Array topic index

Arrays: syntax and basic use

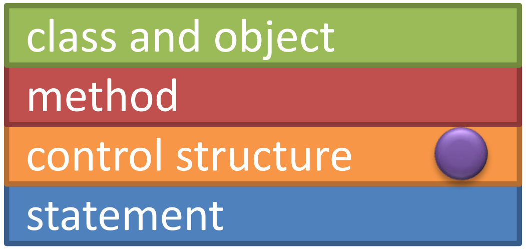
Tracing array code

Methods for working with arrays

Multidimensional arrays

Arrays

How to store and work with large amounts of data



10 Managing Collections with Arrays

Dr James Montgomery, james.montgomery@utas.edu.au

The need for arrays

Task: Calculate the average age of a group of people

Possible (partial) solution:

```
int age1, age2, age3, age4;
int sum;
double average;
//Would read ages from user
//    (~8 lines of code)
//Would add those together,
//    storing total in sum
average = (double) sum / 4;
```

But what if fewer than 4 ages?

What if more?

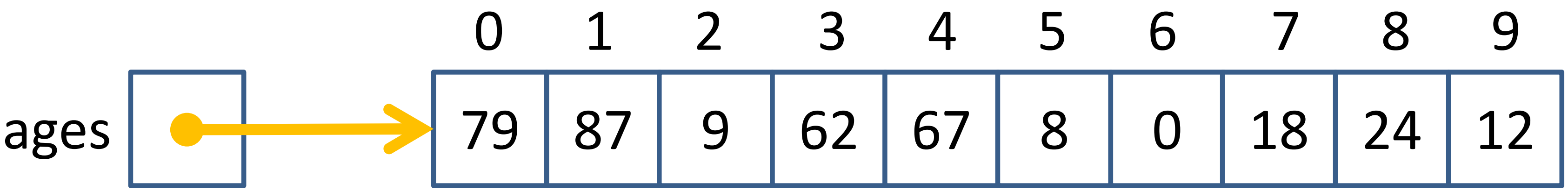
This solution can't scale and requires a lot of duplication

We need one variable name for a collection of ages

Arrays: the simplest data structure

An array is an ordered (and indexed) list of values of the same type (primitive or object)

Example: a list of 10 integer ages



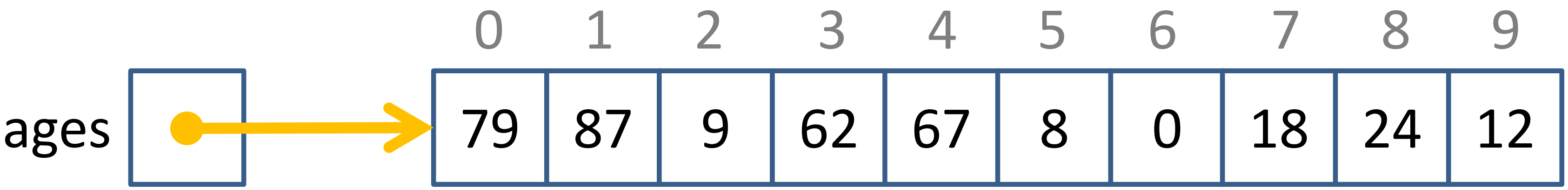
index of each element

	0	1	2	3	4	5	6	7	8	9
ages	79	87	9	62	67	8	0	18	24	12

array reference

the elements

Array elements are like other variables



ages

0	1	2	3	4	5	6	7	8	9
79	87	9	62	67	8	0	18	24	12

ages[*index*] is a single variable at position *index*

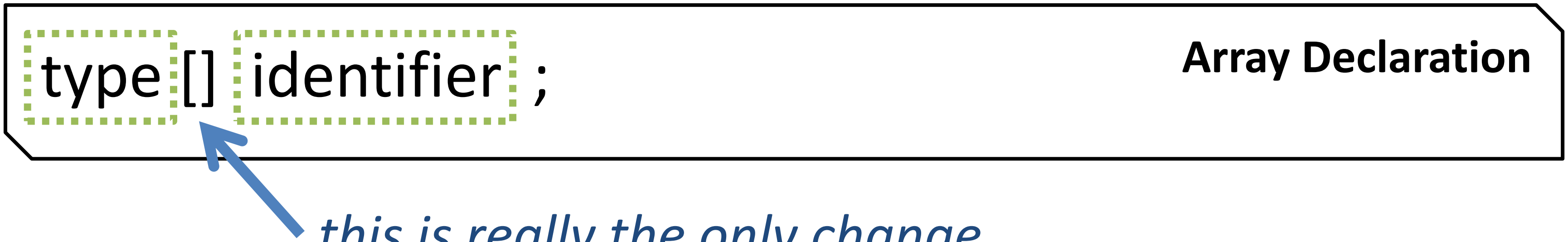
`ages[4]` is 67

```
int a = ages[4]; //a is now 67
ages[4] = ages[4] + 1; //happy birthday
```

