CSSE 220

2D Arrays and Maps

Check out 2DArraysAndMapsInClass from Git Clone

Today’s Agenda

- Academic Honesty
- Quizzes
- Coding Gotchas
- Enhanced for loops
- 2D Arrays
- Maps

An aside: academic honesty in CS

- Please do not collaborate on homework assignments beyond what is allowed

Definitely not OK

- Looking at someone else’s solution, “just for a reference” as you write your own code
- Pair programming on an individual assignment
- Sitting next to someone as you both work on the assignment, working through any problems you have as a group

How much help is too much help?

“So you loop across the array elements, getting each element and seeing if it’s in the hashtable. If it is, you get the value and increment it…”

Borderline
but OK

“So you write a for loop, 1 to array length. Your key variable is gonna be array[i]. You check if hashMap.get(key) is null, if not, you get the value with get, then hashMap.put(key, oldValue + 1)…”

Giving away the an
swer. Cheating.

Penalties – they are severe

- Automatic F in the course
- Drop 1 letter grade
- -100% score on assignment

Yes, you can get an automatic F for cheating on one assignment one time.

You should always credit anyone you get help from on an assignment. If you do, it lets me assign one of the weaker penalties.

Coding Gotchas

```
int[] numbers = { 2, 4, 8, 16};
int numbersCount = numbers.length;

ArrayList<String> words = new ArrayList<String>();
words.add( “Hello!”);
int wordsCount = words.size();

String word = “Hello”;
int characterCount = word.length();
```

Enhanced For Loops

- Convenient Syntax
- for iterating through collections
- More readable
- Less Typing
- Less Error Prone
- Works for Arrays, ArrayList, Map (later)
- Similar to Python:

```
    for num in nums:
```

Enhanced For Loop and Arrays

- Old school

```
double[] scores = …
double sum = 0.0;
for (int i=0; i < scores.length; i++) {
    sum += scores[i];
}
```

- New, whiz-bang, enhanced for loop

```
double[] scores = …
double sum = 0.0;
for (double score : scores) {
    sum += score;
}
```

Say “in”

- No index variable **(easy, but limited in 2 respects)**
- Gives a name (**score** here) to each element

Enhanced For and ArrayList’s

```
ArrayList<State> states = …
int total = 0;
for (State state : states) {
    total += state.getElectoralVotes();
}
```

2D Arrays – What, When, Why, How?

What:

- Think of them as an array of arrays
- … or as a grid with rows & columns

When:

- – Represent 2 dimensional data
  - Game Boards
  - Tables
  - Multiple lists of items
  - Etc.

2D Arrays – What, When, Why, How?

Why:

