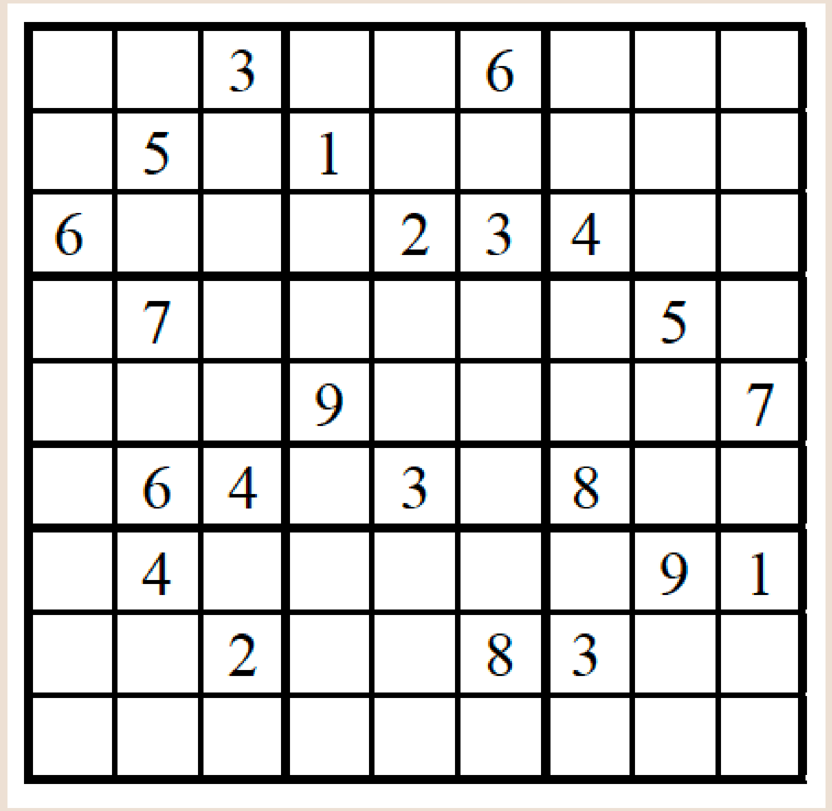
Topic 8.0: Multidimensional Arrays

Learning Goals (Topic 8):

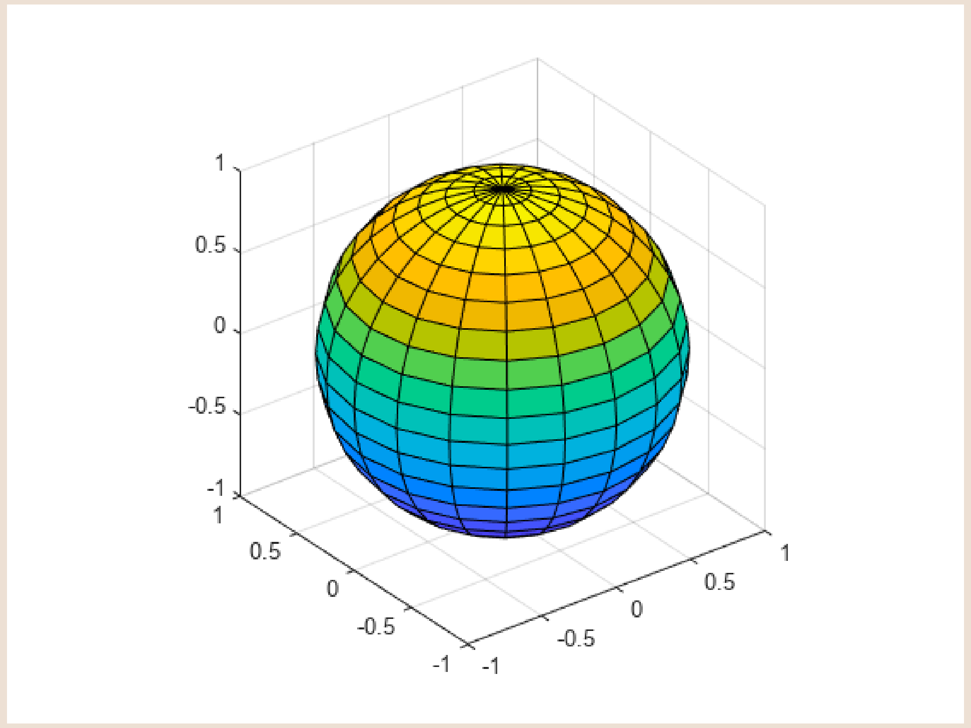
- **Write code to create and manipulate Multidimensional Arrays**
- **Given a piece of code, draw a diagram representing the state of references in a multi-dimensional array;**
- Write code that reads and manipulates multidimensional array information from a file
- **Write code that can create and manipulate ragged arrays**

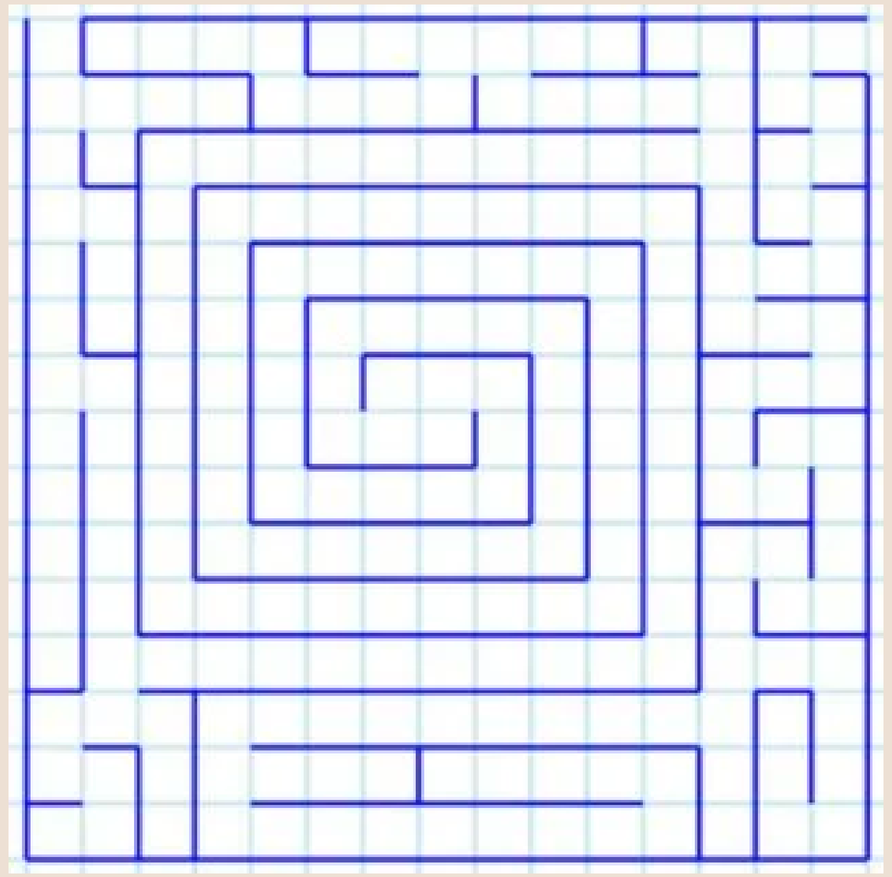
Multidimensional arrays

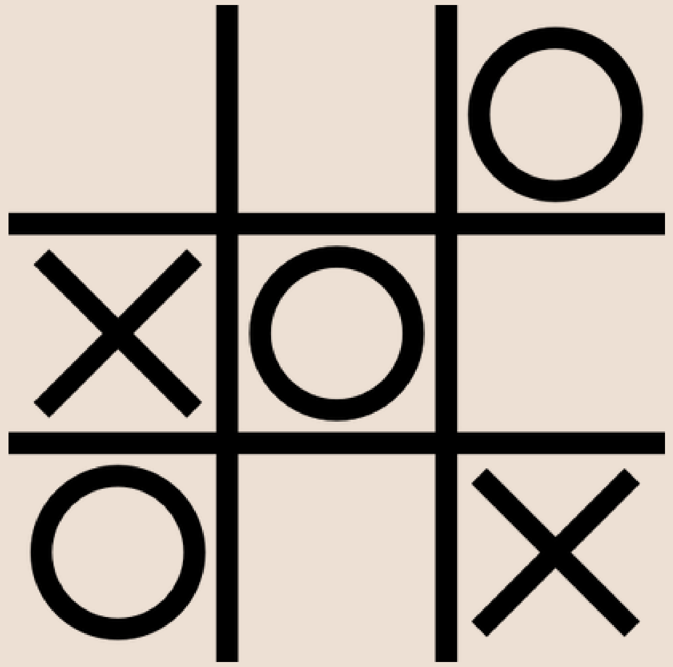
- Data often needs to be organized into a matrix of rows and columns (often 2D but could be more!)



