Multi-Dimensional Arrays

1-Dimentional and 2-Dimentional Arrays

In the previous chapter we used 1-dimensional arrays to model linear collections of elements.

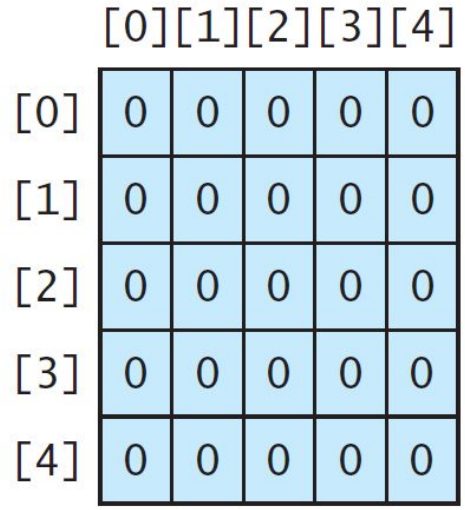
myArray:

6	4	1	9	7	3	2	8

Now think of each element in the array to be a 1-dimentional array. This gives us a matrix.

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

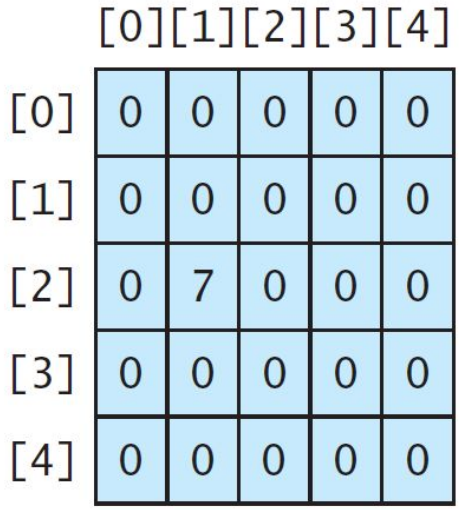
Two-dimensional Array Illustration



	[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];
```

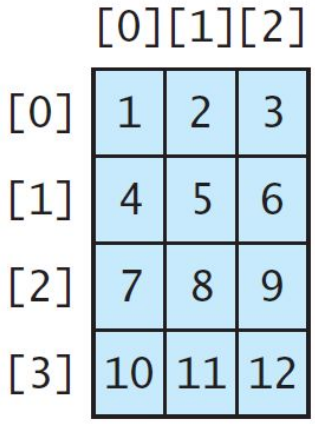
(a)



	[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;
```

(b)



	[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}
};
```

(c)

matrix.length? 5

matrix[0].length? 5

array.length? 4

array[0].length? 3

Declare/Create Two-dimensional Arrays

```
// Declare array reference variable
dataType[][] refVar;  //each [] represents one dimension

// 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];
```

Code Examples

```
// Note that a matrix has rows and columns. First index
// is for rows and second index for columns.

double[][] distance; //declare matrix distance

distance[0][0] = 295; //assign 295 to position [0,0]


int[][] grades = new int[10][10]; //declare & create

for (int i = 0; i < grades.length; i++) //rows
  for (int j = 0; j < grades[i].length; j++) //columns
    grades[i][j] = (int)(Math.random() * 100);

