



Chapter 8

Multidimensional Arrays

Motivations

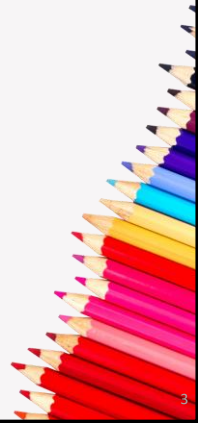
- Thus far, you have used one-dimensional arrays to model linear collections of elements. You can use a two-dimensional array to represent a matrix or a table. For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

Distance Table (in miles)							
	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

Motivations

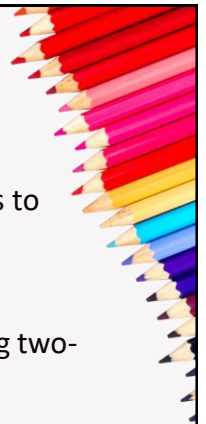
- *Data in a table or a matrix can be represented using a two-dimensional array.*

```
double[][] distances = {
    {0, 983, 787, 714, 1375, 967, 1087},
    {983, 0, 214, 1102, 1763, 1723, 1842},
    {787, 214, 0, 888, 1549, 1548, 1627},
    {714, 1102, 888, 0, 661, 781, 810},
    {1375, 1763, 1549, 661, 0, 1426, 1187},
    {967, 1723, 1548, 781, 1426, 0, 239},
    {1087, 1842, 1627, 810, 1187, 239, 0},
};
```



Objectives

- To give examples of representing data using two-dimensional arrays (§8.1).
- To declare variables for two-dimensional arrays, create arrays, and access array elements in a two-dimensional array using row and column indexes (§8.2).
- To program common operations for two-dimensional arrays (displaying arrays, summing all elements, finding the minimum and maximum elements, and random shuffling) (§8.3).
- To pass two-dimensional arrays to methods (§8.4).
- To write a program for grading multiple-choice questions using two-dimensional arrays (§8.5).
- To solve the closest-pair problem using two-dimensional arrays (§8.6).
- To check a Sudoku solution using two-dimensional arrays (§8.7).
- To use multidimensional arrays (§8.8).



8.2 Two-Dimensional Array Basics

- The syntax for declaring a two-dimensional array is:

```
elementType[][] arrayRefVar;  
elementType arrayRefVar[][];
```

- Create array and assign its reference to variable

```
refVar = new dataType[10][10];
```

- Combine declaration and creation in one statement

```
dataType[][] refVar = new dataType[10][10];  
dataType refVar[][] = new dataType[10][10];
```

Two-dimensional Array Illustration

- An element in a two-dimensional array is accessed through a row and column index.

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```
matrix = new int[5][5];
```

matrix.length? 5

matrix[0].length? 5

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	7	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```
matrix[2][1] = 7;
```

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9
[3]	10	11	12

```
int[][] array = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9},  
    {10, 11, 12}  
};
```

array.length? 4

array[0].length? 3

Declaring, Creating, and Initializing Using Shorthand Notations

- You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

```
int[][] array = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9},
    {10, 11, 12}
};
```

(a)

Equivalent

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

(b)

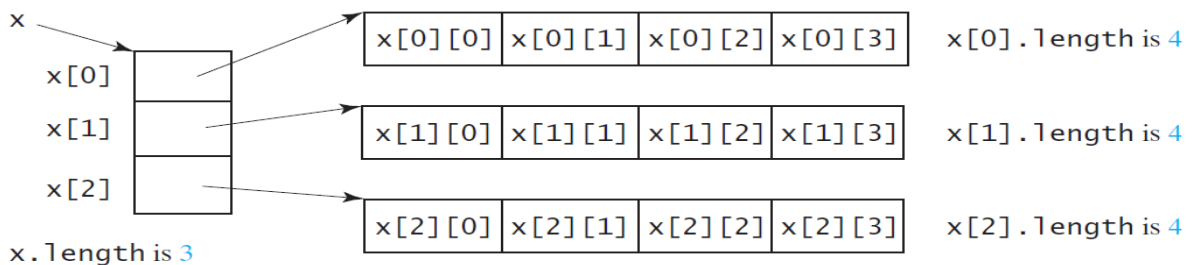
7

Lengths of Two-dimensional Arrays

- A two-dimensional array is a one-dimensional array in which each element is another one-dimensional array.

```
int[][] x = new int[3][4];
```

