Data Structures and Algorithms

Arrays

Dr. Lina Xu
lina.xu@ucd.ie

School of Computer Science,
University College Dublin

September 10, 2018

Table of Contents

Arrays in Java

Arrays in Java are used in the exact same way as in C, but there are some differences in how they are declared, created and some extra functionality

- Arrays in Java are objects
- With arrays in Java there is a difference between declaring an array and constructing an array
- When constructing them we have to use the new keyword just like with objects
- Arrays contain an instance variable called length, which stores the size of the array

Declaring Arrays

Declaring an array variable does not allocate the memory for the array.

- An array is not given a size until we create it
- Arrays should be declared with the type and brackets together then followed by a name

Syntax of array declaration

```
type[] name;
int[] numbers;
```

- This only creates a reference for storing the location of the array, it does not allocate any memory for the array

Constructing Arrays

- If we try to use and array before it is constructed, the program will crash
- Just like creating objects we use the new keyword to create an array

Syntax of array construction

```
new type[size];
new double[10];
```

- This allocates the memory for the array but does not store the reference to the location of the array

Declaration and construction

```
type[] vName = new type[size];
double[] values = new double[10];
```

Alternative Array Construction

- Alternatively, we can create an array in Java by passing it a list of values
- The values must be of the correct type

Array list declaration

```
type[] arr = { val, val, ..., val};
int[] a = {1, 2, 3, 4, 5, 6, 7};
```

- When created this way, Java will automatically allocate the exact amount of memory required to store the data
- This means you cannot add any new information to the array without replacing the original data

Arrays of Objects

Arrays can be created to store and primitive type as well as any type of objects

- `String[] names = new String[10];`
- `Rectangle[] rects = new Rectangle[100];`
- `Point[] points = new Point[123];`

Default Values

When an array is first constructed, the value of each element in the array is set to the default value for that type

- For primitive numbers such as int and double this is the value 0 or 0.0
- For the boolean type the this is the value false
- For all objects this is the special reference called null

Using Arrays in Java

Using an array in Java is the same as in C

Assigning a value to an element in the array

```
varName[index] = newValue;
names[0] = "John";
```

Using a value in the array

```
int x = varName[index];
System.out.println(names[0]);
```

Size of an Array

When arrays are constructed in Java its size is stored in an instance variable named length

- This instance variable can be used in the same an instance variable of any other object

Syntax

```
int size = varName.length;
int size = names.length;
```

Table of Contents

Iterating Through an Array

- Iterating through each element in an array is a very common task
- It can be achieved using any type of loop
- It is generally easier using a for loop
- It is even easier to complete using a for-each loop
  - We will learn about these later

Array with for loop

- We start at index 0
- We stop when we get to `arrayName.length`
- We increment by one every step of the loop

Syntax

```
1 for(int i = 0; i < arr.length; i++){
2   //Some code here
3 }
```

Table of Contents

2D Arrays

- The arrays that we have used so far have been one dimensional
  - This means a single line of elements
- Often data comes in a form that is similar to a table
  - We can store this in a one dimensional array
  - But it is easier to understand if we store it in a two dimensional array

Declaring Arrays

Declaring an array variable does not allocate the memory for the array.

- Declaring 2D arrays in Java is very similar to one dimensional arrays
- We just need to add a second set of brackets

Syntax of 2D array declaration

```
type[][] name;
int[][] numbers;
```

- This only creates a reference for storing the location of the array, it does not allocate any memory for the array

Constructing 2D Arrays

- Constructing 2D arrays in Java is very similar to one dimensional arrays
- We need to add another set of brackets and another size
  - It is easier if we think of the first size as the number of rows and the second as the number of columns

Syntax of 2D array construction

```
new type[size1][size2];
new double[10][15];
```

- This allocates the memory for the array but does not store the reference to the location of the array

Declaration and construction

```
type[][] vName = new type[size1][size2];
double[][] values = new double[10][15];
```

Arrays of Arrays

- When we are creating a two dimensional array in Java we are really creating an array where every element is another array
- Each element in the first array is a one dimensional array that we can use in the same way as before
- The values are stored in the one dimensional arrays

Array Types

If we have the following declaration

```
double[][] values = new double[3][3];
```

- What is the type of `values[0][0]`?
  - double
- What is the type of `values[0]`?
  - Array of doubles
- What is the type of `values`?
  - An array of arrays of doubles