`ages[4]` is now

Tip: *index* can be any integer expression

Declaring arrays



Array Declaration

type [] identifier ;

this is really the only change

```
int[] ages;
String[] names;
```

How would you declare an array of doubles?

Allocating space

Arrays are like objects: when declared they refer to **null**

identifier = new type [size];

Array Initialisation

```
ages = new int[10];
```

initially, every element is zero (or false if array of boolean, or null if array of objects)

or (better)

```
final int SIZE = 10;
ages = new int[SIZE];
```

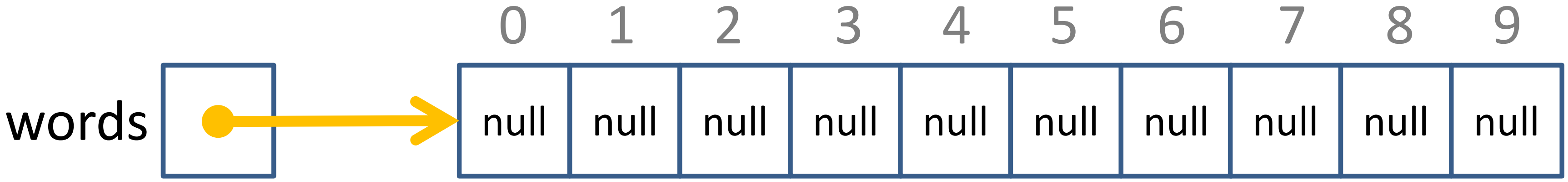
Can also declare and initialise all at once:

int[] ages = { 79, 87, 9, 62, 67, 8, 0, 18, 24, 12 };

Arrays of objects

```
String[] words = new String[10];
```

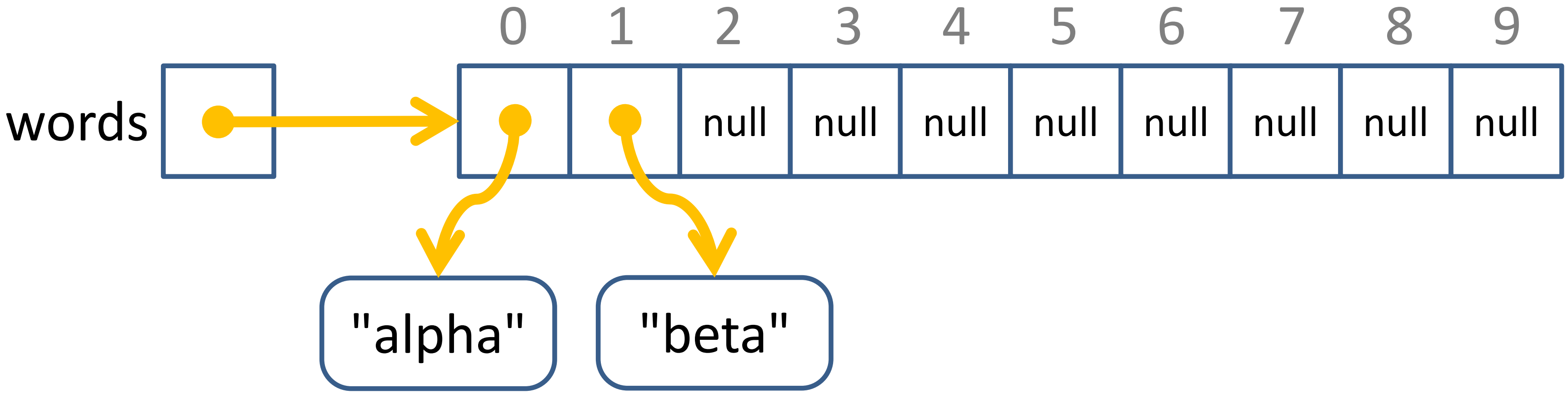
allocates space for 10 String references, but doesn’t create them



words

0	1	2	3	4	5	6	7	8	9
null	null	null	null	null	null	null	null	null	null

```
words[0] = "alpha";
words[1] = "beta";
```



words

0	1	2	3	4	5	6	7	8	9
		null	null	null	null	null	null	null	null

"alpha"

