the color amount for red.
<code>color.getGreen();</code>	Returns the color amount for green.
<code>color.getBlue();</code>	Returns the color amount for blue.

# Lecture Outline

- Announcements
- 2D Arrays Review
- Images
- **Images with 2D Arrays!** 



# Practice : Think

[sli.do](https://sli.do)

#cse-122

## What loop bounds will create the desired rectangle?

Draw a blue rectangle with **top-left** at row 200, column 400.  
With **width** of 100 and **height** of 180

```
for (int row = ?? ; row < ?? ; row++) {  
    for (int col = ?? ; col < ?? ; col++) {  
        pixels[row][col] = new Color(0, 0, 255);  
    }  
}
```

- A) 

```
for (int row = 400; row < 580; row++) {  
    for (int col = 200; col < 300; col++) {
```
- B) 

```
for (int row = 200; row < 380; row++) {  
    for (int col = 400; col < 500; col++) {
```
- C) 

```
for (int row = 200; row < 100; row++) {  
    for (int col = 400; col < 180; col++) {
```



# Practice : Pair

[sli.do](https://sli.do)

#cse-122

## What loop bounds will create the desired rectangle?

Draw a blue rectangle with **top-left at row 200, column 400**.  
With **width of 100** and **height of 180**

```
for (int row = ?? ; row < ?? ; row++) {  
    for (int col = ?? ; col < ?? ; col++) {  
        pixels[row][col] = new Color(0, 0, 255);  
    }  
}
```

- A) 

```
for (int row = 400; row < 580; row++) {  
    for (int col = 200; col < 300; col++) {
```
- B) 

```
for (int row = 200; row < 380; row++) {  
    for (int col = 400; col < 500; col++) {
```
- C) 

```
for (int row = 200; row < 100; row++) {  
    for (int col = 400; col < 180; col++) {
```



# Practice : Pair



sli.do

#cse-122

## What loop bounds will create the desired rectangle?

Draw a blue rectangle with **top-left at row 200, column 400**.  
With **width of 100** and **height of 180**

```
for (int row = ?? ; row < ?? ; row++) {  
    for (int col = ?? ; col < ?? ; col++) {  
        pixels[row][col] = new Color(0, 0, 255);  
    }  
}
```

A) 

```
for (int row = 400; row < 580; row++) {  
    for (int col = 200; col < 300; col++) {
```



B) 

```
for (int row = 200; row < 380; row++) {  
    for (int col = 400; col < 500; col++) {
```

C) 

```
for (int row = 200; row < 100; row++) {  
    for (int col = 400; col < 180; col++) {
```