JAVA NOTES

DATA STRUCTURES AND ALGORITHMS

Terry Marris July 2001

2 ARRAYS

2.1 LEARNING OUTCOMES

By the end of this lesson the student should be able to

- generate random positive integer numbers
- explain meanings of the terms array, element and index
- write Java code to process integers stored in an array
- dry run code that manipulates integers stored in an array

2.2 INTRODUCTION

In Part One - Fundamentals - we looked at variables. A *char* variable is a location in memory that can hold just one *char* value at a time. An *array of char* variable can hold a sequence of *char* values.

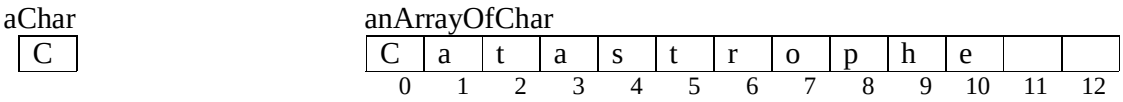
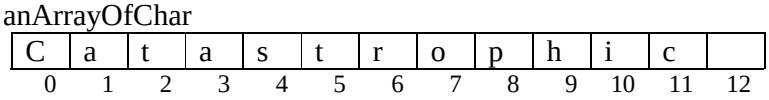


Figure 2.1 A char variable and an array of char

The array of char shown in Figure 2.1 has 13 elements, numbered (we say indexed) from zero up to 12. Each element holds just a single value. The element numbered eight has the value 'p'. The value stored in element numbered 11 is undefined.

We refer to an element in an array by its index. To store 'i' in element numbered 10 and 'c' in element 11 we write

```
anArrayOfChar[10] = 'i';
anArrayOfChar[11] = 'c';
```



The value stored in element numbered six is 'r'.

```
anArrayOfChar[6] == 'r'
```
is true

Arrays are usually processed using *for* loops (*for* loops were introduced in the last lesson). The following fragment of code will display the contents of the array *anArrayOfChar*.

```
for (int i = 0; i < 12; i++)
 System.out.println(anArrayOfChar[i]);
```

i ranges from zero up to (but not including) 12. For each value of *i* in the sequence 0..11 inclusive, the value stored in location *i* is displayed. So, when *i* is zero, *C* is displayed. When *i* is one, *a* is displayed. When *i* is two, *t* is displayed. And so on up to when *i* is 11.

Of course, not just *char*s can be stored in an array; *int*s, *double*s, *booleans* and (references to) objects can also.

2.3 ARRAYS

An array is an indexed sequence of storage locations.

We shall use an array to store and analyse a collection of marks obtained by ten students in an exam. The marks range from zero up to 100.

To create an array that can store up to ten *int* values we write

```
final int arrayCapacity = 10; // indexed 0..9
int[] markArray = new int[arrayCapacity];
```

The keyword *final* denotes a constant; the value of *arrayCapacity* is not going to change during program execution.

It is always useful to write a comment reminding you of the index values of your array. Arrays are always indexed from zero upwards.

int[] denotes an array that stores *int* values only. *markArray* is the name of the array.

new int[arrayCapacity] creates an array with the given capacity, that is, the number of elements (storage locations) it has. *arrayCapacity* represents the maximum number of values that can be stored in the array. This value is fixed at compile time.

To fill the array with random numbers between 0 and 100 (to represent the students' marks) we create a new *Random* object (that generates random numbers)

```
Random generator = new Random();
```

then set up an index, *i*, that ranges from zero up to (but not including) the capacity of the array and, for each value of *i* in that range, assign the next random number to location *i* in the array.

```
for (int i = 0; i < arrayCapacity; i++) {
  markArray[i] = generator.nextInt(101);
}
```

So, if the first three random numbers generated were 36, 18, 34, then, when *i* is zero, 36 is stored in location zero in *markArray*, when *i* is one 18 is stored in location one, and when *i* is two, 34 is stored in location two in *markArray*.

nextInt(101) returns the next random number in the range 0..100.

To print out the contents of the array we write

```
System.out.println("The marks recorded are");
System.out.print("[ ");
for (int i = 0; i < arrayCapacity; i++) {
  System.out.print(markArray[i]);
  if (i + 1 < arrayCapacity)
    System.out.print(", ");
}
System.out.println(" ]");
```

A typical result would be

```
The marks recorded are
[ 36, 18, 34, 94, 78, 3, 52, 14, 66, 21 ]
```

Exercise: explain each line of the code fragment, shown above, that displays the contents of the array enclosed within square brackets and where each item is separated from the next by a comma.

