FCDS

Programming I

Lecture 11: 2D Arrays

Two-Dimensional Arrays

Two-Dimensional Arrays

- A one-dimensional array stores a list of elements.
- A two-dimensional array can be thought of as a ***table*** of elements, with rows and columns.
- For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

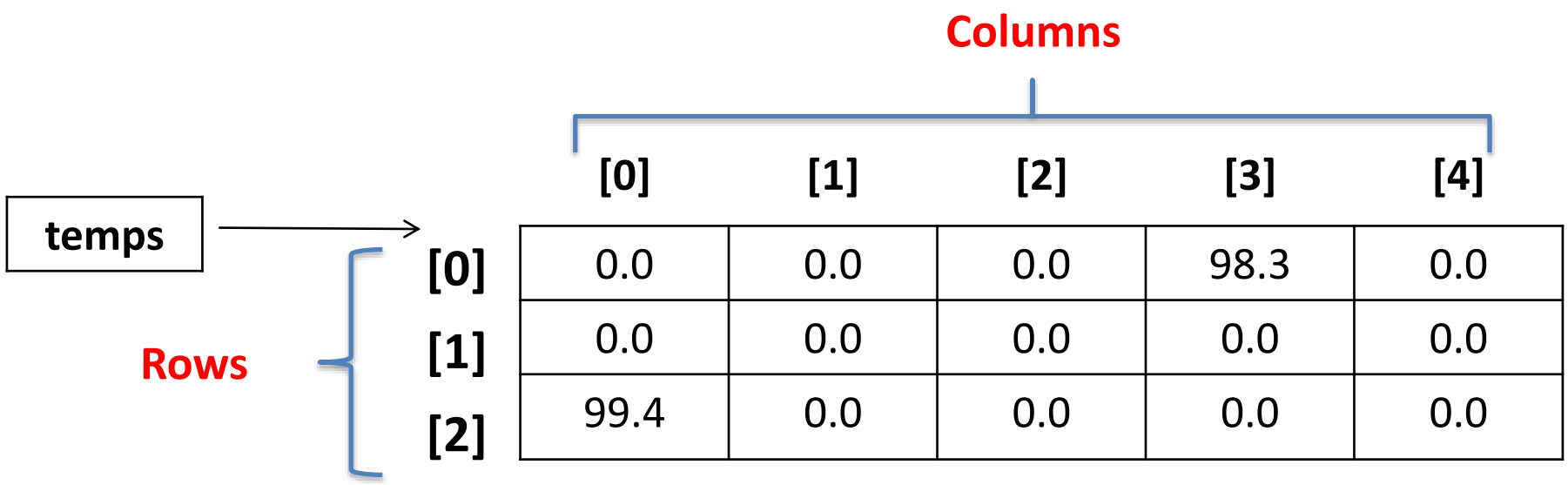
Distance Table (in miles)

	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-Dimensional Arrays

```
double[][] temps = new double[3][5];

temps[0][3] = 98.3;
temps[2][0] = 99.4;
```



Two-Dimensional Arrays (Quick Initialization)

```
double[][] temps = {
        {0.1, 0.2, 0.3, 0.4, 1.0},
        {0.1, 0.2, 0.3, 0.4, 2.0},
        {0.1, 0.2, 0.3, 0.4, 3.0},
};
```

