Arrays

Software Development 1 (F27SA)

Michael Lones

Week 4, lecture 1

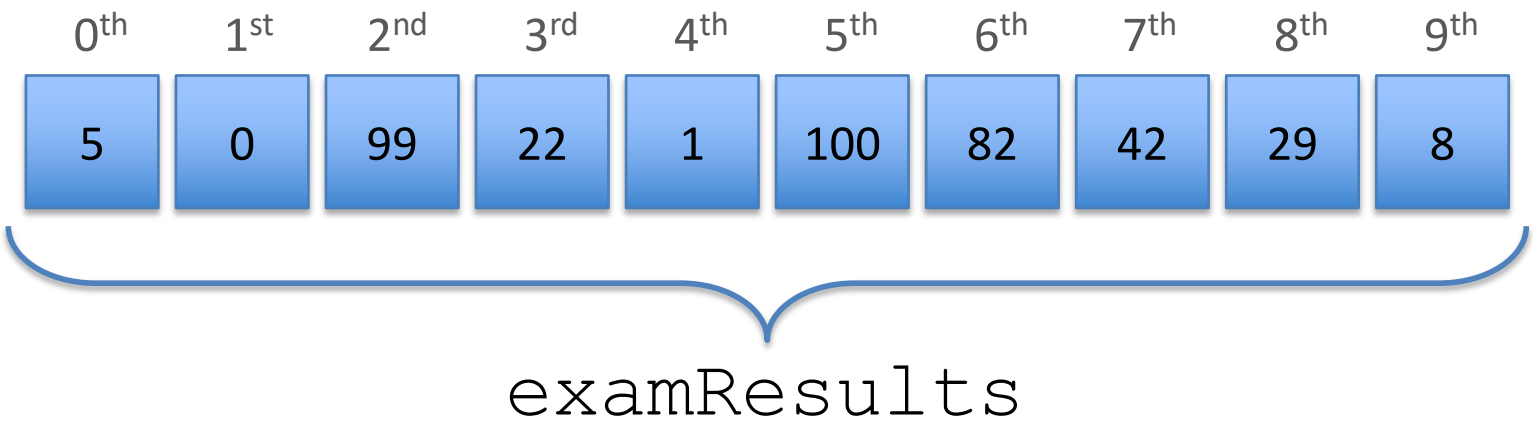
Today’s Lecture

- What is an array?
- Creating and initialising arrays
- Iterating through arrays
- Some examples

What is an array?

An array is an ordered set of variables, all of the same type and sharing the same name

- It's a way of grouping together related things
- They are used extensively in programming
- e.g. an array of integers:



Declaring an array

This is the syntax for declaring an array:

```
type[] name;
```

Some examples:

```
int[] examResults;      // array of ints
String[] words;         // array of Strings
```

However, this does not create the array. It just tells Java that you are going to create one.

Initialising an array

This is the syntax for initialising an array:

```
type[] name = {value1, value2, …};
```

Some examples:

```
int[] examResults = {5, 0, 99, 22};
String[] words = {"I", "like", "cheese"};
```

This creates an array and adds the specified values to it. You would use this syntax when you know in advance the initial values that you want to be in the array.

Creating an array

However, often you do not know the values that will be in the array, just how many there are.

This is the syntax for creating such an array:

```
type[] name = new type[length];
```

Some examples:

```
int[] examResults = new int[10];
String[] words = new String[20];
```

Creating an array

However, often you do not know the values that will be in the array, just how many there are.

This is the syntax for creating such an array:

```
type[] name = new type[length];
```

The `new` keyword allocates a block of memory in which to store something. You'll learn more about this in the second part of SD2. For now, just use it and don't worry about what it means.

Creating an array

However, often you do not know the values that will be in the array, just how many there are.

This is the syntax for creating such an array:

```
type[] name = new type[length];
```

The array will initially be filled with the default value for the specified type. For numeric types, this is 0. For booleans, it is false. For object types, it is "null".

Setting values

Once you've created an array, you can then **set** or modify the value of each element:

```
name[index] = value;
```

An example:

```
int[] examResults = new int[3];
examResults[0] = 5;
examResults[1] = 0;
examResults[2] = 99;
```

Setting values

Once you've created an array, you can then **set** or modify the value of each element:

```
name[index] = value;
```

An example:

```
int[] examResults = new int[3];
examResults[0] = 5;
examResults[1] = 0;
examResults[2] = 99;
```

The indices are numbered from 0 to length-1

Setting values

Once you've created an array, you can then **set** or modify the value of each element:

```
name[index] = value;
```

An example:

```
int[] examResults = new int[3];
examResults[0] = 5;
examResults[1] = 0;
examResults[2] = 99;

int[] examResults = {5, 0, 99};
```

These two program fragments produce the same array

Any Questions?