2D Array size

- Finding the size of a two dimensional array is similar to finding the size of a one dimensional array
- We just have to find both the number of arrays and the size of the arrays
- The code `values.length` will tell us the size of the overall array
  - This really means the number of arrays that it holds
- To find the size of the internal arrays we used `values[0].length`
  - This tells us the size of the array `values[0]`
- The total number of elements that can be stored in the 2D array is the product of these numbers
- If the 2D array was declared using lists of values, not all of the arrays will be the same size

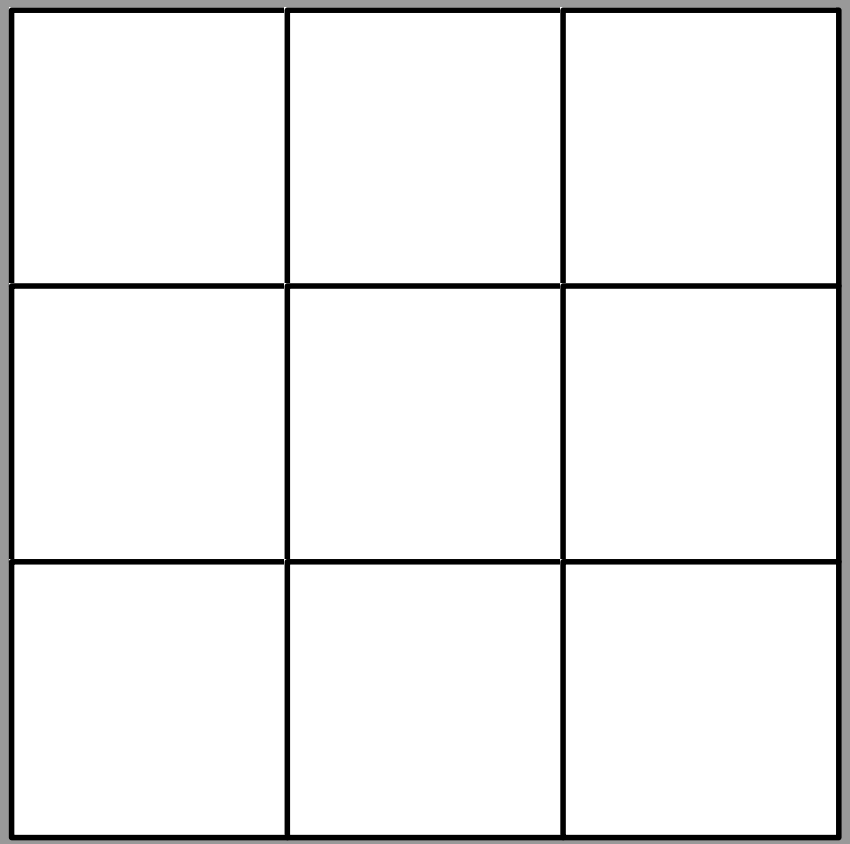
Iterating Through 2D Arrays

Looping through a 2D array requires a pair of nested loops

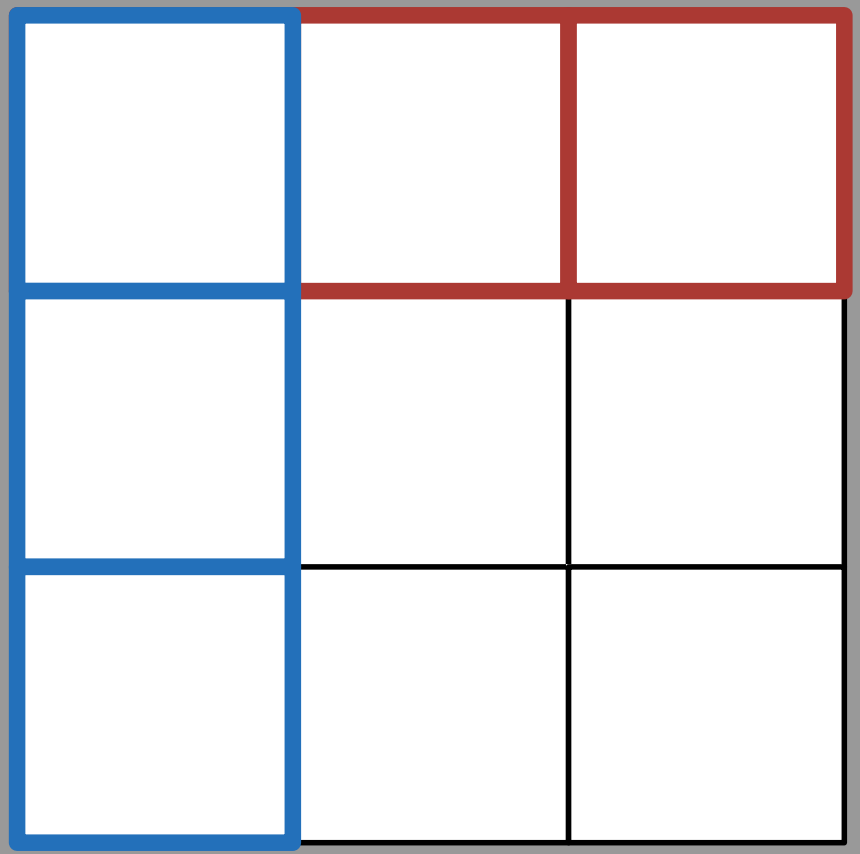
- The first loop is responsible for iterating through each of the arrays
- The second loop is responsible for iterating through each value in the array

```
1 for (int i = 0; i < values.length; i++) {
2   for (int j = 0; j < values[i].length; j++) {
3     System.out.print( values[i][j] + " ");
4   }
5   System.out.println();
6 }
```