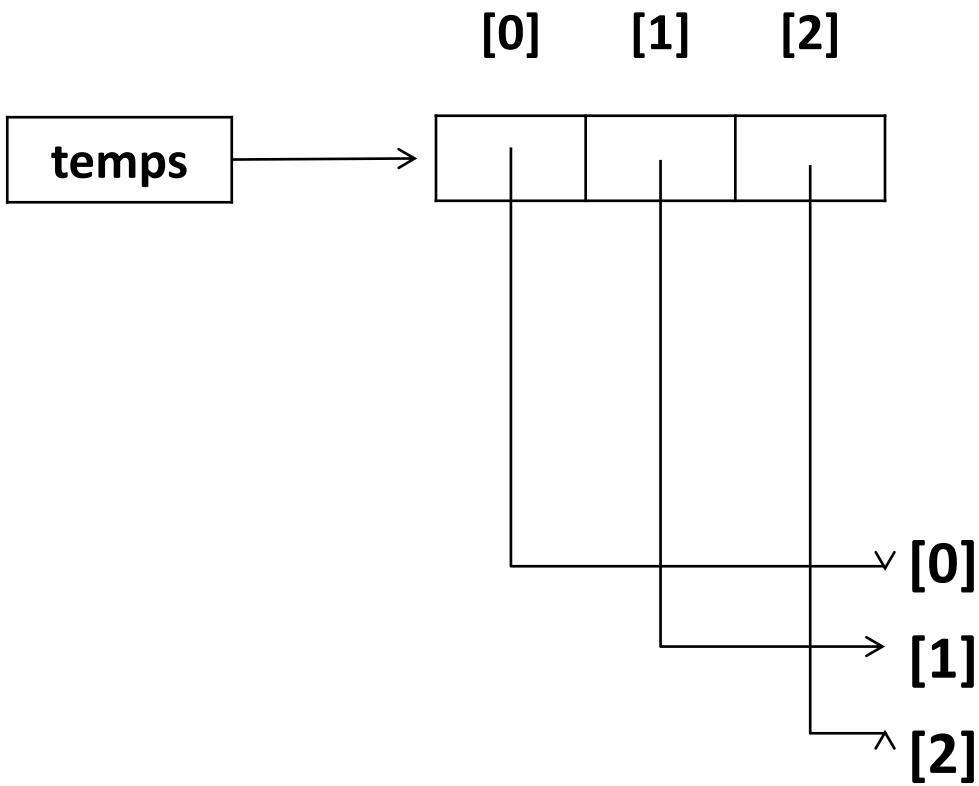
temps →

	[0]	**[1]**	**[2]**	**[3]**	**[4]**
[0]	0.1	0.2	0.3	0.4	1.0
[1]	0.1	0.2	0.3	0.4	2.0
[2]	0.1	0.2	0.3	0.4	3.0

Two-Dimensional Arrays

```
double[][] temps = new double[3][5];
```

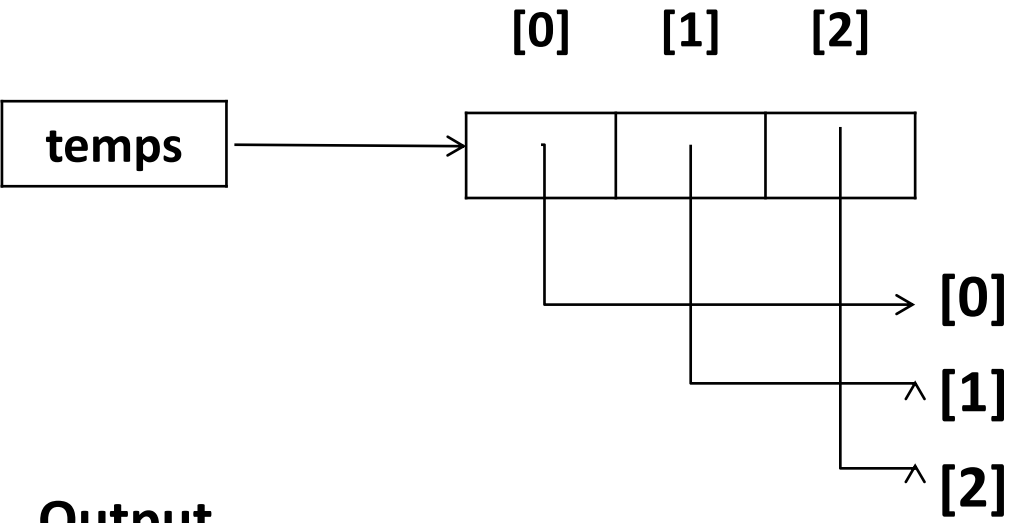
temps the entire grid
temps[2] the entire third row
temps[2][0] the first element of the third row



	[0]	[1]	[2]	[3]	[4]
[0]	0.0	0.0	0.0	0.0	0.0
[1]	0.0	0.0	0.0	0.0	0.0
[2]	0.0	0.0	0.0	0.0	0.0

Two-Dimensional Arrays

```
public static void main(String[] args) {
    double[][] temps = new double[3][5];
    System.out.println(temps);
    System.out.println(temps[0]);
    System.out.println(Arrays.toString(temps));
    System.out.println(Arrays.toString(temps[0]));
    System.out.println(Arrays.deepToString(temps));
}
```



[0]	[1]	[2]	[3]	[4]
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0

Output

```
[[D@4b71bbc9
[D@17dfafd1
[[D@17dfafd1, [D@5e8fce95, [D@3343c8b3]
[0.0, 0.0, 0.0, 0.0, 0.0]
[[0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0]]
```