		3			6			
	5		1					
6				2	3	4		
	7						5	
			9					7
	6	4		3		8		
	4						9	1
		2			8	3		



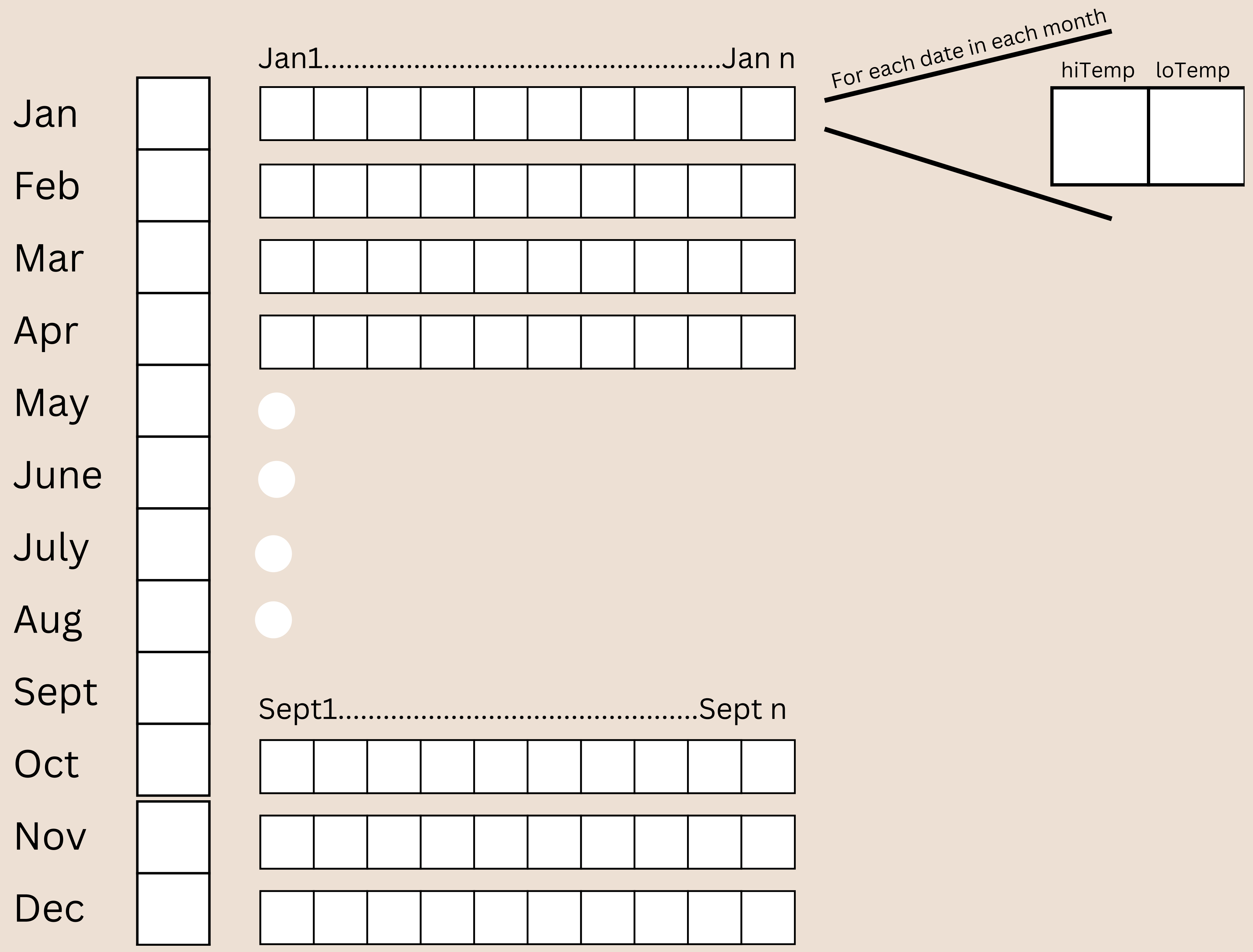




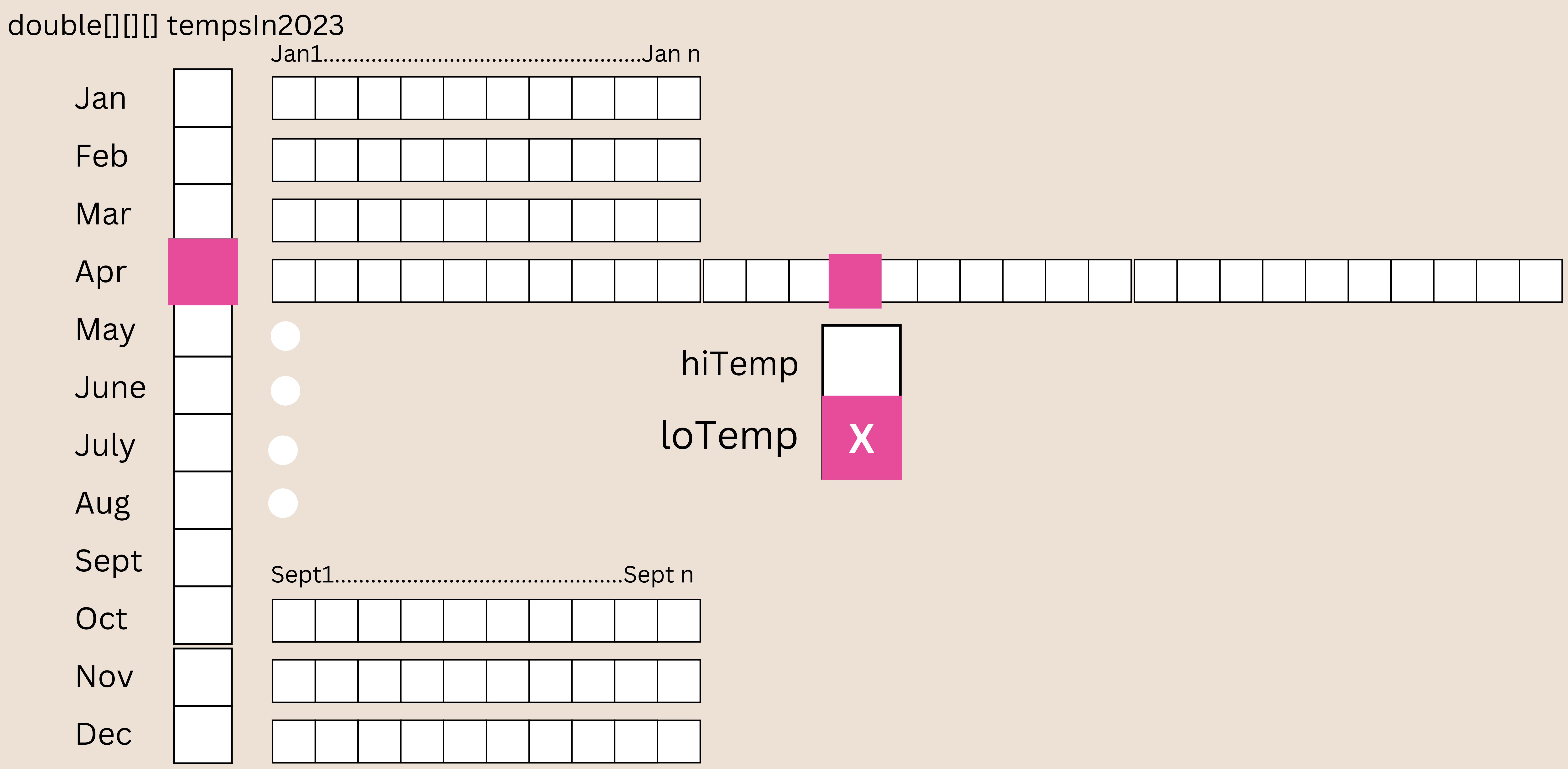
Multidimensional arrays

- Let’s say you want to record high and low temperatures in Winnipeg for every day in 2023:
- You would need:
  - An array of arrays of arrays (huh?)
  - For each HI/LO in a Day in a Month in a Year

Multidimensional arrays

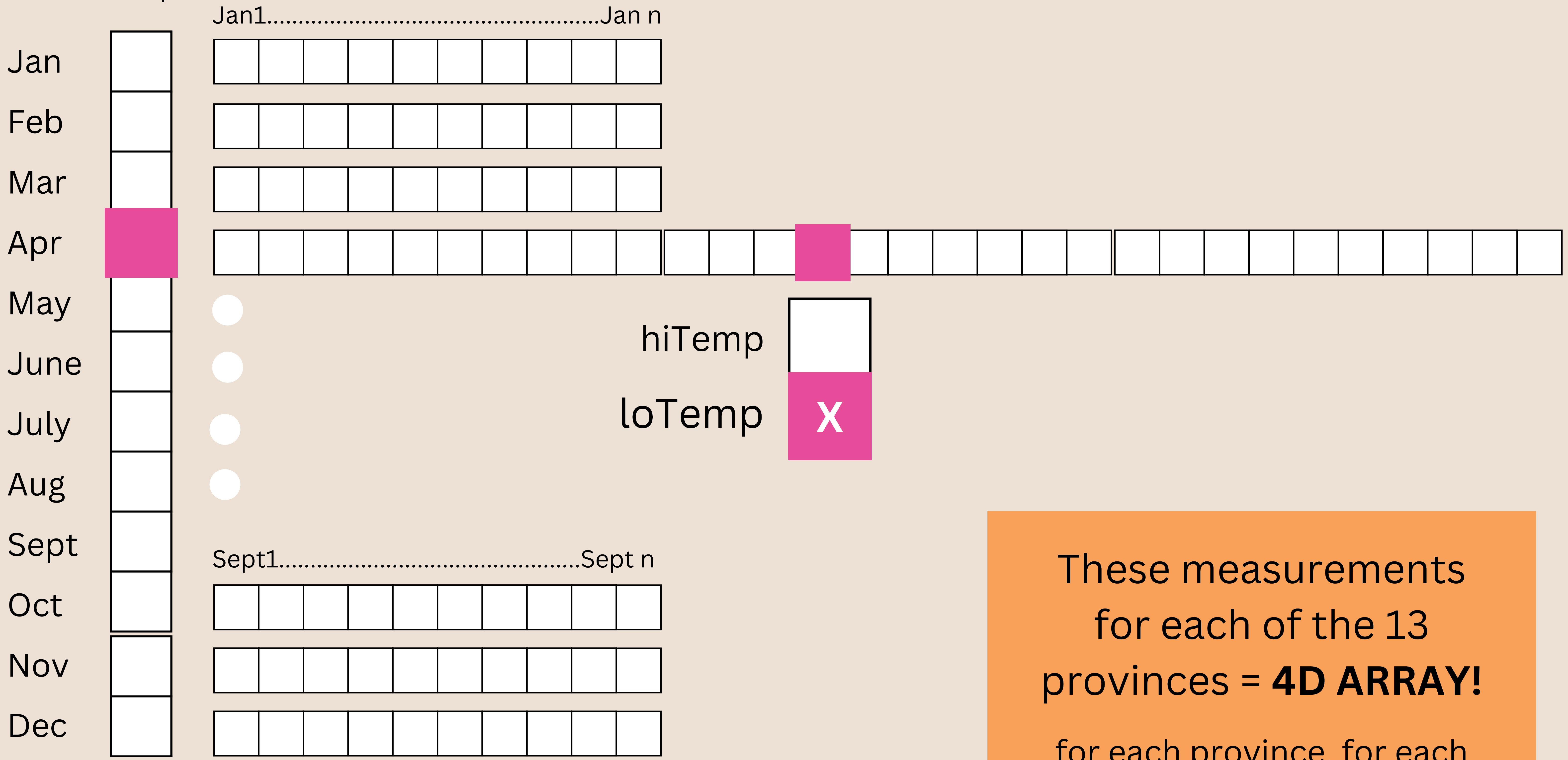

Jan
Feb
Mar
Apr
May
June
July
Aug
Sept
Oct
Nov
Dec
Jan1...Jan n
Sept1...Sept n
For each date in each month
hiTemp
loTemp