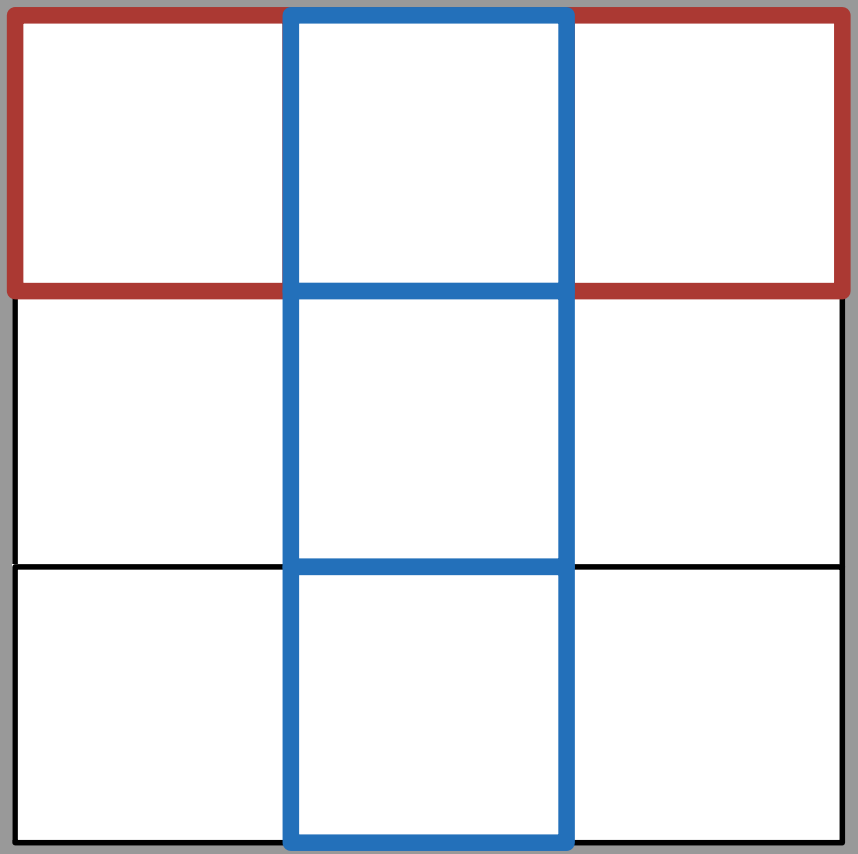
Iterating Through 2D Arrays



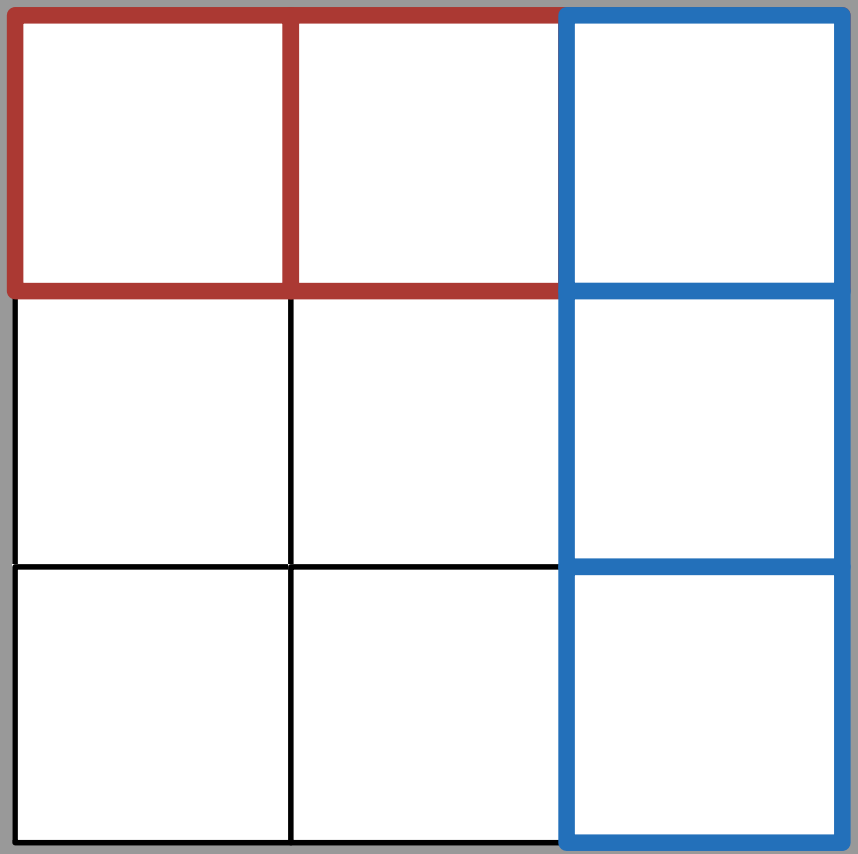
Iterating Through 2D Arrays



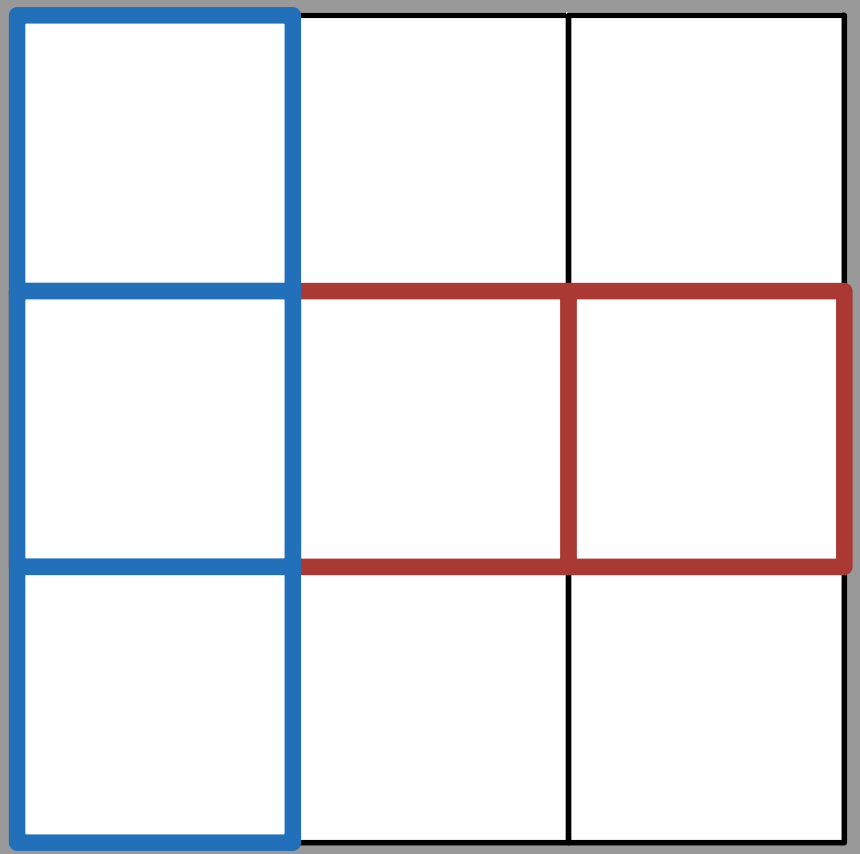
Iterating Through 2D Arrays



