+

Multidimensional Dimensional Arrays

Introduction to Computer Science CSCI UA.0101

Lecture10

+

Agenda Day 10

- Two-dimensional arrays basics
- Processing two-dimensional arrays
- Multidimensional arrays
- Examples

+

Which is the output?

```
public class PrePostInc
{
       public static void main (String [] args)
       {
              int i = 0;
              System.out.println(i++);
              i = 0;
              System.out.println(++i);
              i = 0;
              System.out.println(i--);
              i = 0;
              System.out.println(--i);
       }
}
```

Which is the output? Solution

```
public class PrePostInc
{
        public static void main (String [] args)
        {
                int i = 0;
                System.out.println(i++);
                i = 0;
                System.out.println(++i);
                i = 0;
                System.out.println(i--);
                i = 0;
                System.out.println(--i);
        }
}
Output: 0
        1
        0
       -1
```

+

Which is the output?

```
public class MoreAssignment
{
   public static void main(String[] args)
   {
      int a, b, c;
      a = b = c = 1;
      System.out.println("a=" + a + “ b=" + b + “ c=" + c);
                                              //a, b, c?
      a = (b = 2) * 2;
      System.out.println("a=" + a);           //a?

      a = (b = c) * (a = b);
      System.out.println("a=" + a);           //a?
   }
}
```

+ Which is the output? Solution

```
public class MoreAssignment
{
   public static void main(String[] args)
   {
      int a, b, c;
      a = b = c = 1;
      System.out.println("a=" + a + “ b=" + b + “ c=" + c);
                                        //a=1, b=1, c=1
      a = (b = 2) * 2;
      System.out.println("a=" + a);     //a=4, b=2, c=1

      a = (b = c) * (a = b);
      System.out.println("a=" + a);     //a=1
   }
}
```

+

The Enhanced For Loop

The Enhanced For Loop

- The “enhanced for loop” allows you to iterate through an array or collection without having to create an iterator or without having to calculate beginning and end conditions for a counter.

```
double [] numbers = {1, 2, 3, 4, 5};

for (double n: numbers)
{
        System.out.println(n);
}

//is equivalent to

for(int i = 0; i < numbers.length; i++)
{
        System.out.println(numbers[i]);
}
```

+

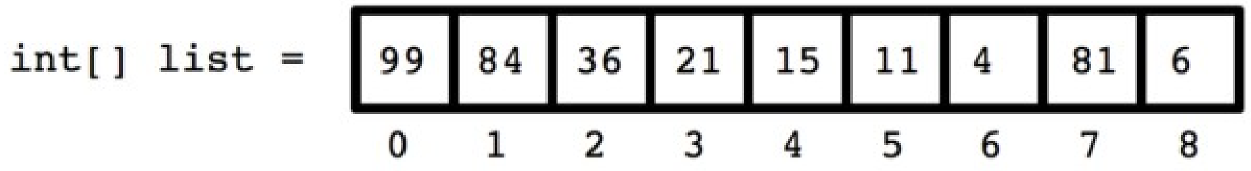
Programming Challenge (ForEachDemo.java)

+

Multidimensional Arrays

Multidimensional Arrays

- So far we have studied how to store linear collections of data using a single dimensional array.
- However, sometimes we need to store more complex structures, such as matrices and tables. We can do this using a multidimensional array.



+
Multidimensional Arrays

	Chicago	Boston	New York
Chicago	0	983	787
Boston	983	0	214
New York	787	214	0

```
int[][] distances = {
                       {0, 983, 787},
                       {983, 0, 214},
                       {787, 214, 0}
                     };
```

Two Dimensional Array Basics

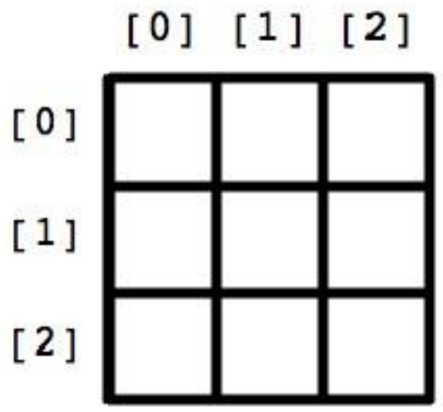
- You can declare a two dimensions array using almost the same syntax you would use to declare a single dimensional array. For example:

```
int[][] myList = new int[5][5]; //declaration and
                                //creation
```

- This would create a 5 x 5 matrix of integers, all defaulted to a zero value upon creation. We generally think of the first number as the number of "rows" in your array and the second as the number of "columns"

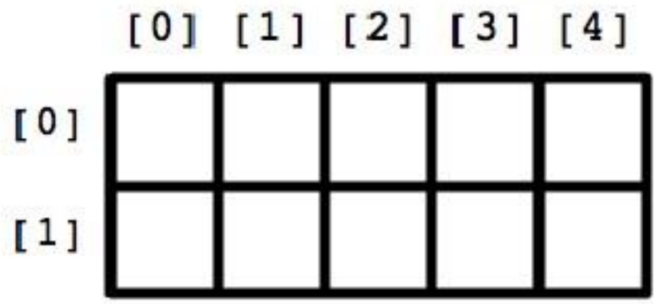
+ Two Dimensional Array Basics

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



Two Dimensional Array Basics

```
int[][] a = new int[2][5]
```



Two Dimensional Array Basics

- You can access the two dimensional array values using the same syntax as you would with a single dimensional array. For example, the first element in the first row and first column is at position:

```
myList[0][0]
```

- Which would the element on the third row and second column?

```
myList[?][?]
```

+
Two Dimensional Array Basics

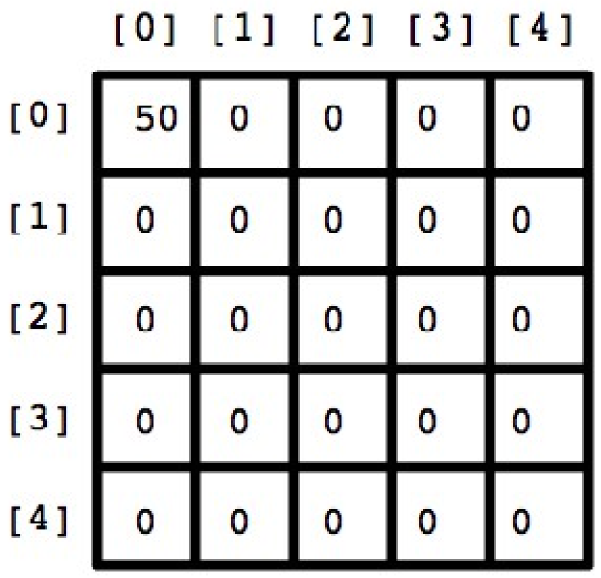
```
int[][] list = new int[5][5];

list[0][0] = 50;

list[4][4] = 99;

list[2][1] = 11;
```