Simple examples

Initialise and then output an array:

ArrayDemo1.java

```
public class ArrayDemo1 {
   public static void main(String[] args) {
      int[] scores = {5,7,5,8,2,8,0,4,6,7};
      System.out.println("Lab 2 scores:");
      for(int i=0; i<10; i++) {
         System.out.print(scores[i] + " ");
      }
   }
}
```

Terminal

```
$ java ArrayDemo1
Lab 2 scores:
5 7 5 8 2 8 0 4 6 7
```

Simple examples

Initialise and then output an array:

ArrayDemo1.java

```
public class ArrayDemo1 {
   public static void main(String[] args) {
      int[] scores = {5,7,5,8,2,8,0,4,6,7};
      System.out.println("Lab 2 scores:");
      for(int i=0; i<scores.length; i++) {
         System.out.print(scores[i] + " ");
      }
   }
}
```

This is how you find the length of an existing array.
It is better to do this than hard-code the number.

Terminal

```
$ java ArrayDemo1
Lab 2 scores:
5 7 5 8 2 8 0 4 6 7
```

Simple examples

Declare empty array, then fill using user input:

ArrayDemo2.java

```
public class ArrayDemo2 {
    public static void main(String[] args) {
        int[] scores = new int[10]; // lab scores
        Scanner scan = new Scanner(System.in);

        System.out.println("Please input scores:");
        for(int i=0; i<scores.length; i++)
            scores[i] = scan.nextInt();

        System.out.println("Lab 2 scores:");
        for(int i=0; i<scores.length; i++) {
            System.out.print(scores[i] + " ");
        }
    }
}
```

Simple examples

Calculate the sum and average of the array:

ArrayDemo3.java

```
public class ArrayDemo3 {
    public static void main(String[] args) {
        int[] scores = new int[10]; // lab scores
        int sum = 0; // sum of scores
        Scanner scan = new Scanner(System.in);

        System.out.println("Please input scores:");
        for(int i=0; i<scores.length; i++)
            scores[i] = scan.nextInt();

        for(int i=0; i<scores.length; i++) // calculate sum
            sum += scores[i];

        // calculate and display average
        System.out.println("Average:"+ (sum/scores.length));
    }
}
```

“for each” syntactic sugar

You can also iterate through an array using:

```
for(int name : array)
```

So, in the previous example, we can rewrite

```
for(int i=0; i<scores.length; i++)
    sum += scores[i];
```

as:

```
for(int score : scores)
    sum += score;
```

“for each” syntactic sugar

You can also iterate through an array using:

```
for(int name : array)
```

So, in the previous example, we can rewrite

```
for(int i=0; i<scores.length; i++)
    sum += scores[i];
```

as:

```
for(int score : scores)
    sum += score;
```

You can give this part (circled in red) any name you like. It will behave as a variable, containing the value of each subsequent element of the array on each subsequent iteration of the loop

“for each” syntactic sugar

You can also iterate through an array using:

```
for(int name : array)
```

So, in the previous example, we can rewrite

```
for(int i=0; i<scores.length; i++)
    sum += scores[i];
```

as:

```
for(int score : scores)
    sum += score;
```

This can only be used when reading values from the array, not when setting values.

Any Questions?

A larger example

LabScoreAnalysis.java

```
/*
 *  This program analyses student lab scores.
 *  It calculates the lowest, highest and mean scores.
 */
public class LabScoreAnalysis {
   public static void main(String[] args) {
      double[] scores; // lab scores
      int students;    // number of students
      double lowest;   // lowest score
      double highest;  // highest score
      double mean;     // mean score

      // first, find out how many students there are
      System.out.println("How many students?");
      Scanner scan = new Scanner(System.in);
      students = scan.nextInt();

      // make the scores array the correct size
      scores = new double[students];
```

A larger example

LabScoreAnalysis.java

```
/*
 *  This program analyses student lab scores.
 *  It calculates the lowest, highest and mean scores.
 */
public class LabScoreAnalysis {
   public static void main(String[] args) {
      double[] scores; // lab scores
      int students;    // number of students
      double lowest;   // lowest score
      double highest;  // highest score
      double mean;     // mean score

      // first, find out how many students there are
      System.out.println("How many students?");
      Scanner scan = new Scanner(System.in);
      students = scan.nextInt();

      // make the scores array the correct size
      scores = new double[students];
```

Note that the array length can be specified by a variable value

A larger example

```
// read scores from user
System.out.println("Please input scores:");
for(int i=0; i<scores.length; i++)
    scores[i] = scan.nextInt();

// find lowest score
lowest = Double.POSITIVE_INFINITY;
for(double score : scores)
    if(score<lowest)
        lowest = score;

// find highest score
highest = Double.NEGATIVE_INFINITY;
for(double score : scores)
    if(score>highest)
        highest = score;
```

