CHAPTER: 06

Multi-Dimensional Arrays

Objectives:

By the end of this chapter you should be able to:

- *distinguish between **one-dimensional** arrays and **multi-dimensional** arrays;*
- *create and process **two-dimensional** arrays.*

6.8 Multi-dimensional arrays

In the temperature reading example we used at the beginning of this chapter we used an array to hold seven temperature readings (one for each day of the week). Creating an array allowed us to use loops when processing these values, rather than having to repeat the same bit of code seven times – once for each different temperature variable.

Now consider the situation where temperatures were required for the four weeks of a month. We could create four arrays as follows:

```
double[] temperature1 = new double [7]; // to hold week 1 temperatures
double[] temperature2 = new double [7]; // to hold week 2 temperatures
double[] temperature3 = new double [7]; // to hold week 3 temperatures
double[] temperature4 = new double [7]; // to hold week 4 temperatures
```

How would the temperatures for these four months be entered? The obvious solution would be to write four loops, one to process each array. Luckily there is a simpler approach – create a **multi-dimensional** array.

A multi-dimensional array is an array that has *more than one* index. So far, the arrays that we have shown you have had only one index – for this reason they are very often referred to as **one-dimensional** arrays. However, an array may have as many indexes as is necessary (up to the limit of the memory on your machine). In this particular example we need *two* indexes to access a temperature reading (one for the week number the other for the day number). If we required temperatures for each month of the year we may require *three* indexes (one for the month number, one for the week number and one for the day number) and so on. The number of dimensions an array has corresponds to the number of indexes required. Usually, no more than two indexes will ever need to be used. An array with two indexes is called a **two-dimensional** array.

6.9 Creating a two-dimensional array

To create a two-dimensional (2D) array, simply provide the size of both indexes. In this example we have four lots of seven temperatures:

```
double [][] temperature ; // declares a 2D array
temperature = new double [4][7]; // creates memory for a 4 by 7 array
```

As you can see, this is very similar to creating a one-dimensional array except that we have *two* pairs of brackets for a two-dimensional array. For larger dimensions we can have more pairs of brackets, 3 pairs for 3 dimensions, 4 for 4 dimensions and so on. In this example we have chosen to treat the first index as representing the number of weeks (4) and the second representing the number of days (7), but we could have chosen to treat the first index as the number of days and the second as the number of weeks.

While you would think of a one-dimensional array as a list, you would probably visualize a two-dimensional array as a table with rows and columns (although actually it is implemented in Java as an array of arrays). The name of each item in a two-dimensional array is the array name, plus the row and column index (see figure 6.4).

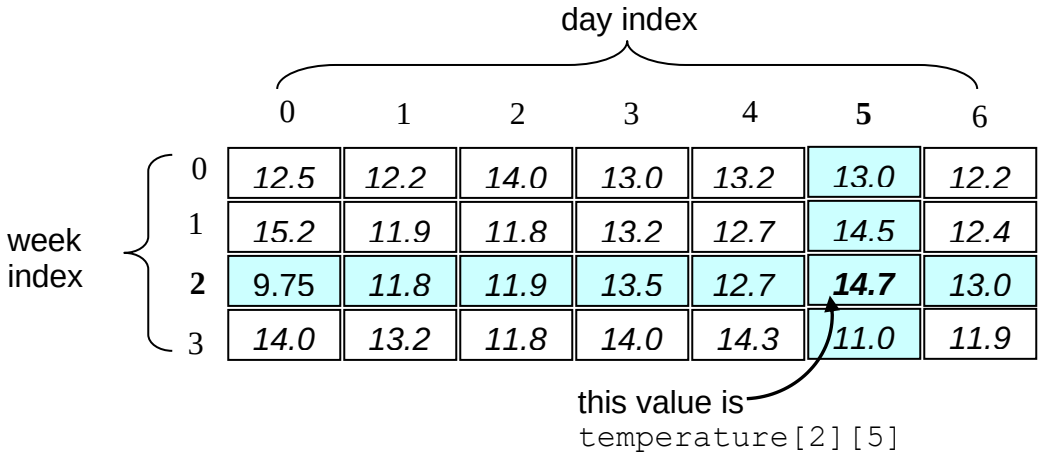

Fig 6.4 To access an element in a 2D array requires both a row and a column index

Note again that both indexes begin at zero, so that the temperature for the *third* week of the month and the *sixth* day of the week is actually given as `temperature[2][5]`.

