### **Two Dimensional Arrays**

Data in a table or matrix can be stored in a 2-D array

|  | **Chicago** | **Boston** | **New York** |
| --- | --- | --- | --- |
| **Chicago** | 0 | 983 | 787 |
| **Boston** | 983 | 0 | 210 |
| **New York** | 787 | 210 | 0 |

Such data can stored as

int[][] distances = {

{ 0, 983, 787 },

{ 983, 0, 210 },

{ 787, 210, 0 }

};

### **2D Array Basics**

Declaring 2D Array:

int[][] distances;

or

int distances[][]; // but this one is not recommend

Creating 2D Arrays:

distances = new int[3][3];

Accessing 2D Arrays:

You need to use 2 indexes.

The first represents the row and the second the column.

distances[2][1] = 214;

distances[1][2] = 214;

A better way to do this is to declare constants:

final int CHICAGO = 0;

final int BOSTON = 1;

final int NEWYORK = 2;

Then set as follows:

distances[NEWYORK][BOSTON] = 214;

distances[BOSTON][NEWYORK] = 214;

### **Obtaining Lengths**

A 2D array is actually a 1D array of 1D arrays.

Assuming

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

Then the following equals 3:

x.length

and any of the following equal 4:

x[0].length

x[1].length

x[2].length

### **Ragged Arrays**

A 2D array can have rows of different sizes

This is know as a **ragged array**.

For example:

int[][] triangleArray = {

{ 1, 2, 3, 4 },

{ 2, 3, 4 },

{ 3, 4 },

{ 4 }

};

Note that triangleArray.length is 4.

And triangleArray[0].length is 4, triangleArray[1].length is 3, triangleArray[2].length is 2, triangleArray[3].length is 1.

Another way to initialize this is

int[][] triangleArray = new int[4][];

triangleArray[0] = new int[4];

triangleArray[1] = new int[3];

triangleArray[2] = new int[2];

triangleArray[3] = new int[1];

Note that with this approach the number of rows needs to specified in the declaration (with int[4][]).

### **Processing 2D Arrays**

It is most natural to use two nested for loops to iterate over 2D arrays.

Initializing a 2D array with a value, val:

for (int r = 0; i < distances.length; r++)

for (int c = 0; c < distances[r].length; c++)

distances[r][c] = val;

Printing a 2D array:

for (int r = 0; i < distances.length; r++) {

for (int c = 0; c < distances[r].length; c++)

System.out.printf("%d\t", distances[r][c]);

System.out.println();

}

How would you count all values in a 2D array?

How would sum the values of each row of a 2D array?

### **Passing 2D Arrays to Methods**

2D arrays are passed into methods by reference.

2D array parameters are declared as:

public static void someMethod(int[][] arr) {

...

}

2D arrays are returned as:

public static int[][] someOtherMethod() {

...

ret arr;

}

### **Multi-dimensional Arrays**

A 3D array is an 1D array of 2D arrays

For example

double[][][] scores = new double[6][5][2];