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

+

Two Dimensional Array Basics

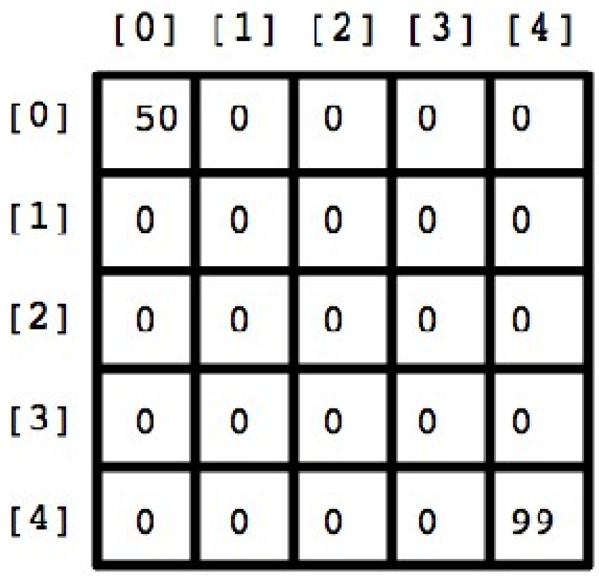
```
int[][] list = new int[5][5];

list[0][0] = 50;

list[4][4] = 99;

list[2][1] = 11;
```



	[0]	[1]	[2]	[3]	[4]
[0]	50	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

+ Two Dimensional Array Basics

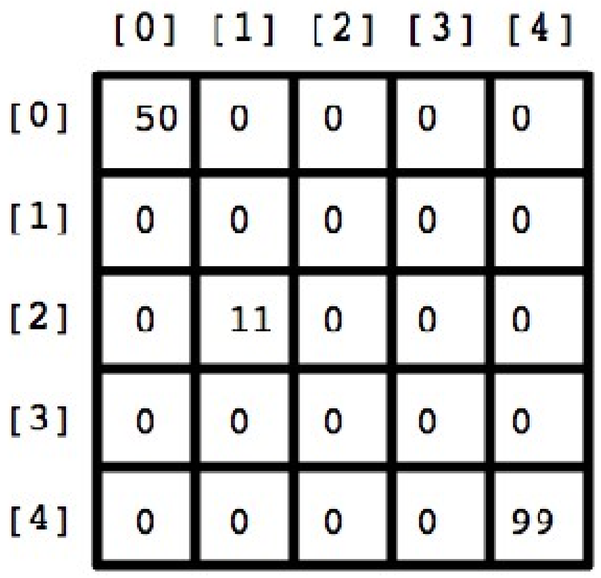
```
int[][] list = new int[5][5];

list[0][0] = 50;

list[4][4] = 99;

list[2][1] = 11;
```



	[0]	[1]	[2]	[3]	[4]
[0]	50	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	99

+
Two Dimensional Array Basics

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

list[0][0] = 50;

list[4][4] = 99;

list[2][1] = 11;
```



	[0]	[1]	[2]	[3]	[4]
[0]	50	0	0	0	0
[1]	0	0	0	0	0
[2]	0	11	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	99

Two Dimensional Array Basics

- You can create a two dimensional array with pre-specified values by using almost the same syntax as a single dimensional array. For example:

```
int[][] list = {
                  { 1, 2, 3 },
                  { 4, 5, 6 },
                  { 7, 8, 9 }
                };
```

- Which are the dimensions of the array? Rows? Columns?

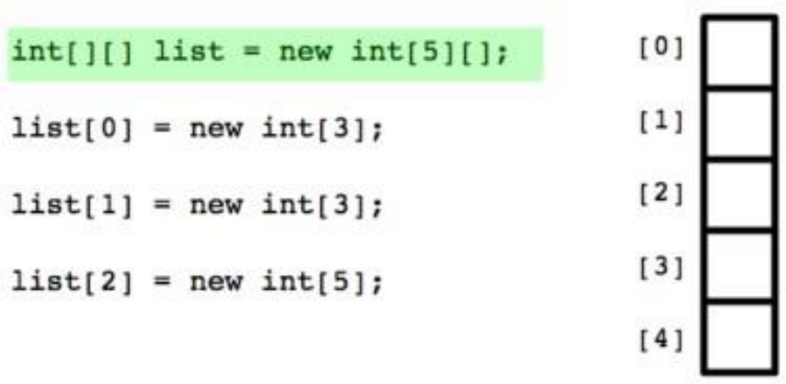
Getting the lengths of a two dimensional array

- Two dimensional arrays are really just one dimensional arrays that have been "chained" together. For example:

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

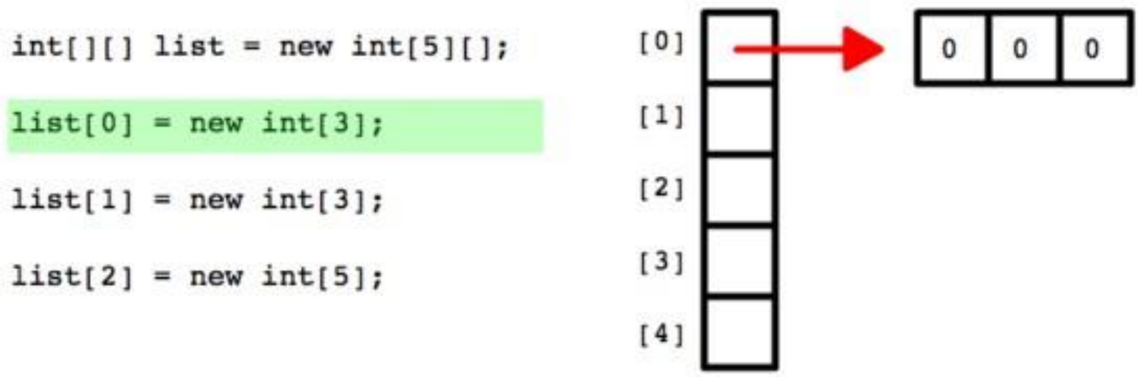
- Is a list of 5 elements. Each of those elements, however, references another single dimensional array, each of which is 5 elements long as well.

Getting the length of a two dimensional array



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

list[0] = new int[3];

list[1] = new int[3];

list[2] = new int[5];
```

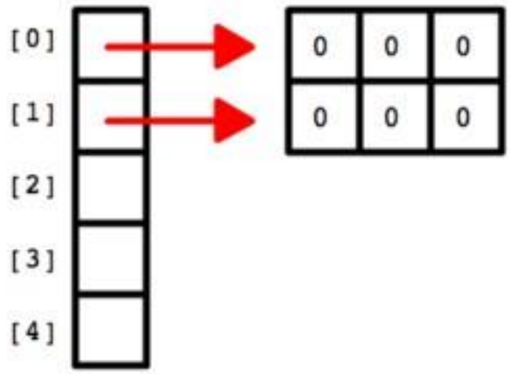
+
Getting the length of a two dimensional array