What was the low temp on April 14? **tempsIn2023[3][13][1]**


double[][][] tempsIn2023
Jan1...Jan n
Jan
Feb
Mar
Apr
May
June
July
Aug
Sept
Oct
Nov
Dec
hiTemp
loTemp
X
Sept1..Sept n

What if you want these for each province?

double[][][] tempsIn2023


Jan1...Jan n
Jan
Feb
Mar
Apr
May
June
July
Aug
Sept
Oct
Nov
Dec
hiTemp
loTemp
X
Sept1...Sept n

These measurements for each of the 13 provinces = **4D ARRAY!**

for each province, for each month, for each day, the hi/low)

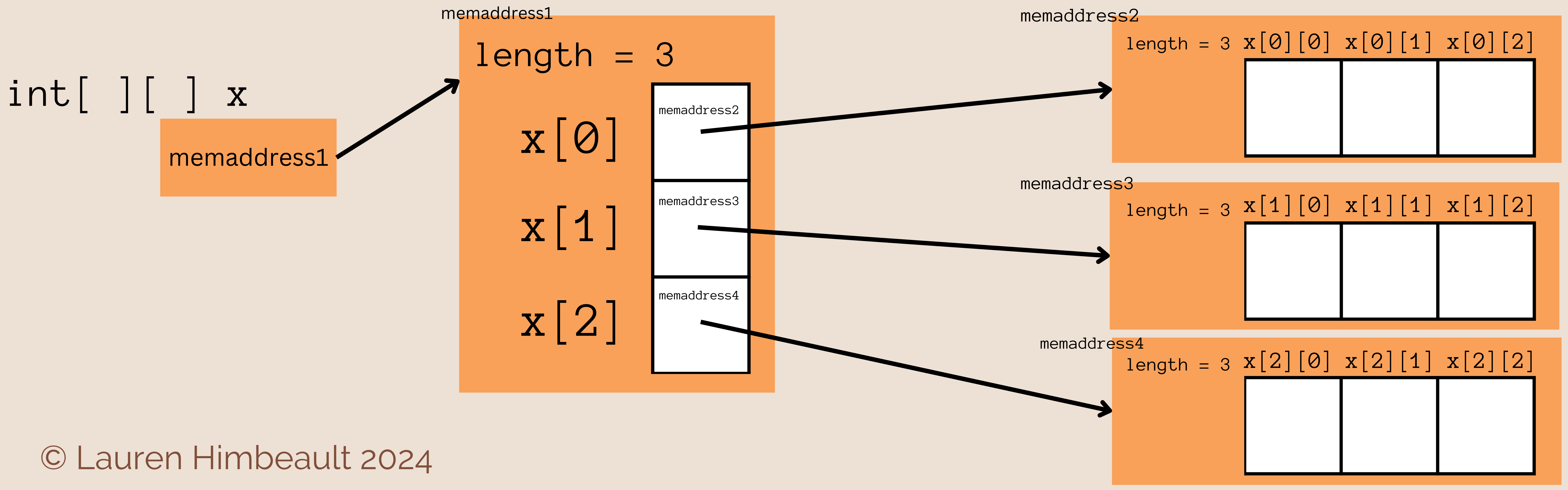
Multidimensional arrays

- In Java, we can have an array of ints, doubles, booleans, Strings, people
- **Why not an array of arrays?**

```
any-type[ ] x = new any-type[n]; //n any-type's
```

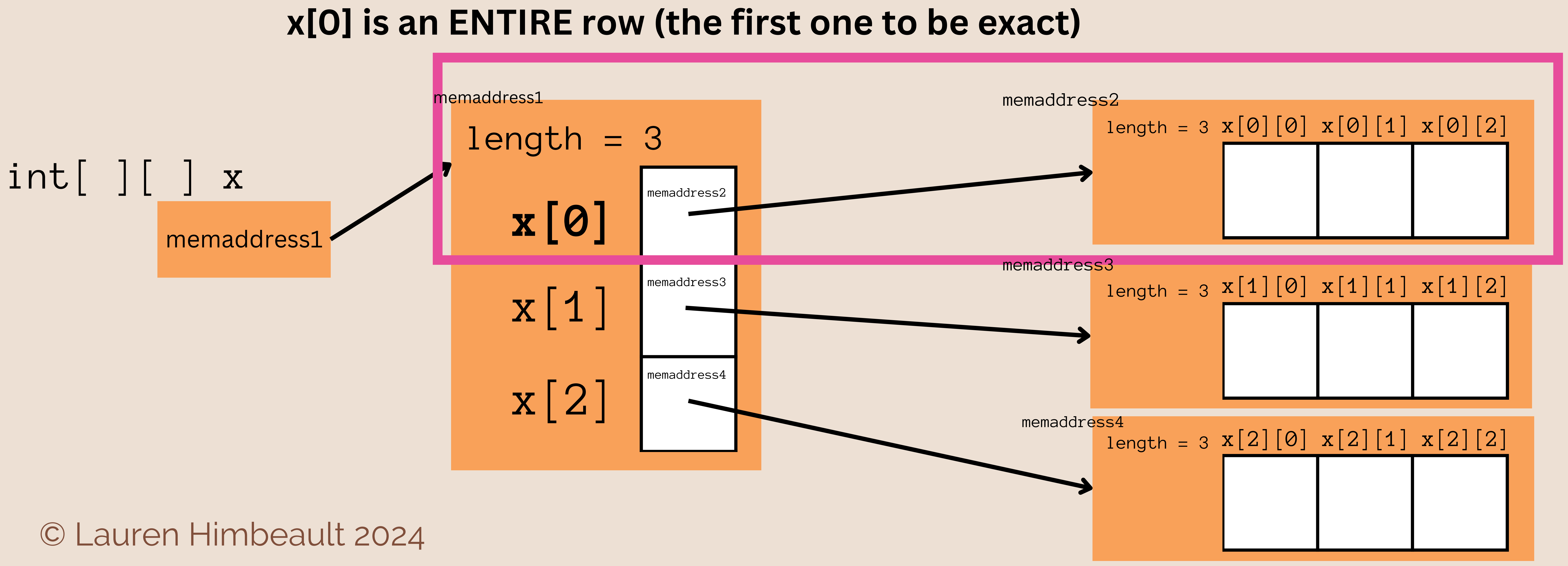
- "int[ ]" is a type, so you could have:

```
int[ ][ ] x = new int[3][3]; //a 3x3 array
```



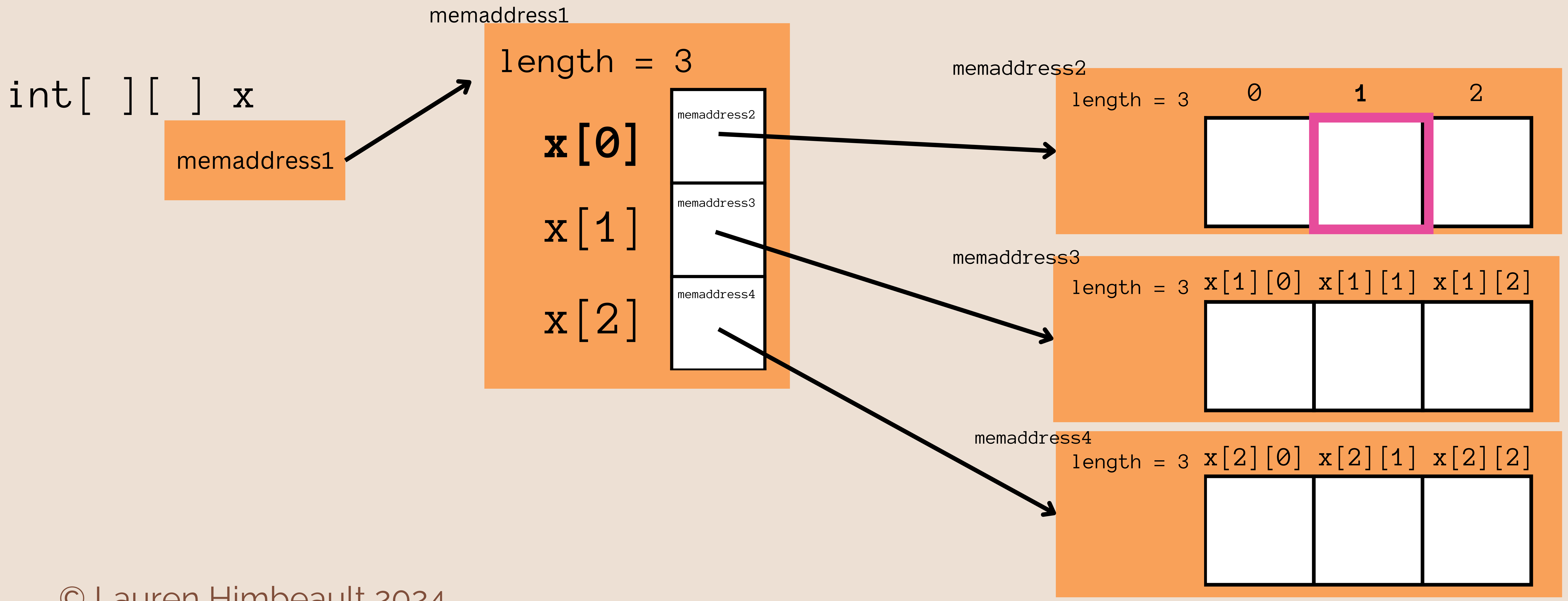
Multidimensional arrays

- `x is the reference to the whole matrix`



Multidimensional arrays

x[0][1] is the cell at index 1 in row 0



Multidimensional arrays

This can be redrawn a bit like a matrix

```
x[0][0] =  3;
x[0][2] = -5;
x[1][1] =  7;
x[2][0] =  1;
x[2][1] = 10;
x[1][0] = -6;
```

$$\begin{array}{c|c|c|c|}
 & 0 & 1 & 2 \\ \hline
0 & 3 & 0 & -5 \\ \hline
1 & -6 & 7 & 0 \\ \hline
2 & 1 & 10 & 0 \\ \hline
\end{array}$$

When thinking of it as a matrix, x[r][c] is the element in row r and column c.