"beta"

Length of an array

Arrays have a `length` property (a read-only integer data member, not a method)

Given

```
int[] counts = new int[15];
String[] labels = new String[25];
```

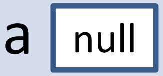
`counts.length` is 15 and
`labels.length` is

Tip: Attempting to access an element below 0 or above array's length – 1 results in an error

Quick summary

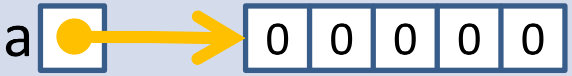
Declare an array reference

- syntax: *type*[] *identifier*;
- example: int[] a;



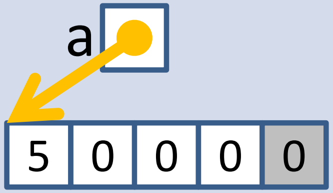
Allocate space

- syntax: *identifier* = new *type*[*size*];
- example: a = new int[5];



Access a specific element

- syntax: *identifier*[*index*];
- examples:
  - a[0] = 5;
  - int x = a[4];



And…

- *array*.length is length of array, as in a.length
- Array contents can be modified by methods

Tracing array code

- class and object
- method
- control structure ●
- statement

10 Managing Collections with Arrays

Dr James Montgomery, james.montgomery@utas.edu.au

Two alternatives

If the code appears to be reading the array only…

draw it off to the side of the tracing table

Line	sum	i
2	0	
3		0
6	1	
7		1
6	3	
7		2
6	7	
7		3
6	15	

	0	1	2	3
a	1	2	4	8

If it's modifying the array's contents…

incorporate it into the table

		a			
Line	i	0	1	2	3
1		1	2	4	8
2	0				
5		2			
6	1				
5			4		
6	2				
5				8	
6	3				

Demonstration

```
1.  int[] a = { 1, 2, 4, 8};
2.  int sum = 0;
3.  int i = 0;
4.
5.  while (i < a.length) {
6.      sum += a[i];
7.      i++;
8.  }
```

Line	sum	i
2	0	
3		0
6	1	
7		1
6	3	
7		2
6	7	
7		3
6	15	
7		4

	0	1	2	3
a	1	2	4	8

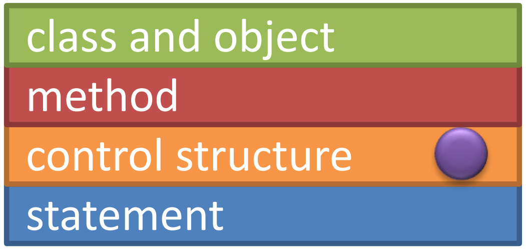
Demonstration

```
1.   int[] a = { 1, 2, 4, 8};
2.   int i = 0;
3.
4.   while (i < a.length) {
5.       a[i] = 2 * a[i];
6.       i++;
7.   }
```

		a			
Line	i	0	1	2	3
1		1	2	4	8
2	0				
5		2			
6	1				
5			4		
6	2				
5				8	
6	3				
5					16
6	4				

	0	1	2	3
a	2	4	8	16

Methods for working with arrays



class and object

method

control structure

statement

10 Managing Collections with Arrays

Dr James Montgomery, james.montgomery@utas.edu.au

Arrays as method arguments

Arrays are references, so when passed to a method the *reference* is copied

Changes to the elements persist after the method has finished

```
public static void main(String[] args) {
  int[] vals = { 2, 4, 6, 8, 10 };
  change(vals, 2, 100);
  System.out.println(vals[2]);
}

public static void change(int[] a, int index, int toVal) {

    a[index] = toVal;
}
```

What value is printed?

