**2 Dimensional Arrays**

**Declaring a 2D Array** xxxx

int[][] nums = new int [3][4]; This creates a 2D array with 3 rows and 4 columns xxxx

xxxx

int[][] nums = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }; This creates a 3x3 2D array which looks like: 123

456

789

You can think of 2D arrays as an *array* *of arrays.*

**Number of Rows**

int numRows = nums.length; (how many arrays are there?)

**Number of Columns**

int numCols = nums[0].length (How many elements in the first array (row)?

**Populating/Traversing a 2D Array**

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

{

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

{ 123

System.out.println( nums[r][c] ); 456

} 789

}

**int**[][] nums = **new** **int**[5][6];

**for**(**int** r=0; r < nums.length; r++)

**for**(**int** c=0; c < nums[0].length; c++)

nums[r][c]= (r + (2\*c)) % 10;

The resulting 2D array looks like:

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

**Commonly Used Code** - The code you write should work for any sized 2D Array of Integers.

**Outputting a 2D Array**

**Sum all Numbers in a 2D Array**

**Average all Numbers in a 2D Array**

**Add the sum of the first column to the sum of last row**

**Find the sum of the 2 Diagonals, but only count the middle value once. (Assume same number of rows and cols)**

**2D Array Programs**

Write a method called isWinner that receives a 3 x 3 2D array of characters and determines if there is a winner for Tic Tac Toe. You can assume that the array is either populated with a space, 'X', or 'O'. The method should return a boolean value of true if there is a winner, otherwise false;

**WordSearch**

Create a 20 x 20 2-dimensional array of random upper-case letters.

Using the dictionary ArrayList, try to find all occurrences of words in the dictionary in the 2D array.

Starting by looking horizontally forward, as if you were normally reading.

Then try vertically from top to bottom and progressively try to add in the other six ways that a word could appear in the 2D Array.

**PointDistance**

The user enters a size of an array, with even numbered rows and columns. The program will build a miniature graph from it, starting at a negative value equal to half the number of row, and similarly with the columns. It will go up to half the number of rows and columns. Don't forget 0!

Example: If the user entered 8 rows and 10 columns, a graph with -4 to 4 would be stored in the rows and -5 to 5 would be stored in the columns.

The user is then prompted to enter a distance to search for between points. The program will output all pairs of coordinates that are that distance apart.