Multidimensional arrays

This can be redrawn a bit like a matrix

```
x[0][0] =  3;
x[0][2] = -5;
x[1][1] =  7;
x[2][0] =  1;
x[2][1] = 10;
x[1][0] = -6;
```

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 3 & 0 & -5 \\ 1 & -6 & 7 & 0 \\ 2 & 1 & 10 & 0 \end{array}$$

REMINDER:
This is not how the data is stored in memory.

It is stored as an array of arrays (pictures with the arrows)

Multidimensional arrays

Sometimes it is a bit easier to think of this way (as a matrix)
But remember this is not actually how memory works so if you are asked to draw how it is stored in memory, a matrix like this is **incorrect**

```
int[][] x = new int[2][3];
```

$$\begin{array}{c|c|c|c|} & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}$$

REMINDER:
This is not how the data is stored in memory.

It is stored as an array of arrays (pictures with the arrows)

Reminder for › 2 Dimensions

- More than 2 dimensions are really rows/columns anymore.
- It becomes much harder to represent it as some kind of matrix.
- Recognizing how it is actually stored makes > 2D arrays easier to visualize (see our weather example from before)

- Just make sure you remember which dimension is which when you declare and use your multidimensional array...

- ... and always remember that a multidimensional array is an array of arrays of [anything]
- We will look at other ways to help us visualize this shortly

Creates & Accessing Multidimensional Arrays

- We’ve seen the creation:
  - Declare the datatype and the number of array levels:
    - String [] [] [] // 3D array of Strings or an array of arrays which contain an array of strings :)
    - int [] [] // 2D array of ints or an array of integer arrays
  - Just like a 1D array, the right side ‘[]’ need a number inside them to define the number of elements
    - this just needs to happen for each level:

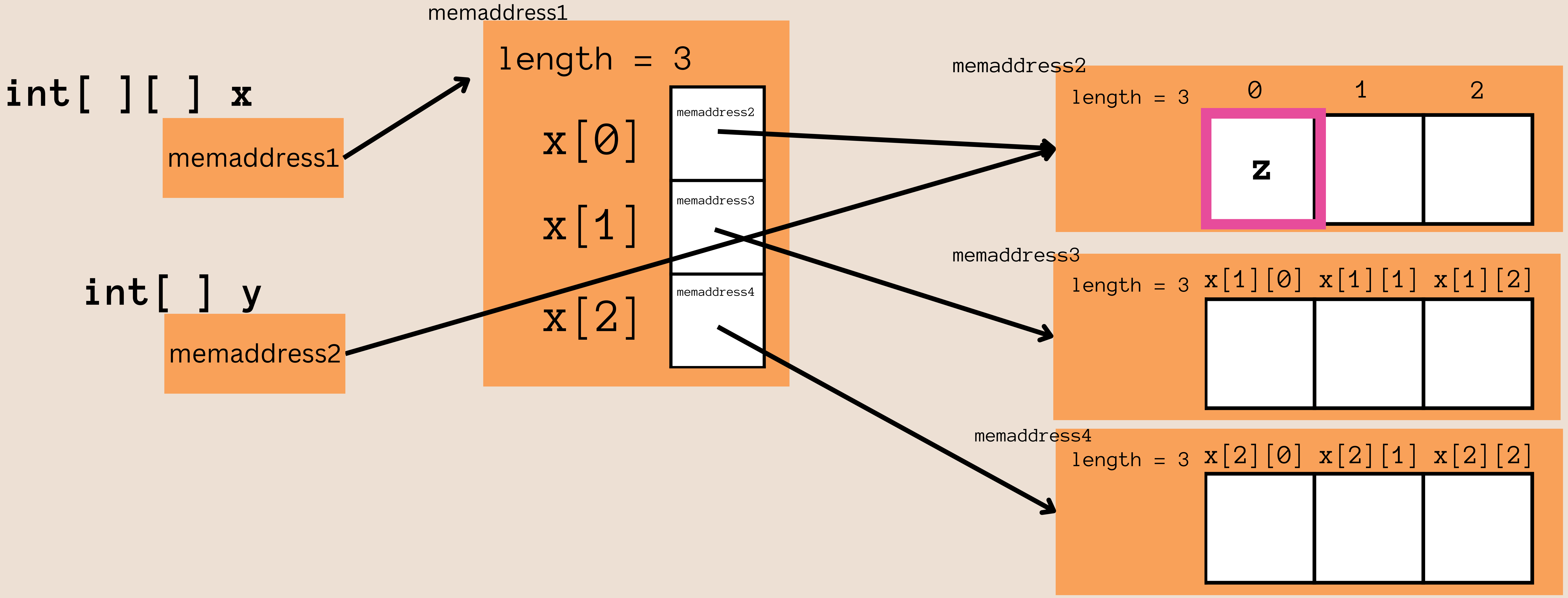
```
String[][][] a = new String[4][10][2];
/*
 * An array with 4 spots. Each index has an array with 10 elements in
 * it. Each of those elements is a String[] with 2 spots. Each of those
 * spots is a String
 */

int[][] b = new int[2][2];
/*
 * An array with 2 spots (rows). Each element stores an array with 2 ints in
 * it. Each of those elements are the ints
 */
```

Creates & Accessing Multidimensional Arrays

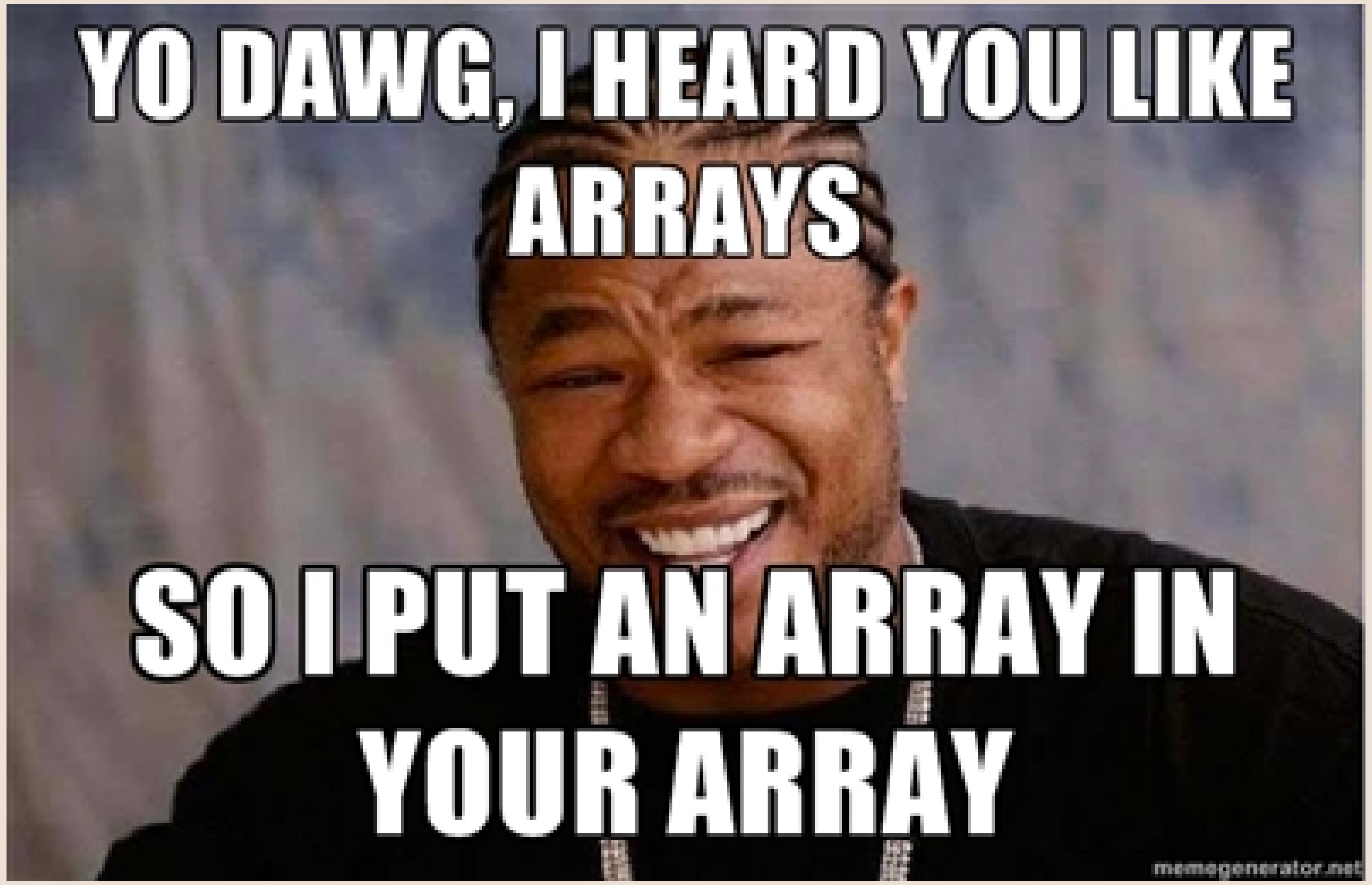
- Accessing Elements Walks you Down that Path of arrays to the value.

```
int[][] x = new int[3][3];
int[]   y = x[0];   // first row -> alias for memaddress2
int     z = y[0];   // first element from first row
```



