**6 kyu**

**Pascal's Triangle #2**

1401080% of 357829of 3,684[Jeroko](https://www.codewars.com/users/Jeroko)

Java

* [TRAIN AGAIN](https://www.codewars.com/kata/pascals-triangle-number-2/train/java)
* [NEXT KATA](https://www.codewars.com/trainer/java)

Details

[Solutions](https://www.codewars.com/kata/pascals-triangle-number-2/solutions/java)

[Forks (1)](https://www.codewars.com/kata/pascals-triangle-number-2/forks/java)

[Discourse (58)](https://www.codewars.com/kata/pascals-triangle-number-2/discuss/java)

* Add to Collection
* |
* Share this kata:

Here you will create the classic pascal's triangle. Your function will be passed the depth of the triangle and you code has to return the corresponding pascal triangle upto that depth

The triangle should be returned as a nested array.

Note: For JavaScript version, your final result is a string representation of the array. Sorry for the inconvenience, test cases are locked so this is the best I could do.

- V

for example:

PascalsTriangle.pascal(5); // should return [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]

To build the triangle, start with a single 1 at the top, for each number in the next row you just take the two numbers above it and add them together (except for the edges, which are all "1"). eg

[1]

[1 1]

[1 2 1]

[1 3 3 1]

here you get the 3 by adding the 2 and 1 above it.

<https://www.codewars.com/kata/pascals-triangle-number-2/java>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

public static List<List<int>> PascalsTriangle(int n)

{

int[,] matriz = new int[n, n];

for (int i = 0; i < n; i++)

{

matriz[i, 0] = 1;

matriz[0, i] = 1;

}

for (int i = 1; i < n; i++)

for (int j = 1; j < n; j++)

matriz[i, j] = matriz[i, j - 1] + matriz[i - 1, j];

List<List<int>> ans = new List<List<int>>();

for (int i = 0; i < n; i++)

{

List<int> f = new List<int>();

int fila = i;

int col = 0;

while (fila >= 0 && col < n)

{

f.Add(matriz[fila, col]);

fila--;

col++;

}

ans.Add(f);

}

return ans;

}

static void Main(string[] args)

{

List<List<int>> ans = PascalsTriangle(6);

for(int i =0; i<ans.Count; i++)

{

for(int j =0; j<ans[i].Count; j++)

{

Console.Write(ans[i][j] + " ");

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}

*/\**

*\* To change this license header, choose License Headers in Project Properties.*

*\* To change this template file, choose Tools | Templates*

*\* and open the template in the editor.*

*\*/*

**package** javaapplication46;

**import** java.util.ArrayList;

***/\*\****

***\****

***\* @author Usuario***

***\*/***

**public** **class** JavaApplication46 {

**public** **static** **int**[][] pascal(**int** depth) {

*//return new int[][] { { 1 } };*

**int**[][] matriz = **new** **int**[depth] [depth];

**for** (**int** i = 0; i < depth; i++)

         {

             matriz[i][0] = 1;

             matriz[0][i] = 1;

         }

**for** (**int** i = 1; i < depth; i++)

**for** (**int** j = 1; j < depth; j++)

                 matriz[i][j] = matriz[i][j - 1] + matriz[i - 1][j];

         ArrayList<ArrayList<Integer>> ans =

**new** ArrayList();

**for** (**int** i = 0; i < depth; i++)

         {

             ArrayList<Integer> f = **new** ArrayList();

**int** fila = i;

**int** col = 0;

**while** (fila >= 0 && col < depth)

             {

                 f.add(matriz[fila][col]);

                 fila--;

                 col++;

             }

             ans.add(f);

         }

**int**[][] r = **new** **int**[ans.size()][];

**for**(**int** i = 0; i<ans.size(); i++) {

             r[i] = **new** **int**[ans.get(i).size()];

**for**(**int** j =0; j < ans.get(i).size(); j++) {

                 r[i][j] = ans.get(i).get(j);

             }

         }

**return** r;

    }

**public** **static** **void** main(String[] args) {

*// TODO code application logic here*

**int**[][] r = pascal(7);

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

**for**(**int** j =0; j<r[i].length; j++) {

                System.out.print(r[i][j] + " ");

            }

            System.out.println();

        }

    }

}

---------------------------------------------

**public** **class** PascalsTriangle {

**public** **static** **int**[][] pascal(**int** depth) {

**int**[][] triangle = **new** **int**[depth][];

**for** (**int** i = 0; i < depth; i++) {

      triangle[i] = **new** **int**[i+1];

      triangle[i][0] = 1;

      triangle[i][i] = 1;

**for** (**int** j = 1; j < i; j++) {

        triangle[i][j] = triangle[i-1][j] + triangle[i-1][j-1];

      }

    }

**return** triangle;

  }

}