Multiple Dimensions

by Sophia

In this lesson, you will learn about multiple **dimension** arrays. A dimension of an array is an axis along which the elements are organized. The data in an array can be organized along 1, 2, or more axes or dimensions. Specifically, this lesson covers:

Table of Contents

1. Multidimensional Arrays

Java multidimensional arrays are arranged as an array of arrays. The elements of multi-dimensional arrays are seen in rows and columns. Before discussing multiple dimensions in Java arrays, It is important to explore what multiple dimensions look like.

1a. One Dimensional Arrays

Recall what you learned about one dimension arrays in a previous tutorial. It resembles a list data collection type.

It appears as a straight line on a flat plane as demonstrated in this diagram:

PLACEHOLDER JAV 152 Alt Text: A double sided arrow.

A one dimensional listing would look like this on paper:

element value	"sun"	"fun"	run	bun	nun	pun
Index	0	1	2	3	4	5

The index position is one integer. To locate "run" from this listing you can use the single index position.

🖊 **CONCEPT TO KNOW**

Remember that list elements start at 0.

"run" is at index value 2.

📄 **TERM TO KNOW**

Dimension
A dimension of an array is an axis along which the elements are organized.

1b. Two Dimensional Arrays

When nesting an array within an array you are, in a sense, adding a second dimension. A nested list is a two dimensional list; or 2D for short.

Here is what it looks like graphically:

Placeholder JAV 153 Alt Text: An X/Y chart.

You have likely seen this previously in an X/Y chart. It has two dimensions. One dimension is running diagonally and the other vertically; though they are still on a flat plane.

A two dimensional listing would look like this on paper:

X/Y indexes		X ➡		
		0	1	2
Y	0	fun	sun	run
⬇	1	bun	nun	pun
	2	fan	van	man
	3	can	tan	ran

There are two index positions for each element. To locate "fan" from this two dimensional listing identify both index positions, starting with the row then column.

⤳ EXAMPLE
"fan" is at position 2, 0 (Y,X or Row/Column

There can be more than one dimension when it comes to an array in Java. Consider how teachers store their grades for a class in a grade book. They have a list of students and within the list of students and they have a list of assignments. For each of those assignments, they will have a grade assigned.

The scores in Row 0 represent the scores for one student. This person's scores, for the different assignments, are seen in Column 0, Column 1, Column 2, and Column 3. The scores in Row 1 and Row 2 represent scores for other students. Since all of the values in an array have to be of the same data type, you can not mix names or letter grades in with the integer scores.

	Column 0	Column 1	Column 2	Column3
Row 0	100	92	99	85
Row 1	100	95	88	91

Row 2	99	100	100	100

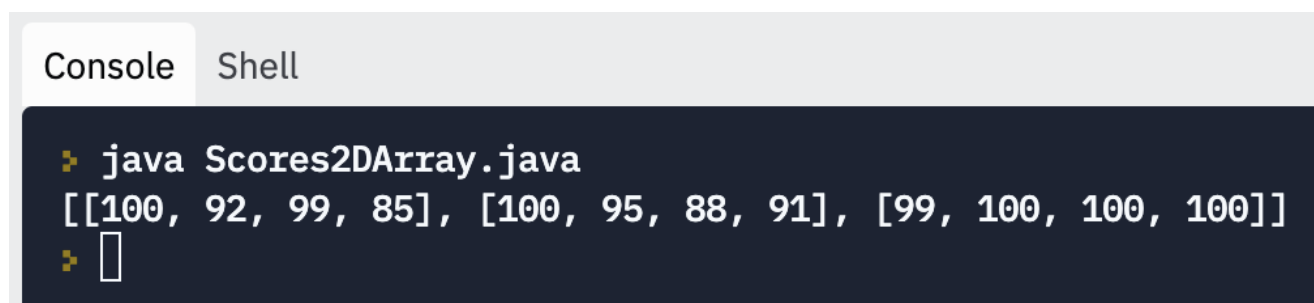Placeholder JAV 154 Alt Text: Four inner lists within an outer list making this a 2D list.

Let's see this 2D array in Java code. To print out a 2-dimensional array, use the Arrays.deepToString() method. This method has to "go deep" because it needs to work down through both dimensions in the array to display the values, not just 1.

Here is what a 2D array in Java code would look like:

```java
import java.util.Arrays;

class Scores2DArray {
  public static void main(String[] args) {
    // Array of 3 rows & 4 columns
    // 2 pairs of square brackets declare 2D array of int
    int[][] scores = {
      {100, 92, 99, 85},
      {100, 95, 88, 91},
      {99, 100, 100, 100}
    };
    System.out.println(Arrays.deepToString(scores));
  }
}
```

The output should look like this:


```
Console   Shell

> java Scores2DArray.java
[[100, 92, 99, 85], [100, 95, 88, 91], [99, 100, 100, 100]]
> []
```

As demonstrated above, this 2D list is a long row of elements separated by the square brackets. Keep in mind that a 2-dimensional array in Java can be thought of as an array of arrays.