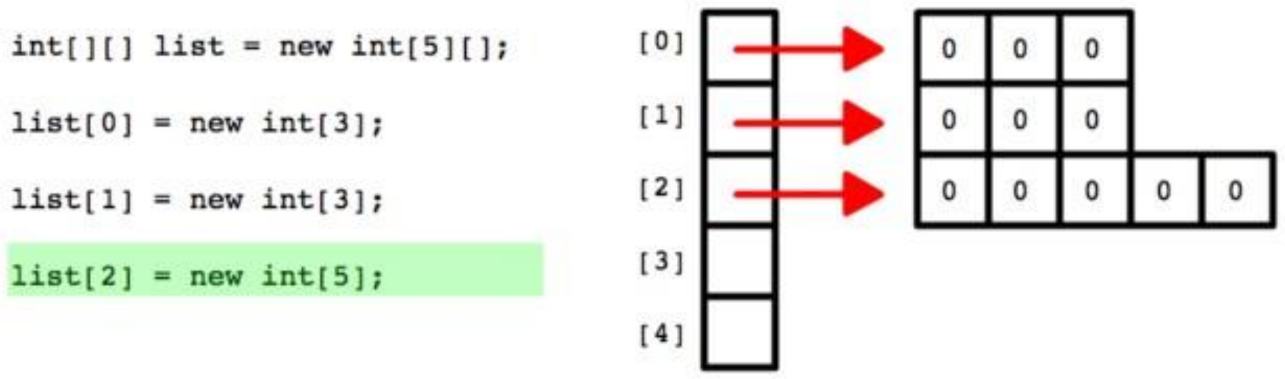
```
int[][] list = new int[5][];

list[0] = new int[3];

list[1] = new int[3];

list[2] = new int[5];
```

+
Getting the length of a two dimensional array

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

list[0] = new int[3];

list[1] = new int[3];

list[2] = new int[5];
```



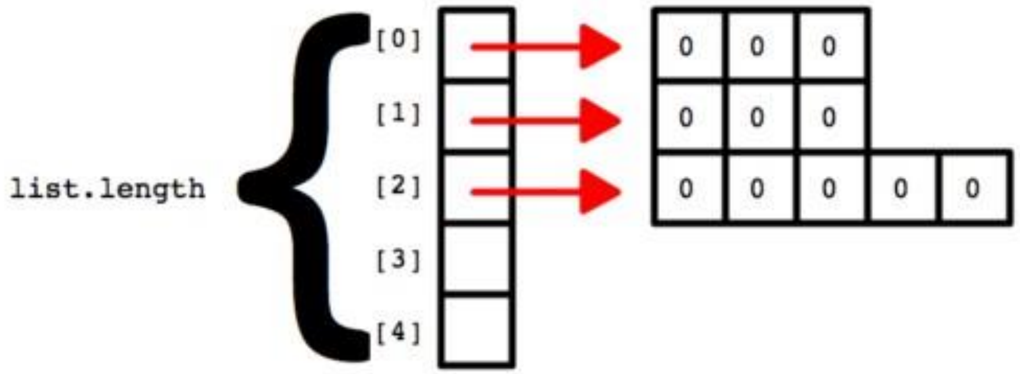
+ Getting the length of a two dimensional array



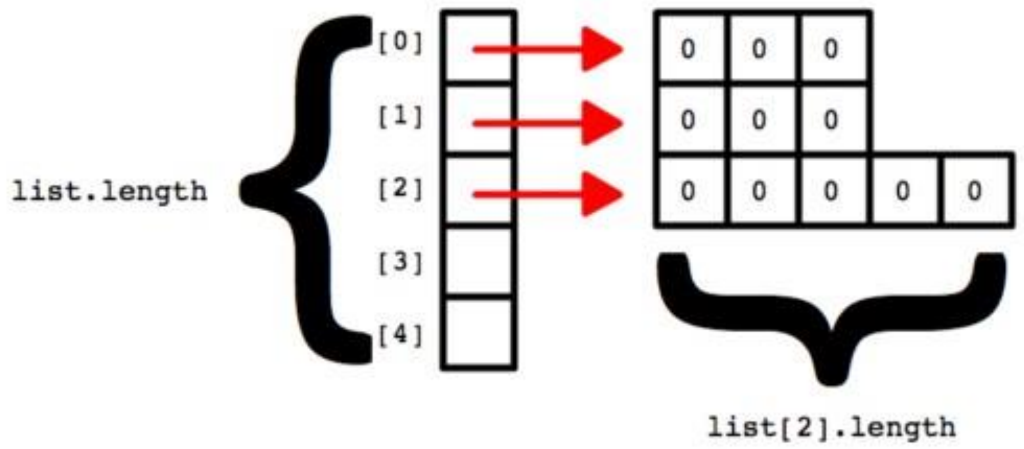
```
int[][] list = new int[5][];

list[0] = new int[3];

list[1] = new int[3];

list[2] = new int[5];
```

+

Getting the length of a two dimensional array



Getting the length of a two dimensional array



Getting the length of a two dimensional array

- The length of your “main” array in a 2 dimensional array can be obtained by referencing the following. We usually refer to this as the number of “rows” in the array

```
list.length
```

- The length of each sub-array in a two dimensional array can be obtained by referencing the following. We usually refer to this number as the “columns” in the array.

```
list[element].length
```

- Note that each row in a two dimensional array could have a different number of columns. We sometimes refer to these kinds of arrays as “ragged” arrays

+
Programming Challenge (TwoDimArrayBasics.java)

- Declare, create and initialize 2D arrays:

Processing Two Dimensional Arrays

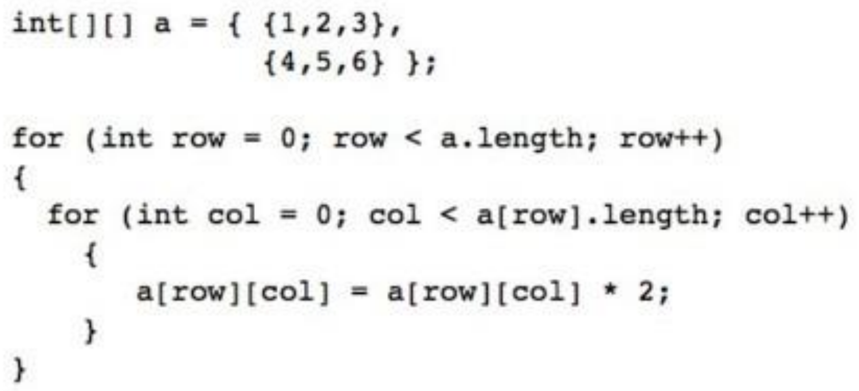
+ Iterating over all elements in a two dimensional array

- In order to iterate over all elements in a two dimensional array you will need to maintain two indexes - one for the row and one for the column
- This is usually done by setting up a nested for loop like this:

