Array Recap and Lab Solutions

Produced
by:
Ms. Mairead Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Topics list

- RECAP of **Arrays**

- Topic09 - Lab Solutions

- **Length** Property

Arrays (fixed-size collections)

- Arrays are a way to collect associated values.

- Programming languages usually offer a special fixed-size collection type: an *array*.

- Java arrays can store
  - objects
  - primitive-type values.

- Arrays use a special syntax.
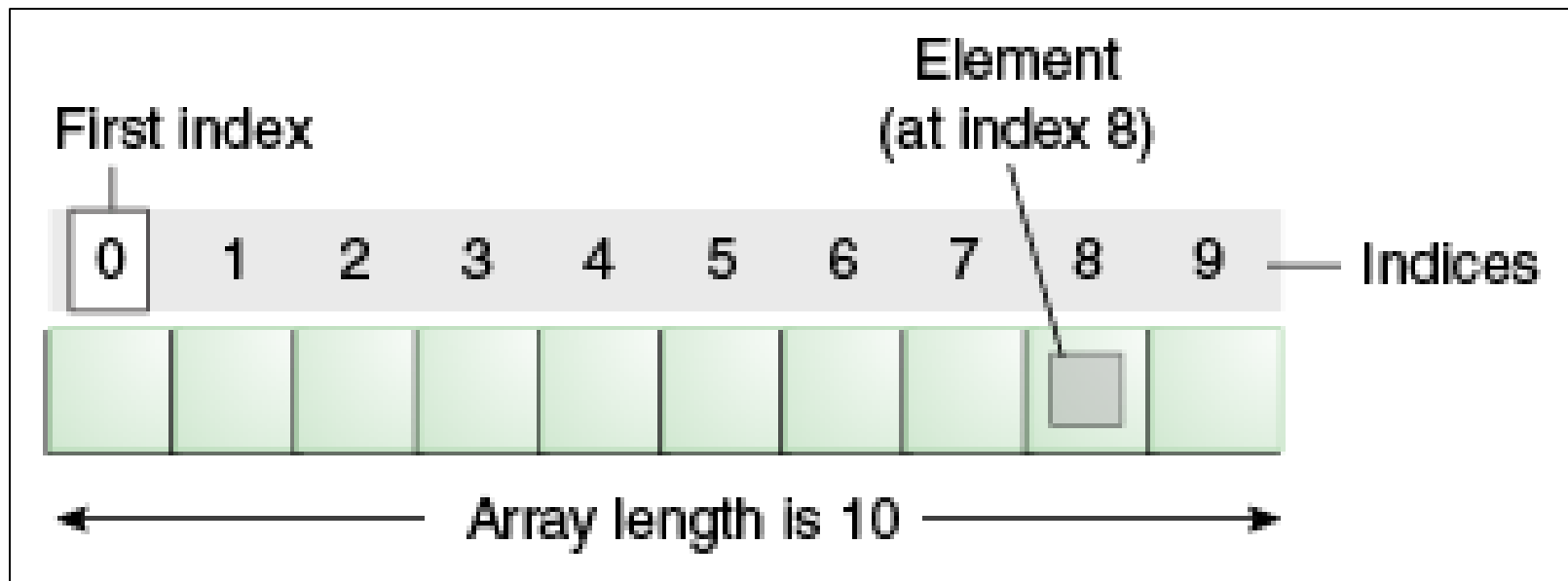
Primitive types

Primitive type

`int num = 17;`

Directly stored
in memory…

17

- We are now going to look at a **structure** that can **store many values** of the **same type**.

- Imagine a structure made up of sub-divisions or sections…

- Such a structure is called an **array** and would look like:

Structure of a primitive array
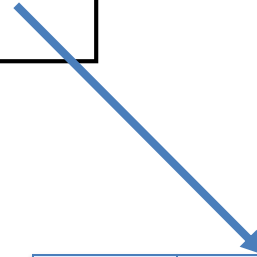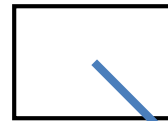
Structure of a primitive array

int[] numbers;

numbers

null

Structure of a primitive array

int[] numbers;

numbers = new int[4];

numbers

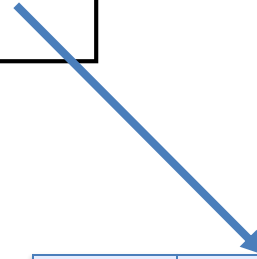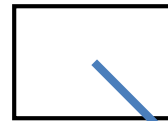| 0	0
1	0
2	0
3	0

Structure of a primitive array

`int[]  numbers;`

numbers = new int[4];

We have declared an array of int, with a capacity of four.