The array scores could have been declared like this:

```java
import java.util.Arrays;

class Scores2DArray {
  public static void main(String[] args) {
    // Array of 3 rows & 4 columns
    // 2 pairs of square brackets declare 2D array of int
    int[][] scores = new int[3][4];
    // Assign a 1D array to each row in the 2D array
```

```
    scores[0] = new int[] {100, 92, 99, 85};
    scores[1] = new int[] {100, 95, 88, 91};
    scores[2] = new int[] {99, 100, 100, 100};
    // Print out just the first row as a 1-dimensional array
    System.out.println(Arrays.toString(scores[0]));
  }
}
```

Either way, the first list can be accessed by using the index for the first row (index 0):
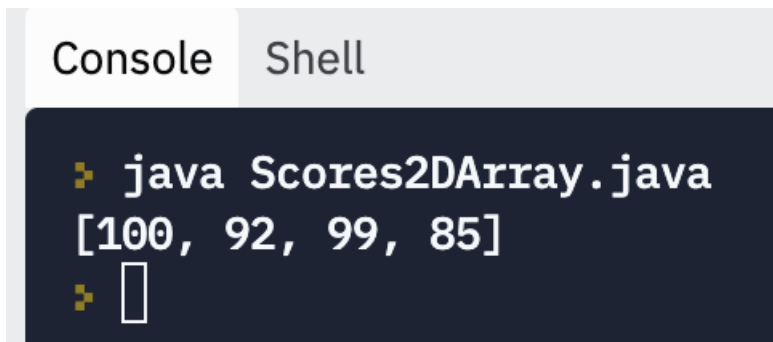
```
import java.util.Arrays;

class Scores2DArray {
  public static void main(String[] args) {
    // Array of 3 rows & 4 columns
    // 2 pairs of square brackets declare 2D array of int
    int[][] scores = {
      {100, 92, 99, 85},
      {100, 95, 88, 91},
      {99, 100, 100, 100}
    };
    // Print out just the first row as a 1-dimensional array
    System.out.println(Arrays.toString(scores[0]));
  }
}
```

The output is just the first row from the 2D array as a 1D array as seen below:

To access a specific element within that first row, use an index in a second pair square brackets after the first set. To select the third element from the first row, use the 2 as the column index. Each row in the 2D array represents the grades for one student.

The following code shows how to access the 3rd score for the 1st student, since each column (the 2nd dimension or index) represents a different assignment score:
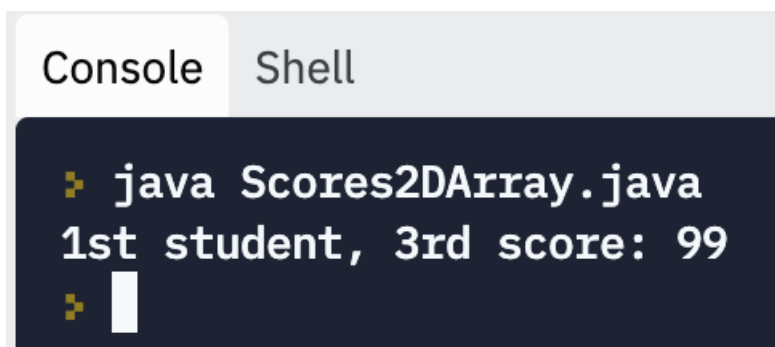
```
class Scores2DArray {
  public static void main(String[] args) {
    // Array of 3 rows & 4 column
```

```
      // 2 pairs of square brackets declare 2D array of int
      int[][] scores = new int[3][4];
      // Assign a 1D array to each row in the 2D array
      scores[0] = new int[] {100, 92, 99, 85};
      scores[1] = new int[] {100, 95, 88, 91};
      scores[2] = new int[] {99, 100, 100, 100};
      // Print out just the element in the first row,
      // third column as int. No toString() needed.
      System.out.println("1st student, 3rd score: " + scores[0][2]);
  }
}
```

The output should look like this:

```
Console   Shell

 ┣ java Scores2DArray.java
 1st student, 3rd score: 99
 ┣ ▊
```

📄 **TERMS TO KNOW**

Two-Dimensional (or 2D) Array
A two-dimensional array is an array of arrays with data laid out in a grid-like pattern of rows and columns.

Arrays.deepToString()
A Java method that converts data in a two-dimensional array to a format that can be displayed on the screen.
[Text]

1c. Three Dimensional Arrays

Using **three-dimensional arrays** can be complicated. Here, you will consider X,Y, and Z index positions. At this point it becomes difficult to see a representation of it on "paper" or a flat plane. Utilizing three dimensional collections of data is rarely used unless it is for 3D object modeling or mapping.

When defining a 3D array representing some sort of values in points in 3-dimensional space, the declaration would look like this:

```
int[][][] temperatures3D = new int[100][100][100];
```

The values in the elements of this array could be assigned and accessed using 3 indices. The first index is the "layer" in the 3-dimensional structure. The second index is the row within that layer, and the third index is the column within that row.

Three-Dimensional (or 3D) Array

A three-dimensional array adds a 3rd index or axis to the organization of the array elements.

☑ SUMMARY

In this lesson, you learned about using arrays with **multiple dimensions**. You learned about the difference between **one, two, and three dimensions**. Finally, you learned that three-dimensional arrays are complex and not often used in basic programming in Java.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source **cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf**

It has also been adapted from "Python for Everybody" By Dr. Charles R. Severance. Source **py4e.com/html3/**

📄 TERMS TO KNOW

Arrays.deepToString()

A Java method that converts data in a two-dimensional array to a format that can be displayed on the screen.

Dimension

A dimension of an array is an axis along which the elements are organized.

Three-Dimensional (or 3D) Array

A three-dimensional array adds a 3rd index or axis to the organization of the array elements.

Two-Dimensional (or 2D) Array

A two-dimensional array is an array of arrays with data laid out in a grid-like pattern of rows and columns.