A larger example

```
// read scores from user
System.out.println("Please input scores:");
for(int i=0; i<scores.length; i++)
    scores[i] = scan.nextInt();

// find lowest score
lowest = Double.POSITIVE_INFINITY;
for(double score : scores)
    if(score<lowest)
        lowest = score;

// find highest score
highest = Double.NEGATIVE_INFINITY;
for(double score : scores)
    if(score>highest)
        highest = score;
```

Here we are setting the values of array members, so we can **not** use "for each" syntax

A larger example

```
// read scores from user
System.out.println("Please input scores:");
for(int i=0; i<scores.length; i++)
    scores[i] = scan.nextInt();

// find lowest score
lowest = Double.POSITIVE_INFINITY;
for(double score : scores)
    if(score<lowest)
        lowest = score;

// find highest score
highest = Double.NEGATIVE_INFINITY;
for(double score : scores)
    if(score>highest)
        highest = score;
```

Here we are only reading values of array members, so we **can** use "for each" syntax

A larger example

```
// read scores from user
System.out.println("Please input scores:");
for(int i=0; i<scores.length; i++)
    scores[i] = scan.nextInt();

// find lowest score
lowest = Double.POSITIVE_INFINITY;
for(double score : scores)
    if(score<lowest)
        lowest = score;

// find highest score
highest = Double.NEGATIVE_INFINITY;
for(double score : scores)
    if(score>highest)
        highest = score;
```

This is how you specify ∞ and -∞ in Java

A larger example

```
        // find mean score
        mean = 0;
        for(double score : scores)
            mean += score;
        mean /= students;

        // output info
        System.out.println("The scores were between"
                           + lowest + " and " + highest
                           + " with a mean of " + mean);
    }
}
```

Terminal

```
$ java LabScoreAnalysis
How many students?
5
Please input scores:
1 2 3 4 5
The scores were between 1.0 and 5.0 with a mean of 3.0
```

Historical Note

You may also come across array declarations like this:

```
int myArray[];
```

This form (with the [] after the name) was inherited from C, but is discouraged in Java. So, stick to:

```
int[] myArray;
```

However, you're likely to find old code, or code by older coders, that uses the C version.

Any Questions?

Some exercises

DoubleArray.java

```
/* Reverse an input array */
public class DoubleArray {
   public static void main(String[] args) {
      int[] input = {1,2,3,4,5,6,7,8,9,10};
      int[] output = new int[input.length];

      // what goes here?

      for(int o : output)
         System.out.print(o + " ");
      System.out.println();
   }
}
```

Terminal

```
$ java DoubleArray
2 4 6 8 10 12 14 16 18 20
```

Some exercises

SumArrays.java

```
/* Join two arrays together */
public class SumArrays {
    public static void main(String[] args) {
        int[] in1 = {1,2,3,4,5,6,7,8,9,10};
        int[] in2 = {1,1,1,1,1,0,0,0,0,0};
        int[] output = new int[in1.length];

        // what goes here?

        for(int o : output)
            System.out.print(o + " ");
        System.out.println();
    }
}
```

Terminal

```
$ java SumArrays
2 3 4 5 6 6 7 8 9 10
```

Some exercises

ArrayReversal.java

```
/* Reverse an input array */
public class ArrayReversal {
   public static void main(String[] args) {
      int[] input = {1,2,3,4,5,6,7,8,9,10};
      int[] output = new int[input.length];

      // what goes here?

      for(int o : output)
         System.out.print(o + " ");
      System.out.println();
   }
}
```

Terminal

```
$ java ArrayReversal
10 9 8 7 6 5 4 3 2 1
```

Some exercises

RunningTotal.java

```
/* Running total of input values */
public class RunningTotal {
    public static void main(String[] args) {
        int[] input = {1,2,3,4,5,6,7,8,9,10};
        int[] output = new int[input.length];

        // what goes here?

        for(int o : output)
            System.out.print(o + " ");
        System.out.println();
    }
}
```

Terminal

```
$ java RunningTotal
1 3 6 10 15 21 28 36 45 55
```

1+2=3

1+2+3+4+5+6+7+8+9+10=55

See separate file on Vision for solution slides

Summary

- Arrays are sets of variables with a single name
- Array **indices** are used to access each array **member**
- Indices are numbered from 0 to (length-1)
- Arrays can have any number of **dimensions**
- An array must be created before use, either using the `new` keyword or `{…}` if the values are known
- Its length can be found using *`array`*`.length`
- Sometimes the "for each" syntax can be used to loop through its values: `for(`*`type name`*`: `*`array`*`)`

Next Lecture

- Sub-programs
  - Also known as methods, functions, procedures and subroutines in different programming languages