iClicker
Chapter 8 Multidimensional Arrays

CS1: Java Programming
Colorado State University

Original slides by Daniel Liang
Modified slides by Chris Wilcox
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.

<table>
<thead>
<tr>
<th>Distance Table (in miles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
</tr>
<tr>
<td>Boston</td>
</tr>
<tr>
<td>New York</td>
</tr>
<tr>
<td>Atlanta</td>
</tr>
<tr>
<td>Miami</td>
</tr>
<tr>
<td>Dallas</td>
</tr>
<tr>
<td>Houston</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Chicago</th>
<th>Boston</th>
<th>New York</th>
<th>Atlanta</th>
<th>Miami</th>
<th>Dallas</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>0</td>
<td>983</td>
<td>787</td>
<td>714</td>
<td>1375</td>
<td>967</td>
<td>1087</td>
</tr>
<tr>
<td>Boston</td>
<td>983</td>
<td>0</td>
<td>214</td>
<td>1102</td>
<td>1763</td>
<td>1723</td>
<td>1842</td>
</tr>
<tr>
<td>New York</td>
<td>787</td>
<td>214</td>
<td>0</td>
<td>888</td>
<td>1549</td>
<td>1548</td>
<td>1627</td>
</tr>
<tr>
<td>Atlanta</td>
<td>714</td>
<td>1102</td>
<td>888</td>
<td>0</td>
<td>661</td>
<td>781</td>
<td>810</td>
</tr>
<tr>
<td>Miami</td>
<td>1375</td>
<td>1763</td>
<td>1549</td>
<td>661</td>
<td>0</td>
<td>1426</td>
<td>1187</td>
</tr>
<tr>
<td>Dallas</td>
<td>967</td>
<td>1723</td>
<td>1548</td>
<td>781</td>
<td>1426</td>
<td>0</td>
<td>239</td>
</tr>
<tr>
<td>Houston</td>
<td>1087</td>
<td>1842</td>
<td>1627</td>
<td>810</td>
<td>1187</td>
<td>239</td>
<td>0</td>
</tr>
</tbody>
</table>
Other Representations?

What are some other representations of multi-dimensional arrays?
Motivations

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},
};
Declare/Create Two-dimensional Arrays

// Declare array ref var
dataType[][] refVar;

// 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];

// Alternative syntax
dataType refVar[][] = new dataType[10][10];
Declaring Variables of Two-dimensional Arrays and Creating Two-dimensional Arrays

```java
int[][] matrix = new int[10][10];
    or
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
    for (int j = 0; j < matrix[i].length; j++)
        matrix[i][j] = (int)(Math.random() * 1000);

double[][] x;
```
Two-dimensional Array Illustration

matrix = new int[5][5];

matrix[2][1] = 7;

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

matrix.length? 5
matrix[0].length? 5

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}
};
```

Same as

```
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;
```
Lengths of Two-dimensional Arrays

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

The `x` array is a two-dimensional array with 3 rows and 4 columns. Each `x[i][j]` represents an element in the array, with `x.length` indicating the number of rows and `x[0].length` indicating the number of columns for each row. For example, `x[0][3]` represents the element at the first row and the third column.
Lengths of Two-dimensional Arrays, cont.

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

array.length
array[0].length
array[1].length
array[2].length
array[3].length
array[4].length

ArrayIndexOutOfBoundsException
Ragged Arrays

Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as a *ragged array*. For example,

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

- `matrix.length` is 5
- `matrix[0].length` is 5
- `matrix[1].length` is 4
- `matrix[2].length` is 3
- `matrix[3].length` is 2
- `matrix[4].length` is 1

Redundant text: Ragged Arrays

Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as a *ragged array*. For example,

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

- `matrix.length` is 5
- `matrix[0].length` is 5
- `matrix[1].length` is 4
- `matrix[2].length` is 3
- `matrix[3].length` is 2
- `matrix[4].length` is 1
```java
int[][] triangleArray = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```
Initializing arrays with input values

```java
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

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

```java
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

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

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);
}
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;
    }
}
}
Passing Two-Dimensional Arrays to Methods
Problem: Grading Multiple-Choice Test

Students’ answer

<table>
<thead>
<tr>
<th>Student</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A B A C C D E E A D</td>
</tr>
<tr>
<td>1</td>
<td>D B A B C A E E A D</td>
</tr>
<tr>
<td>2</td>
<td>E D D A C B E E A D</td>
</tr>
<tr>
<td>3</td>
<td>C B A E D C E E A D</td>
</tr>
<tr>
<td>4</td>
<td>A B D C C D E E A D</td>
</tr>
<tr>
<td>5</td>
<td>B B E C C D E E A D</td>
</tr>
<tr>
<td>6</td>
<td>B B A C C D E E A D</td>
</tr>
<tr>
<td>7</td>
<td>E B E C C D E E A D</td>
</tr>
</tbody>
</table>

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

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

PassTwoDimensionalArray Run
Problem: Finding Two Points Nearest to Each Other

http://www.cs.armstrong.edu/liang/animation/web/ClosestPair.html
Multidimensional Arrays

Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays for any integer n.

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 $\geq 3$. 
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}}
};
Problem: Calculating Total Scores

Objective: write a program that calculates the total score for students in a class. Suppose the scores are stored in a three-dimensional array named scores. The first index in scores refers to a student, the second refers to an exam, and the third refers to the part of the exam. Suppose there are 7 students, 5 exams, and each exam has two parts--the multiple-choice part and the programming part. So, scores[i][j][0] represents the score on the multiple-choice part for the i’s student on the j’s exam. Your program displays the total score for each student.
Problem: 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.

```
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)
Problem: 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.
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).
What is Sudoku?

http://www.cs.armstrong.edu/liang/animation/web/Sudoku.html
Every row contains the numbers 1 to 9
Every column contains the numbers 1 to 9

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td>7</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.
Every $3 \times 3$ box contains the numbers 1 to 9

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>
Checking Whether a Solution Is Correct

PassTwoDimensionalArray

Run
Processing Two-Dimensional Arrays

See the examples in the text.

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. (Finding the smallest index of the largest element)
7. (Random shuffling)