Each element is of type **int**.
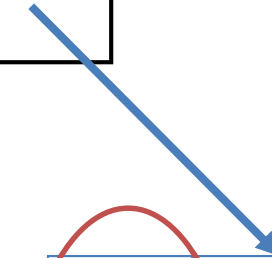
The array is called **numbers**.

numbers

0	**0**
1	**0**
2	**0**
3	**0**

Structure of a primitive array

int[] numbers;

numbers = new int[4];

numbers

0	**0**
1	**0**
2	**0**
3	**0**
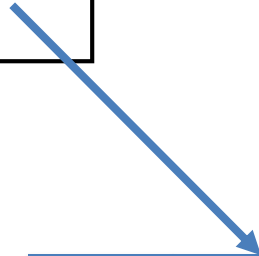
Index of each element in the array

Structure of a primitive array

`int[]  numbers;`

numbers = new int[4];
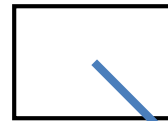
numbers

0	**0**
1	**0**
2	**0**
3	**0**

Default value for each element of type **int**.

Structure of a primitive array
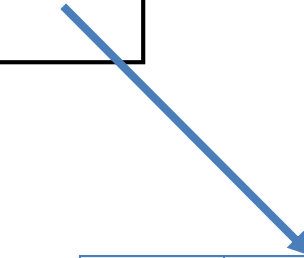
int[] numbers;

numbers = new int[4];

numbers[2] = 18;

We are directly accessing the element at index **2** and setting it to a value of **18**.

numbers

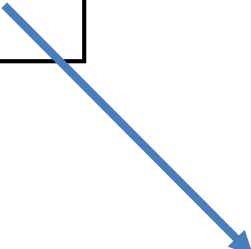|	
0	0
1	0
2	18
3	0

Structure of a primitive array

int[] numbers;

numbers = new int[4];

numbers[2] = 18;

numbers[0] = 12;

We are setting the element at index **0** and to a value of **12**.

numbers

0	12
1	0
2	18
3	0

Structure of a primitive array
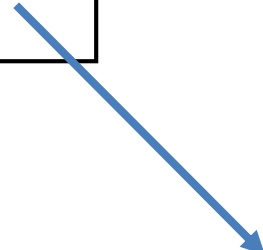
int[] numbers;

numbers = new int[4];

numbers[2] = 18;

numbers[0] = 12;

print(numbers[2]);

numbers
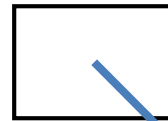
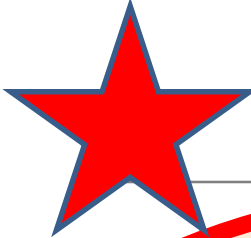| 0	12
1	0
2	18
3	0

Here we are printing the contents of index location 2
i.e. 18 will be printed to the console.
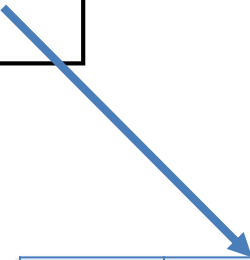
Declaring a primitive array

```
int[]  numbers;

//somecode

numbers = new int[4];
```

This is how we previously declared our array of four **int**, called **numbers**.

numbers
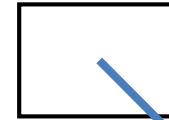
0	0
1	0
2	0
3	0
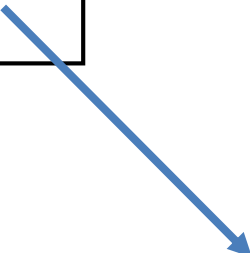
Declaring a primitive array

```
int[]  numbers;

//somecode

numbers = new int[4];
```

We can also declare it like this…

int[] numbers = new int[4];

numbers
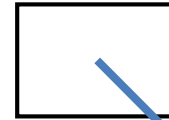
0	0
1	0
2	0
3	0
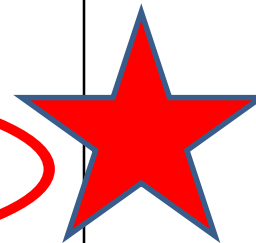
An array can store ANY TYPE of data.