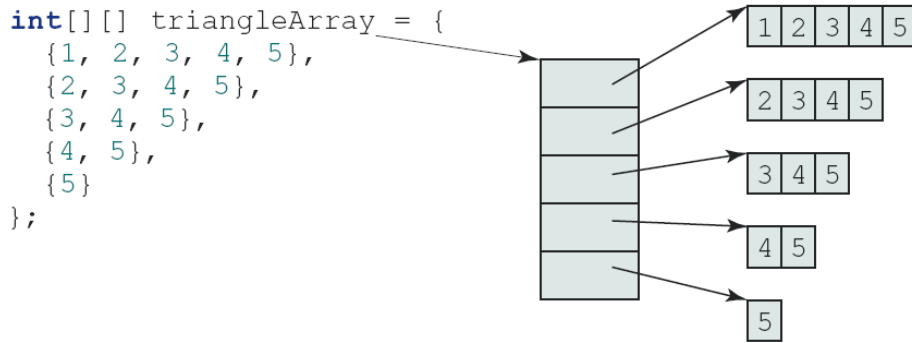
ArrayIndexOutOfBoundsException



8

Ragged Arrays

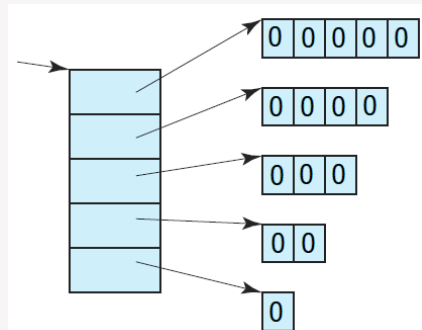
- Each row in a two-dimensional array is itself an array. Thus, the rows can have different lengths. An array of this kind is known as a *ragged array*.



Ragged Arrays

- If you don't know the values in a ragged array in advance, but do know the sizes, you can create a ragged array using the following syntax:

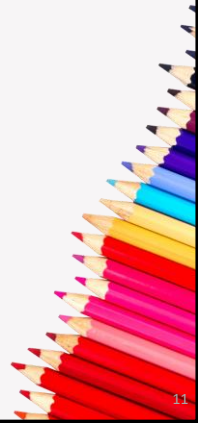
```
int[][] triangleArray = new int[5][];
triangleArray[0] = new int[5];
triangleArray[1] = new int[4];
triangleArray[2] = new int[3];
triangleArray[3] = new int[2];
triangleArray[4] = new int[1];
```



8.3 Processing Two-Dimensional Arrays

- *Nested **for** loops are often used to process a two-dimensional array.*

1. Initializing arrays with input values
2. Printing arrays
3. Summing all elements
4. Summing all elements by column
5. Which row has the largest sum
6. Random shuffling



11

Initializing arrays with input values

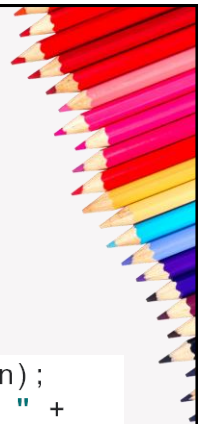
- Suppose an array matrix is created as follows:

```
int[][] matrix = new int[10][10];
```

- The following loop initializes the array with user input values:

```
java.util.Scanner input = new java.util.Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
    matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        matrix[row][column] = input.nextInt();
    }
}
```

12



Initializing arrays with random values

- The following loop initializes the array with random values between 0 and 99:

```
for (int row = 0; row < matrix.length; row++) {  
    for (int column = 0; column < matrix[row].length; column++) {  
        matrix[row][column] = (int)(Math.random() * 100);  
    }  
}
```

13

Printing arrays

- To print a two-dimensional array, you have to print each element in the array using a loop like the following:

```
for (int row = 0; row < matrix.length; row++) {  
    for (int column = 0; column < matrix[row].length; column++) {  
        System.out.print(matrix[row][column] + " ");  
    }  
  
    System.out.println();  
}
```

14

Summing all elements

- Use a variable named total to store the sum. Initially total is 0. Add each element in the array to total using a loop like this:

```
int total = 0;
for (int row = 0; row < matrix.length; row++) {
    for (int column = 0; column < matrix[row].length; column++) {
        total += matrix[row][column];
    }
}
```