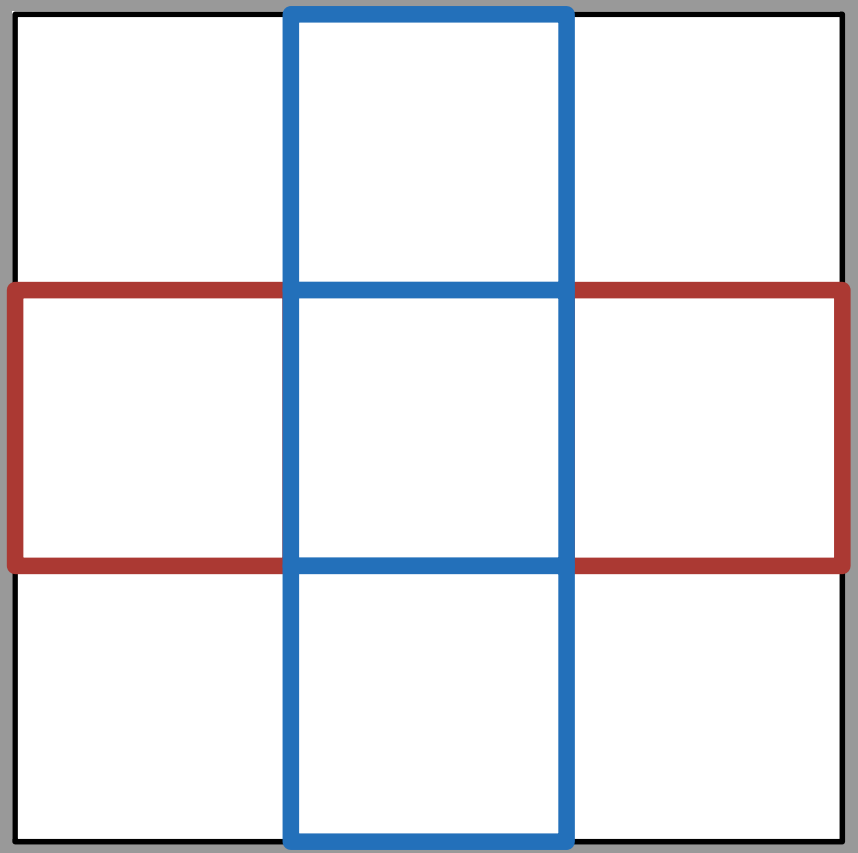
Iterating Through 2D Arrays



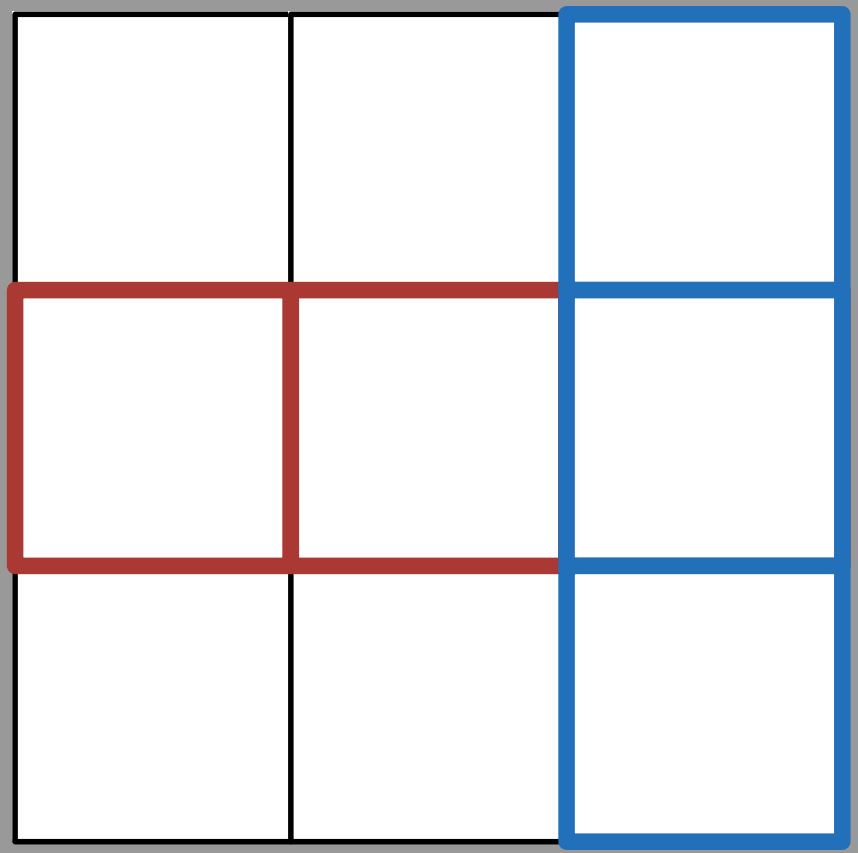
Iterating Through 2D Arrays



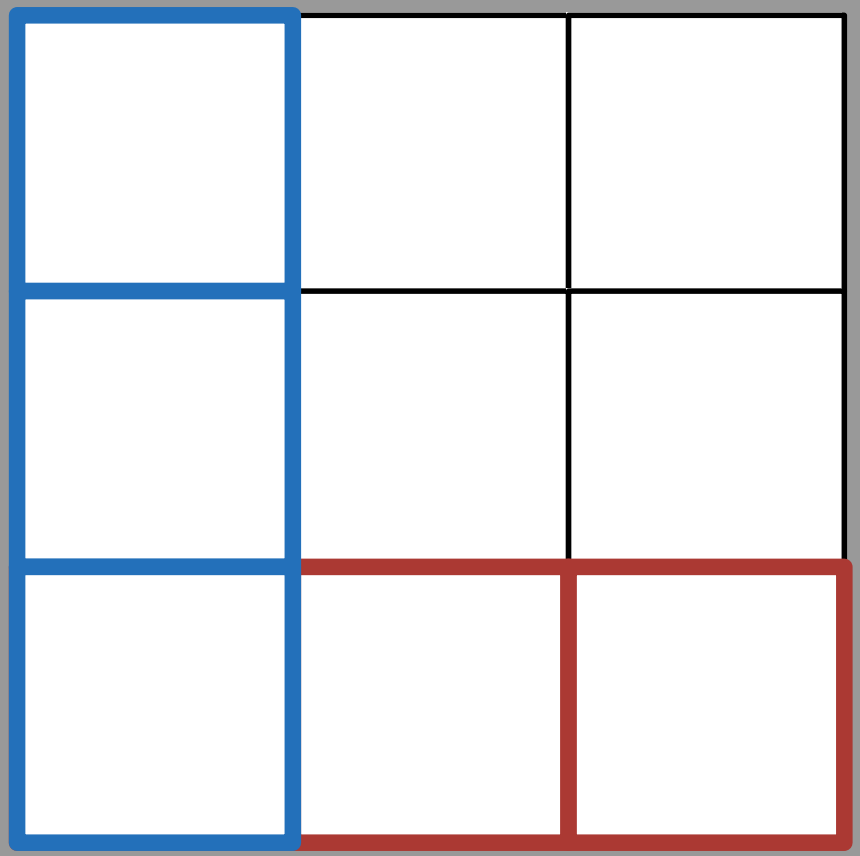
Iterating Through 2D Arrays