An interesting note

- There are ***practically*** no limits to the number of dimensions (2D,3D,4D,5D,6D, etc)
  - The limit **does** exist (255) but that's cuckoo banana pants. Nobody needs that


YO DAWG, I HEARD YOU LIKE ARRAYS
SO I PUT AN ARRAY IN YOUR ARRAY
memegenerator.net

stackoverflow (check this out it's got a silly billion D array)

Hardcoding Multidimensional Arrays

- Just like 1D arrays we can hardcode values:

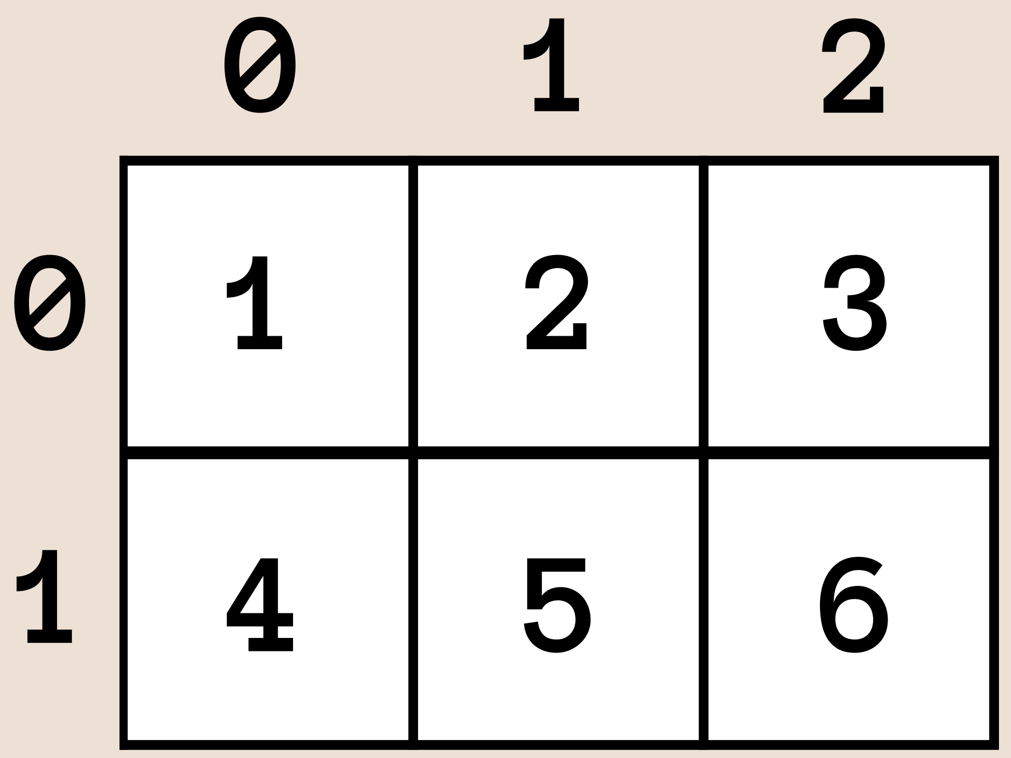
```
int[][] x = { {1,2,3}, {4,5,6} };
```

	0	1	2
0	1	2	3
1	4	5	6

Hardcoding Multidimensional Arrays

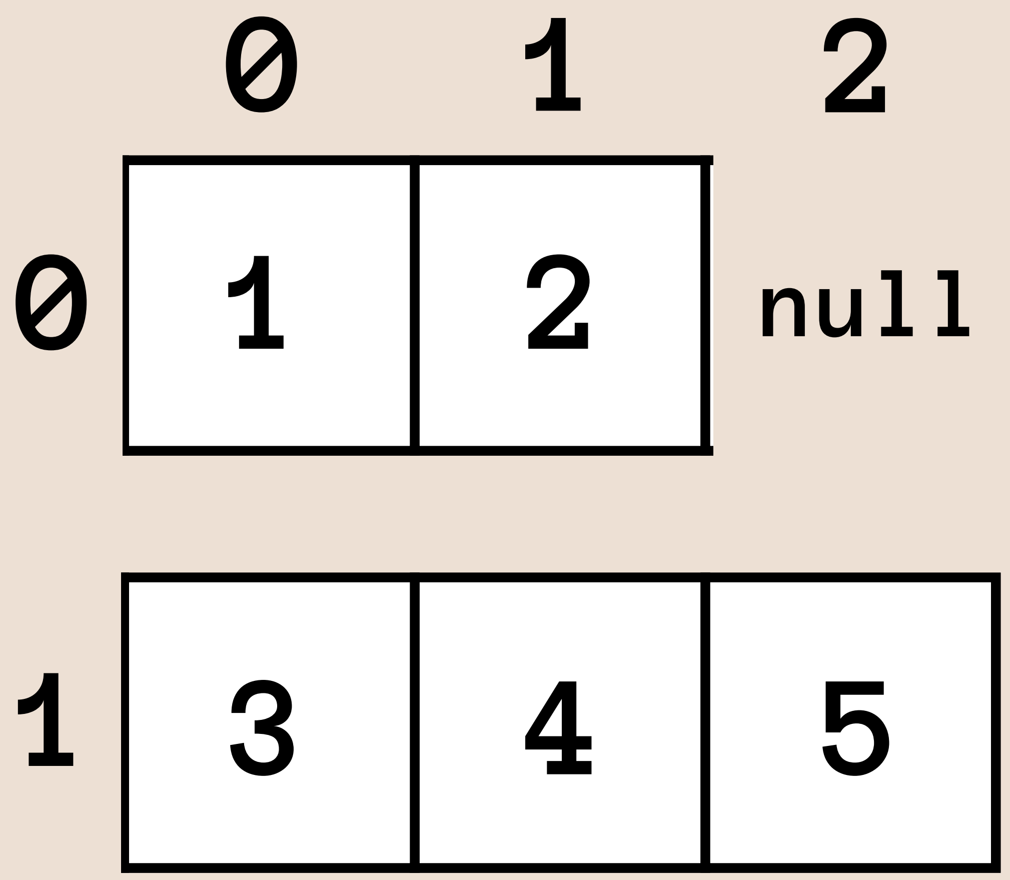
- Just like 1D arrays we can hardcode values:

```
int[][] x = { {1,2,3}, {4,5,6} };
```



- Rows can be different sizes (more on this later)

```
int[][] y = { {1,2}, {3,4,5} };
```



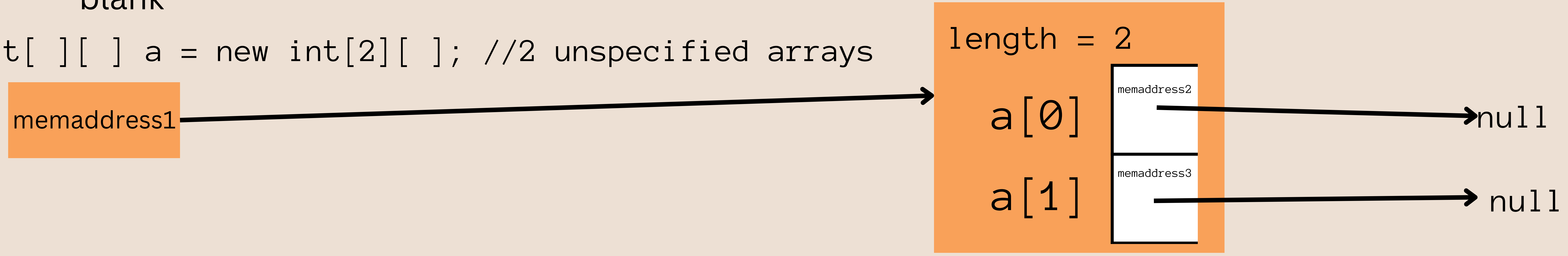
Using the ‘new’ keyword

- When you use **new**, like this:

```
int[ ][ ] a = new int[2][4];  //2 arrays, each an int[4] object
```

- **You can leave the smaller arrays unallocated**
  - the last dimensions only, i.e. you must give the first $k \geq 1$ sizes, then you can leave the rest blank

```
int[ ][ ] a = new int[2][ ]; //2 unspecified arrays
```