```
for (int row = 0; row < list.length; row++)
{
    for (int col = 0; col < list[row].length; col++)
        {
            // statements
            // list[row][col] = …
        }
}
```

+

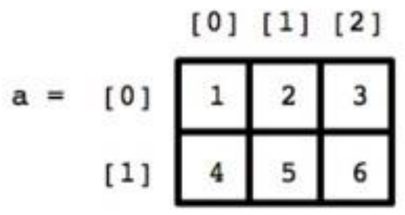
Iterating over all elements in a two dimensional array



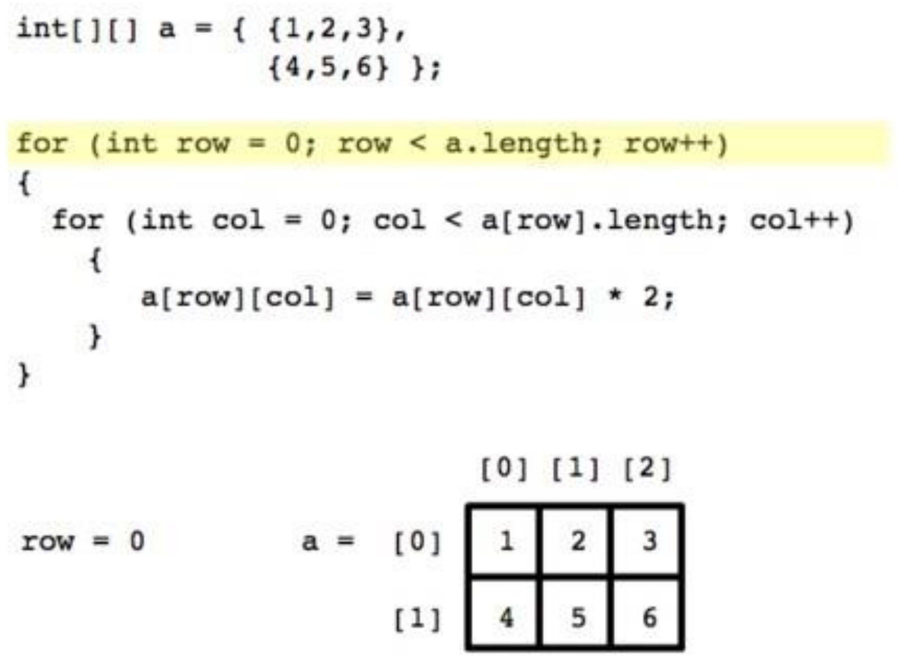
```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

+ Iterating over all elements in a two dimensional array

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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
        a[row][col] = a[row][col] * 2;
    }
}
```



+ Iterating over all elements in a two dimensional array

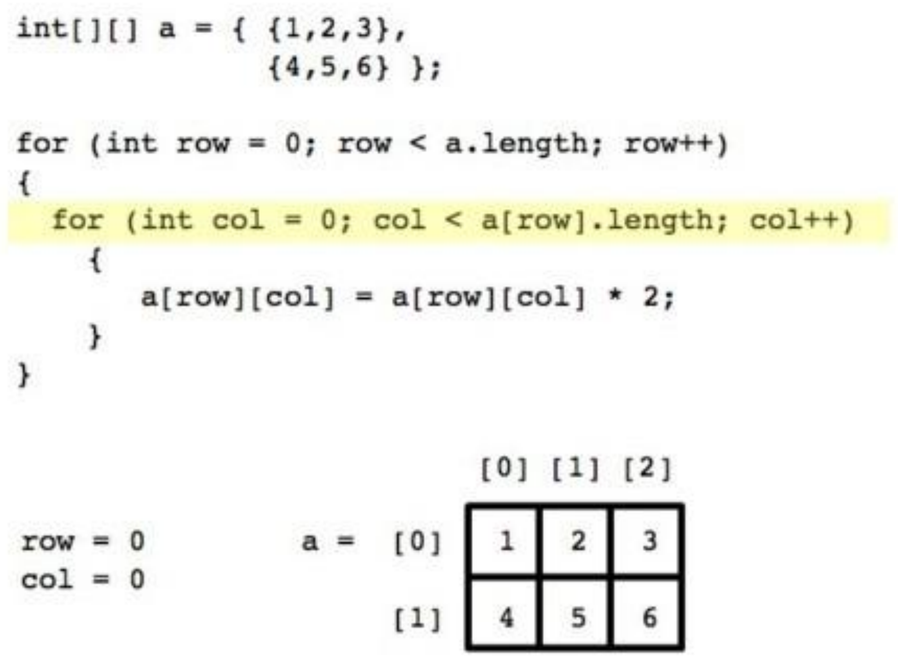


```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
        a[row][col] = a[row][col] * 2;
    }
}
```

row = 0

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 1 & 2 & 3 \\ \hline [1] & 4 & 5 & 6 \end{array}$$

Iterating over all elements in a two dimensional array



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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
        a[row][col] = a[row][col] * 2;
    }
}
```

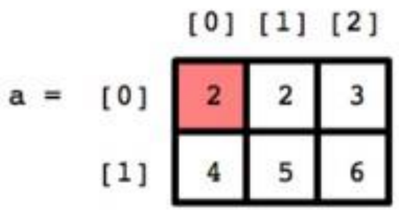
```
row = 0
col = 0
```

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 1 & 2 & 3 \\ \hline [1] & 4 & 5 & 6 \end{array}$$

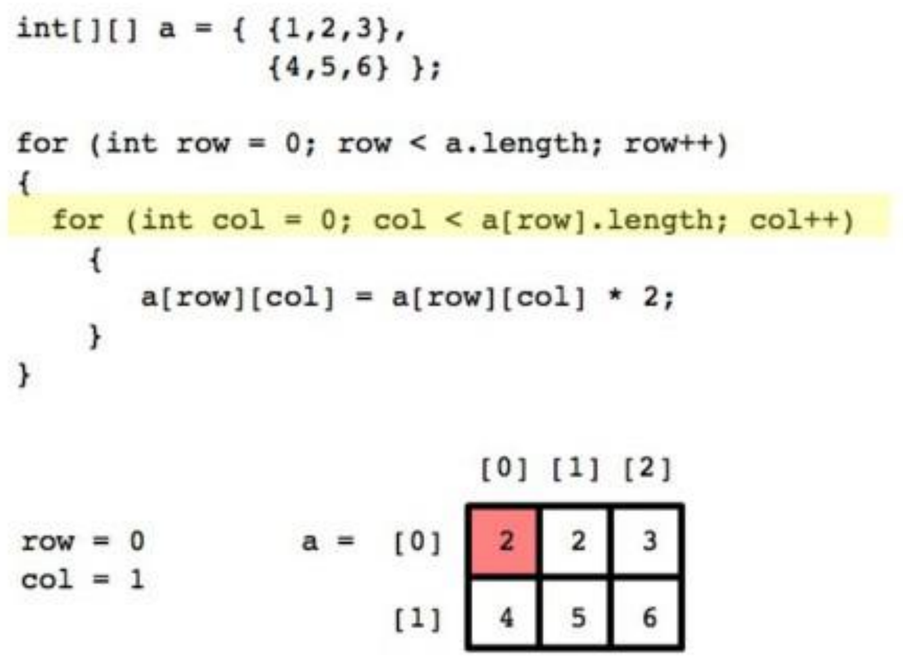
+ Iterating over all elements in a two dimensional array