Arrays as method arguments

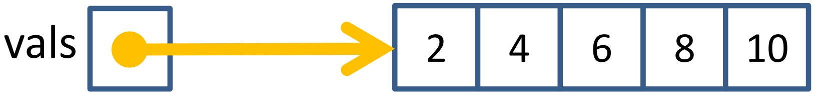
Arrays are references, so when passed to a method the *reference* is copied

Changes to the elements persist after the method has finished

```
public static void main(String[] args) {
    int[] vals = { 2, 4, 6, 8, 10 };
    change(vals, 2, 100);
    System.out.println(vals[2]);
}

public static void change(int[] a, int index, int toVal) {

        a[index] = toVal;
}
```



Arrays as method arguments

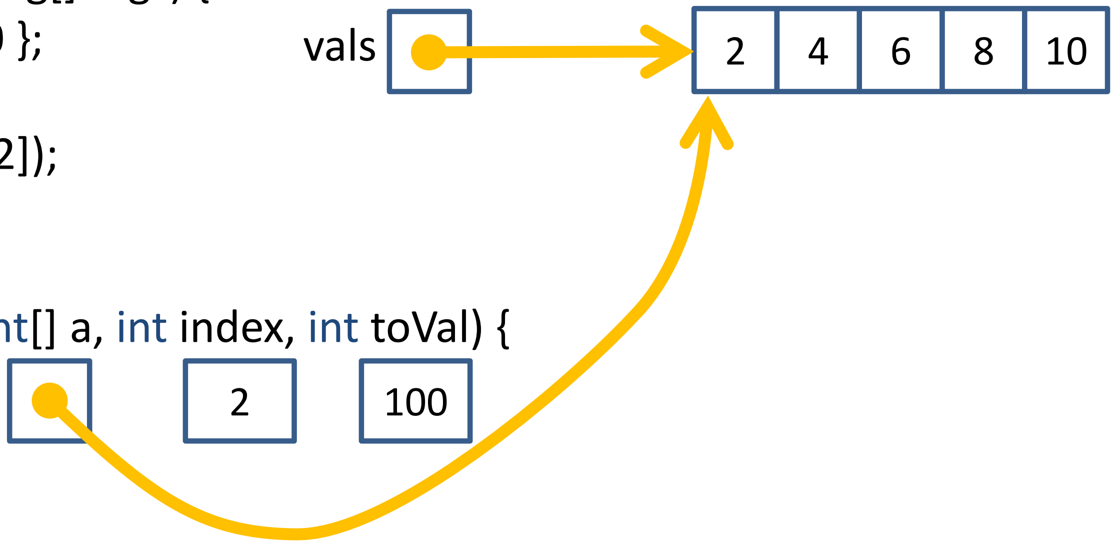
Arrays are references, so when passed to a method the *reference* is copied

Changes to the elements persist after the method has finished

```
public static void main(String[] args) {
   int[] vals = { 2, 4, 6, 8, 10 };
   change(vals, 2, 100);
   System.out.println(vals[2]);
}

public static void change(int[] a, int index, int toVal) {

      a[index] = toVal;
}
```



Arrays as method arguments

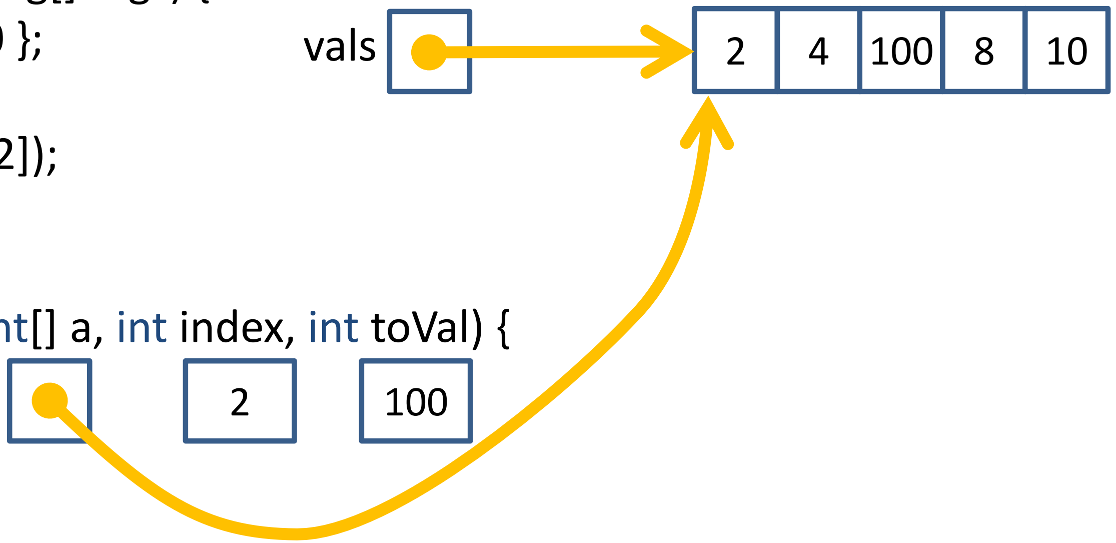
Arrays are references, so when passed to a method the *reference* is copied