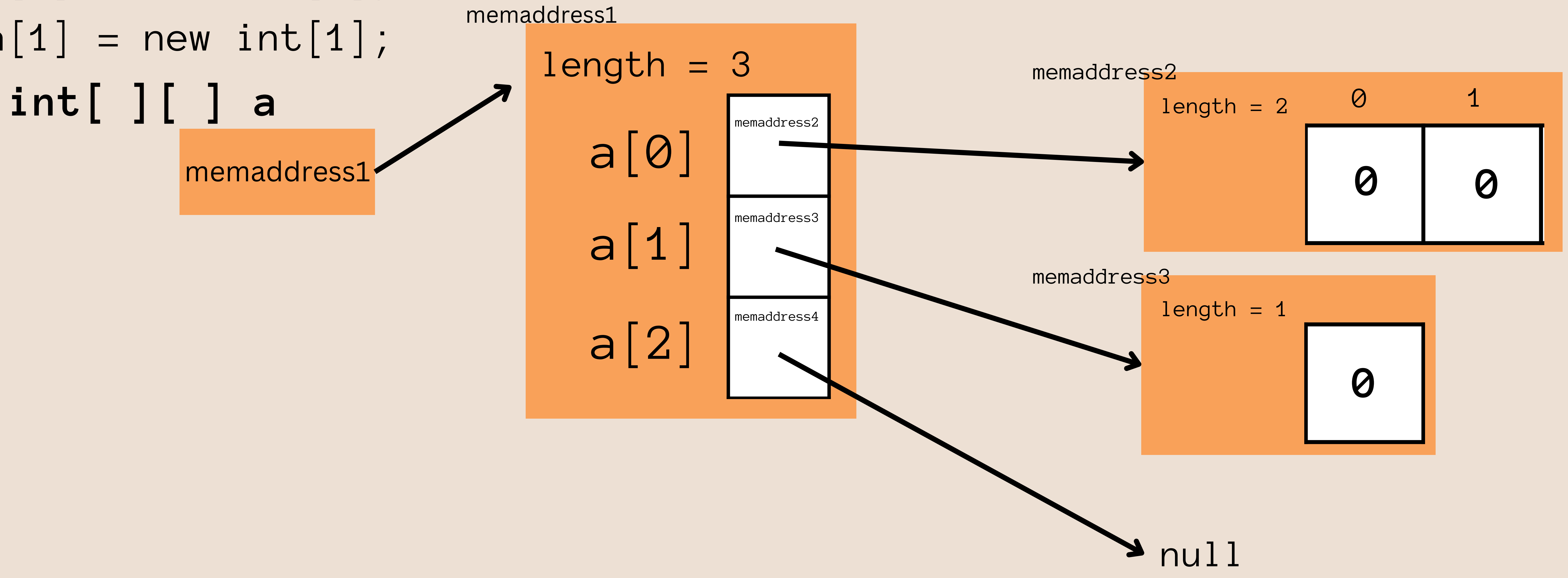


```
int[][][][][] a = new int[3][5][][][]; //this works too
```

Ragged Arrays

- You can create arrays containing smaller arrays of assorted sizes:
  - these are known as **ragged arrays (the ends are ragged)**

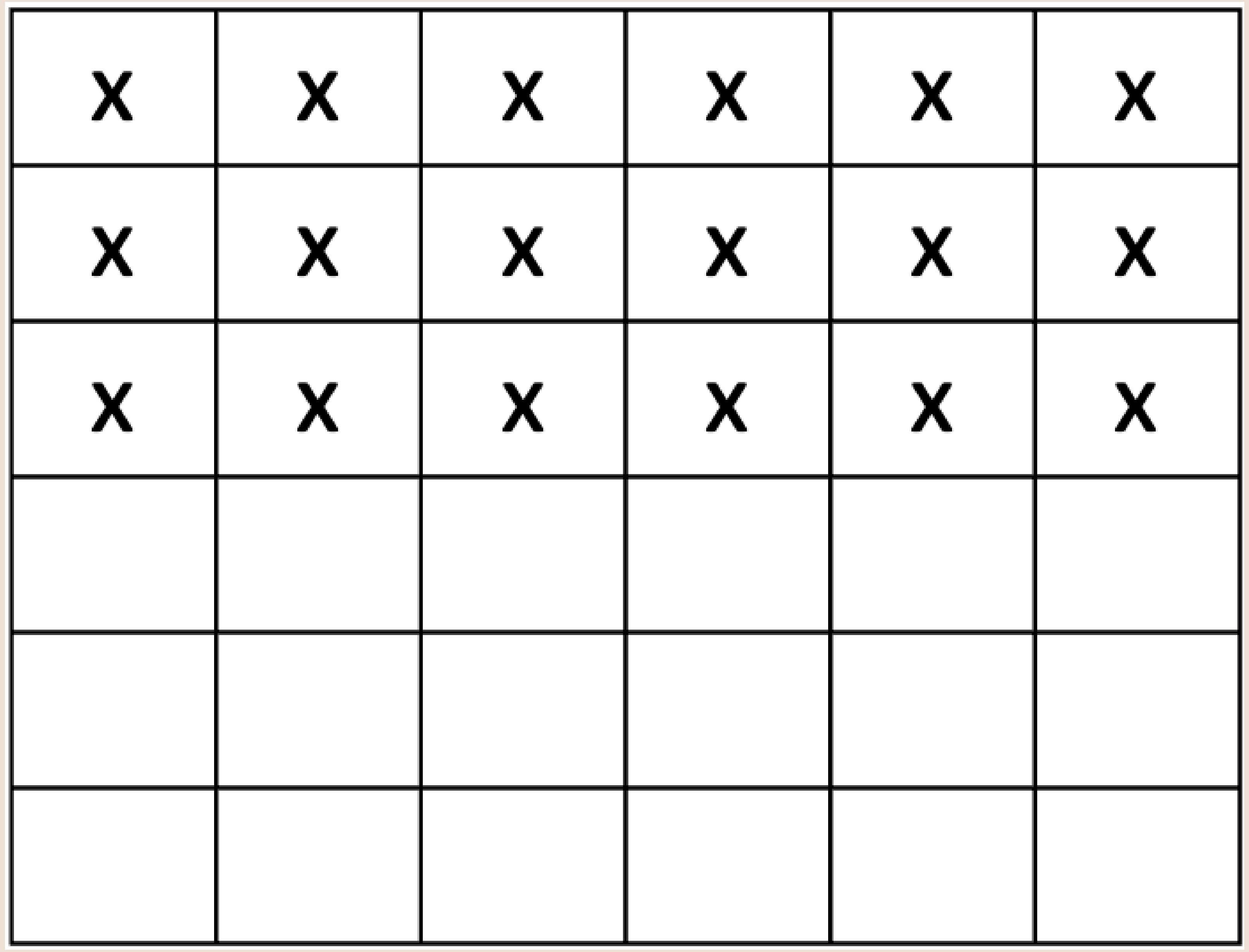
```
int[ ][ ] a = new int[3][ ];
a[0] = new int[2];
a[1] = new int[1];
```



Pause & Practice (With Me)

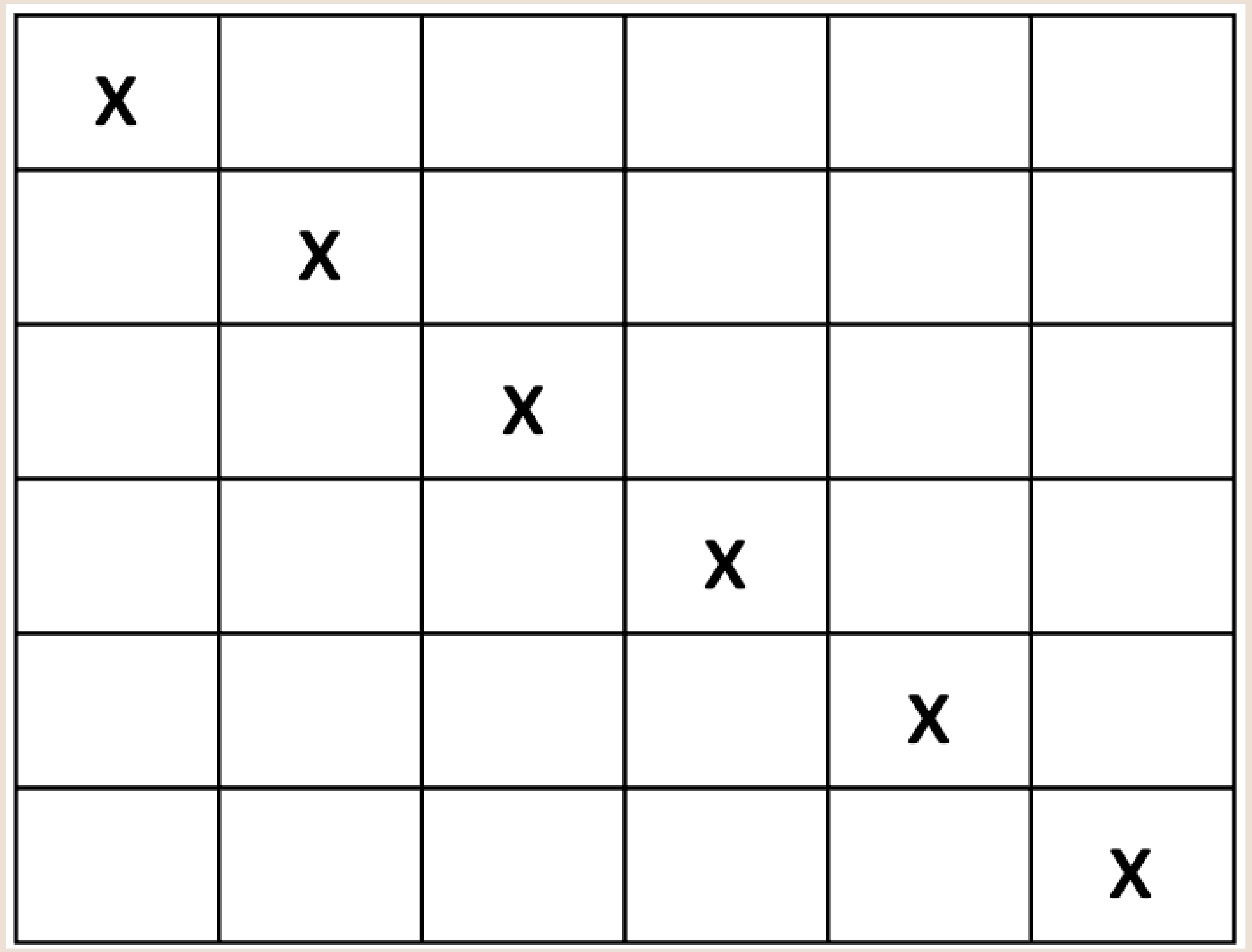
- Let’s build a CharMatrix object, which will contain a 2D matrix of chars as the instance variable
    - We can have a constructor that sets the size of the board and the symbol we will use to fill the spots with
    - We can also have a constructor that prints out the matrix (nicely)
    - Then we will built the following methods
        - fillTopHalf()
        - fillFrontDiagonal()
        - fillAboveDiagonal()
        - fillBelowDiagonal()
        - fillEvenRows()
        - fillEvenCols()
        - fillChessBoard()

fillTopHalf()