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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

```
row = 0
col = 0
```



+ ## Iterating over all elements in a two dimensional array



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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

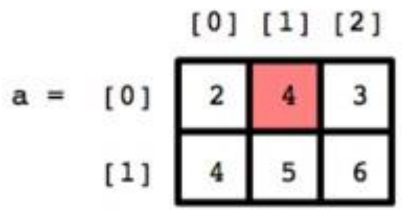
```
row = 0
col = 1
```

$$
a = \begin{array}{c|ccc} & [0] & [1] & [2] \\ \hline [0] & 2 & 2 & 3 \\ [1] & 4 & 5 & 6 \end{array}
$$

+
Iterating over all elements in a two dimensional array

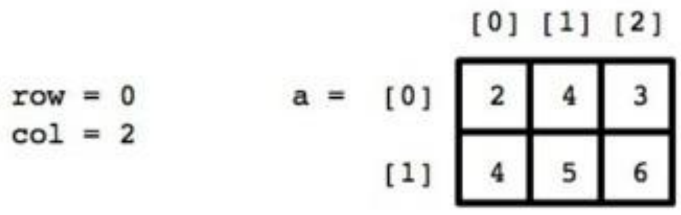
```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

```
row = 0
col = 1
```



+
Iterating over all elements in a two dimensional array

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

for (int row = 0; row < a.length; row++)
{
   for (int col = 0; col < a[row].length; col++)
      {
         a[row][col] = a[row][col] * 2;
      }
}
```

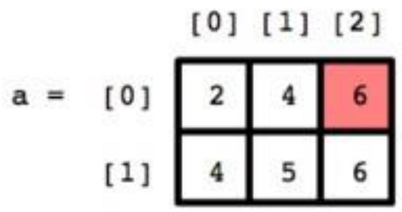


$$\begin{array}{c|ccc} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 3 \\ [1] & 4 & 5 & 6 \end{array}$$

+ Iterating over all elements in a two dimensional array

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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

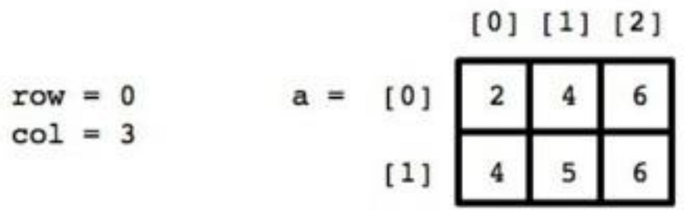
```
row = 0
col = 2
```



$$a = \begin{array}{c|ccc} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & \boxed{6} \\ [1] & 4 & 5 & 6 \end{array}$$

+ Iterating over all elements in a two dimensional array

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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```



```
row = 0
col = 3
```

$$a = \begin{array}{c|ccc} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ [1] & 4 & 5 & 6 \end{array}$$

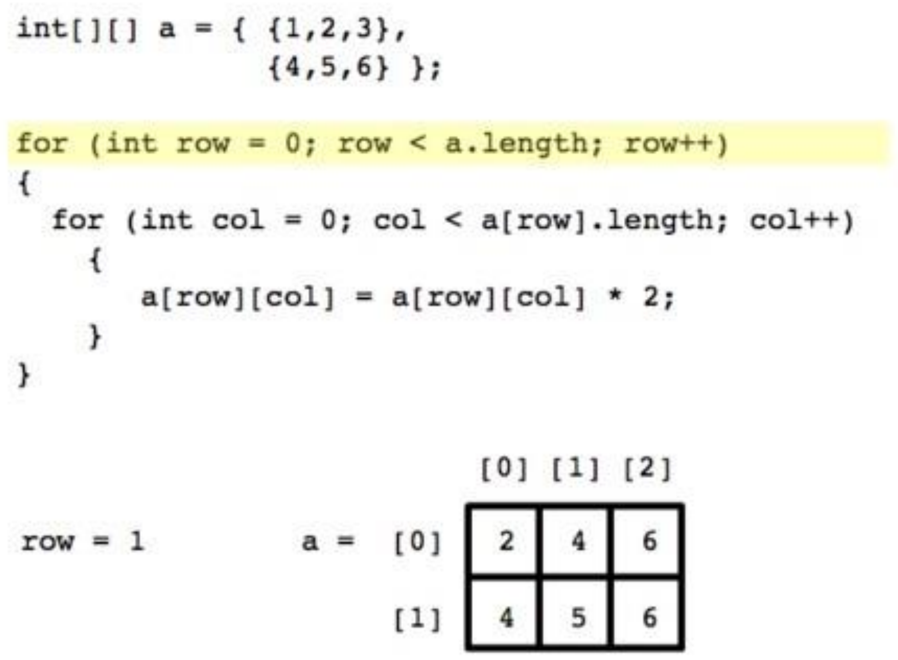
+
Iterating over all elements in a two dimensional array

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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

row = 0

$$a = \begin{array}{c|c|c|c|} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 4 & 5 & 6 \\ \hline \end{array}$$

Iterating over all elements in a two dimensional array

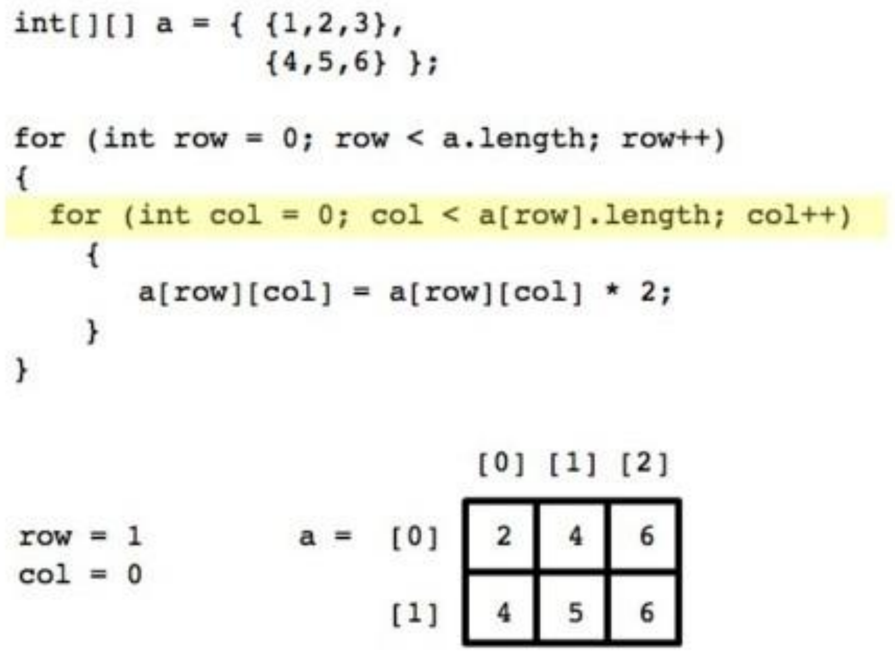


```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
        a[row][col] = a[row][col] * 2;
    }
}
```

