# Arrays

Introduction to Programming

1

## Arrays

- There are times when a programmer wants to work with groups or sequences of values
- For example, a programmer might want to store several exam results, or several student names.
- Arrays are used to store a number of values of the same type.
- Arrays allow us to manipulate a group of values at the same time.

2

## Seven temperature readings

```
public class TemperatureReadings
{
      public static void main(String [] args)
      {
         double temp1,temp2,temp3,temp4,
                temp5,temp6,temp7;

         // more code will go here

      }
}
```

3

## Read in temps for 7 days

```
System.out.println("Temperature for day 1 ?");
temp1 = keyboardIn.nextDouble( );

System.out.println("Temperature for day 2 ?");
temp2 = keyboardIn.nextDouble( );
...
...
System.out.println("Temperature for day 7 ?");
temp7 = keyboardIn.nextDouble( );
```

4

## Using a loop?

```
for (int i=1; i<=7; i++)
{
   System.out.println("Temperature for day " + i);
   temp1 = keyboardIn.nextDouble( );
}
```

**What is the problem here?**

5

## Arrays

- An ***array*** is a sequence of values of the same type
- The data values stored in an array are called the array ***elements***.
- The variables that form an array always have the same type called the ***base type***
- Like variables, arrays are given a *name.* We can refer to the entire array by this *name* or to an individual *element* by appending a number in square brackets to the name

6

# Declaring arrays

- Array declaration:

```
arrayType[] arrayName = new arrayType[length];
```

```
// declare an array of 8 ints
int[] ages = new int[8];

// declare an array of 10 doubles
double[] data = new double[10];

// declare an array of 6 Strings
String[] subjectTitles = new String[6];
```

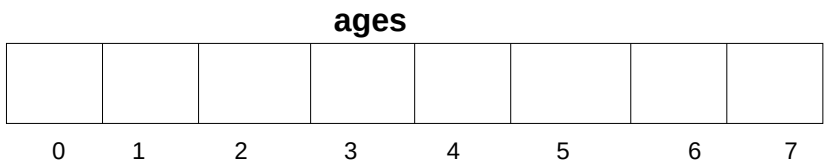
7

# Array Subscript / Index

- In Java, all arrays start with *subscript* 0
- The first element in the array named `myArray` would be `myArray[0]`
- Next would be `myArray[1]`
- If the array had six elements, the last one would be `myArray[5]`

8

```
int[] ages = new int[8];
```

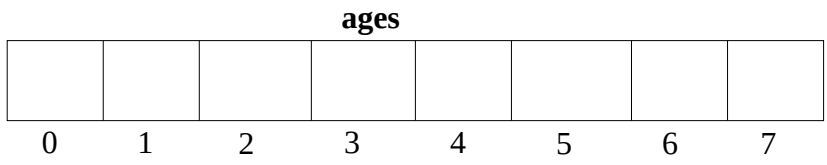



- The position of an element in an array is called the *index* or *subscript.* The first element in an array of 8 elements has an index value of 0, and the last element has an index value of 7.
- To refer to a particular element of an array – give the array name and the index in square brackets

  E.g. `ages[0]` refers to first element in the array

9

# Assigning values




- To assign a value to an array element

```
ages[0] = 133;
ages[4] = -9;
ages[7] = 12;
ages[3] = ages[7];
ages[3] = keyIn.nextInt();
```

10

## Starting at zero

- **Remember** an array starts at 0, so an index of 1 references the second element
- There is a difference between *array element 7* and the *7th element in an array*
- An array

```
int[] arrayName = new int[n];
```

  has *n* elements with indexes from 0 to n-1

11

## Initialisation

- When array is created, all elements are initialised with values depending on the array type:
  - Numbers: 0
  - Boolean: `false`
  - Object References: `null`

12

## Out of bounds errors

- Accessing a nonexistent element results in a bounds error

```
double[] data = new double[10];
data[12] = 29.95; // ERROR
```

- This error will occur at runtime. It will not be flagged by the compiler.
- Arrays have fixed length. This is their major limitation.

13

## Example - array of temperatures

Since each temperature is of type **`double`** an array of temperature readings is declared as follows:

```
double [ ] temps;
```

Since there will be seven temperature readings, memory is reserved for this array as follows:

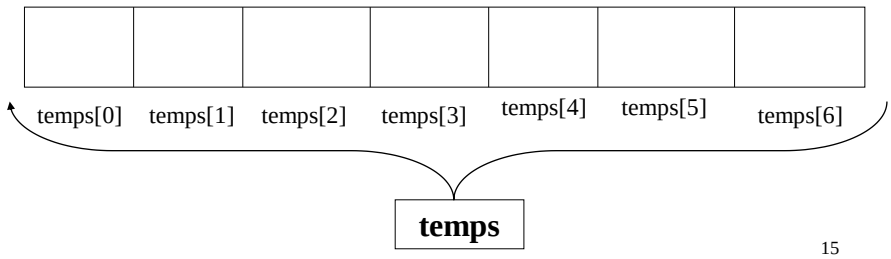
```
temps = new double [7];
```

**OR in one step**

```
double[ ]temps = new double [7];
```

14

```
double[ ]temps = new double [7];
```




15

# Initialising an array

- In situations where you know the contents in advance, it is possible to initialise or populate an array at the time it is declared.
- When doing this, it is not necessary to state the size of the array. This is dictated by the elements you initialise it with.

```
double [] temps = new double [] {9, 11.5, 11, 8.5, 7,
9, 8.5};
      OR
double [] temps =  {9, 11.5, 11, 8.5, 7, 9, 8.5};
```

16

## Array length property

- The array `.length` property contains the size of an array.

```
// declare an array of 6 Strings
String[] subjectTitles = new String[6];

// print out the size of the array
System.out.print(subjectTitles.length);

// assign the size of the array to a variable
int noOfSubjects = subjectTitles.length;
```

17

## Using a loop to access array elements

- Programs which use an array usually involve stepping through the array to examine or manipulate the contents.
- `for` loops are used for this purpose as they are designed to repeat an action a set number of times, and they have a built in counter ( `i` or `counter` ) which can be used to track the index of an array.
- Most programs using arrays will use a `for` loop for this purpose.

18

## Use `for` to load array with values.

```
// declare an array of 7 doubles
double[] temps = new double[7];

// step through the array
// reading in values and assigning values
// to the array elements
for(int i = 0; i < temps.length; i++)
{
   System.out.println("Temperature: ");
   temps[i] = keyboardIn.nextDouble();
}
```

19

## Use `for` to display array values.

```
// step through the array, printing out
// the value stored in each element
for(int i = 0; i < temps.length; i++)
{
  System.out.println(temps[i] + " ");
}
```

20

```
import java.util.*;
public class Temperatures
{
  public static void main(String[] args)
  {
      Scanner keyboardIn = new Scanner(System.in);
      double[] temps = new double[7];

      //Get daily temperatures for the week
      System.out.println("Enter temperature:  ");
      for(int i = 0; i< temps.length; i++)
      {
         System.out.print("Day " +(i+1)+": ");
         temps[i] = keyboardIn.nextDouble();
      }
      //display daily temperatures for week
      System.out.println("Daily temperatures");
      for(int i = 0; i< temps.length; i++)
      {
         System.out.println("Day" +(i+1)+": " +temps[i]);
      }
  }
}
```

21