15

Summing elements by column

- For each column, use a variable named total to store its sum. Add each element in the column to total using a loop like this:

```
for (int column = 0; column < matrix[0].length; column++) {
    int total = 0;
    for (int row = 0; row < matrix.length; row++)
        total += matrix[row][column];
    System.out.println("Sum for column " + column + " is "
        + total);
}
```

16

Which row has the largest sum?

```
int maxRow = 0;
int indexOfMaxRow = 0;

// Get sum of the first row in maxRow
for (int column = 0; column < matrix[0].length; column++) {
    maxRow += matrix[0][column];
}

for (int row = 1; row < matrix.length; row++) {
    int totalOfThisRow = 0;
    for (int column = 0; column < matrix[row].length; column++)
        totalOfThisRow += matrix[row][column];

    if (totalOfThisRow > maxRow) {
        maxRow = totalOfThisRow;
        indexOfMaxRow = row;
    }
}

System.out.println("Row " + indexOfMaxRow
    + " has the maximum sum of " + maxRow);
```

Random shuffling

```
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        int i1 = (int)(Math.random() * matrix.length);
        int j1 = (int)(Math.random() * matrix[i].length);

        // Swap matrix[i][j] with matrix[i1][j1]

        int temp = matrix[i][j];
        matrix[i][j] = matrix[i1][j1];
        matrix[i1][j1] = temp;
    }
}
```

8.4 Passing Two-Dimensional Arrays to Methods

- When passing a two-dimensional array to a method, the reference of the array is passed to the method.

Enter 3 rows and 4 columns:

1	2	3	4	<input type="button" value="Enter"/>
5	6	7	8	<input type="button" value="Enter"/>
9	10	11	12	<input type="button" value="Enter"/>

Sum of all elements is 78

PassTwoDimensionalArray

Run

19

Case Study: Grading Multiple-Choice Test

- Objective: write a program that grades multiple-choice test.
- Students' answer

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

Key to the Questions:

0 1 2 3 4 5 6 7 8 9

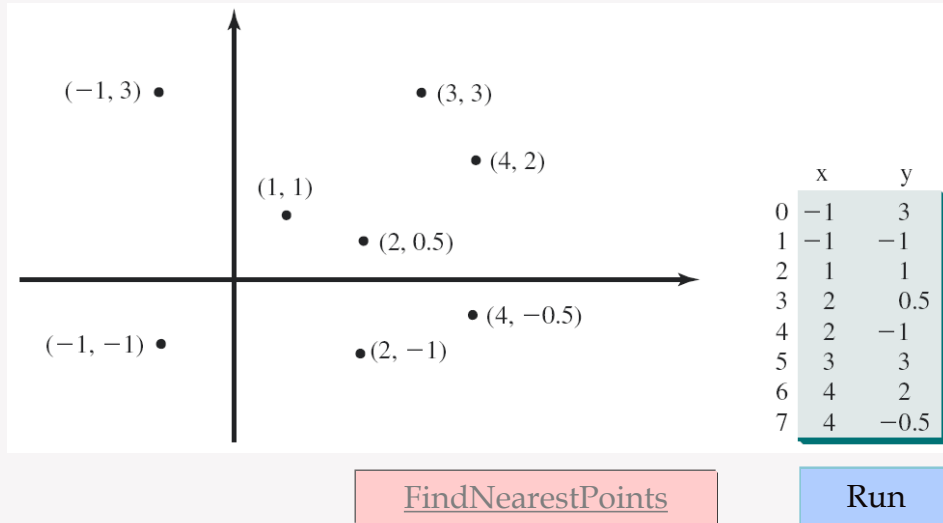
Key D B D C C D A E A D

GradeExam

Run

20

Case Study: Finding Two Points Nearest to Each Other



21

What is Sudoku?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

(a) Puzzle

Solution →

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(b) Solution

22

A grid can be represented using a two-dimensional array

5	3	0	0	7	0	0	0	0
6	0	0	1	9	5	0	0	0
0	9	8	0	0	0	0	6	0
8	0	0	0	6	0	0	0	3
4	0	0	8	0	3	0	0	1
7	0	0	0	2	0	0	0	6
0	6	0	0	0	0	0	0	0
0	0	0	4	1	9	0	0	5
0	0	0	0	8	0	0	7	9

