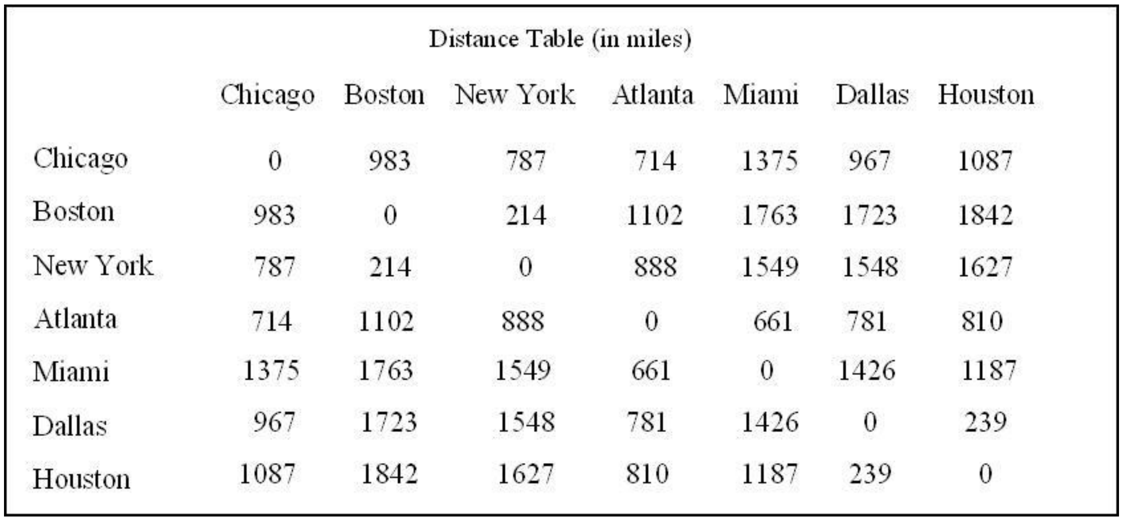
Multidimensional Arrays

Introduction

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

Two-Dimension Array Basics

You can use a two-dimensional array to represent a matrix or a table.
Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays for any integer n.

Declaring Variables of Two-Dimensional Arrays and Creating Two-Dimensional Arrays

Here is the syntax for declaring a two-dimensional array:

```
    dataType [][] arrayRefVar;
or
    dataType arrayRefVar[][];     // This style is correct, but not preferred
```

As an example, here is how you would **declare** a two-dimensional array variable matrix of int values

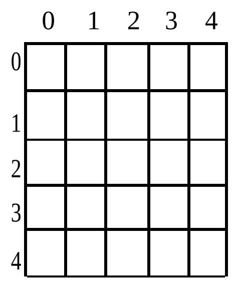
int [][] matrix;

or

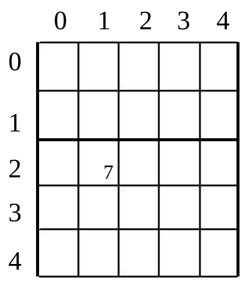
int matrix[][]; // *This style is correct, but not preferred*

You can **create** a two-dimensional array of 5 by 5 int values and assign it to matrix using this syntax:

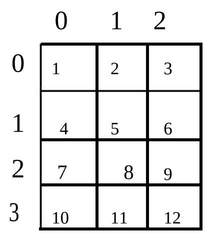
matrix = **new** int[5][5];



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



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



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

The index of each subscript of a multidimensional array is an int value starting from 0.

Caution

It is a common mistake to use matrix[2,1] to access the element at row 2 and column 1.
In Java, each subscript must be enclosed in a pair of square brackets.
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}
};
```

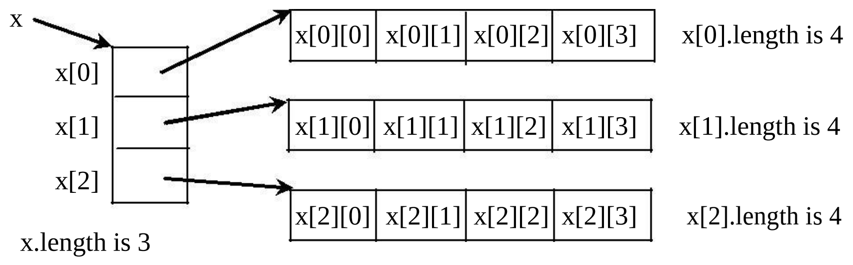
Equivalent

```
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;
```

Obtaining the Lengths of Two-Dimensional Arrays

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

x.length is 3
x[0].length is 4, x[1].length is 4, x[2].length is 4

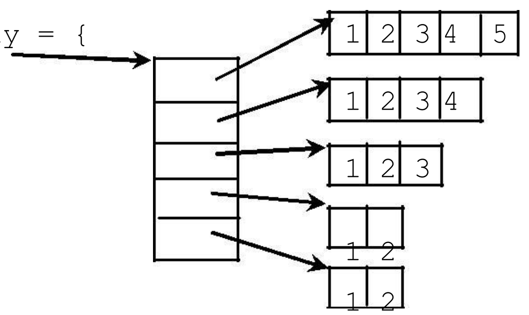

x
x[0]
x[1]
x[2]
x.length is 3
x[0][0] x[0][1] x[0][2] x[0][3]
x[1][0] x[1][1] x[1][2] x[1][3]
x[2][0] x[2][1] x[2][2] x[2][3]
x[0].length is 4
x[1].length is 4
x[2].length is 4

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

Ragged Arrays

Each row in a two-dimensional array is itself an array. Thus, the rows can have different lengths.