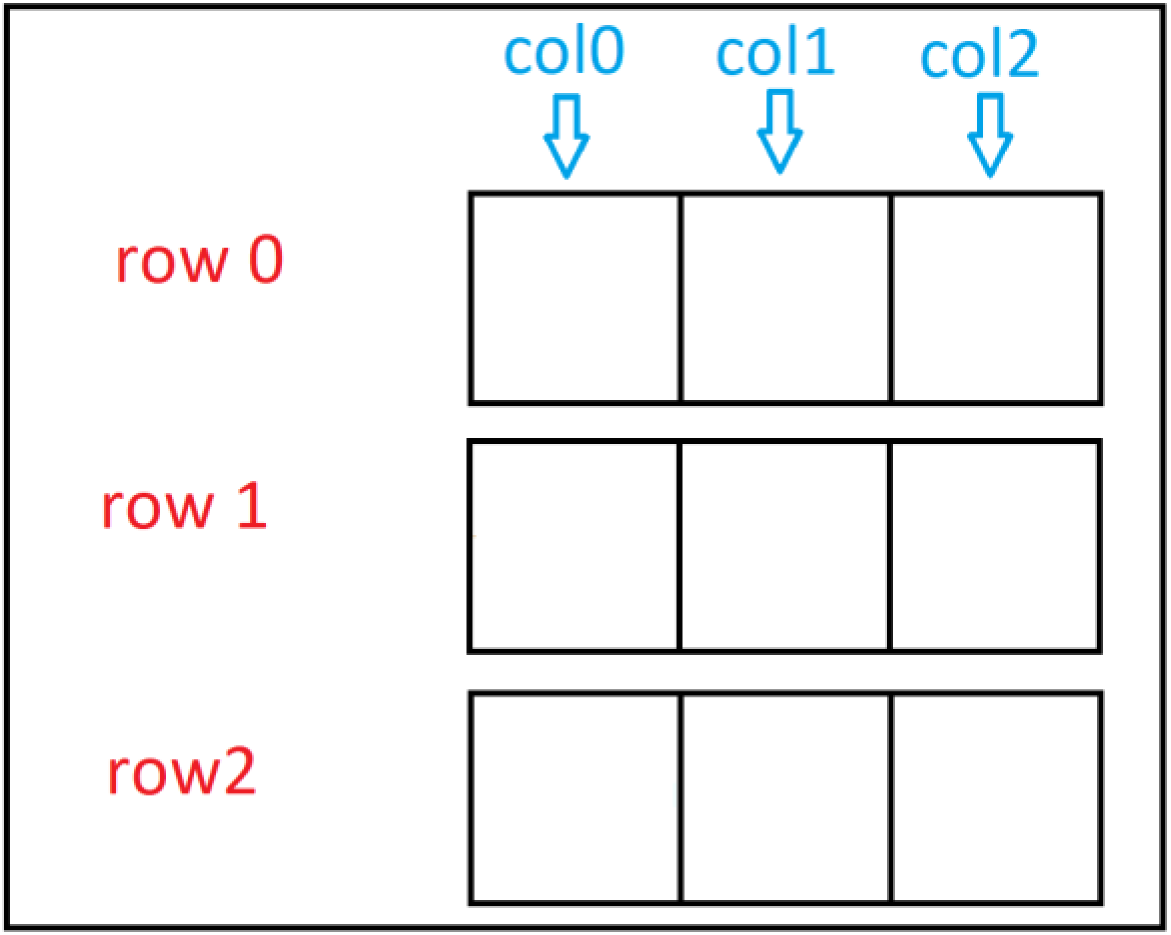
- Match your data representation as closely as possible to the real-world

How:

- char[][] ticTacToe = new char[3][3];
- Retrieving data
  - ticTacToe[0] → Gets the first char[]
  - ticTacToe[1][2] →Gets the second array's third item

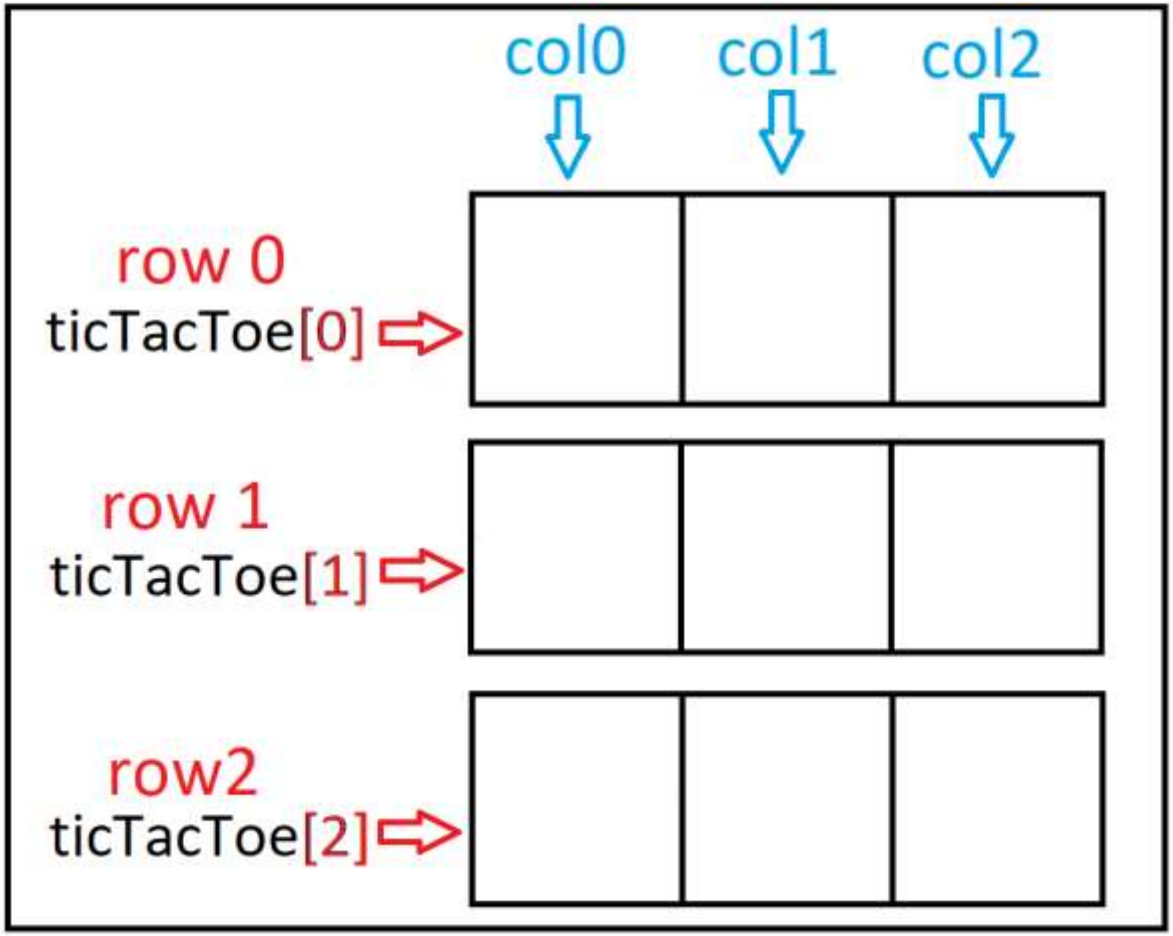
Creating a 2D char array

```
char[][] ticTacToe = new char[3][3];
```