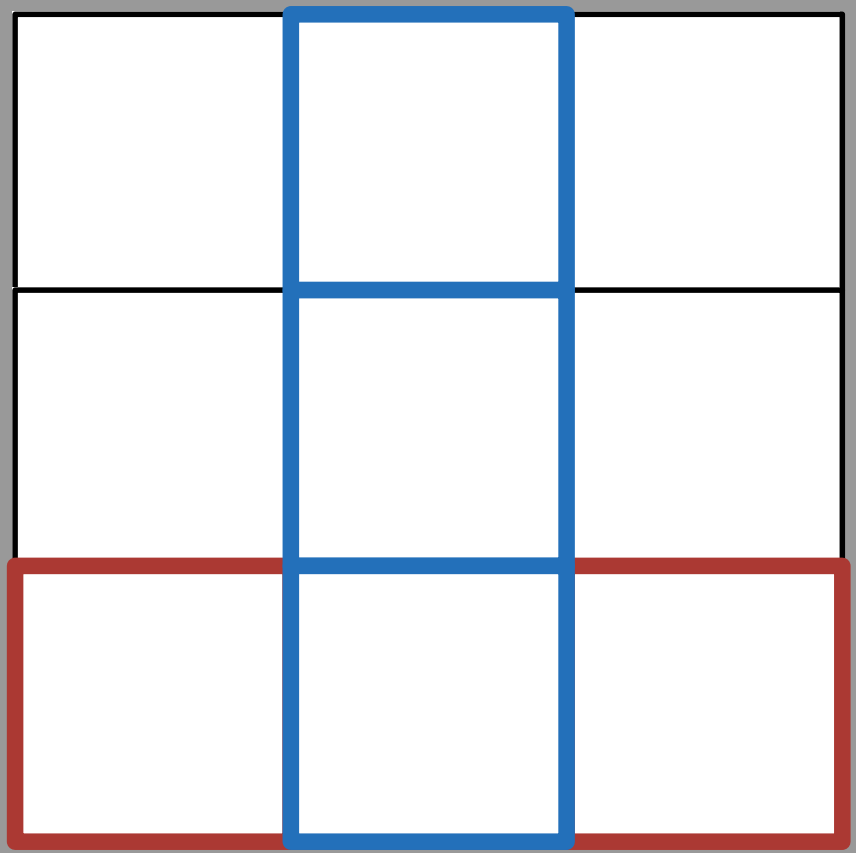
Iterating Through 2D Arrays



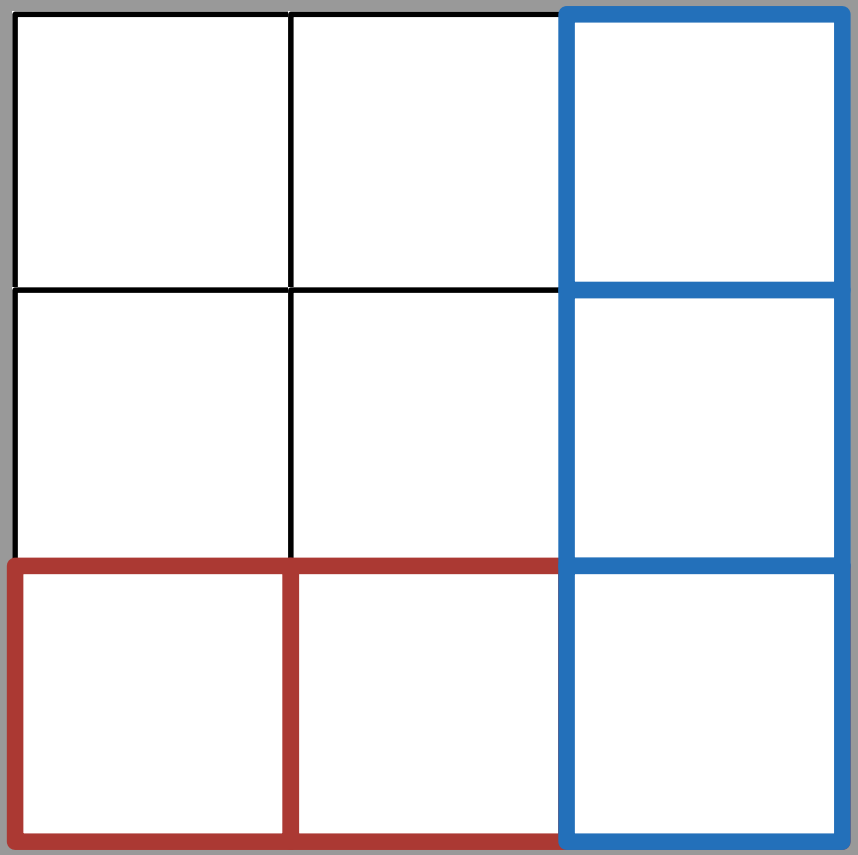
Iterating Through 2D Arrays



Iterating Through 2D Arrays



Iterating Through 2D Arrays



Rows and Columns

	Column 1	Column 2	Column 3	Column 4
Row 1	`a[0][0]`	`a[0][1]`	`a[0][2]`	`a[0][3]`
Row 2	`a[1][0]`	`a[1][1]`	`a[1][2]`	`a[1][3]`
Row 3	`a[2][0]`	`a[2][1]`	`a[2][2]`	`a[2][3]`

Further Information and Review

- Uneven 2D array
- If you wish to review the materials covered in this lecture or get further information, read the following section in Data Structures and Algorithms textbook.
  - 3.1 - Using Arrays