Multidimensional Arrays
Chapter 8

Course: CPSC 1150

Instructor: Dr. Bita Shadgar

Lecture 17

Learning Outcomes

- Declare variables for two-dimensional arrays,
- Create and access array elements in a two-dimensional array using row and column indexes
- Program common operations for two-dimensional arrays
- Pass two-dimensional arrays to methods
- Return two-dimensional arrays from a method

The Arrays class

- Use by importing java.util.Arrays
- Contains methods that perform some common actions on (one-dimensional) arrays
- These methods can be used on an array of any primitive type

Useful methods in the Arrays class

Method in `Arrays` class	What it does...
`sort(arr)`	sorts `arr`
`parallelSort(arr)`	sorts `arr` more efficiently[1]
`sort(arr, start, end)`	partial sort, beginning at `start`, until `end - 1`
`parallelSort(arr, start, end)`	
`binarySearch(arr, key)`	returns index of `key` in `arr` or `-(ins + 1)` if not found
`equals(arr1, arr2)`	returns `true` if all elements equal
`fill(arr, val)`	sets each element of `arr` to `val`
`fill(arr, start, end, val)`	partial fill, from index `start` until `end - 1`
`toString(arr)`	returns a `String` representation

[1] **Only if the computer has multiple processors**

Problem– Distance Table

- The following table that describes the distances between the cities.
- **Question :** How can we store these data on computer?
- **Answer:** use 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

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 and Creating Two-dimensional Arrays

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

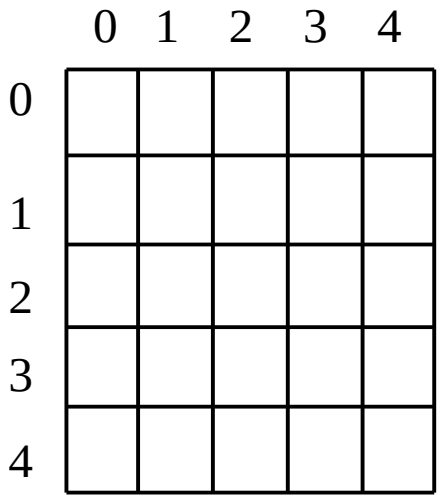
Or

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

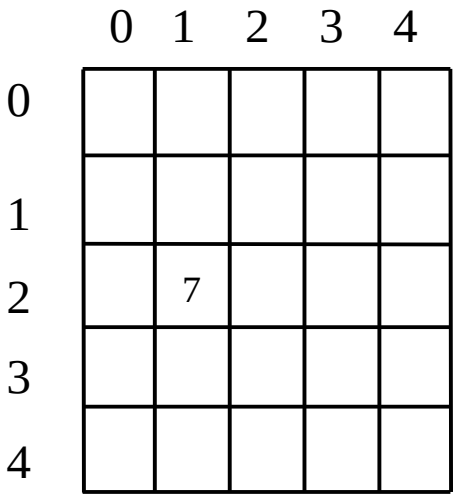
Example – initializing 2D array with input values

```
Scanner iput = new  Scanner(System.in);
System.out.println("Enter " + m.length + " rows and " +
                                        m[0].length + " columns: ");
for (int i = 0; i < matrix.length; i++)
   for (int j = 0; j < matrix[i].length; j++)
         matrix[i][j] = input.nextInt();
```