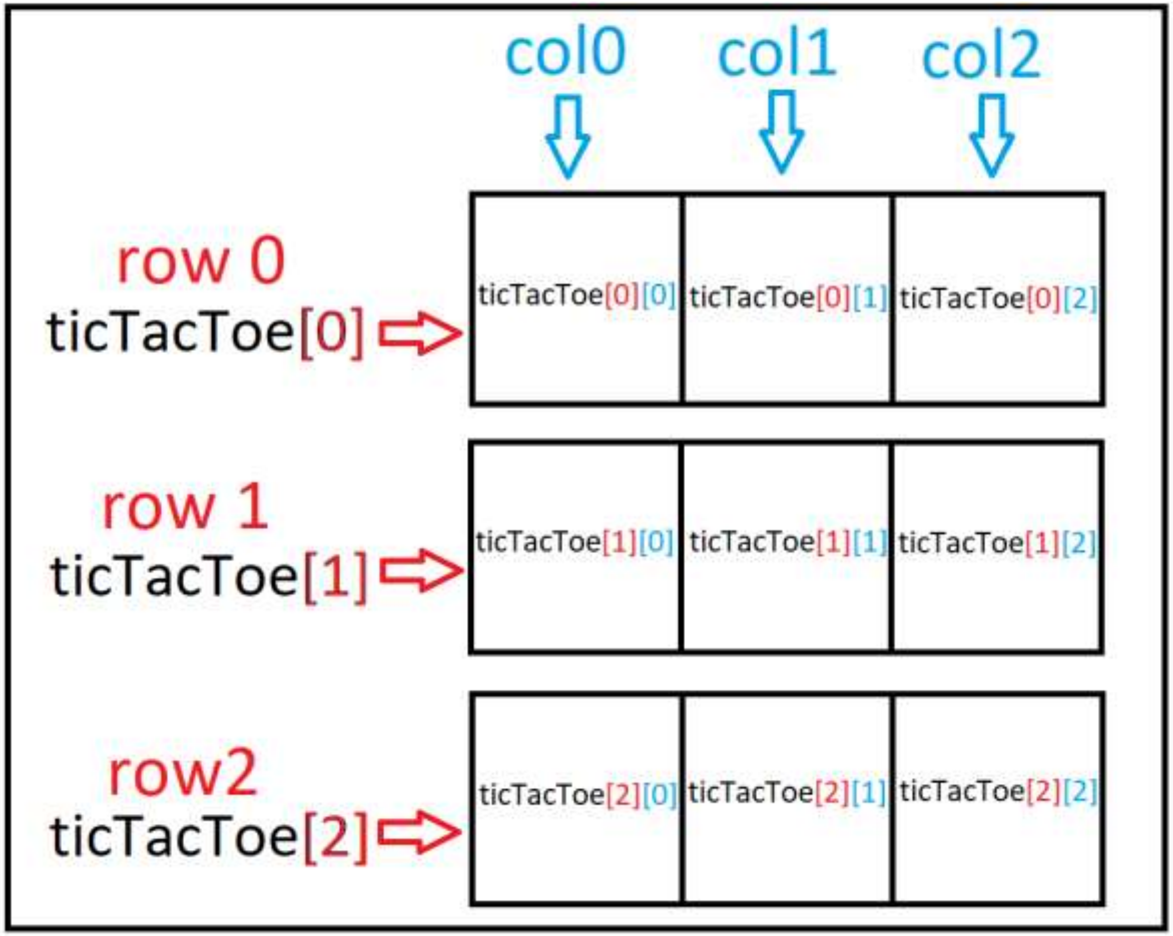
Creating a 2D char array

```
char[][] ticTacToe = new char[3][3];
```