row = 1

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 4 & 5 & 6 \end{array}$$

Iterating over all elements in a two dimensional array

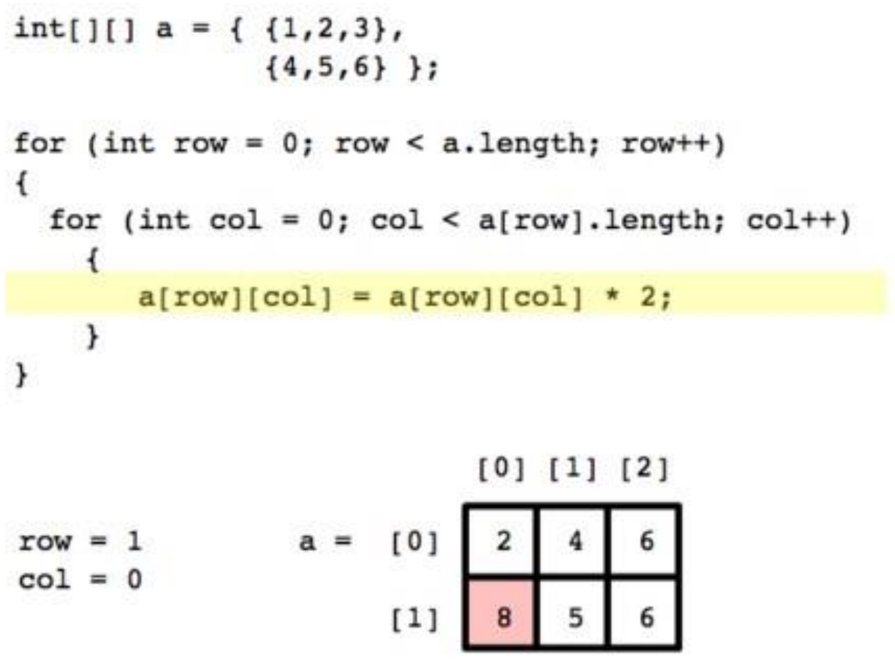


```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

```
row = 1
col = 0
```

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 4 & 5 & 6 \end{array}$$

+ Iterating over all elements in a two dimensional array



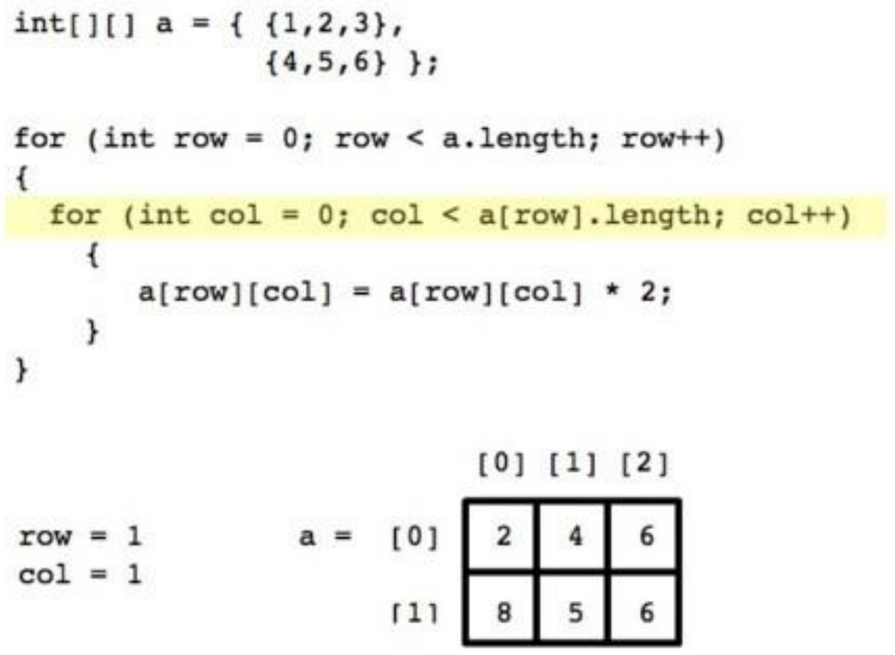
```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

```
row = 1
col = 0
```

a =

	[0]	[1]	[2]
[0]	2	4	6
[1]	8	5	6

Iterating over all elements in a two dimensional array



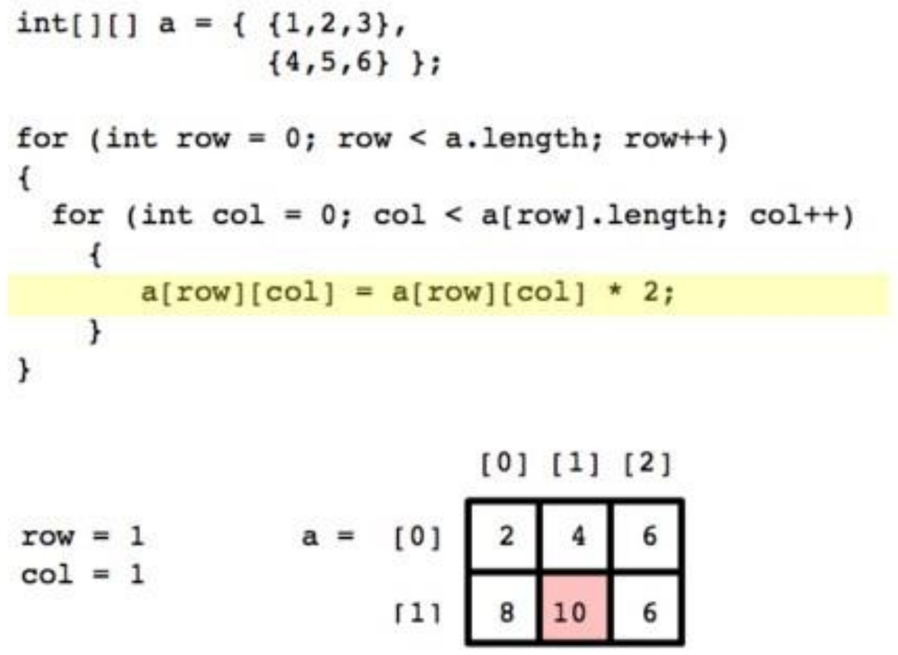
```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