6.10 Initializing two-dimensional arrays

As with one-dimensional arrays, it is possible to declare and initialize a multi-dimensional array with a collection of values all in one instruction. With a one-dimensional array we separated these values by commas and enclosed these values in braces. For example:

```
// this array of integers is initialized with four values
int[] a1DArray = { 11, -25, 4, 77};
```

This creates an array of size 4 with the given elements stored in the given order. A similar technique can be used for multi-dimensional arrays. With a two-dimensional array the sets of values in each row are surrounded with braces as above, then these row values are themselves enclosed in another pair of braces. A two-dimensional array of integers might be initialized as follows, for example:

```
// this creates a 2 dimensional array with two rows and four columns
int[][] a2DArray =   {
                            { 11, -25, 4, 77};
                            {-21, 55, 43, 11};
                        };
```

This instruction creates the same array as the following group of instructions:

```
int[][] a2DArray = new int[2][4]; // size array
// initialize first row of values
a2DArray[0][0] = 11;
a2DArray[0][1] = -25;
a2DArray[0][2] = 4;
a2DArray[0][3] = 77;
// initialize second row of values
a2DArray[1][0] = -21;
a2DArray[1][1] = 55;
a2DArray[1][2] = 43;
a2DArray[1][3] = 11;
```

As with one dimensional arrays, however, it is not common to initialize two-dimensional arrays in this way. Instead, once an array has been created, values are added individually to the array once the program is running.

6.11 Processing two-dimensional arrays

With the one-dimensional arrays that we have met we have used a single **`for`** loop to control the value of the single array index. How would you process a two-dimensional array that requires two indexes?

With a two-dimensional array, a *pair* of nested loops are commonly used to process each element – one loop for each array index. Let's return to the two-dimensional array of temperature values. We can use a pair of nested loops, one to control the week number and the other the day number. As with the example of the one-dimensional array of programs 6.1 – 6.3, in the following code fragment we've started our day

and week counters at 1, and then taken 1 off these counters to get back to the appropriate array index:

```
// create Scanner object for user input
Scanner keyboard = new Scanner (System.in);
// the outer loop controls the week row
for (int week = 1; week <= temperature.length; week++)
{
   // the inner loop controls the day column
   for (int day = 1; day <= temperature[0].length; day++)
   {
      System.out.println("enter temperature for week " + week +
                                                        " and day " + day);
      /* as array indices start at zero not 1, we must take one off the loop
         counters */
      temperature[week-1][day-1] = keyboard.nextDouble();
   }
}
```

Notice that in a two-dimensional array, the `length` attribute returns the length of the *first* index (this is, what we have visualized as the number of rows):

```
// here, the length attribute returns 4 (the number of rows)
for (int week = 1; week <= temperature.length; week++)
```

The number of columns is determined by obtaining the length of a particular row. In the example below we have chosen the first row but we could have chosen any row here:

```
// the length of a row returns the number of columns (7 in this case)
for (int day = 1; day <= temperature[0].length; day++)
```

Here we have used a pair of nested loops as we wish to process the entire two-dimensional array. If, however, you just wished to process part of the array (such as one row or one column) then a single loop may still suffice. In the next section we present a program that demonstrates this.

6.12 The *MonthlyTemperatures* program

Program 6.5 provides the user with a menu of options. The first option allows the user to enter the temperature readings for the 4 weeks of the month. The second option allows the user to display *all* these readings. The third option allows the user to display the reading for a particular week (for example all the temperatures for week 3). The final option allows the user to display the temperatures for a particular day of the week (for example all the readings for the first day of each week). Take a look at it and then we will discuss it.