Creating a 2D char array

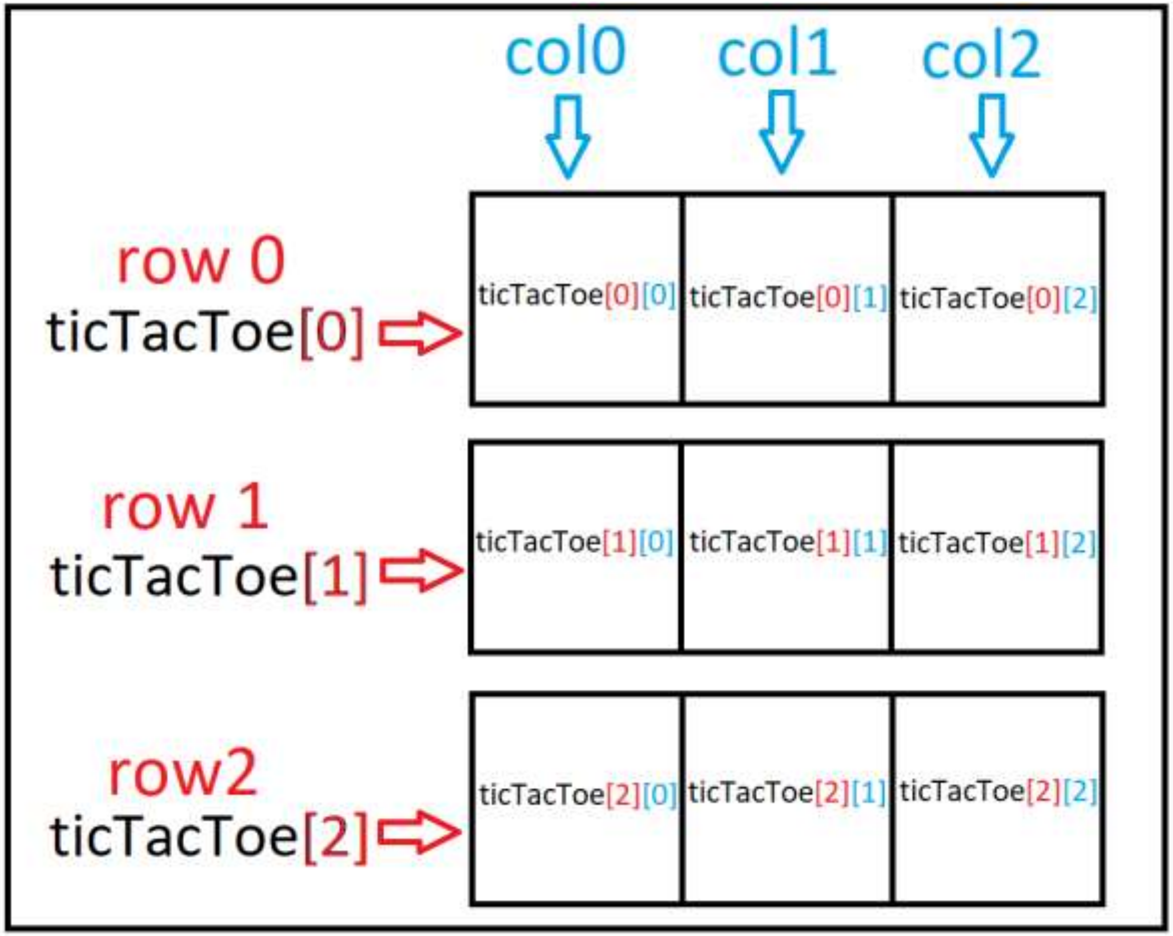
```
char[][] ticTacToe = new char[3][3];
```



	col0	col1	col2
row 0 ticTacToe[0]	ticTacToe[0][0]	ticTacToe[0][1]	ticTacToe[0][2]
row 1 ticTacToe[1]	ticTacToe[1][0]	ticTacToe[1][1]	ticTacToe[1][2]
row2 ticTacToe[2]	ticTacToe[2][0]	ticTacToe[2][1]	ticTacToe[2][2]