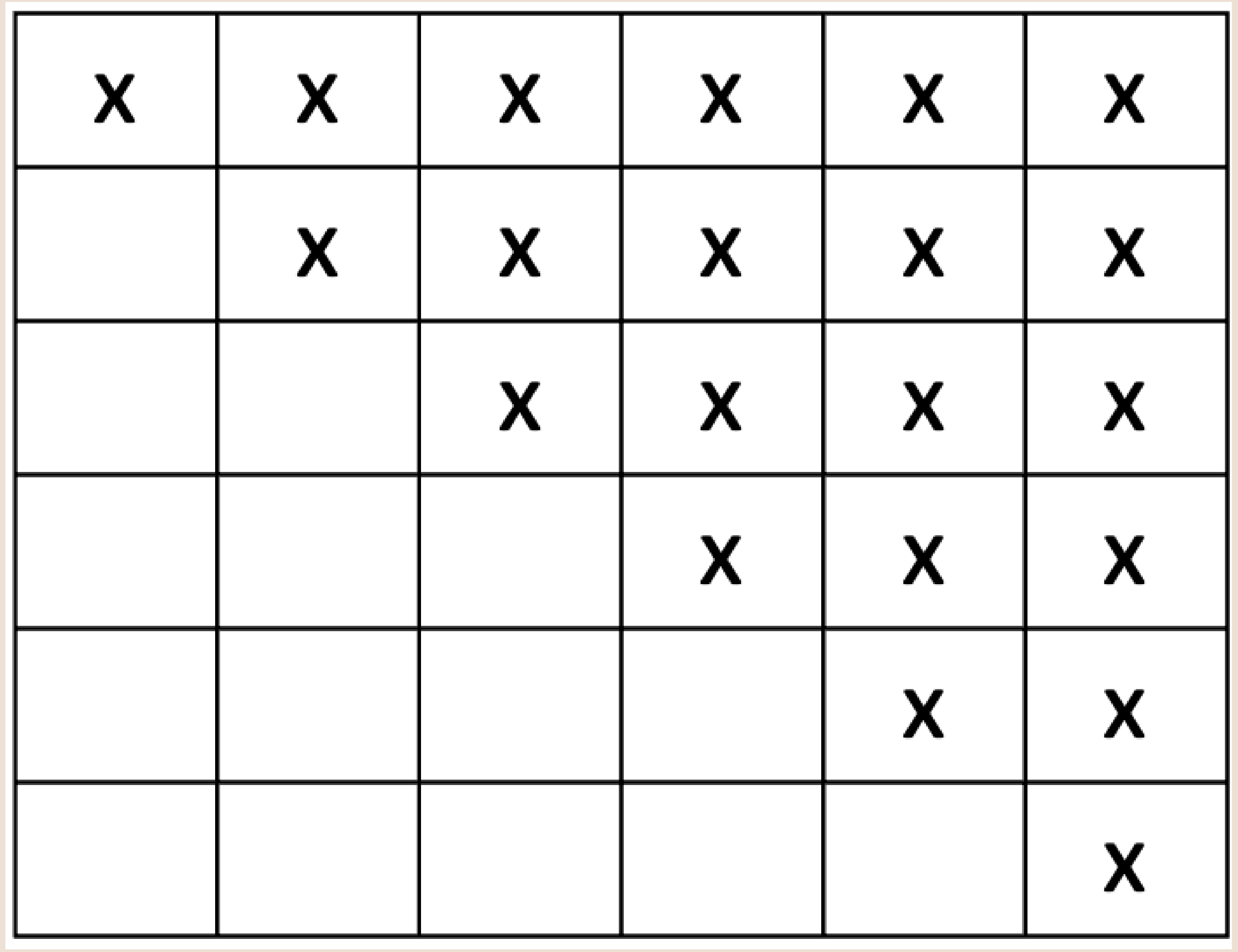
X	X	X	X	X	X
X	X	X	X	X	X
X	X	X	X	X	X

fillFrontDiagonal()



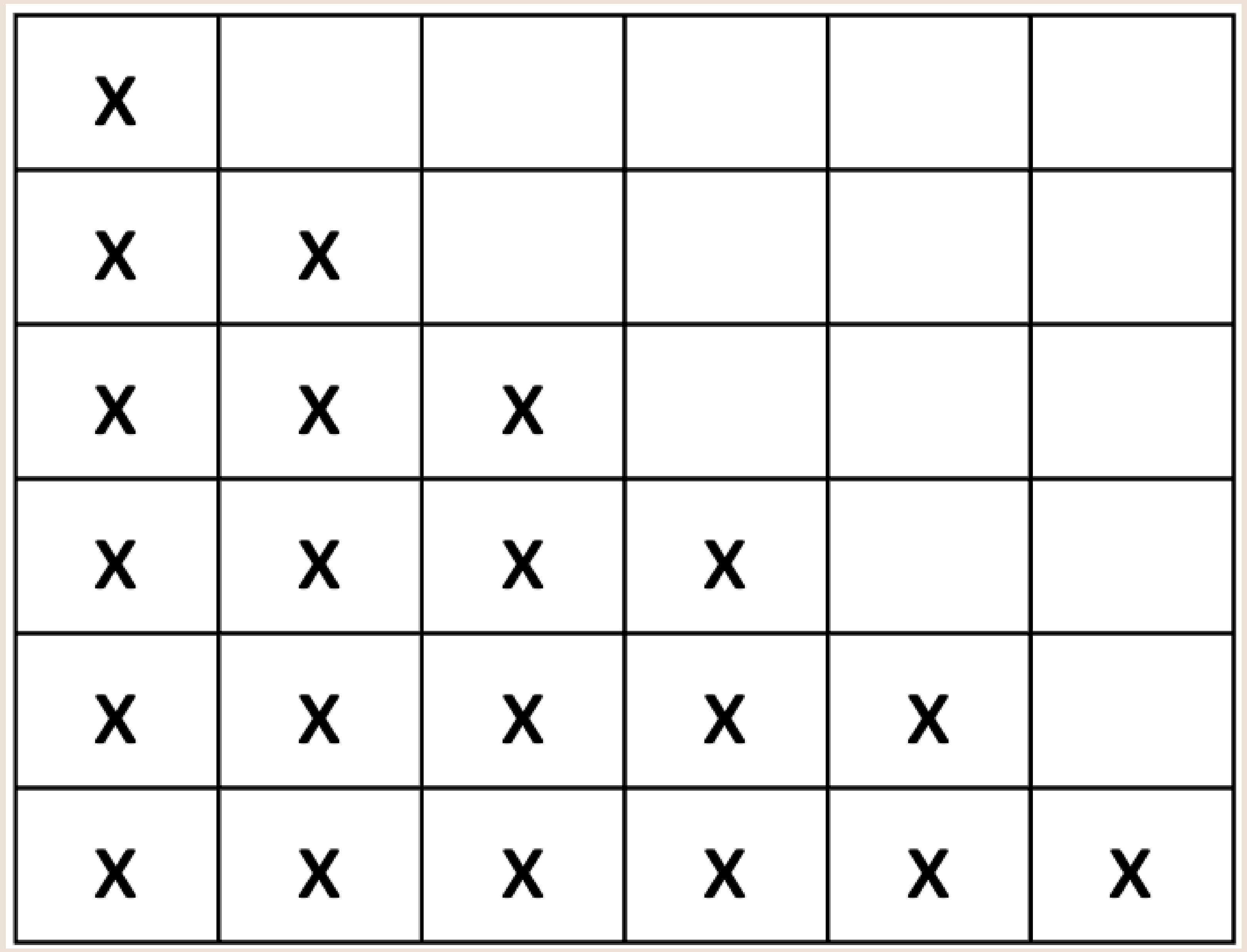
X					
	X				
		X			
			X		
				X	
					X

fillAboveDiagonal()



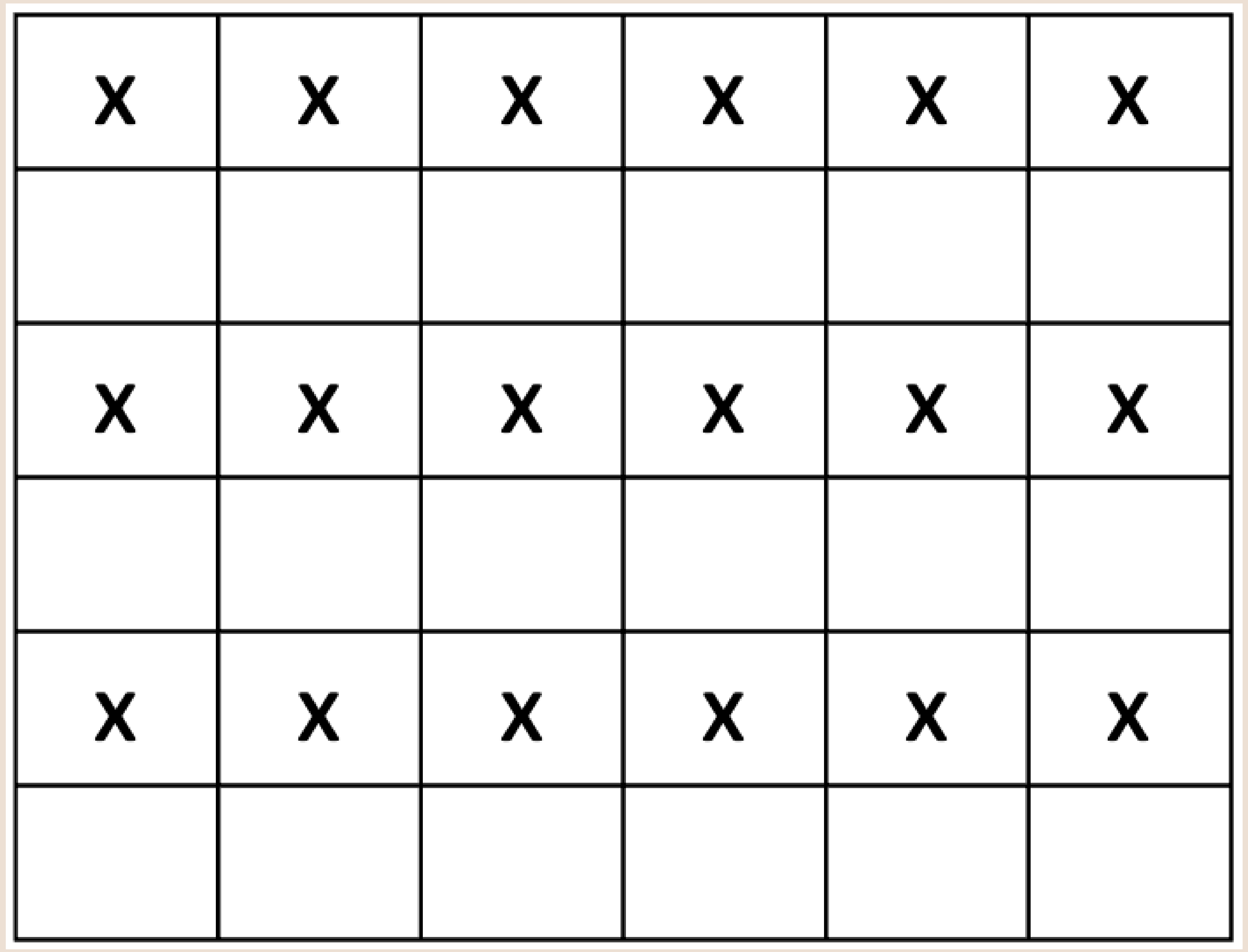
X	X	X	X	X	X
	X	X	X	X	X
		X	X	X	X
			X	X	X
				X	X
					X