for (int i = 0; i < 10; i++) //process rows
{ for (int j = 0; j < 10; j++) //process columns
     System.out.print ("  " + grades[i][j]);
  System.out.println();
}
```

Initialization Using Shorthand Notations

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

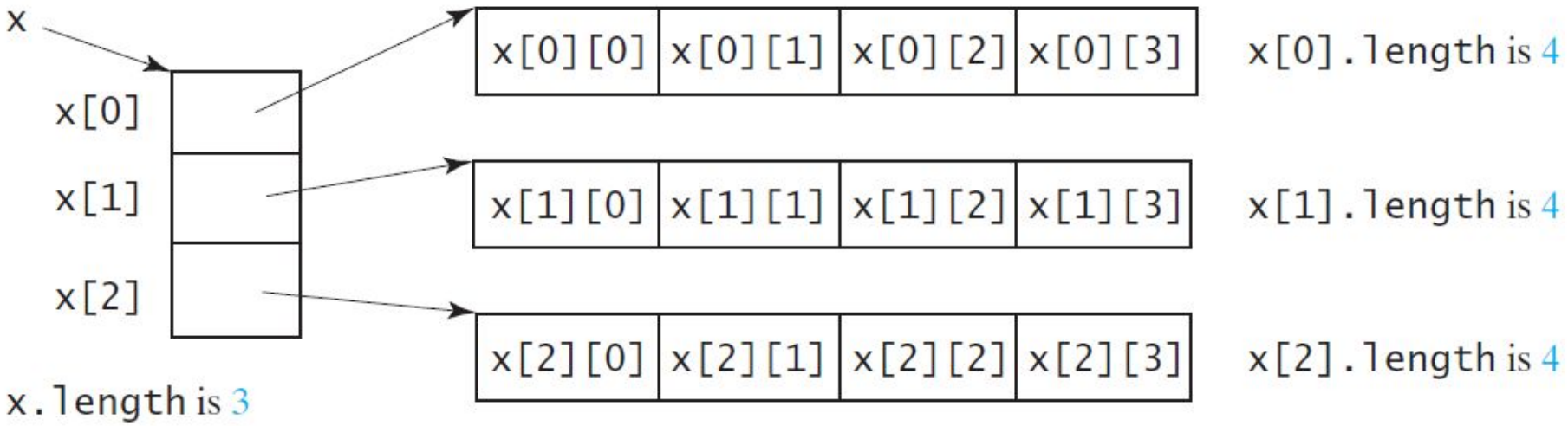
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

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];
```



Lengths of Two-dimensional Arrays

```
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
```

Runtime Error:

```
array[4].length; //ArrayIndexOutOfBoundsException
```

Ragged Arrays

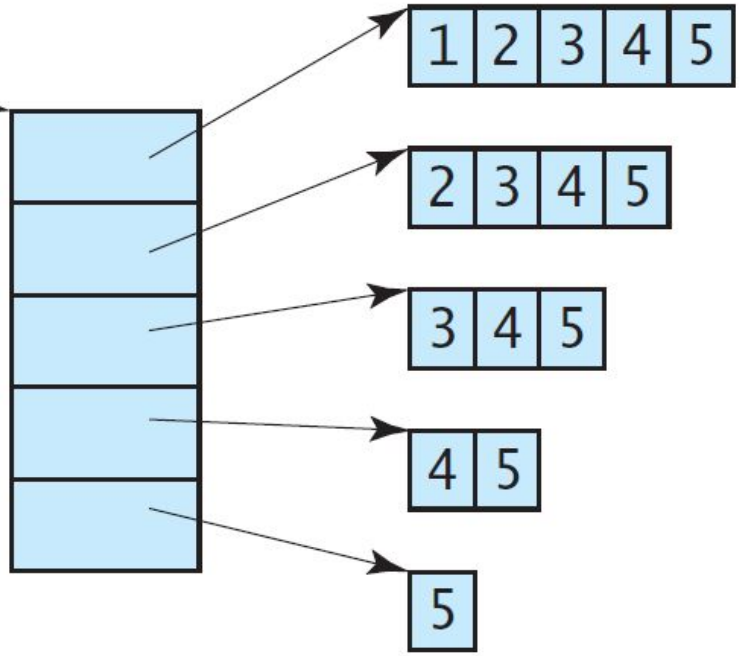
Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as ***ragged array***. For example,

```
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
```

Ragged Arrays, cont.