Creating a 2D char array

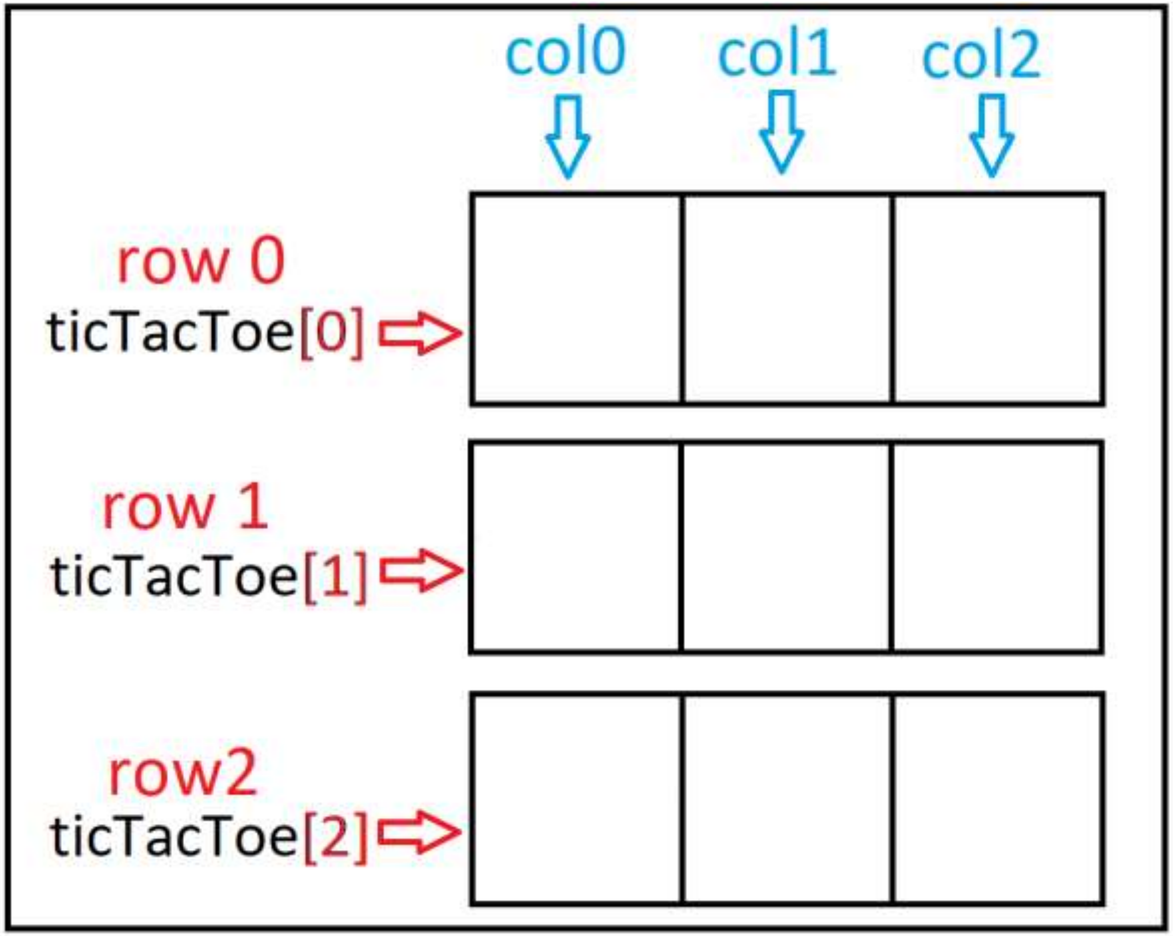
ticTacToe[row] [column] -> refers to individual cell



	col0	col1	col2
row 0 ticTacToe[0]	ticTacToe[0][0]	ticTacToe[0][1]	ticTacToe[0][2]
row 1 ticTacToe[1]	ticTacToe[1][0]	ticTacToe[1][1]	ticTacToe[1][2]
row2 ticTacToe[2]	ticTacToe[2][0]	ticTacToe[2][1]	ticTacToe[2][2]

