Arrays

Programming Methodology and Design

Dr Ioanna Stamatopoulou

Arrays

- Declaration
- Initialisation
- Use of Arrays
- Arrays and Strings
- Arrays as method arguments

Programming Methodology & Design - 06 - Arrays

The need for Arrays

- The problem: we want to store 100 integers
 - Declaring 100 integer variables is not really practical...

```
int num1, num2, num3, ..., num100 ;
```

■ Neither is initialising them and using them:

```
num1 = 0 ;
...
num100 = 0 ;
```

Arrays solve this problem:

```
int num[] = new int[100];
for (int i = 0 ; i < 100 ; i++)
    num[i] = i ;

for (int i = 0; i < num.length; i++)
    System.out.println(num[i]);</pre>
```

Programming Methodology & Design - 06 - Arrays

3

Array Declaration

To declare an array I declare its type followed by a pair of square brackets and its name:

```
String[] stringArray; //an array for Strings
int[] integerArray; //an array for integers
```

- Declaring an array does not reserve memory space and does initialise its values
 - ☐ If you attempt to use an array immediately after declaring it you will get a compiler error: Variable <array name> may not have been initialized

Programming Methodology & Design - 06 - Arrays

ŀ

Array Creation and Initialization

To create an array you use the following syntax and also declare its length (size/number of positions):

```
stringArray = new String[10]; //an array for 10 Strings
integerArray = new int[5]; //an array for 5 integers
```

Declaration and creation in one step:

```
char[] alphabet = new char[26];
double[] dnumbers = new double[10];
```

 Creation and initialisation (when all array elements are known at the time of creation)

```
int[] numbers = {10, 25, 43, 14, 5};
```

Programming Methodology & Design - 06 - Arrays

5

Array indices

- Every position inside an array is identified by its index
- Index numbering starts at 0
 - ☐ So the last index of any array is always: length-1

45	65	87	999	651	56	1	2369	444	78
0	1	2	3	4	5				length-1

Programming Methodology & Design - 06 - Arrays

Value assignment

To assign/store a value inside the array we have to specify the exact position where it is to be stored:

```
stringArray[0] = "petros";
numberArray[3] = 12;
alphabet[1] = 'b';
dnumbers[2] = 10.2;
```

Programming Methodology & Design - 06 - Arrays

7

Accessing stored values

An array element is accessed by specifying the name of the array and the index (position) of the particular element:

```
//print the 4<sup>th</sup> element
System.out.println(integerArray[3]);

//print the 1<sup>st</sup> element
System.out.println("Element in 1st position: " + integerArray[0]);

//add the 4<sup>th</sup> and 6<sup>th</sup> elements and store result in x
int x = integerArray[3] + integerArray[5];
```

Programming Methodology & Design - 06 - Arrays

Going through an entire array

- To do something with all elements of an array we typically use a for loop, with I representing the index of the array
- e.g. to prompt the user to fill the array:

```
for (int i = 0; i < numbers.length; i++) {
   System.out.println("Enter value " + i);
   numbers[i] = scanner.nextInt();
}</pre>
```

To go through the entire array always start at 0 and go up to less than length

e.g. to print all elements:

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

Programming Methodology & Design - 06 - Arrays

9

char arrays → strings

A character array may be used to initialise a string:

```
char[] name = {'J', 'a', 'v', 'a' };
String s = new String(name);
//s will get the value "Java"
```

Programming Methodology & Design - 06 - Arrays

.0

Disadvantage

- Biggest disadvantage of arrays is that they cannot be resized according to our needs
- We will see in the near future what alternatives we have to address this problem

Programming Methodology & Design - 06 - Arrays

11

Additional issues related to Arrays

Arrays and Methods

Passing arrays to methods

A method can be declared to have an array parameter(s):

```
public static void printArray (int[] array){
   for (int i = 0; i < array.length; i++){
      System.out.println(array[i]);
}</pre>
```

When the method is called ONLY the array name is passed as an argument (without the brackets):

```
int[] numbers = new int[10];
...
printArray(numbers); //calling the printArray method
```

Programming Methodology & Design - 06 - Arrays

13

Methods returning arrays

A method may also be declared to return an array:

```
public static int[] promptForAllGrades(int numberOfStudents){
    Scanner scanner = new Scanner(System.in);
    int[] allGrades = new int[numberOfStudents];

    for (int i = 0; i < allGrades.length; i++){
        System.out.println("Enter the grade of student " + i);
        allGrades[i] = scanner.nextInt();
    }

    scanner.close();
    return allGrades;
}</pre>
```

Programming Methodology & Design - 06 - Arrays

Additional issues related to Arrays

Multidimensional Arrays

Multidimensional Arrays

- A 2-dimensional array is one that contains rows and columns
- Declaring and creating a 2d array:

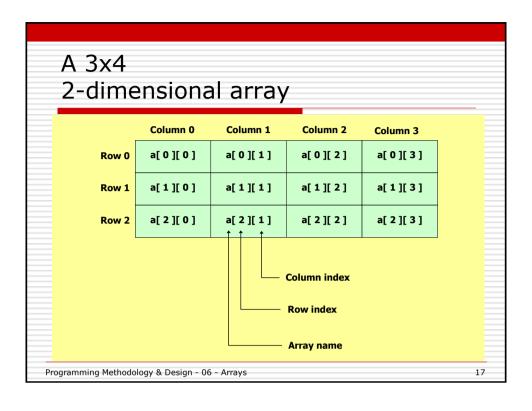
```
//an array with 2 rows and 4 columns
int a[][] = new int[3][4];
```

