Announcements

• PS 5 is due 11:59pm this Thursday

• Reading assignment for this lecture: Chapter 8
Multidimensional Arrays
Motivation

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></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>
Motivation (cont.)

def 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, 1549, 781, 1426, 0, 239},
    {1087, 1842, 1627, 810, 1187, 239, 0},
};
Declare/Create 2D 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];
Declaring variables of 2D arrays and creating 2D arrays

```java
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);
    }
}

// another example
double[][] x;
```
2D array illustration

(a) matrix = new int[5][5];
matrix.length? 5
matrix[0].length? 5

(b) matrix[2][1] = 7;

(c) 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 2D array. For example,

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

Same as

```java
int[][] a = new int[4][3];
a[0][0] = 1; a[0][1] = 2; a[0][2] = 3;
a[1][0] = 4; a[1][1] = 5; a[1][2] = 6;
a[2][0] = 7; a[2][1] = 8; a[2][2] = 9;
a[3][0] = 10; a[3][1] = 11; a[3][2] = 12;
```
Lengths of 2D arrays

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

- `x[0]`: Length is 4
- `x[1]`: Length is 4
- `x[2]`: Length is 4
- `x.length`: Length is 3
Lengths of 2D arrays (cont.)

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

a[4].length  // ArrayIndexOutOfBoundsException
```
Ragged arrays (optional)

Each row in a 2D 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[][] m = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```

m.length is 5
m[0].length is 5
m[1].length is 4
m[2].length is 3
m[3].length is 2
m[4].length is 1
**Ragged arrays** (optional) (cont.)

```java
int[][][] triangleArray = {
    {1, 2, 3, 4, 5},
    {2, 3, 4, 5},
    {3, 4, 5},
    {4, 5},
    {5}
};
```
Processing 2D 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)
Initializing arrays with input values

```java
java.util.Scanner input = new Scanner(System.in);
System.out.println("Enter " + m.length + " rows and " +
m[0].length + " columns: ");

for (int i = 0; row < m.length; i++) {
    // i for row
    for (int j = 0; j < m[i].length; j++) {
        // j for column
        m[i][j] = input.nextInt();
    }
}
```
Initializing arrays with random values

```java
for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < m[i].length; j++) {
        m[i][j] = (int)(Math.random() * 100);
    }
}
```
Printing arrays

for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < m[i].length; j++) {
        System.out.print(m[i][j] + " ");
    }
    System.out.println();
}
int total = 0;
for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < m[i].length; j++) {
        total = total + m[j][j];
    }
}

Summing elements by column

```java
for (int j = 0; j < m[0].length; j++) {
    int total = 0;
    for (int i = 0; i < m.length; i++) {
        total = total + m[i][j];
    }
    System.out.println("Sum for column " + j + " is " + total);
}
```
Random shuffling

for (int i = 0; i < m.length; i++) {
    for (int j = 0; j < m[i].length; j++) {
        int i1 = (int)(Math.random() * m.length);
        int j1 = (int)(Math.random() * m[i].length);
        // Swap m[i][j] with m[i1][j1]
        int temp = m[i][j];
        m[i][j] = m[i1][j1];
        m[i1][j1] = temp;
    }
}
}
### Grading multiple-choice test (next)

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

<table>
<thead>
<tr>
<th>Students’ answers</th>
<th>Objective: write a program that grades multiple-choice test.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 0</td>
<td>A B A C C D E E A D</td>
</tr>
<tr>
<td>Student 1</td>
<td>D B A B C A E E A D</td>
</tr>
<tr>
<td>Student 2</td>
<td>E D D A C B E E A D</td>
</tr>
<tr>
<td>Student 3</td>
<td>C B A E D C E E A D</td>
</tr>
<tr>
<td>Student 4</td>
<td>A B D C C D E E A D</td>
</tr>
<tr>
<td>Student 5</td>
<td>B B E C C D E E A D</td>
</tr>
<tr>
<td>Student 6</td>
<td>B B A C C D E E A D</td>
</tr>
<tr>
<td>Student 7</td>
<td>E B E C C D E E A D</td>
</tr>
</tbody>
</table>

**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
```
Finding 2 points nearest to each other

\[ (-1, 3) \]
\[ (1, 1) \]
\[ (2, 0.5) \]
\[ (3, 3) \]
\[ (4, 2) \]
\[ (4, -0.5) \]
\[ (-1, -1) \]
\[ (2, -1) \]
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
```
Every row contains the numbers 1 to 9
Every column contains the numbers 1 to 9
Every 3×3 box contains the numbers 1 to 9
Checking whether a solution is correct
nD arrays (optional)

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 2D array variables and create 2D arrays can be generalized to declare nD array variables and create nD arrays for n > 2.
3D arrays (optional)

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

scores[i][j][k]
Calculating total scores (optional)

Write a program that calculates the total score for students in a class. Suppose the scores are stored in a 3D 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.
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 4 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
```
Guessing birthday (optional)

Listing 4.3, `GuessBirthday.java`, gives a program that guesses a birthday. The program can be simplified by storing the numbers in 5 sets in a 3D array, and it prompts the user for the answers using a loop.