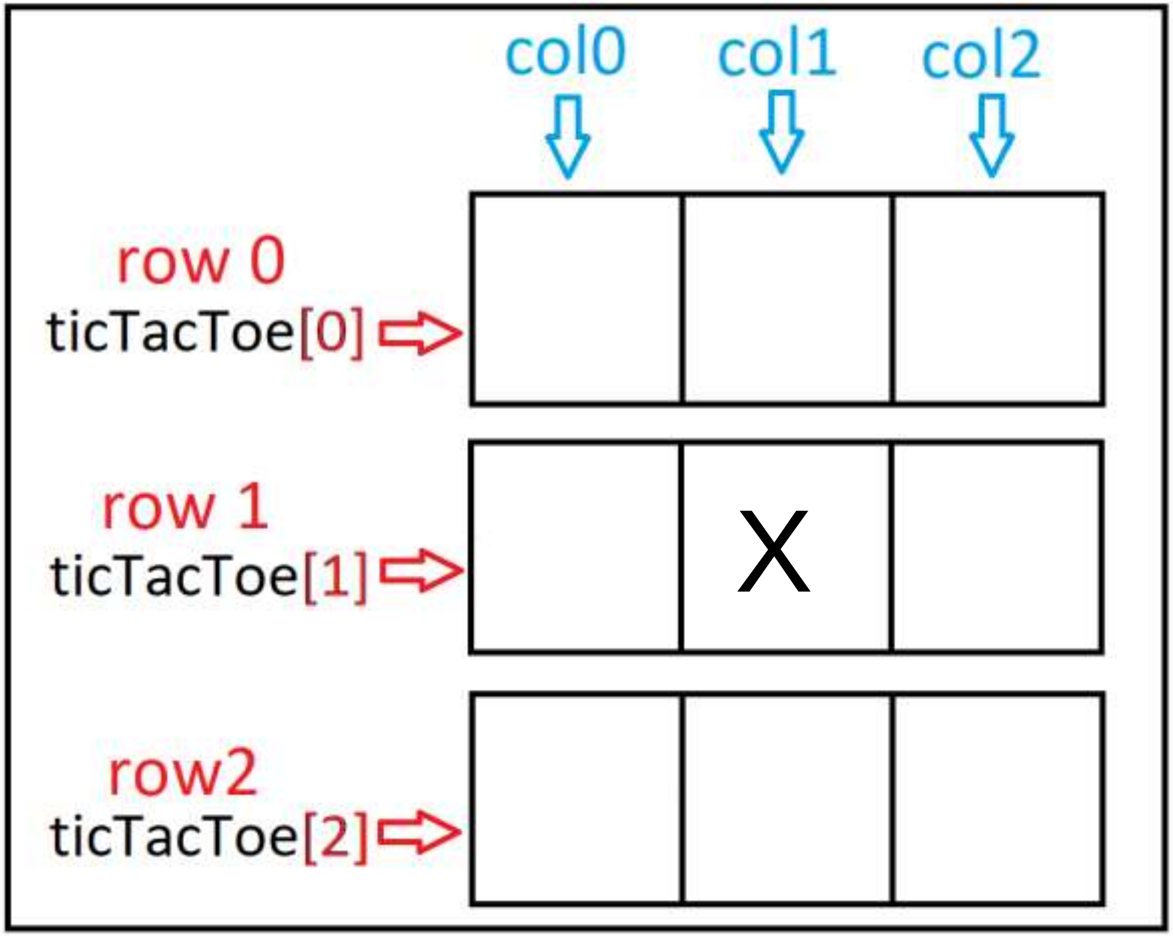
Creating a 2D char array

```
ticTacToe[ 1 ][  1 ] = ‘X’;
```



Creating a 2D char array

```
ticTacToe[ 1 ][  1 ] = ‘X’;
```



2D array dimensions

```
int[][] numArray = new int[2][4];
int rows = numArray.length;
int cols = numArray[0].length;
```