Changes to the elements persist after the method has finished

```
public static void main(String[] args) {
   int[] vals = { 2, 4, 6, 8, 10 };
   change(vals, 2, 100);
   System.out.println(vals[2]);
}

public static void change(int[] a, int index, int toVal) {

→      a[index] = toVal;
}
```



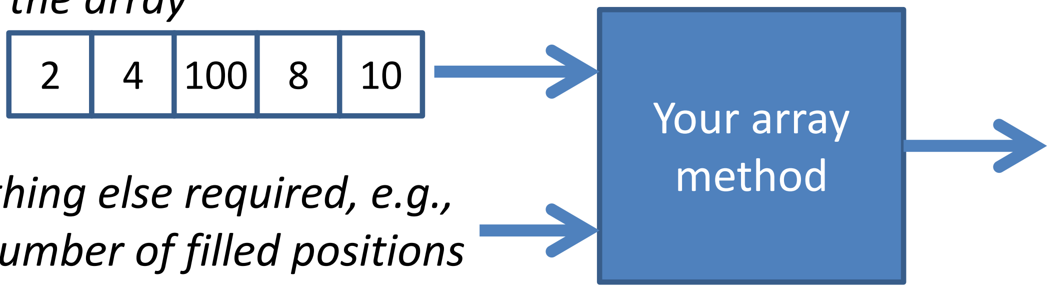
see ElementChange.java

Manipulating Arrays

If working on the entire array, write a method (it can be reused on different arrays of the same type)

Data in

Data out



the array

2	4	100	8	10

Anything else required, e.g.,

- *number of filled positions*
- *a value to search for*
- *value to multiply each element by*

Depends on the task, e.g.,

- *sum of elements*
- *index of found value*
- *maybe nothing*

The algorithm: simplest is to traverse the array and do “something” with each element

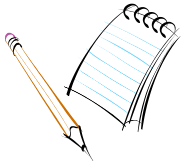
I – Set index to 0

T – index < length of array

B – Do “something” with element at that index

U – increment index

Print an array

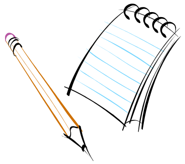


Task: Write a method to display all the elements of an array of ints

```
public static ______ display(int[] a, __________) {
  for (int i = 0; i < ______________ ; i++) {
    __________________________________________ ;
  }
}
```

Can also be done by using java.util.Arrays.toString()

Print an array



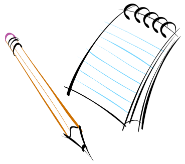
Task: Write a method to display all the elements of an array of ints

```java
public static void display(int[] a) {
    for (int i = 0; i < a.length; i++) {
        System.out.println("element " + i + ":" + a[i]);
    }
}
```

see DisplayFillAndSum.java

Can also be done by using java.util.Arrays.toString()

Fill an array



Task: Write a method to fill an entire array of ints with a given value

```
public static ______ fill(int[] a, ___________) {
    for (int i = 0; i < _____________ ; i++) {
        __________________________________________ ;
    }
}
```

Can also be done by using java.util.Arrays.fill()

Fill an array

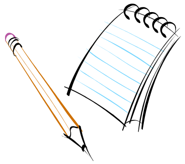
Task: Write a method to fill an entire array of ints with a given value

```
public static void fill(int[] a, int value) {
    for (int i = 0; i < a.length; i++) {
        a[i] = value;
    }
}
```

see DisplayFillAndSum.java

Can also be done by using java.util.Arrays.fill()

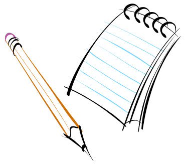
Sum the values in an array



Task: Write a method to calculate the sum of values in an array of integers

```
public static ______ sum(int[] a, _________) {
    _________________ ;
    for (int i = 0; i < _____________ ; i++) {
        ________________________________________ ;
    }
    __________________ ;
}
```