fillBelowDiagonal()



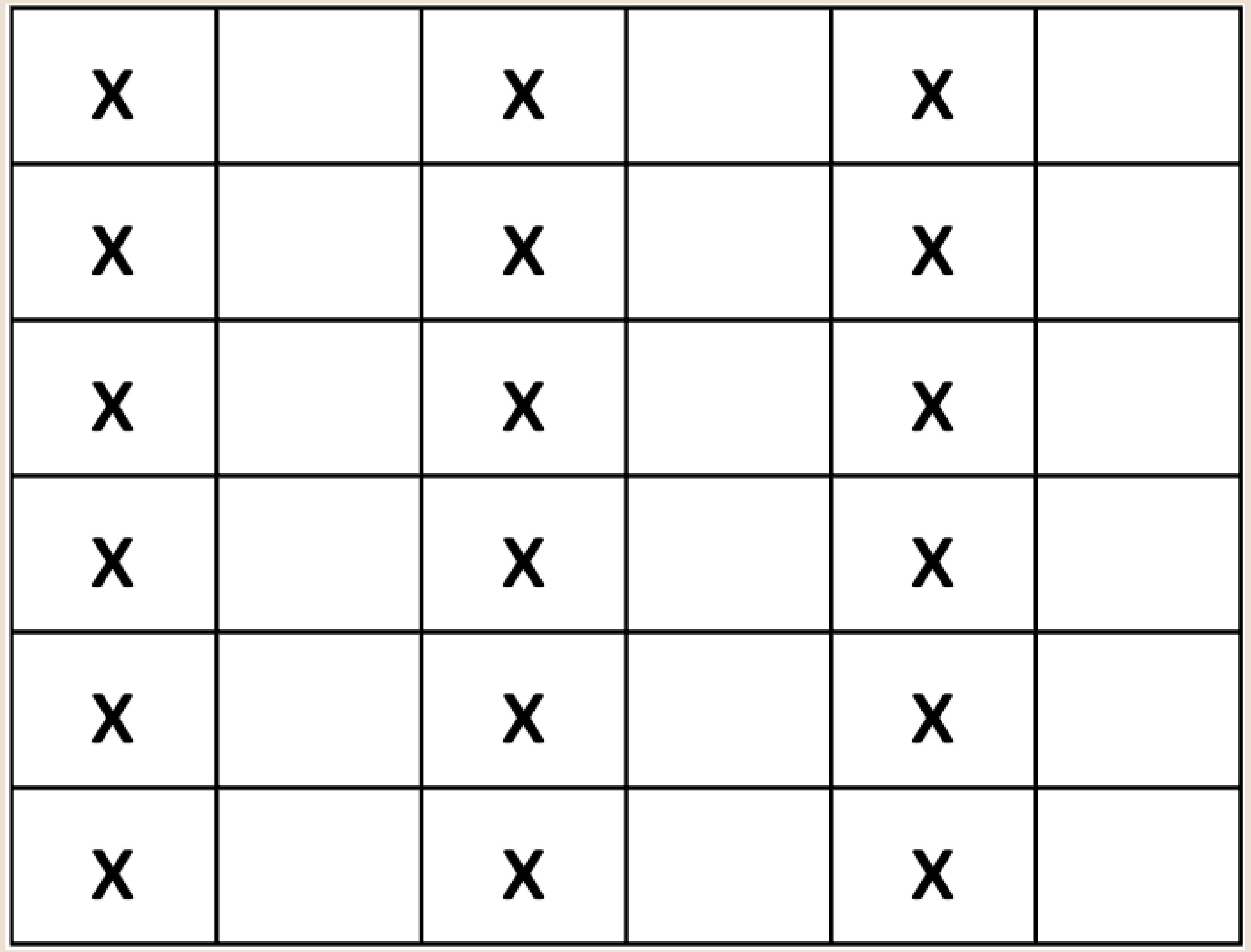
X					
X	X				
X	X	X			
X	X	X	X		
X	X	X	X	X	
X	X	X	X	X	X

fillEvenRows()



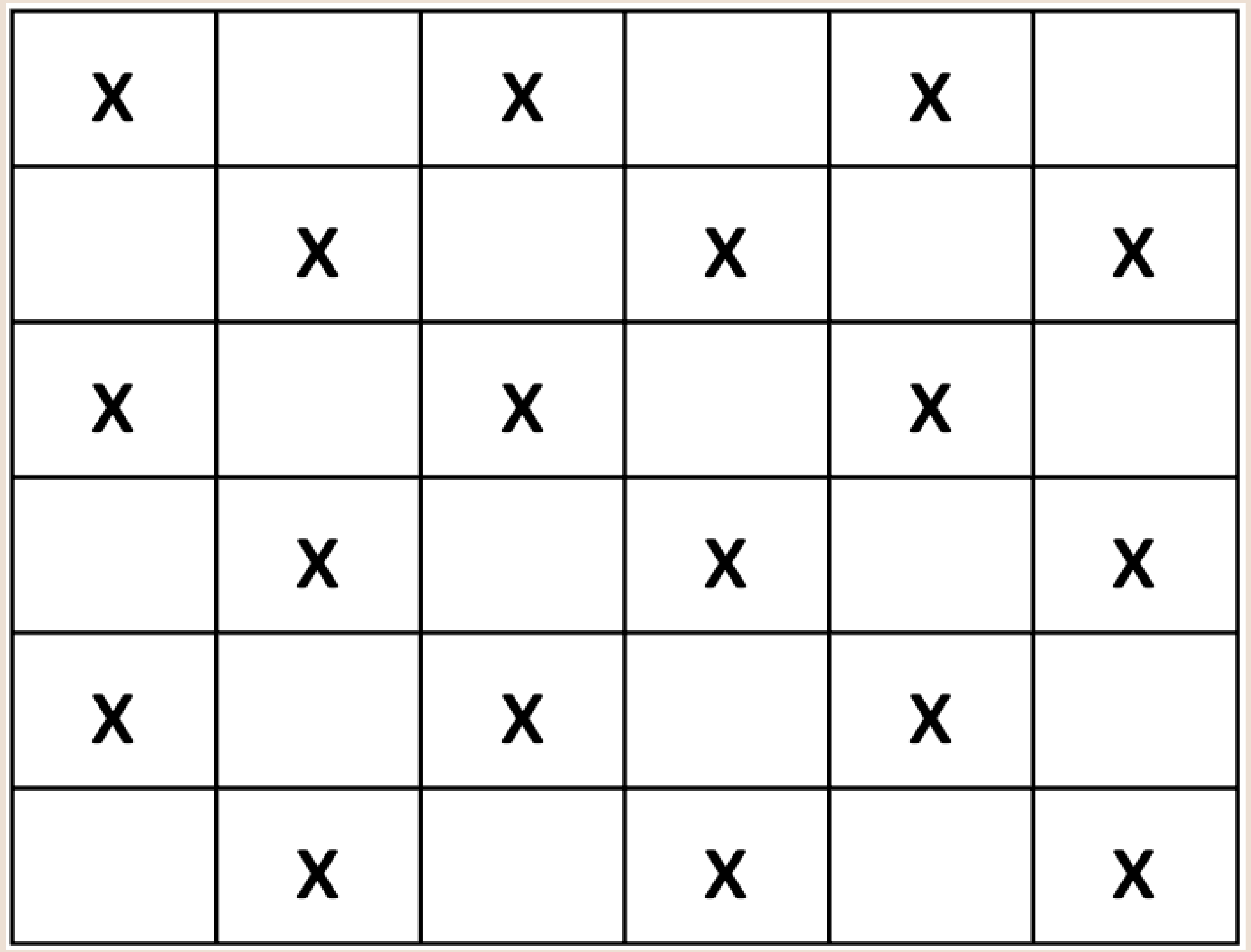
X	X	X	X	X	X
X	X	X	X	X	X
X	X	X	X	X	X

fillEvenCols()



X		X		X	
X		X		X	
X		X		X	
X		X		X	
X		X		X	
X		X		X	

fillChessBoard()



X		X		X	
	X		X		X
X		X		X	
	X		X		X
X		X		X	
	X		X		X

Pause & Practice

- Consider practicing your 2D array manipulation with creating a TIC-TAC-TOE game or a Sudoku game validator.
- The next video will touch on Reading in from a file into a multi-dimensional array (specifically a 3D array example)