

Worksheet: 2D Array Practice 2

1. Write the statement that will declare a variable and initialize it to a **two-dimensional array**:

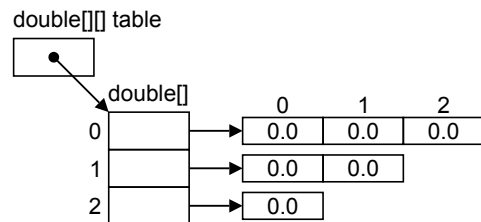
a) of `int` with five rows and six columns.

```
int[][] arr = new double[5][6];
```

b) of `String` with seven rows and no space allocated for columns.

```
String[] arr = new String[7][];
```

2. Using a `for` loop, write a method named `upperTriangular` that takes a single integer parameter, `numRows`, and allocates a two-dimensional array where the number of rows is equal to `numRows`. The first row of the array is equal in length to `numRows`, and each subsequent row is one less than the previous row. For example, if `numRows` is equal to 3, the table to the right is produced.



```
1 public static double[][] upperTriangular(int numRows) {  
2     double[][] table = new double[numRows][];  
3     for(int i = 0; i < numRows; i++) {  
4         table[i] = new double[numRows - i];  
5     }  
6     return table;  
7 }
```

3. Write a **method** named `sumRows` that takes a two-dimensional array of `double` named `matrix` and returns a one-dimensional array that is equal in length to the number of rows of `matrix` where each element in the array contains the sum of all the elements in the corresponding row of `matrix`. You can assume the precondition that all row lengths are equal.

```
1 public static double[] sumRows(double[][] matrix) {  
2     double[] rowSum = new double[matrix.length];  
3     for(int row=0; row<matrix.length; row++) {  
4         for(int col=0; col<matrix[row].length; col++) {  
5             rowSum[row] += matrix[row][col];  
6         }  
7     }  
8     return rowSum;  
9 }
```

Worksheet: 2D Array Practice 2

4. Write a **method** named `sumCols` that takes a two-dimensional array of `double` named `matrix` and returns a one-dimensional array that is equal in length to the number of columns of `matrix` where each element in the array contains the sum of all the elements in the corresponding column of `matrix`. You can assume the precondition that there is at least one row in the matrix, and that all row lengths are equal.

```
1 public static double[] sumCols(double[][] matrix) {
2     double[] colSum = new double[matrix[0].length];
3     for(int row=0; row<matrix.length; row++) {
4         for(int col=0; col<matrix[row].length; col++) {
5             colSum[col] += matrix[row][col];
6         }
7     }
8     return colSum;
9 }
```

5. Write a **method** named `verifyRectangular` that takes a two-dimensional array of `double` named `matrix` and verifies that every row in the two-dimensional array has the same length, returning `true` if it is a rectangular matrix (all row lengths equal), otherwise `false`. You can assume the precondition that there is at least one row in the matrix.

```
// enhanced for loop version
1 public static boolean verifyRegular(double[][] matrix) {
2     int numCols = matrix[0].length;
3     for(double[] row: matrix) {
4         if(row.length != numCols) {
5             return false;
6         }
7     }
8     return true;
9 }

// for loop version
1 public static boolean verifyRegular(double[][] matrix) {
2     int numCols = matrix[0].length;
3     for(int row = 0; row < matrix.length; row++) {
4         if(matrix[row].length != numCols) {
5             return false;
6         }
7     }
8     return true;
9 }
```