JAVA NOTES

DATA STRUCTURES AND ALGORITHMS

Terry Marris 19 July 2001

6 TABLES

6.1 LEARNING OUTCOMES

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

- describe the features of a two-dimensional array
- state a scenario that may be implemented using a two-dimensional array
- write code to access data stored in a two-dimensional array

6.2 INTRODUCTION

Up until now we have been using one-dimensional arrays. For example, we might record the exam mark obtained by five students in a one-dimensional array like this

examMark

43	27	84	76	59
0	1	2	3	4

In this lesson we look at two-dimensional arrays.

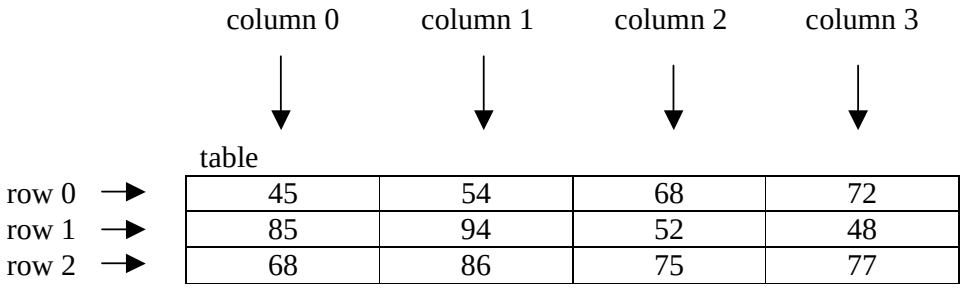
The marks obtained by three students in four different modules might be recorded in a table like this

resultsTable

	Maths	Programming	Sys Anal	Comms
Smith	45	54	68	72
Kline	85	94	52	48
French	68	86	75	77

6.3 TWO-DIMENSIONAL ARRAYS

We might store the marks obtained by the three students in their four modules in a two-dimensional array.



A two-dimensional array, also known as a table, is organised into rows and columns. Rows are always horizontal (You might *row* your boat towards the *horizon*.) Columns are always upright.

In a two-dimensional array, both rows and columns are always numbered from zero upwards.

We might want to relate column 0 to Maths, column 1 to Programming, column 2 to Sys Anal, ..., row 0 to Smith, row 1 to Kline, and so on, because the elements in a two-dimensional array can store values of the same type only, e.g. all ints or all strings but not both at the same time.

We refer to a particular element by its [row][column] co-ordinates. 52 is stored in row one column two i.e. in *table[1][2]*. We always quote the row before the column.

We create a table with three rows and four columns like this.

```
final int rowCount = 3;
final int colCount = 4;
int[][] table = new int[rowCount][colCount];
```

int[][] specifies a two-dimensional array that holds *int* values. *new int[rowCount][colCount]* returns a new two-dimensional array comprising three rows and four columns.

We visit elements in a table row by row.

```
for (int row = 0; row < rowCount; row++) {
  for (int col = 0; col < colCount; col++) {
    table[row][col] = generator.nextInt(101);
  }
}
```

So, when *row* is zero, *col* ranges from zero to three inclusive. Values are stored in elements

table[0][0], table[0][1], table[0][2], table[0][3]

in that order.

Then *row* is incremented. When *row* is one, *col* ranges from zero to three inclusive. Values are stored in elements

table[1][0], table[1][1], table[1][2], table[1][3]

Then *row* is incremented. When *row* is two, *col* ranges from zero to three inclusive. Values are stored in elements

table[2][0], table[2][1], table[2][2], table[2][3]

We look at one row at a time. For each row we visit all the columns in that row.

Exercise: explain each line of the Java code fragment shown below

```
for (int row = 0; row < rowCount; row++) {
  for (int col = 0; col < colCount; col++) {
    System.out.print(table[row][col] + "\t  ");
  }
  System.out.println();
}
```

The entire program and sample output is shown below.

```
/* Table.java
   Terry Marris 19 July 2001
*/

import java.util.*;

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

  public static void main(String[] s)
  {
    final int rowCount = 3;
    final int colCount = 4;
    int[][] table = new int[rowCount][colCount];

    for (int row = 0; row < rowCount; row++) {
      for (int col = 0; col < colCount; col++) {
        table[row][col] = generator.nextInt(101);
      }
    }

    System.out.println(
                    "Displaying the contents of the table");
    for (int row = 0; row < rowCount; row++) {
      for (int col = 0; col < colCount; col++) {
        System.out.print(table[row][col] + "\t  ");
      }
      System.out.println();
    }
  }
}
```

```
Displaying the contents of the table
56     79   67   30
33     26   89   81
22     7    51   14
```

6.4 REVIEW

6.5 FURTHER READING

HORSTMANN & CORNELL *Core Java 2 Volume 1* pp 98

In the next lesson we look at stacks.

6.6 EXERCISES

1 Explain the term two-dimensional array. Your explanation should mention rows and columns, how an element in the array is accessed and be illustrated with diagrams.

2 Write Java code to create a two-dimensional array named *sales* that can store four quarterly sales figures for each of five product lines.

3 Write fragments of Java code that extend program *Table.java* shown in §3.3 above to

(i) add an extra column to hold the average (rounded of to the nearest *int*)
(ii) find the average mark for each student and store it in the appropriate place in the array
(iii) find the student, 0, 1 or 2 which has the highest average.