```
int[][] triangleArray = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```



If you **don't** know the values in a raged array in advance, but know the sizes, say the same as before, you can create a ragged array using the syntax that follows:

```
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];
```

Processing Two-Dimensional Arrays

Suppose an array matrix is declared as follows:

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

Here are some examples of processing two-dimensional arrays:

- (Initializing arrays with input values) The following loop initializes the array with user input values:

```
java.util.Scanner input = new 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();
      }
}
```

(Initializing arrays with random values) You can now assign random values to the array using the following loop:

```
for (int row = 0; row < triangleArray.length; row++)
      for (int column = 0; column < triangleArray[row].length; column++)
            triangleArray[row][column] = (int) (Math.random( ) * 1000);
```

- (Printing arrays)

```
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();
}
```

(Summing all elements)
(Summing elements by column)
(Which row has the largest sum?)

Passing Two-Dimensional Arrays to Methods

You can pass a two-dimensional array to a method just as you pass a one-dimensional array. Following example with a method that returns the sum of all the elements in a matrix.

PassTwoDimensionalArray.java

```
import java.util.Scanner;

public class PassTwoDimensionalArray {
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);

    // Enter array values
    int[][] m = new int[3][4];
    System.out.println("Enter " + m.length + " rows and
      " + m[0].length + " columns: ");
    for (int i = 0; i < m.length; i++)
      for (int j = 0; j < m[i].length;j++)
        m[i][j] = input.nextInt();

    // Display result
    System.out.println("\nSum of all elements is " + sum(m));
  }

  public static int sum(int[][] m)
    { int total = 0;
    for (int row = 0; row < m.length; row++) {
      for (int column = 0; column < m[row].length; column++)
        { total += m[row][column];
      }
    }

    return total;
  }
}
```

```
Enter 3 rows and 4
columns: 1 2 3 4 5 6 7 8

9 10 11 12
Ï
Sum of all elements is 78
```

Example: Grading a Multiple-Choice Test

Objective: write a program that grades multiple-choice test.
Suppose there are **eight** students and **ten** questions, and the answers are stored in a two-dimensional array.
Each row records a student's answers to the questions, as shown in the following array:

Students' Answers to the Questions:

	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.java: Grading a Multiple-Choice Test

```java
public class GradeExam
  { /** Main method */
  public static void main(String args[]) {
    //Students' answers to the questions
    char[][] answers = {
      {'A', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'D', 'B', 'A', 'B', 'C', 'A', 'E', 'E', 'A', 'D'},
      {'E', 'D', 'D', 'A', 'C', 'B', 'E', 'E', 'A', 'D'},
      {'C', 'B', 'A', 'E', 'D', 'C', 'E', 'E', 'A', 'D'},
      {'A', 'B', 'D', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'B', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'B', 'B', 'A', 'C', 'C', 'D', 'E', 'E', 'A', 'D'},
      {'E', 'B', 'E', 'C', 'C', 'D', 'E', 'E', 'A', 'D'}};

       //Key to the questions
    char[] keys = {'D', 'B', 'D', 'C', 'C', 'D', 'A', 'E', 'A', 'D'};

    // Grade all answers
    for (int i = 0; i < answers.length; i++)
      { // Grade one student
      int correctCount = 0;
      for (int j = 0; j < answers[i].length; j++)
        { if (answers[i][j] == keys[j])
          correctCount++;
      }

      System.out.println("Student " + i + "'s correct count is "

        + correctCount);
    }
  }
}
```

```
Student 0's correct count is 7
Student 1's correct count is 6
Student 2's correct count is 5
Student 3's correct count is 4
Student 4's correct count is 8
Student 5's correct count is 7
Student 6's correct count is 7
Student 7's correct count is 7
```

Multidimensional Arrays

The following syntax declares a three-dimensional array variable scores, creates an array, and assigns its reference to scores:

```
double [ ] [ ] [ ] x = new double[2][3][4];
```

double[ ][ ] [ ] x = new double[2][3][4];

x.length is 2
x[0].length is 3, x[1].length is 3
x[0][0].length is 4, x[0][1].length is 4, x[0][2].length is 4,
x[1][0].length is 4, x[1][1].length is 4, x[1][2].length is 4