Program 6.5

```
import java.util.*;

public class MonthlyTemperatures
{
   public static void main(String[] args)
   {
      Scanner keyboard = new Scanner (System.in);
      char choice;

      double[][] temperature = new double[4][7]; // create 2D array
      // offer menu
      do
      {
         System.out.println();
         System.out.println("[1] Enter temperatures");
         System.out.println("[2] Display all");
         System.out.println("[3] Display one week");
         System.out.println("[4] Display day of the week");
         System.out.println("[5] Exit");
         System.out.print("Enter choice [1-5]: ");
         choice = keyboard.next().charAt(0);
         System.out.println();
         // process choice by calling additional methods
         switch(choice)
         {
            case '1': enterTemps(temperature);
                      break;
            case '2': displayAllTemps(temperature);
                      break;
            case '3': displayWeek(temperature);
                      break;
            case '4': displayDays(temperature);
                      break;
            case '5': System.out.println ("Goodbye");
                      break;
            default:  System.out.println("ERROR: options 1-5 only!");
         }
      } while (choice != '5');
   }


   // method to enter temperatures into the 2D array requires a nested loop
   static void enterTemps(double[][] temperatureIn)
   {
      Scanner keyboard = new Scanner (System.in);
      // the outer loop controls the week number
      for (int week = 1; week <= temperatureIn.length; week++)
      {
         // the inner loop controls the day number
         for (int day = 1; day <= temperatureIn[0].length; day++)
         {
            System.out.println("enter temperature for week " + week + " and day " + day);
            temperatureIn[week-1][day-1] = keyboard.nextDouble();
         }
      }
   }

   // method to display all temperatures in the 2D array requires a nested loop
```

```
    static void displayAllTemps(double[][] temperatureIn)
    {
        System.out.println();
        System.out.println("***TEMPERATURES ENTERED***");
        // the outer loop controls the week number
        for (int week = 1; week <= temperatureIn.length; week++)
        {
            // the inner loop controls the day number
            for (int day = 1; day <= temperatureIn[0].length; day++)
            {
                  System.out.println("week " +week+" day "+day+": "+ temperatureIn[week-1][day-1]);
            }
        }
    }

    // method to display temperatures for a single week requires a single loop
    static void displayWeek(double[][] temperatureIn)
    {
        Scanner keyboard = new Scanner (System.in);
        int week;
        // enter week number
        System.out.print("Enter week number (1-4): ");
        week = keyboard.nextInt();
        // input validation: week number should be between 1 and 4
        while (week<1 || week > 4)
        {
            System.out.println("Inavlid week number!!");
            System.out.print("Enter again (1-4 only): ");
            week = keyboard.nextInt();
        }
        // display temperatures for given week
        System.out.println();
        System.out.println("***TEMPERATURES ENTERED FOR WEEK "+week+"***");
        System.out.println();
        // week number is fixed so loop required to process day numbers only
        for (int day = 1; day <= temperatureIn[0].length; day++)
        {
              System.out.println("week "+week+" day "+day+": "+ temperatureIn[week-1][day-1]);
        }
    }

    // method to display temperatures for a single day of each week requires a single loop
    static void displayDays(double[][] temperatureIn)
    {
        Scanner keyboard = new Scanner (System.in);
        int day;
        // enter day number
        System.out.print("Enter day number (1-7): ");
        day = keyboard.nextInt();
        // input validation: day number should be between 1 and 7
        while (day<1 || day > 7)
        {
            System.out.println("Inavlid day number!!");
            System.out.print("Enter again (1-7 only): ");
            day = keyboard.nextInt();
        }
        // display temperatures for given day of the week
        System.out.println();
        System.out.println("***TEMPERATURES ENTERED FOR DAY "+day+"***");
        System.out.println();
        // day number is fixed so loop required to process week numbers only
        for (int week = 1; week <= temperatureIn.length; week++)
        {
              System.out.println("week "+week+" day "+day+": " + temperatureIn[week-1][day-1]);
        }
    }
}
```

Here you can see that the first menu option contains the code we have just discussed for entering values into a two-dimensional array. Notice how you pass a two-dimensional array to a method. As with a one-dimensional array you do not refer to the size of the array, just its dimensions:

```
/* As with a standard 1D array, to pass a 2D array to a method the number of dimensions
needs to be indicated but not the size of these dimensions */
static void enterTemps(double[][] temperatureIn)
{
    // code for entering temperatures goes here
}
```

This method uses a pair of nested loops as we wish to process the entire two-dimensional array. Similarly, when we wish to display the entire array we use a pair of nested loops to control the week and day number:

```
// method to display all temperatures in the 2D array requires a nested loop
static void displayAllTemps(double[][] temperatureIn)
{
    System.out.println();
    System.out.println("***TEMPERATURES ENTERED***");
    // the outer loop controls the week number
    for (int week = 1; week <= temperatureIn.length; week++)
    {
        // the inner loop controls the day number
        for (int day = 1; day <= temperatureIn[0].length; day++)
        {
               System.out.println("week " +week+" day "+day+": "+
                                                temperatureIn[week-1][day-1]);
        }
    }
}
```