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



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

Initializing arrays with input values

```
java.util.Scanner input = new Scanner(System.in);

int[][] grades = new int[10][10];
System.out.println("Enter " + grades.length + " rows
      and " + grades[0].length + " columns: ");
for (int row = 0; row < grades.length; row++) {
  for (int column = 0; column < grades[row].length; column++)
  {
    grades[row][column] = input.nextInt();
  }
}
```

Initializing arrays with random values

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

Printing arrays

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

Summing all elements

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

```
import java.util.Scanner;
public class PassTwoDimensionalArray {
  public static void main(String[] args) {
    int[][] table = getArray(); // call method getArray()
    // Display sum of elements
    System.out.println("\nSum of all elements is " + sum(table));
  }

  public static int[][] getArray() {
    Scanner input = new Scanner(System.in);// Create a Scanner
    int[][] m = new int[3][4];// declare and create array m
    System.out.println("Enter " + m.length + " rows and " +
                       m[0].length + " columns: "); //prompt
    for (int i = 0; i < m.length; i++)
      for (int j = 0; j < m[i].length; j++)
        m[i][j] = input.nextInt();
    return m;
  }
// code continues next slide
```

Passing Two-Dimensional Arrays to Methods

```
// code continues from previous slide

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

    return total;
  }
}
```

Problem: Grading 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

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

Problem: Grading Multiple-Choice Test

```
public class GradeExam {
  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'};

    // code continue next slide
```

Problem: Grading Multiple-Choice Test

```
  // code continues from previous slide

    // Grade all students
    for (int i = 0; i < answers.length; i++)
    {
       // Grade one student
       int correctCount = 0; // reset count for each student
       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);
    }
  }
}
```

Multidimensional Arrays

Occasionally, we 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 >= 3.

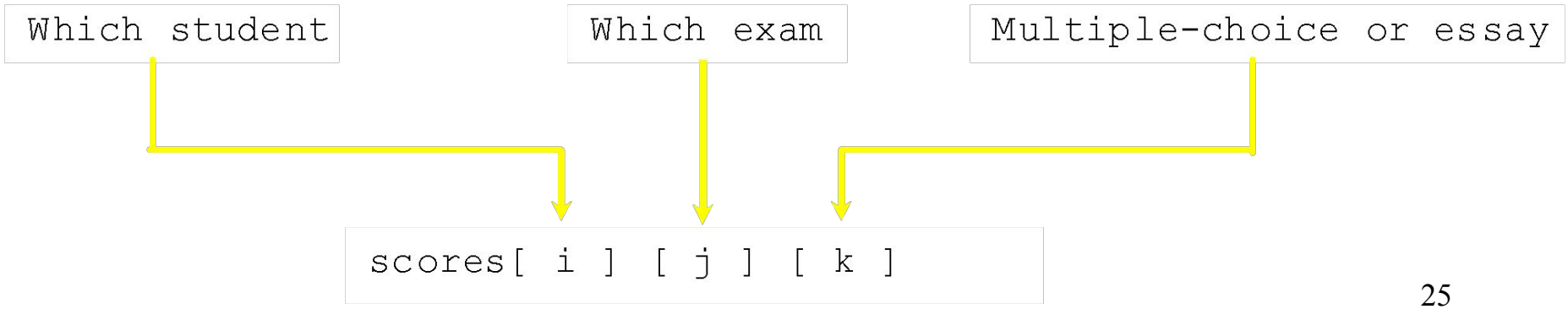
Problem: Calculating Total Scores

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.

The program displays the total score for each student.

3-Dimensional 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

```
public class TotalScore
{
 //Main method
 public static void main(String args[]) {
   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} },
     { {1.5, 29.5}, {6.4, 22.5}, {14, 30.5}, {10, 30.5}, {16, 6.0} } };

  // Calculate and display total score for each student
  for (int i = 0; i < scores.length; i++) {
    double totalScore = 0;
    for (int j = 0; j < scores[i].length; j++)
      for (int k = 0; k < scores[i][j].length; k++)
        totalScore = totalScore + scores[i][j][k];

    System.out.println("Student " + i + "'s score is " + totalScore);
    }
  }
}
```

See Listing 8.5 for another example - Weather data (day/hour/temperature/Humidity