Primitive Types

```java
int numbers[] = new int[10];

byte smallNumbers[] = new byte[4];

char characters[] = new char[26];
```

Object Types

```java
String words = new String[30];

Spot spots[] = new Spot[20];
```

Summary - Arrays

- Arrays are structures that can store many values of the same type
- Rule – Never lose input data
  - Arrays enable us to store the data efficiently
  - We can use loops with arrays
- Arrays can store ANY type
- Declaring arrays

```
int[]  arryName;
//somecode
arryName= new int[4];
```

OR

```
int[]  arryName= new int[4];
```

Topics list

- Recap of Arrays

- Topic09 - Lab Solutions

- Length Property

Exercise 1 – what's required?

- Write a program to **declare and construct an int array** (called numbers) of **size 10**.

- **Initialise** the array by putting 20 in each of the elements of the array.

- **Print out** the values to the console (each value should be printed to a new line).

```
Number 1 is: 20
Number 2 is: 20
Number 3 is: 20
Number 4 is: 20
Number 5 is: 20
Number 6 is: 20
Number 7 is: 20
Number 8 is: 20
Number 9 is: 20
Number 10 is: 20
```

Exercise 1 – solution

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

// initialise each element to 20.
for (int i = 0; i < 10 ; i ++)   {
  numbers[i] = 20;
}


// now we print each value
for (int i = 0; i < 10 ; i ++)    {
     println("Number " + (i+1) + " is: " + numbers[i]);
}
```

```
Number 1 is: 20
Number 2 is: 20
Number 3 is: 20
Number 4 is: 20
Number 5 is: 20
Number 6 is: 20
Number 7 is: 20
Number 8 is: 20
Number 9 is: 20
Number 10 is: 20
```

Exercise 2 – what's required?

- Write a program to **declare and construct an int array (called numbers) of size 5**.

- **Read in** 5 values and store them in the array.

- **Print out** the values to the console (one line at a time) in the **reverse** order to the order they were entered in. For example, if we entered 3, 4, 5, 6 and 7, the output should be:

```
Number 5 is: 7
Number 4 is: 6
Number 3 is: 5
Number 2 is: 4
Number 1 is: 3
```
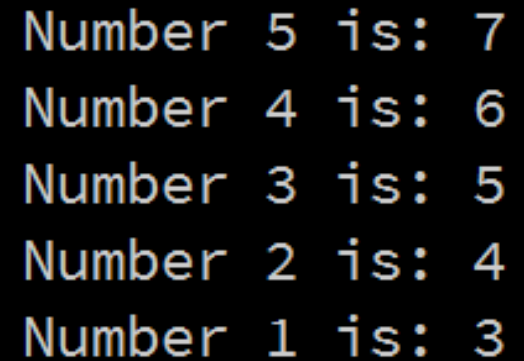
Exercise 2 – solution

```java
import javax.swing.*;

int numbers[] = new int[5];

//populate the array with user input
for (int i = 0; i < 5 ; i ++)  {
   numbers[i] = Integer.parseInt(
     JOptionPane.showInputDialog(
       "Please enter a number ", "3"));
}
// print each value in reverse order
for (int i = 4; i >= 0 ; i --)    {
      println("Number " + (i+1) + " is: " + numbers[i]);
}
```

```
Number 5 is: 7
Number 4 is: 6
Number 3 is: 5
Number 2 is: 4
Number 1 is: 3
```

Exercise 3 – what's required?

- Write a program to **declare and construct an int array** (called numbers) with the **size determined by the user**.

- **Read in** a value for each element in the array and store it.

- Use a for loop to print out **every second value** stored in the array to the console.

  For example, if we choose to enter 8 numbers and then enter the following numbers: 5, 6, 7, 8, 9, 10, 11, 12, we should expect our output to be:

```
Number 1 is: 5
Number 3 is: 7
Number 5 is: 9
Number 7 is: 11
```

Exercise 3 – solution