(a)

```
int[][] grid =
{{5, 3, 0, 0, 7, 0, 0, 0, 0},
 {6, 0, 0, 1, 9, 5, 0, 0, 0},
 {0, 9, 8, 0, 0, 0, 0, 6, 0},
 {8, 0, 0, 0, 6, 0, 0, 0, 3},
 {4, 0, 0, 8, 0, 3, 0, 0, 1},
 {7, 0, 0, 0, 2, 0, 0, 0, 6},
 {0, 6, 0, 0, 0, 0, 2, 8, 0},
 {0, 0, 0, 4, 1, 9, 0, 0, 5},
 {0, 0, 0, 0, 8, 0, 0, 7, 9}};
```

(b)

Checking Whether a Solution Is Correct

- A solution is stored in grid.

```
A solution grid is
{{5, 3, 4, 6, 7, 8, 9, 1, 2},
 {6, 7, 2, 1, 9, 5, 3, 4, 8},
 {1, 9, 8, 3, 4, 2, 5, 6, 7},
 {8, 5, 9, 7, 6, 1, 4, 2, 3},
 {4, 2, 6, 8, 5, 3, 7, 9, 1},
 {7, 1, 3, 9, 2, 4, 8, 5, 6},
 {9, 6, 1, 5, 3, 7, 2, 8, 4},
 {2, 8, 7, 4, 1, 9, 6, 3, 5},
 {3, 4, 5, 2, 8, 6, 1, 7, 9}};
```

CheckSudokuSolution

Run

8.8 Multidimensional Arrays

- Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays for any integer n.
- A two-dimensional array consists of an array of one-dimensional arrays and a three-dimensional array consists of an array of two-dimensional arrays.
- The way to declare two-dimensional array variables and create two-dimensional arrays can be generalized to declare n-dimensional array variables and create n-dimensional arrays for $n \geq 3$.
- For example, you may use a three-dimensional array to store exam scores for a class of six students with five exams, and each exam has two parts (multiple-choice and essay).

25

Multidimensional Arrays

```
double[][][] scores = {
    {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
    {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
    {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
    {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
    {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
    {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}};
```

Which student Which exam Multiple-choice or essay score

scores [i] [j] [k]

Case Study: Weather Information

- Suppose a meteorology station records the temperature and humidity at each hour of every day and stores the data for the past ten days in a text file named `weather.txt`. Each line of the file consists of four numbers that indicate the day, hour, temperature, and humidity. Your task is to write a program that calculates the average daily temperature and humidity for the 10 days.

Day	Hour	Temperature	Humidity
1	1	76.4	0.92
1	2	77.7	0.93
...			
10	23	97.7	0.71
10	24	98.7	0.74

(a)

Day	Hour	Temperature	Humidity
10	24	98.7	0.74
1	2	77.7	0.93
...			
10	23	97.7	0.71
1	1	76.4	0.92

(b)

Weather.txt

Weather

Run

27

Case Study: Guessing Birthday

- Listing 4.3, `GuessBirthday.java`, gives a program that guesses a birthday.
- The program can be simplified by storing the numbers in five sets in a three-dimensional array, and it prompts the user for the answers using a loop.

GuessBirthdayUsingArray

Run

28

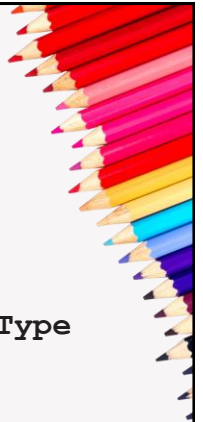


Chapter Summary



Chapter Summary

- A two-dimensional array can be used to store a table.
- A variable for two-dimensional arrays can be declared using the syntax: `elementType[][] arrayVar`.
- A two-dimensional array can be created using the syntax: `new elementType [ROW_SIZE] [COLUMN_SIZE]`.
- Each element in a two-dimensional array is represented using the syntax: `arrayVar[rowIndex] [columnIndex]`.
- You can create and initialize a two-dimensional array using an array initializer with the syntax: `elementType[][] arrayVar = {{row values}, . . . , {row values}}`.
- You can use arrays of arrays to form multidimensional arrays.



Programming Exercises

*1, 2, 4, 5, 6, 10, 11, 13,
16, 22, 23, 25, 26, 27,
29, 32, 35*



31