```
row = 1
col = 1
```

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 8 & 5 & 6 \end{array}$$

+

Iterating over all elements in a two dimensional array



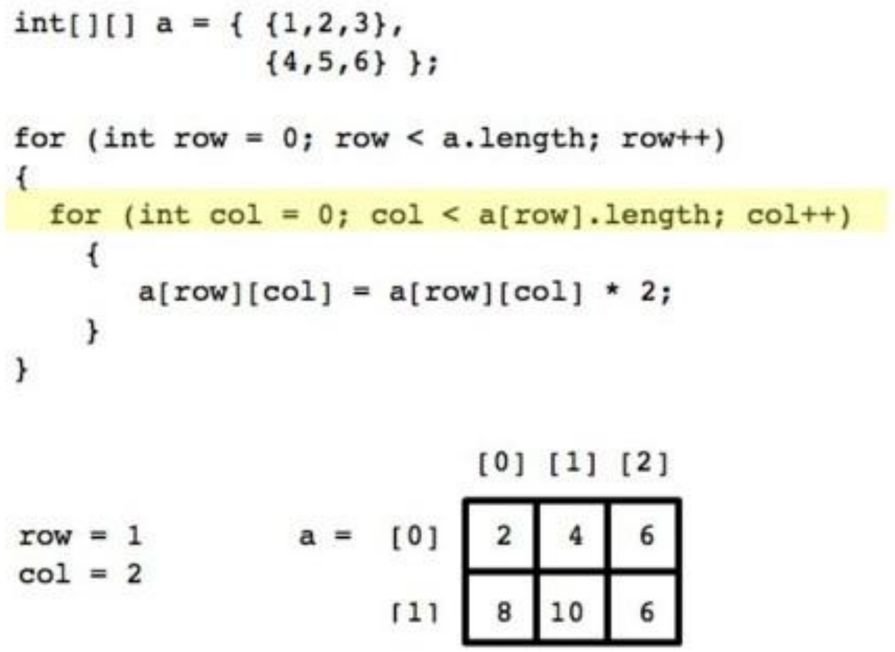
```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

```
row = 1
col = 1
```

a =

	[0]	[1]	[2]
[0]	2	4	6
[1]	8	10	6

+
Iterating over all elements in a two dimensional array

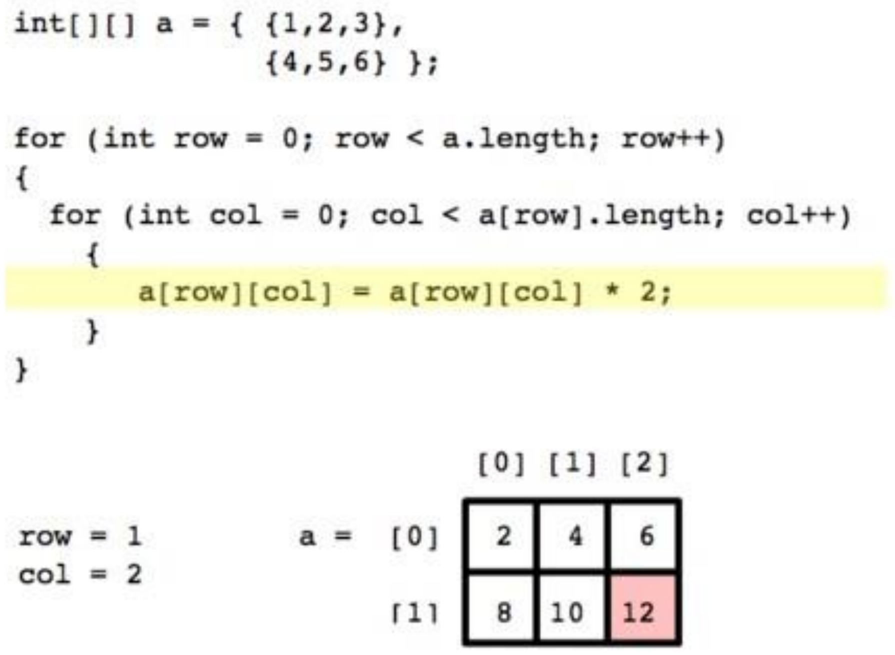


```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
        a[row][col] = a[row][col] * 2;
    }
}
```

```
row = 1
col = 2
```

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 8 & 10 & 6 \end{array}$$

+
Iterating over all elements in a two dimensional array

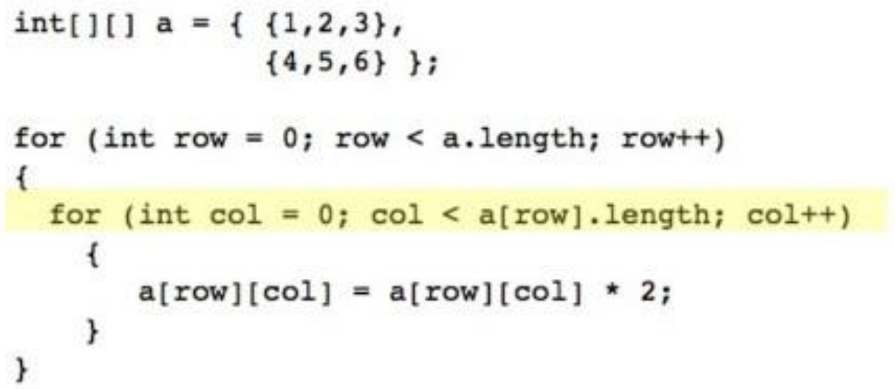


```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

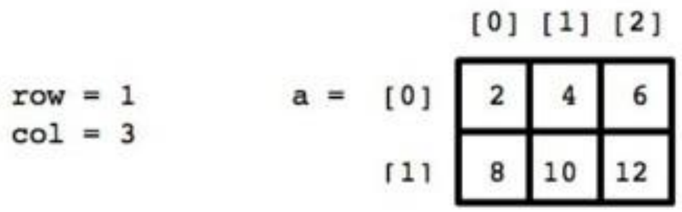
```
row = 1
col = 2
```

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 8 & 10 & 12 \end{array}$$

+ Iterating over all elements in a two dimensional array



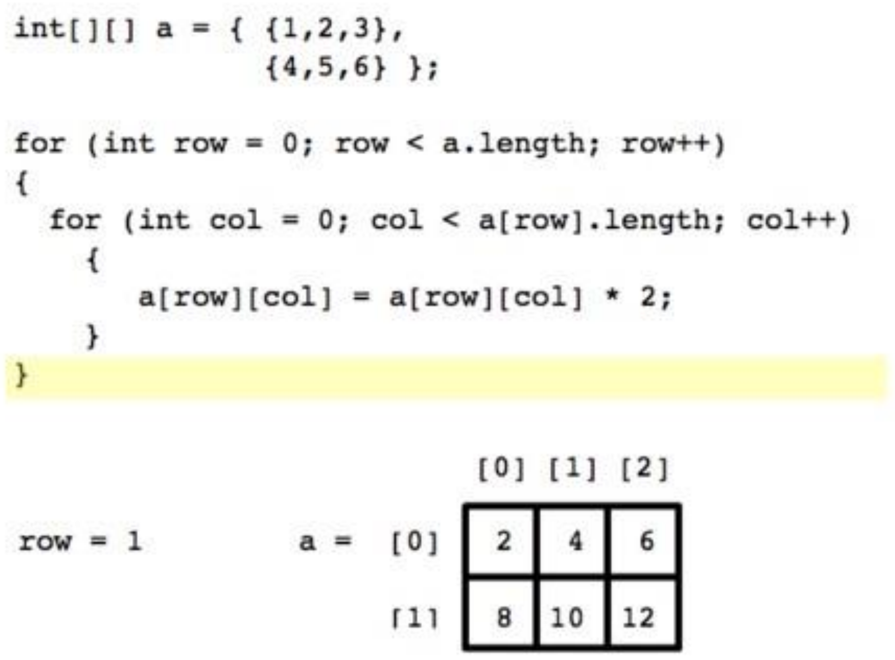
```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
        a[row][col] = a[row][col] * 2;
    }
}
```



row = 1
col = 3

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 8 & 10 & 12 \end{array}$$

+
Iterating over all elements in a two dimensional array

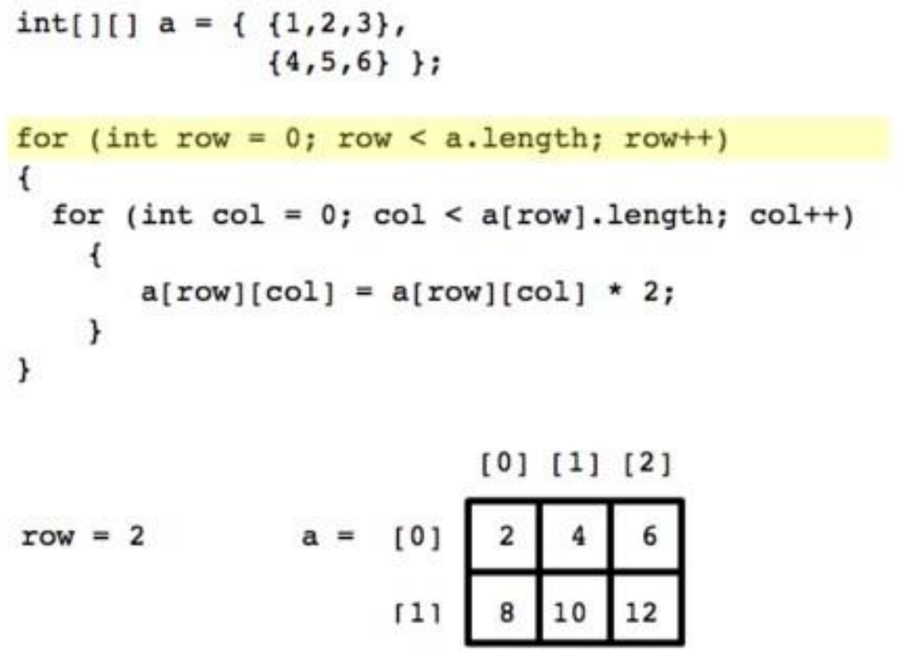