Sum the values in an array



Task: Write a method to calculate the sum of values in an array of integers

```
public static int sum(int[] a) {
    int total = 0;
    for (int i = 0; i < a.length; i++) {
        total += a[i];
    }
    return total;
}
```

see DisplayFillAndSum.java

Tip: working with partially filled arrays

When storing a collection of items it is common to have an array larger than the collection (to have space to add more items)

This leads to this common pattern:

```
type[] identifier = new type[SIZE];
int count = 0;
```

as in

```
int[] data = new int[10];
int count = 0;
```

Then methods to work on the array also take the value of count

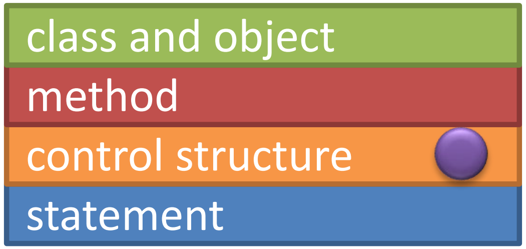
```
public static void display(int[] a, int count)

public static int sum(int[] a, int count)

public static int add(int[] a, int count, int value) //returns the new number of stored values
```

Multidimensional arrays

An advanced topic



class and object

method

control structure ●

statement

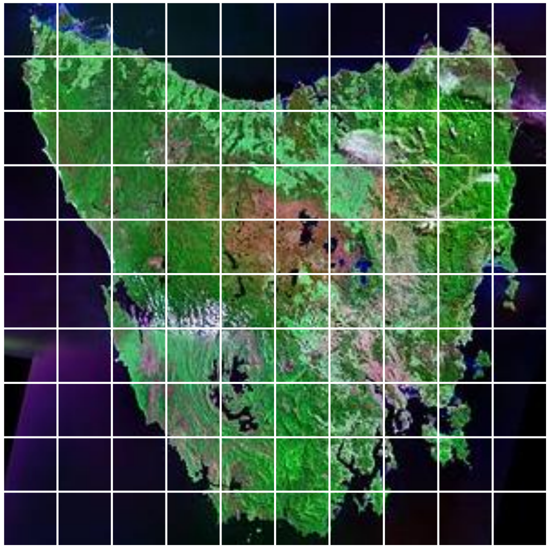
10 Managing Collections with Arrays

Dr James Montgomery, james.montgomery@utas.edu.au

Multidimensional arrays

Arrays can have more than one dimension

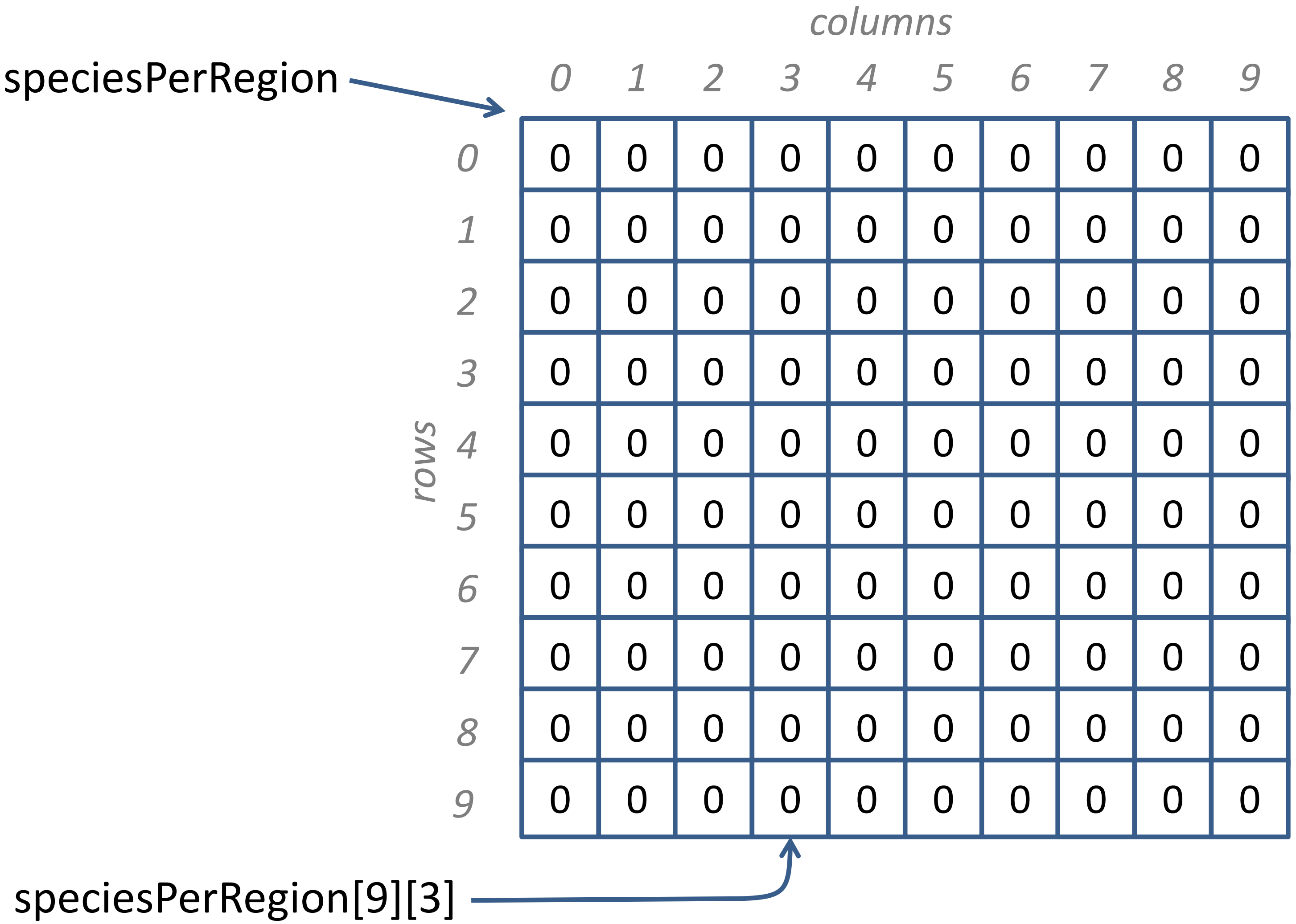
2D arrays (matrices) are common



```
final int ROWS = 10, COLS = 10;
int[][] speciesPerRegion = new int[ROWS][COLS];
```

Logical structure of a 2D array

```
final int ROWS = 10, COLS = 10;
int[][] speciesPerRegion = new int[ROWS][COLS];
```