2D array dimensions

2 rows
4 columns

numArray[row] [column]

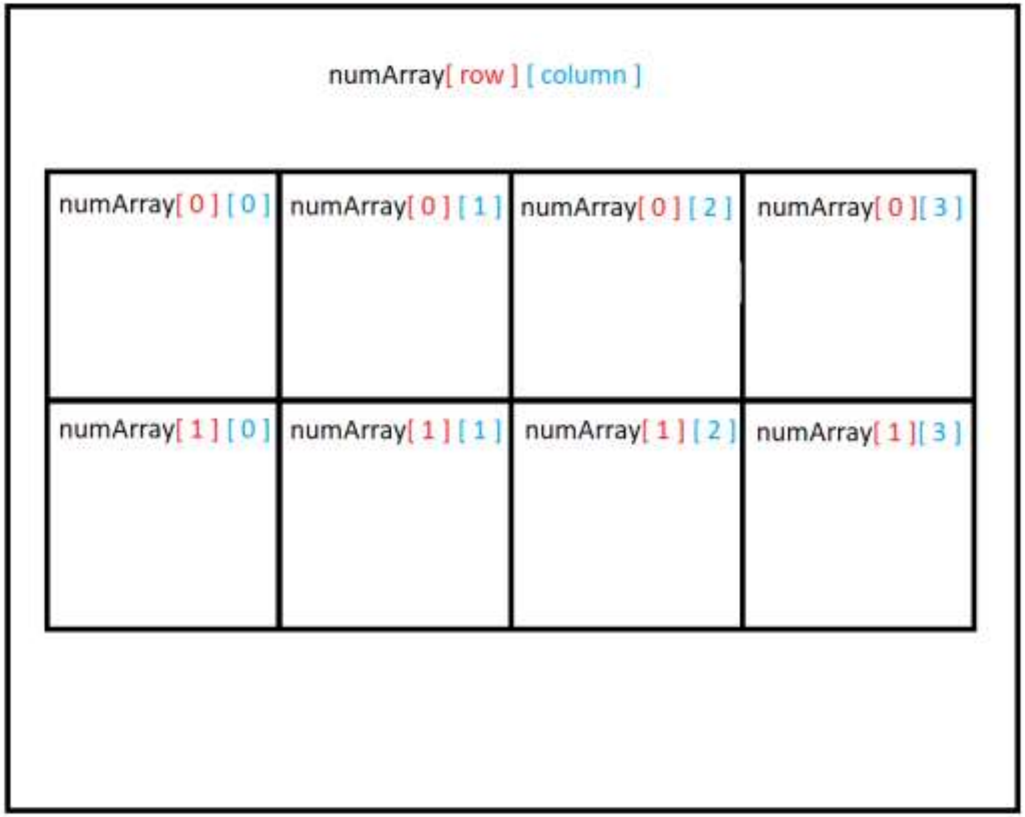
numArray[0] [0]	numArray[0] [1]	numArray[0] [2]	numArray[0][3]
numArray[1] [0]	numArray[1] [1]	numArray[1] [2]	numArray[1][3]

Iterate through 2D array?

```
int[][] numArray = new int[2][4];
int rows = numArray.length;
int cols = numArray[0].length;
int count = 0;
for (int r=0; r < rows; r ++ ) {
        for (int c=0; c< cols; c++) {
                ticTacToe[r][c] = count;
                count ++;
        }
}
```

Order of iteration?

```
int[][] numArray = new int[2][4];
int rows = numArray.length;
int cols = numArray[0].length;
int count = 0;
for (int r=0; r < rows; r ++ ) {
        for (int c=0; c< cols; c++) {
                ticTacToe[r][c] = count;
                count ++;
        }
}
```