Two-Dimensional Arrays - length

```
double[][] temps = new double[3][5];
System.out.println(temps.length);
```

No. of rows

```
System.out.println(temps[0].length);
```

No. of columns

temps

	[0]	[1]	[2]	[3]	[4]
[0]	0.0	0.0	0.0	98.3	0.0
[1]	0.0	0.0	0.0	0.0	0.0
[2]	99.4	0.0	0.0	0.0	0.0

Output

```
3
5
```

Two-Dimensional Arrays – as Parameters

```
public static void main(String[] args) {
    double[][] temps = new double[3][5];
    temps[0][1] = 10;
    temps[2][4] = 11;
    print(temps);
}

public static void print (double[][] grid) {
    for(int i = 0; i< grid.length; i++) {
        for(int j = 0; j < grid[i].length; j++) {
            System.out.print(grid[i][j]+ “  “);
        }
        System.out.println();
    }
}
```

Output

```
0.0   10.0  0.0   0.0   0.0
0.0   0.0   0.0   0.0   0.0
0.0   0.0   0.0   0.0   11.0
```

2D Arrays - Errors

```
public class ArrayTest {

    public static void main(String[] args) {
        int[][] triangle = new int[6][2];
        System.out.println(triangle[6][0]);
    }
}
```

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 6 at ArrayTest.main(ArrayTest.java:5)

```
public class ArrayTest {

    public static void main(String[] args) {
        int[][] triangle = new int[6][];
        System.out.println(triangle[0][0]);
    }
}
```

Exception in thread "main" java.lang.NullPointerException at ArrayTest.main(ArrayTest.java:5)

Two-Dimensional Array parameter questions

- Write a method `ArrayRowSum` that accepts a 2D array of integers and a row number and returns the sum of elements at this row.

```
int[][] a = {{12, 34}, {50, 18}, {30, 26}};
int sum = ArrayRowSum(a, 1);
System.out.println(sum); // 68

// Sum the elements at row r in array a.
public static int ArrayRowSum(int[][] a, int r) {
  int sum = 0;
  for(int j = 0; j < a[r].length; j++) {
        sum += a[r][j];
     }
  return sum;
}
```

Two-Dimensional Array parameter answers

- Write a method `ArraySum` that accepts a 2D array of integers and returns the sum of all its elements.

```
int[][] a = {{12, 34}, {50, 18}, {30, 26}};
int sum = ArraySum(a);
System.out.println(sum); // 170

// Sums the entire contents of array a.
public static int ArraySum(int[][] a) {
  int sum = 0;
  for(int i = 0; i< a.length; i++) {
     for(int j = 0; j < a[i].length; j++) {
        sum += a[i][j];
     }
  return sum;
}
```

Matrix Diagonal Exercise

Matrix diagonal. Given an n-by-n matrix **a**, find the diagonal of **a** and store it in vector **b**.

```
public static void main(String[] args) {
    int[][] a = {{1,2,3},{4,5,6},{7,8,9}};
    int[] b = MatrixDiagonal(a);
    printVector(b);
}
public static int[] MatrixDiagonal(int[][] a) {
    int n = a.length;
    int[] b = new int[n];
    for(int i = 0; i < n; i++) {
        b[i] = a[i][i];
    }
    return b;
}
public static void printVector (int[] vec) {
    for(int i = 0; i< vec.length; i++) {
        System.out.print(vec[i] + " ");
     }
     System.out.println();
}
```

n-1-i

1	2	3
4	5	6
7	8	9

What about finding the secondary diagonal?

1	2	3
4	5	6
7	8	9

Matrix Addition Exercise

Matrix addition. Given two n-by-n matrices **a** and **b**, define **c** to be the n-by-n matrix where **c[i][j]** is the sum **a[i][j] + b[i][j]**.

```
public static void main(String[] args) {
    int[][] a = {{1,2},{3,4}};
    int[][] b = {{1,1},{1,1}};
    int[][] c = addMatrix(a,b);
    print(c);
}
public static int[][] addMatrix(int[][] a, int[][] b) {
    int n = a.length;
    int[][] c = new int[n][n];
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            c[i][j] = a[i][j] + b[i][j];
        }
    }
    return c;
}
public static void print (int[][] matrix) {
    for(int i = 0; i< matrix.length; i++) {
        for(int j = 0; j < matrix[i].length; j++) {
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }
}
```