2 Dimensional Arrays

- A 2D array is really just a 1 dimensional array of arrays

- `int matrix[][];`

  Here is what gets created in memory

  `matrix: [null]   // ref var allocated with a null in it`

  The reference variable indicates it's a ref var for a 2D array by having a double set of brackets [][] in its declaration.

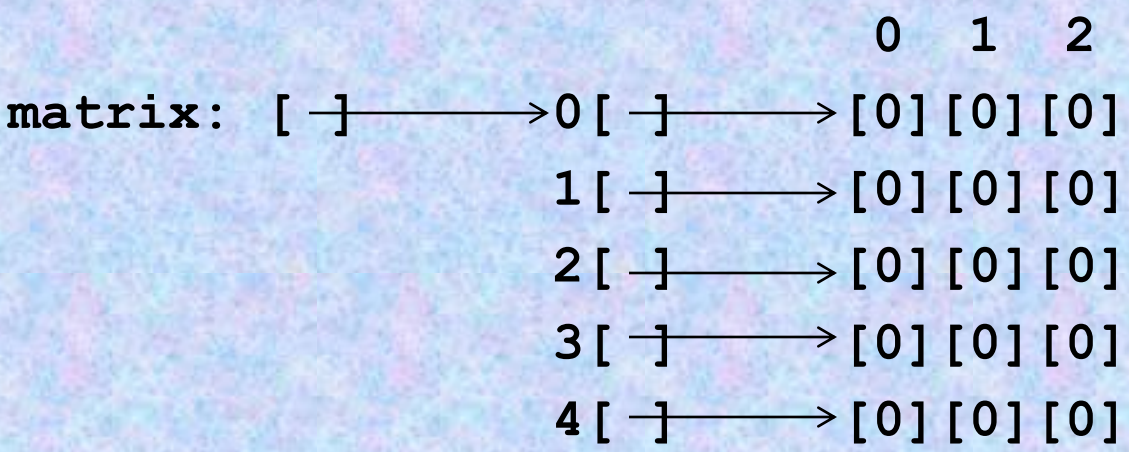
  Note that the above declaration of a ref var does not create any array yet.

- `matrix = new int[5][3];`

  Now we have actually created a 5 row by 3 col array of int.

  The address of where that array lives gets assigned into the ref var.

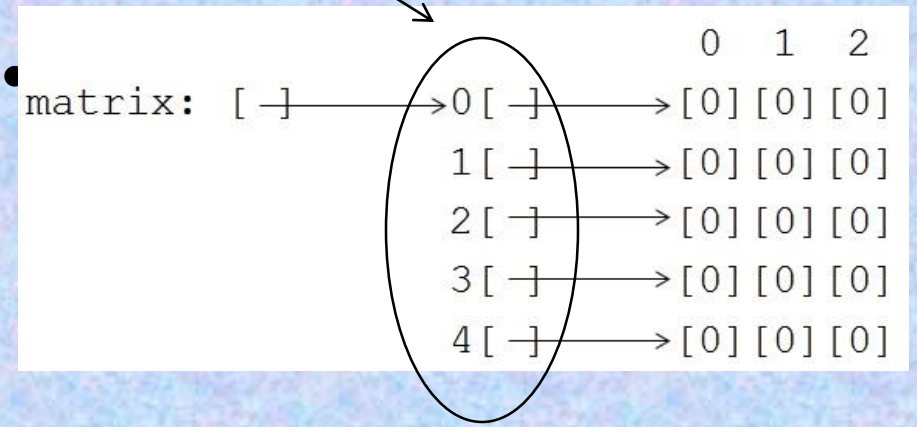
- Here is what memory looks like now:



We usually visualize the array as just the grid part but in fact a 2D array is an array of arrays. Each row of the array is itself a 1D array. You could also say a 2D array is an array of references to 1D arrays

2D Arrays

- The expression matrix refers to this part of memory

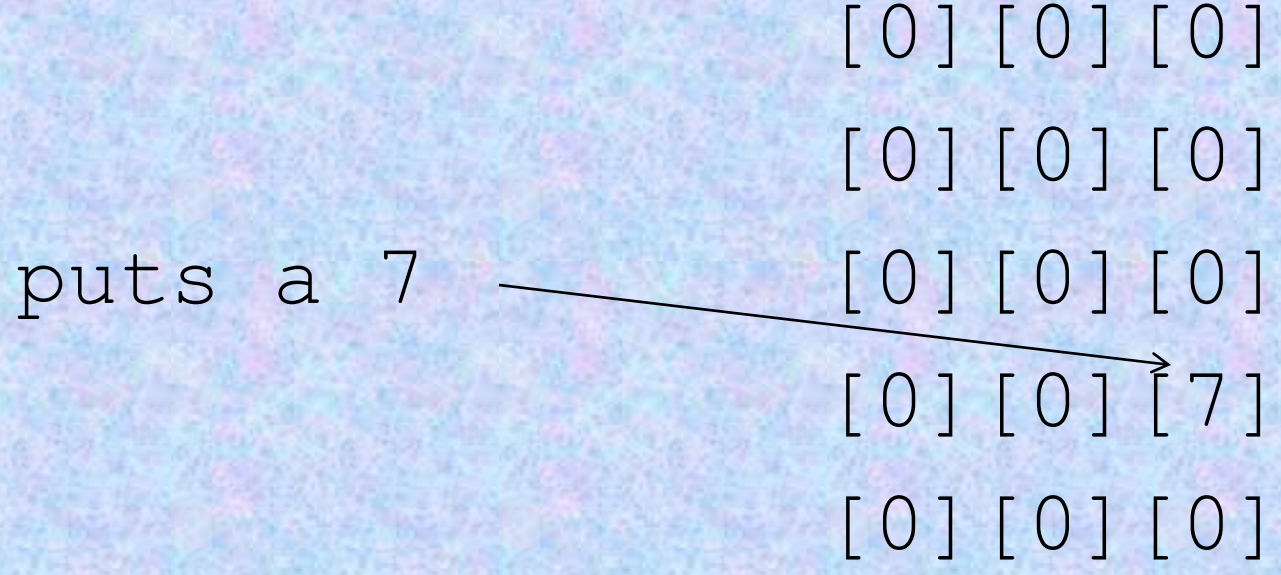


2D Arrays

- matrix.length is 5
- matrix[0] is the ref to the first array.
- matrix[0].length is 3
- To find out how many rows in the entire matrix use matrix.length
- To find the length of a specific row use matrix[row].length

- To access individual elements of the matrix use this syntax:

```
matrix[3][2] = 7
```



2D Arrays

- **Rather than plinking individual values into a matrix with hardcoded indices in solitary assignment statements, arrays and matrices are generally initialized using loops.**

- **Use matrix.length to produce the number of rows. Use matrix[row].length to produce the number of columns in that row.**

2D Arrays

```
int[][] matrix = new int[5][3];

for (int row=0 ; row < matrix.length ; row++ )
       for (int col = 0 ; col < matrix[row].length ; col++)
              matrix[row][col] = row+col;
```