However, when we need to display just one of the dimensions of an array we do not need to use a pair of loops. For example, the `displayWeek` method, that allows the user to pick a particular week number so that just the temperatures for that week alone are displayed, just requires a *single* loop to move through the day numbers, as the week number is fixed by the user:

```
// method to display temperatures for a single week requires a single loop
    static void displayWeek(double[][] temperatureIn)
    {
        Scanner keyboard = new Scanner (System.in);
        int week;
        // enter week number
        System.out.print("Enter week number (1-4): ");
        week = keyboard.nextInt();
        // input validation: week number should be between 1 and 4
        while (week<1 || week > 4)
        {
            System.out.println("Inavlid week number!!");
            System.out.print("Enter again (1-4 only): ");
            week = keyboard.nextInt();
        }
        // display temperatures for given week
        System.out.println();
        System.out.println("***TEMPERATURES ENTERED FOR WEEK "+week+"***");
        System.out.println();
        // week number is fixed so loop required to process day numbers only
        for (int day = 1; day <= temperatureIn[0].length; day++)
        {
               System.out.println("week "+week+" day "+day+": "+
                                          temperatureIn[week-1][day-1]);
        }
    }
```

First the user enters the week number:

```
System.out.print("Enter week number (1-4): ");
week = keyboard.nextInt();
```

We will use this week number to determine the value of the first index when looking up temperatures in the array, so we need to be careful that the user inputs a valid week number (1 to 4) as invalid numbers would lead to an illegal array index being generated. We have used a **`while`** loop here to implement this input validation:

```
// input validation: week number should be between 1 and 4
while (week<1 || week > 4)
{
   System.out.println("Inavlid week number!!");
   System.out.print("Enter again (1-4 only): ");
   week = keyboard.nextInt();
}
```

Once we get past this loop we can be sure that the user has entered a valid week number. For example, assume that we have filled the 2D array with the temperatures given in figure 6.4. Now assume that the user calls the option to display one week's temperature, and chooses week 2. This is illustrated in figure 6.5 below:

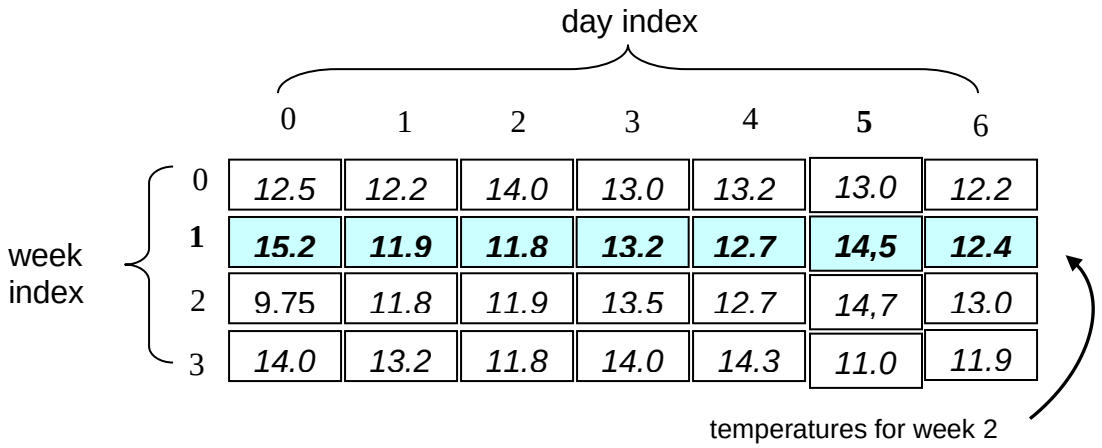

Fig 6.5 To access temperatures for a single week, the week index remains fixed and the day index changes.

Since array indexes in Java begin at zero, the week index (1) is determined by taking one off the week number entered by the user (2). All we need to do now is to iterate through all the day numbers for that

week by using a single **`for`** loop:

```
/* week number is fixed by the user so a single loop is required to process day numbers
only */
for (int day = 1; day <= temperatureIn[0].length; day++)
{
       System.out.println("week "+ week +" day " + day + ": " +
                                        temperatureIn[week-1][day-1]);
}
```

The `displayDays` method works in a similar way but with the day number fixed and the week number being determined by the **`for`** loop.