```
Number 1 is: 5
Number 3 is: 7
Number 5 is: 9
Number 7 is: 11
```

```java
import javax.swing.*;

int numbers[];
int numData = Integer.parseInt(
        JOptionPane.showInputDialog("How many values do you wish to
sum? ", "3"));

//now, use this value to make the array this size.
numbers = new int[numData];

for (int i = 0; i < numData ; i ++)  {
  numbers[i] = Integer.parseInt(
        JOptionPane.showInputDialog("Please enter a number ", "3"));
}

// print out every second value
for (int i = 0; i < numData ; i=i+2)    {
    println("Number " + (i+1) + " is: " + numbers[i]);
}
```

Exercise 4 – what's required?

- Write a program to declare and construct an **int array** (called numbers) with the **size determined by the user**.

- **Read in** a value for each element in the array and store it.

- Print out only the **even numbers** stored in the array to the console (hint: use the **% operator**).

  For example, if we choose to enter 6 numbers and then enter the following numbers: 6, 7, 8, 10, 11, 12, we should expect our output to be:

```
Number 1 is: 6
Number 3 is: 8
Number 4 is: 10
Number 6 is: 12
```

Exercise 4 – solution

```java
import javax.swing.*;

int numbers[];
int numData = Integer.parseInt(JOptionPane.showInputDialog(
            "How many values do you wish to sum? ", "3"));

//now, use this value to make the array this size.
numbers = new int[numData];

for (int i = 0; i < numData ; i ++)  {
  numbers[i] = Integer.parseInt(JOptionPane.showInputDialog(
            "Please enter a number ", "3"));
}

// print out only even numbers
for (int i = 0; i < numData ; i++)    {
    if (numbers[i] % 2 == 0){
        println("Number " + (i+1) + " is: " + numbers[i]);
    }
}
```

```
Number 1 is: 6
Number 3 is: 8
Number 4 is: 10
Number 6 is: 12
```

Topics list

- Recap of Arrays

- 5a - Lab Solutions

- **Length** Property

Returning to Exercise 1

```
Number 1 is: 20
Number 2 is: 20
Number 3 is: 20
Number 4 is: 20
Number 5 is: 20
Number 6 is: 20
Number 7 is: 20
Number 8 is: 20
Number 9 is: 20
Number 10 is: 20
```

We:

- declared an int array (called numbers) of **size 10**.

- initialised the array by putting 20 in each of the elements of the array.

- Printed out the values to the console.

Exercise 1 – solution

```
Number 1 is: 20
Number 2 is: 20
Number 3 is: 20
Number 4 is: 20
Number 5 is: 20
Number 6 is: 20
Number 7 is: 20
Number 8 is: 20
Number 9 is: 20
Number 10 is: 20
```

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

// initialise each element to 20.
for (int i = 0; i < 10 ; i ++)  {
  numbers[i] = 20;
}


// now we print each value
for (int i = 0; i < 10 ; i ++)  {
    println("Number " + (i+1) + " is: " + numbers[i]);
}
```

Q: What changes do we have to make to process 15 elements?

A: We need to change the code in 3 places!!!

There a better way…

length Property

- We will use the **length** property of an array.

```
int numbers[] = new int[15];

// initialise each element to 20.
for (int i = 0; i < numbers.length ; i ++)  {
  numbers[i] = 20;
}

// now we print each value
for (int i = 0; i < numbers.length; i ++)  {
    println("Number " + (i+1) + " is: " + numbers[i]);
}
```

Instead of hard coding the number of elements in the array,
we will use **numbers.length** in place of it.
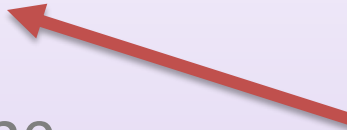
length Property

- We will use the **length** property of an array.

```
int numbers[] = new int[30];

// initialise each element to 20.
for (int i = 0; i < numbers.length ; i ++)  {
  numbers[i] = 20;
}

// now we print each value
for (int i = 0; i < numbers.length; i ++)  {
    println("Number " + (i+1) + " is: " + numbers[i]);
}
```

Then, if we need to change the number of elements, we can simply change it in the declaration and the for loops will still work!

Questions?