```
int[][] a = { {1,2,3},
              {4,5,6} };

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

row = 1

$$a = \begin{array}{c|c|c|c} & [0] & [1] & [2] \\ \hline [0] & 2 & 4 & 6 \\ \hline [1] & 8 & 10 & 12 \end{array}$$

+
Iterating over all elements in a two dimensional array



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

for (int row = 0; row < a.length; row++)
{
  for (int col = 0; col < a[row].length; col++)
    {
       a[row][col] = a[row][col] * 2;
    }
}
```

Printing a two dimensional array

- Just as with single dimensional arrays, you cannot directly print a two dimensional array by using its variable name.

```
int[][] list = { {1,2,3}, {4,5,6} };
System.out.println(list);
// prints memory address Why?
```

- Two dimensional arrays are reference types, so printing out the variable name associated with an array will only print out the memory address where the array is currently stored on the heap.

- Therefore you will need to set up a nested for loop in order to access and print each element in your array.

+
Summing all elements

- You can sum up all elements in an array by establishing a sum accumulator variable outside of the array and then accessing that variable as you visit each element. For example:

```
// set up a running sum
int sum = 0;

// iterate over the array
for (int row = 0; row < list.length; row++)
{
   for (int col = 0; col < list[row].length; col++)
      {
         sum += list[row][col];
      }
}
```

Summing elements by row & column

- Sometimes you will find the need to sum up an entire row or column of an array, especially when working with financial or graphical data.
- You can set up a system to do this by doing the following:
  - Establish new single dimensional counter array(s) to keep track of all sums by row and/or column
  - Iterate over every element in the list
  - Update your counter arrays according to the row or column you are currently examining

+

Programming Challenge (TwoDimArrayProcessing.java)

- Given the array:

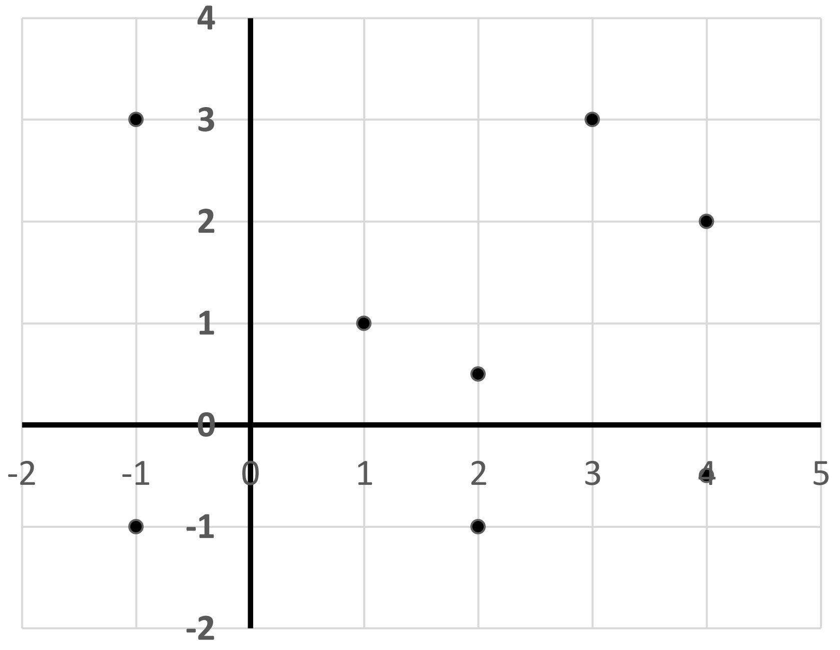
```
int[][] list =
{ {1,2,3}, {4,5,6},
{7,8,9} };
```

- Compute the sum of each row and the sum of each column

- Next, compute the largest row and the largest column

Programming Challenge (FindingNearestPoints.java)

- Given a set of points, calculate the two points that are nearest to each other.



	X	Y
1	-1	3
2	-1	-1
3	1	1
4	2	0.5
5	2	-1
6	3	3
7	4	2
8	4	-0.5

+
Random shuffling of all elements in a two dimensional array

- You can randomly shuffle the elements in a two dimensional array in the same way you would shuffle the elements of a 1 dimensional array. The general algorithm is as follows:
  - Visit every element
  - Pick a random row and column location
  - Store the current element into a temp variable
  - Place the contents at the random location in the current location
  - Place the contents of the temp variable into the random location

+ Two Dimensional Arrays and Methods

Programming Challenge (TwoDimArrayAndMethods.java)

- Passing and returning 2D arrays to and from methods