speciesPerRegion

columns

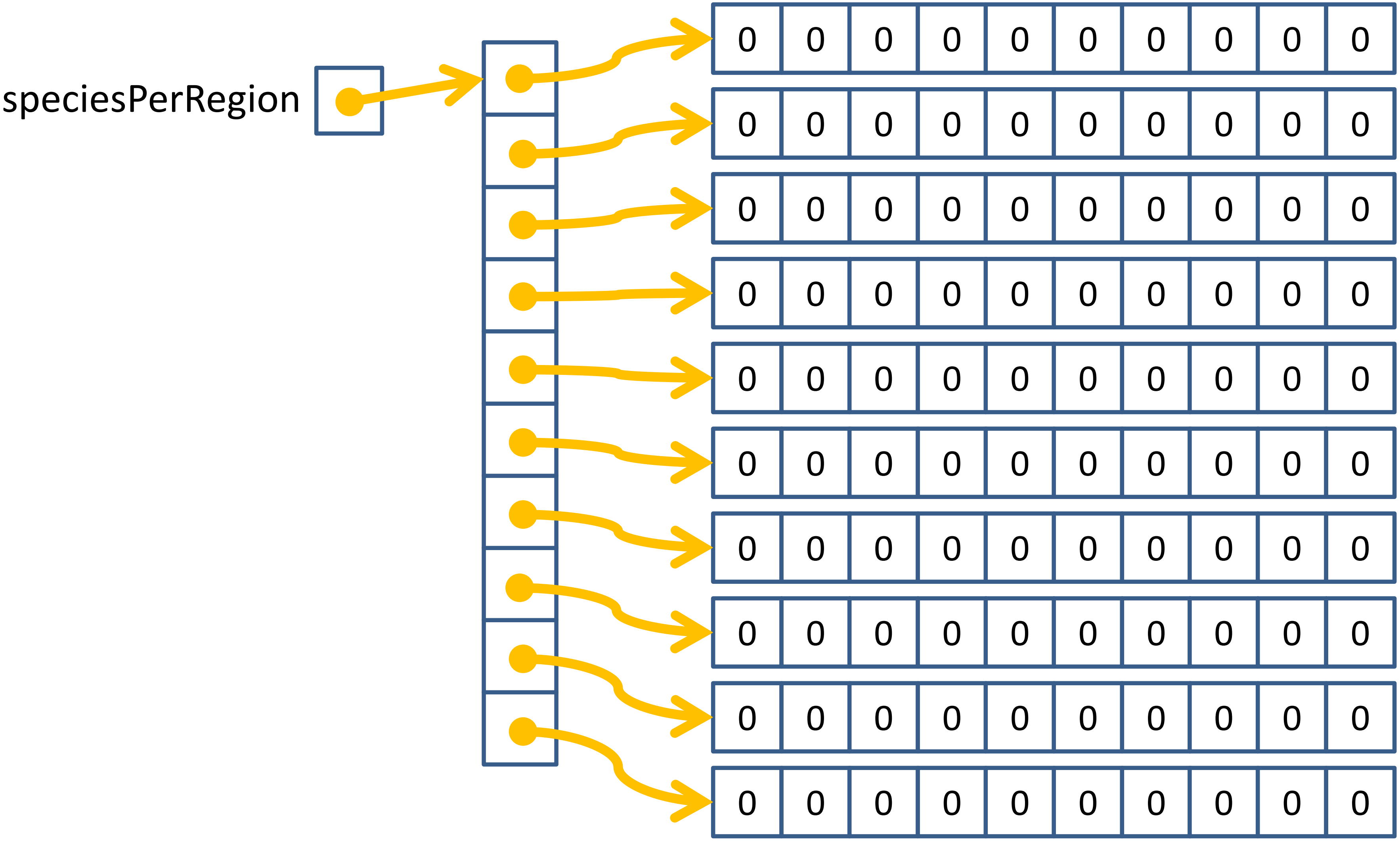
rows

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

speciesPerRegion[9][3]

A 2D array is really a 1D array of arrays

```
final int ROWS = 10, COLS = 10;
int[][] speciesPerRegion = new int[ROWS][COLS];
```



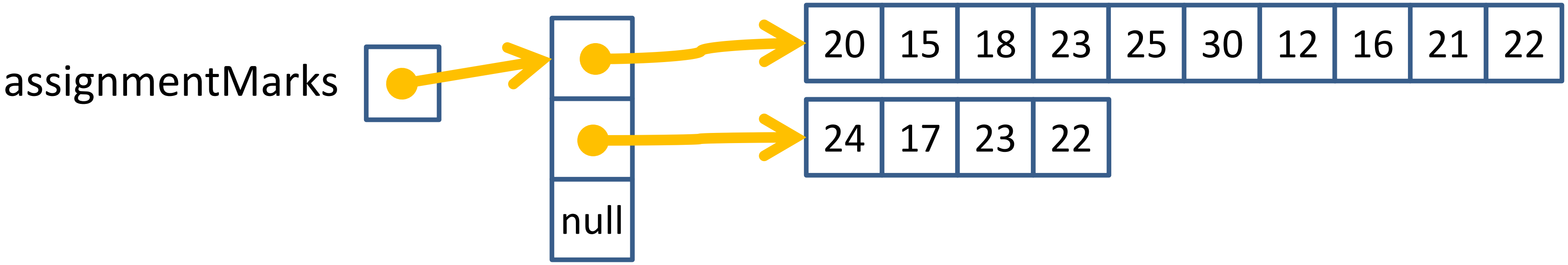
speciesPerRegion

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

An advanced topic: ragged arrays

The rows of a 2D array can instantiated with different lengths

```
int[][] assignmentMarks = new int[3][];
…
assignmentMarks[0] = new int[10];
…
assignmentMarks[1] = new int[4];
```



Tip: ‘ragged’ is pronounced like ra-gid, not rag’d

Displaying a (ragged) 2D array

Use a nested loop for the columns

```
public void display2D(int[][] m) {
    for (int row = 0; row < m.length; row++) {
        for (int col = 0; col < m[row].length; col++) {
            System.out.print(m[row][col] + " ");
        }
        System.out.println(); //adds the newline
    }
}
```