To find the highest mark stored in the array we start with the least possible int value

```
System.out.print("The highest mark is ");
int highest = Integer.MIN_VALUE;
```

then use a subscript (another word for index) *i* to look at the contents of each element in the array in turn. If we find a value in location *i* that is higher than the previously recorded value, then that must become the new highest value.

```
System.out.print("The highest mark is ");
int highest = Integer.MIN_VALUE;
for (int i = 0; i < arrayCapacity; i++) {
  if (markArray[i] > highest)
    highest = markArray[i];
}
System.out.println(highest);
```

A typical output is

```
The highest mark is 94
```

How many students scored fifty or more? Set up a counter and initialise it to zero. Set up an index *i* to look at each element in the array in turn. For each element in the array check its contents; if it is fifty or more then add 1 to the counter.

```
System.out.print("The number of students who scored " +
                 "50 or more is ");
int scoredAtLeastFifty = 0;
for (int i = 0; i < arrayCapacity; i++) {
  if (markArray[i] >= 50)
    scoredAtLeastFifty++;
}
System.out.println(scoredAtLeastFifty);
```

A typical output is

```
The number of students who scored 50 or more is 4
```

The complete program is shown below.

```
/* StudentMarks.java  Terry Marris  17 July 2001 */

import java.util.*;

public class StudentMarks {
  private static Random generator = new Random();

  public static void main(String[] s)
  {
    final int arrayCapacity = 10; // indexed 0..9
    int[] markArray = new int[arrayCapacity];

    for (int i = 0; i < arrayCapacity; i++) {
      markArray[i] = generator.nextInt(101);
    }

    System.out.println("The marks recorded are");
    System.out.print("[ ");
    for (int i = 0; i < arrayCapacity; i++) {
      System.out.print(markArray[i]);
      if (i + 1 < arrayCapacity)
        System.out.print(", ");
    }
    System.out.println(" ]");

    System.out.print("The highest mark is ");
    int highest = Integer.MIN_VALUE;
    for (int i = 0; i < arrayCapacity; i++) {
      if (markArray[i] > highest)
        highest = markArray[i];
    }
    System.out.println(highest);

    System.out.print("The number of students who scored " +
                     "50 or more is ");
    int scoredAtLeastFifty = 0;
    for (int i = 0; i < arrayCapacity; i++) {
      if (markArray[i] >= 50)
        scoredAtLeastFifty++;
    }
    System.out.println(scoredAtLeastFifty);
  }
}
```

Output:

```
The marks recorded are
[ 36, 18, 34, 94, 78, 3, 52, 14, 66, 21 ]
The highest mark is 94
The number of students who scored 50 or more is 4
```

2.4 REVIEW

2.5 FURTHER READING

In the next lesson we look at two-dimensional arrays.

2.6 EXERCISES

1 Explain the meaning of each of the terms **(i)** array **(ii)** element and **(iii)** index. Illustrate your explanations with a diagram (or diagrams).

2 Write Java code to create an array, named *quarterlySales*, that can store up to 4 *double* values.

3 Write a fragment of Java code that will return a random number in the range 1..49 inclusive.

4 Write fragments of Java code that extend program *StudentMarks.java* shown in §2.3 above, to

(i) determine and display the least mark recorded
(ii) determine and display the sum of all marks recorded
(iii) determine and display the number of students who scored 0..24 marks inclusive.
(iv) determine and display the number of students who scored 75..100 marks inclusive.

5 This fragment of code generates a run-time error. What is the error?

```
final int capacity = 5; // indexed 0..4
int[] array = new int[capacity];

System.out.println("The array contains");
System.out.print("[ ");
for (int i = 0; i <= capacity; i++) {
 System.out.print(array[i]);
 if (i + 1 < capacity)
   System.out.print(", ");
}
System.out.println(" ]");
```

6 By using the example array shown below

array

21	66	14	52	3
0	1	2	3	4

dry run the following code fragment where *capacity = 5*;

```
for (int i = 0, j = capacity - 1; i < j; i++, j--) {
  int t = array[i];
  array[i] = array[j];
  array[j] = t;
}
```

Hence give a descriptive name to the code fragment.