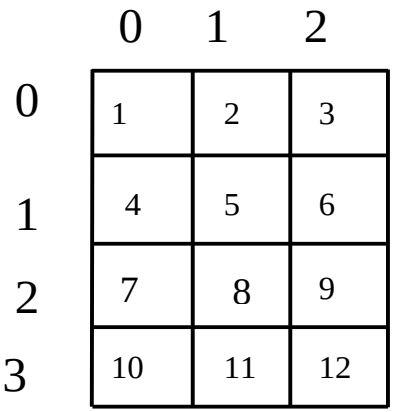
Two-dimensional Array Illustration




```
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,

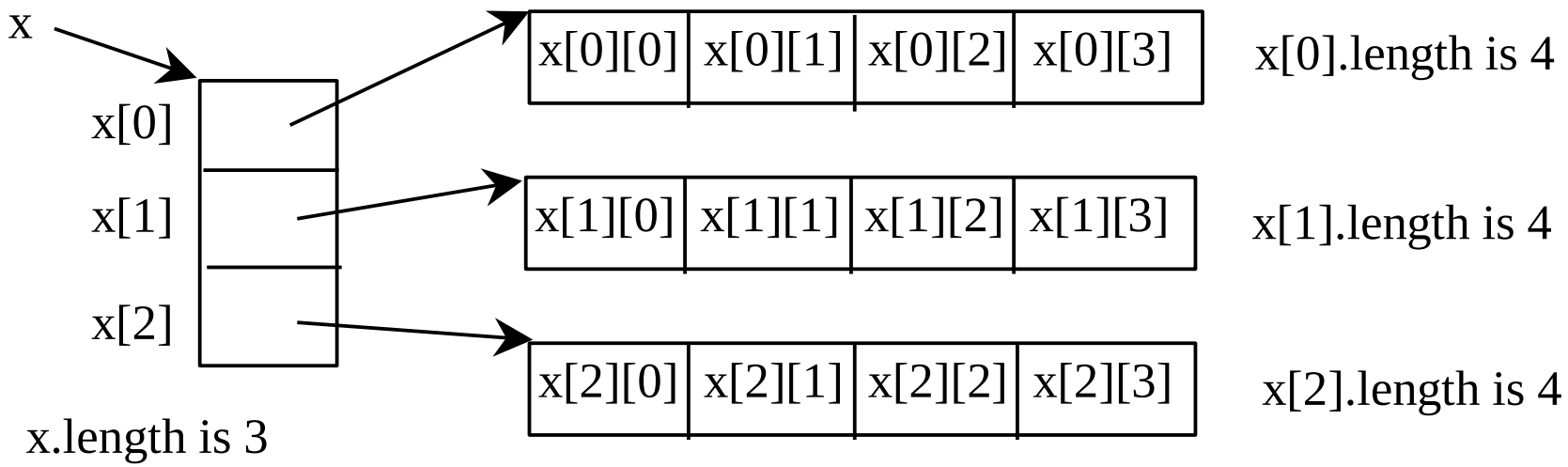
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

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 2D arrays

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



Lengths of 2D 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
- The rows can have different lengths. Such an array is known as *a ragged array*

Example – Initializing and printing ragged array

```
int[][] matrix = {
  {1, 2, 3, 4},
  {2, 3, 4},
  {3, 4},
  {4},
};
// displaying 2D arrays
for (int i=0; i < myList.length; ++i){
   for(int element : myList[i])
      System.out.print(element + " ");
   System.out.println();
}
```

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

Processing Two-Dimensional Arrays

See the examples in the text.

1. (Initializing arrays with random numbers)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)
5. (Which row has the largest sum)
6. (Finding the index of the largest element)
7. (*Random shuffling*)

Passing 2D arrays

Invoking a method with 2D arrays as its argument

```
public static void main(String[] args) {
   int[][] matrix = gen2DArray(); // Get an array
   print2DArray(matrix);
}
```

Defining a method with 2D arrays as its parameter

```
public static void print2DArray(int [][] m) {
     for (int i = 0; i < m.length; ++i){
          for (int j = 0; j < m[i].length; ++j)
               System.out.printf("%5d", m[i][j]);
          System.out.println();
    }
}
```

2D array as return value

Invoking a method with 2D arrays as its return value

```
public static void main(String[] args) {
   int[][] matrix = gen2DArray(); // Get an array
   print2DArray(matrix);
}
```

Defining a method with 2D arrays as its return value

```
public static int[][] gen2DArray() {
   // initialize array values
   int[][] m = new int[3][4];
   for (int i = 0; i < m.length; i++)
      for (int j = 0; j < m[i].length; j++)
         m[i][j] = (int) (Math.random()*100);
  return m;
 }   //see PassTwoDimensionalArray example
```