

# 1B1a

## Arrays

1 Copyright © 2003, Graham Roberts

Department of Computer Science



## Arrays

- A normal variable holds 1 value: 

42
----
- An array variable holds a *collection* of values:

10	22	6	43	19	27
----	----	---	----	----	----

2 Copyright © 2003, Graham Roberts

Department of Computer Science



## Naming arrays

- An array has a single name, so the *elements* are numbered or *indexed*.

myArray	10	22	6	43	19	27
	0	1	2	3	4	5

Numbering starts at zero.

3 Copyright © 2003, Graham Roberts

Department of Computer Science



## Indexing

- An array element is accessed using the name and index number, like this:

myArray[0] - 1st element

myArray[3] - 4th element

myArray	10	22	6	43	19	27
	0	1	2	3	4	5

- The square brackets are the *index operator*.

4 Copyright © 2003, Graham Roberts

Department of Computer Science



## Using indexing

- We can fetch the value of an array element:  
`int n = myArray[2];`
- Or assign to an array element:  
`myArray[3] = 10;`

5 Copyright © 2003, Graham Roberts

Department of Computer Science



## Why?

- Arrays allow a whole collection of values of the same type to be stored using *one* variable.
- We don't have to name lots of variables.
- The array can be as big as we like (within limits).
- Elements can be accessed using loops.

6 Copyright © 2003, Graham Roberts

Department of Computer Science



## Example:

- Add up a collection of numbers in array:
 

```
int sum = 0;
for (int n = 0 ; n < 6 ; ++n)
{
    sum += myArray[n];
}
```

7

Copyright © 2003, Graham Roberts

Department of Computer Science



## Declaring an Array

```
int[] myArray = new int[6];
```

Type array of int

Create an array of 6 integers

- Note the two parts:
  - Declare the variable
  - Create the array

8

Copyright © 2003, Graham Roberts

Department of Computer Science



## Size v. Index

```
int[] myArray = new int[6];
```

myArray	10	22	6	43	19	27
	0	1	2	3	4	5

- This gives 6 elements indexed from 0 to 5.

9

Copyright © 2003, Graham Roberts

Department of Computer Science



## More declarations

```
double[] values = new double[100];
String[] name = new String[50];
boolean[] marks = new boolean[5000];
```

```
int n = <some expression>;
long[] numbers = new long[n];
```

10

Copyright © 2003, Graham Roberts

Department of Computer Science



## Initialisation

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

- The variable array is initialised,
- but what about the array elements?
- They are initialised to default values (e.g., 0,0.0, null)

11

Copyright © 2003, Graham Roberts

Department of Computer Science



## Initialising array elements

- Either write a loop and assign to each element,
- or use an array initialisation expression:
 

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

 or
 

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

 (Create an array of size 5, with each element initialised.)

12

Copyright © 2003, Graham Roberts

Department of Computer Science



## Array length

- Arrays are *really objects*.
- And they know their own size:
 

```
int[] n = new int[10];
      ...
      int size = n.length;
      // size == 10
```

13 Copyright © 2003, Graham Roberts

Department of Computer Science



## Safer loops

```
int sum = 0;
for (int n = 0 ; n < myArray.length ; ++n)
{
    sum += myArray[n];
}
```

- No longer need to have a 'magic number'.
- Instead ask array for its size.

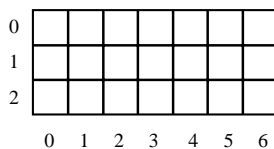
14 Copyright © 2003, Graham Roberts

Department of Computer Science



## 2 dimensions

```
int[][] twoD = new int[3][7];
```



- 3 rows by 7 columns

15 Copyright © 2003, Graham Roberts

Department of Computer Science



## 2D indexing

- Assign to row 1 column 3
 

```
twoD[1][3] = 10;
```

- Fetch row 0 column 6
 

```
int x = twoD[0][6];
```

- (Don't forget we count from zero!)

16 Copyright © 2003, Graham Roberts

Department of Computer Science



## 2D loop

```
int sum = 0;
for (int row = 0 ; row < 3 ; row++)
{
    for (int col = 0 ; col < 7 ; col++)
    {
        sum += twoD[row][col];
    }
}
```

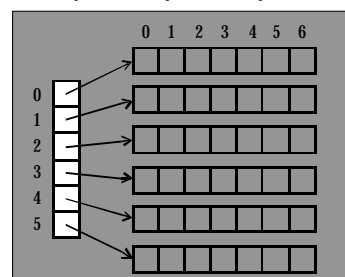
17 Copyright © 2003, Graham Roberts

Department of Computer Science



## 2D Arrays – The Truth!

- A 2D array is really an array of arrays!



18 Copyright © 2003, Graham Roberts

Department of Computer Science



## Array Fun...

```
int[] oned1 = new int[20]; // 1D array
int[][] twod = new int[10][]; // 2D array

twod[0] = oned1; // Add array as row

int i = twod[0][2]; // Can now index

int[] oned2 = new int[50]; // Another 1D array

twod[1] = oned2; // New row different length
twod[1][45] = 10;
int[] n = twod[1]; // Get row from array (slice array)
```

19 Copyright © 2003, Graham Roberts

Department of Computer Science



## N-Dimensional Arrays

- In principle arrays can have as many dimensions as you want:  
`double[][][] d = new double[10][20][30];`
- In practice, rarely need more than 3 dimensions.
- Large arrays also use lots of memory.

20 Copyright © 2003, Graham Roberts

Department of Computer Science



## Array Summary

- Arrays allow collections of values to be stored in a single variable.
- New syntax with square brackets is used.
- Loops are used to work with arrays.

21 Copyright © 2003, Graham Roberts

Department of Computer Science



## Containers

- Arrays are “built-in” to the Java language and supported by special syntax.
- There are also Container classes, providing array-like objects.
  - Also called Collection classes.
- Containers have different properties, variously optimised for convenience, speed and memory use.

22 Copyright © 2003, Graham Roberts

Department of Computer Science



## Containers (2)

- The Java Collections Framework provides various container classes:
  - ArrayList
  - HashMap, HashSet
  - Vector
  - and others

23 Copyright © 2003, Graham Roberts

Department of Computer Science



## ArrayList

```
import java.util.ArrayList;

ArrayList a = new ArrayList();
String s1 = "hello";
String s2 = "world";
a.add(s1);
a.add(s2);
String s = (String)a.get(1);
```

24 Copyright © 2003, Graham Roberts

Department of Computer Science



## ArrayList v. arrays

- Arrays are “manual”, ArrayList is more automated.
- ArrayList allows elements to be added and deleted from any position.
- ArrayLists change size automatically.

25 Copyright © 2003, Graham Roberts

Department of Computer Science



## Summary

- Container classes provide higher level abstractions for dealing with collections.
- But require more knowledge to be used effectively.
- By the end of 1B1b you will now how to construct our own containers.

26 Copyright © 2003, Graham Roberts

Department of Computer Science