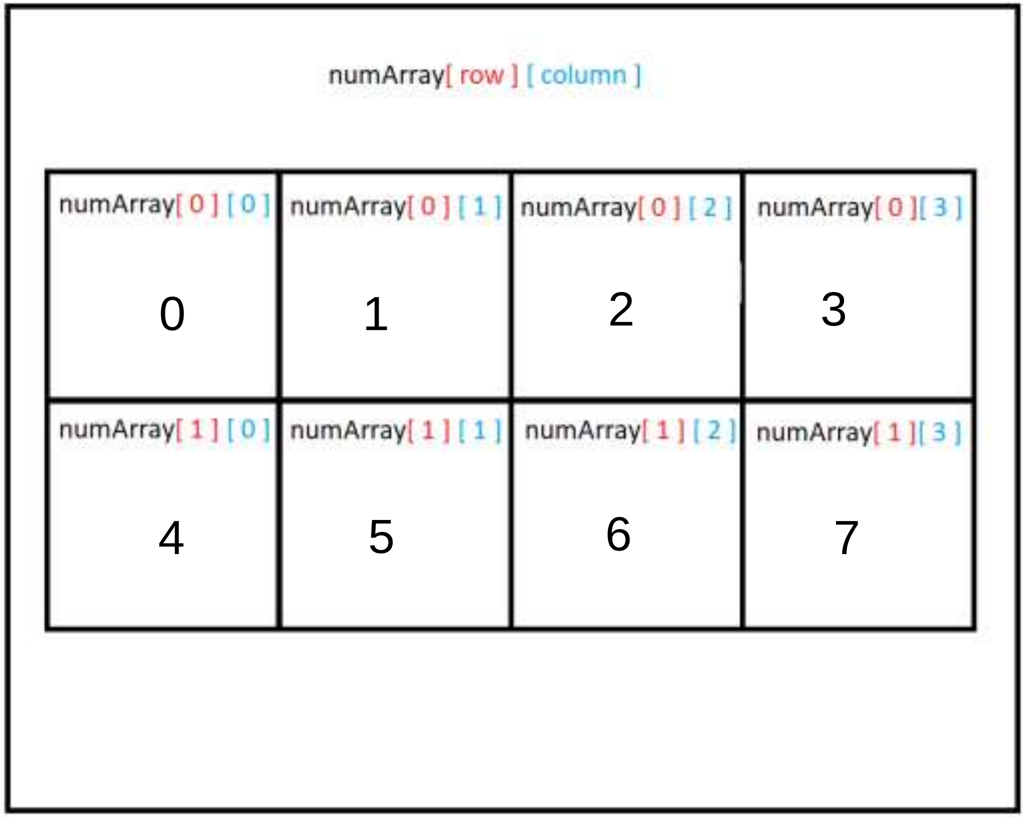


numArray[row] [column]

numArray[0] [0]	numArray[0] [1]	numArray[0] [2]	numArray[0][3]
numArray[1] [0]	numArray[1] [1]	numArray[1] [2]	numArray[1][3]

Order of iteration?

```
int[][] numArray = new int[2][4];
int rows = numArray.length;
int cols = numArray[0].length;
int count = 0;
for (int r=0; r < rows; r ++ ) {
        for (int c=0; c< cols; c++) {
                ticTacToe[r][c] = count;
                count ++;
        }
}
```



numArray[row] [column]

numArray[0] [0] 0	numArray[0] [1] 1	numArray[0] [2] 2	numArray[0][3] 3
numArray[1] [0] 4	numArray[1] [1] 5	numArray[1] [2] 6	numArray[1][3] 7

2D Arrays

- Make groups of two (no more than 3, no one can work alone)
- Read through the 3 2D Array sample problems with your partner and make sure you both understand how they work
- Then use the code as an example to answer the 2D Array quiz questions
- Then do the 2d sample problems
- Call me over when you’re finished

Maps – What, When, Why, How?

What:

- Collection of key-value pairs
  - Key is the identifier
    - i.e. A word in a dictionary, or a student ID number, something that uniquely identifies an item
  - Value is the data for that identifier
    - i.e. The definition of a word in a dictionary, a Student object for an ID, the value associated with an unique ID
- Think of this like a dictionary (in some programming languages they're even called dictionaries)
  - Key: word
  - Value: definition

Maps – What, When, Why, How?

When:

- We use maps when a unique piece of data is used to retrieve additional information

Why:

- Fast access to information based on a unique key

How:

```
HashMap<String, Student> usernameToStudent =
new HashMap<String, Student>();
```

Maps

- Sometimes, it’s difficult to fully understand maps.
- Let’s do an example together:
  - Implement an int array using a map.

Maps

- Make groups of two (no more than 3, no one can work alone)
- Read through the 3 Map sample problems with your partner and make sure you both understand how they work
- Then use the code as an example to answer the Map quiz questions
- Then solve the map problems in today’s code
- Call me over when you’re finished