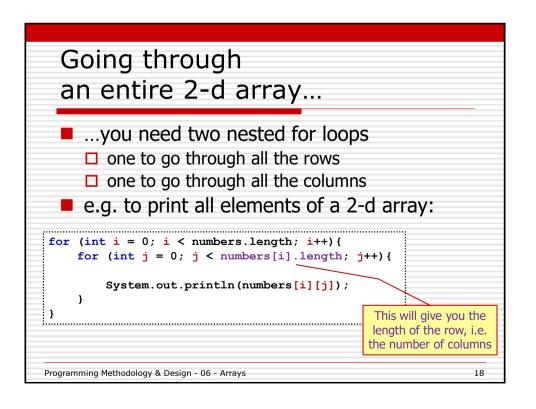
Declaring and initialising a 2d array:

```
int array[][] = {{10, 20}, {30, 40}};
```

- □ 10 and 20 are the values of row 0: array[0][0] and array[0][1]
- □ 30 and 40 are the values of row 1: array[1][0] and array[1][1]

Programming Methodology & Design - 06 - Arrays





Multidimensional Arrays (cont'd)

 Mind that you can create an array in which every row has a different number of columns (cells)

```
//1st way:
int b1[][] = new int[2][]; // b1 has 2 rows
b1[0] = new int[5]; // row 0 of b1 has 5 columns
b1[1] = new int[3]; // row 1 of b1 has 3 columns

// OR 2nd way:
int b2[][] = {{ 1, 2, 3, 4, 5 }, { 9, 10, 11 }};
```

Programming Methodology & Design - 06 - Arrays

10

(More) Advanced issues

References and Reference Parameters

References and Reference Parameters

- Generally, there are two ways to pass arguments to methods
 - □ Pass-by-value
 - Copy of argument's value is passed to called method
 - □ Pass-by-reference
 - Caller gives called method direct access to caller's data
 - ☐ Called method can directly manipulate this data

Programming Methodology & Design - 06 - Arrays

21

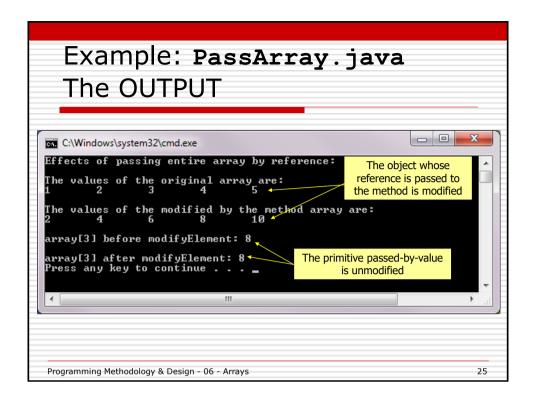
References and Reference Parameters - JAVA

- In Java, everything is pass-by-value, but:
 - ☐ Unlike a primitive data type variable's name, an object name (in our case an array name) is a reference to the object (i.e. its address)
 - ☐ Since object references are passed by value, it means that the data that the reference points to can be changed by a method, but not the reference itself

Programming Methodology & Design - 06 - Arrays

```
Example:
                                              Download
                                          PassArray. java and
  PassArray.java
                                               try it out!
  public class PassArray {
                                                  Declare 5-int array with
    public static void main(String[] args)
      int[] array = {1, 2, 3, 4, 5}; 
      String output = "Effects of passing entire array by
         reference: \n\nThe values of the original array
         are:\n";
      // append original array elements to String output
      for (int i = 0; i < array.length; i++)</pre>
          output += array[i] + "\t";
                                              Pass array reference by value
                                                to method modifyArray
      modifyArray(array); // array passed by reference
      output += "\n\nThe values of the modified array are:\n";
      for (int i = 0; i < array.length; i++)</pre>
          output += array[i] + "\t";
Programming Methodology & Design - 06 - Arrays
```

```
Pass array[3] (int primitive
   Example (cont'd):
                                                  data type) by value to
                                                  method modifyElement
   PassArray.java
                                                   The original primitive is
                                                      left unmodified
    // attempt to modify array[3/]
    output += "\n\narray[3] before modifyElement:
                                                       + array[3];
    modifyElement(array[3]);
    output += "\n\narray[3] after modifyElement: " + array[3];
    System.out.println(output);
  } //end of main method
  // METHOD: multiply each element of an array by 2
 public static void modifyArray(int[] array) {
     for (int i = 0; i < array.length; i++)</pre>
        array[i] *= 2;
                                                   Method modifyArray
                                                  receives array reference
 // METHOD: multiply a number by 2
 public static void modifyElement(int element) {
     element *= 2;
                                    Method modifyElement
} // end class PassArray
                                   receives a primitive's copy
.....
 Programming Methodology & Design - 06 - Arrays
```



```
swap Method

Swapping the position of two values in an array

// method that swaps two elements of an array
static void swap(int[] array, int first, int second)
{
  int temp; // temporary holding area for swap

  temp = array[first];
  array[first] = array[second];
  array[second] = temp;
}

Programming Methodology & Design - 06 - Arrays
```



Check list

- Describe a situation in which you need an array of values
- Can I declare, initialise, and use an array?
- Can I use a for loop to go through an entire array?
- What is biggest disadvantage of arrays?

Programming Methodology & Design - 06 - Arrays

27

Check list (cont'd)



- Do I understand what happens when I pass a primitive data type as an argument and what happens when I pass an object (non-primitive data type) as an argument?
- Can a method modify the contents of a passed array?
- Can I declare, initialise, and use multidimensional arrays?

Programming Methodology & Design - 06 - Arrays