

Jagged Matrices – Pascal’s Triangle

Purpose

The goal for this lab is to use matrices in a different way. Pascal’s Triangle is a good example of where we could apply the concept of a “jagged” matrix.

Description

This assignment involves you creating, initializing, and filling a jagged array based on what is known as Pascal’s Triangle.

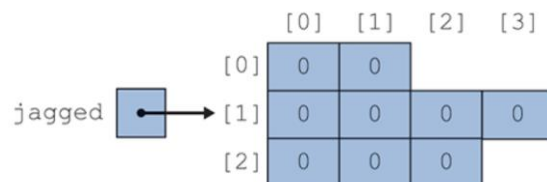
How to Get started

Start with the [Pascals_Triangle.java](#) file, which is a skeleton file with places for you to insert your code.

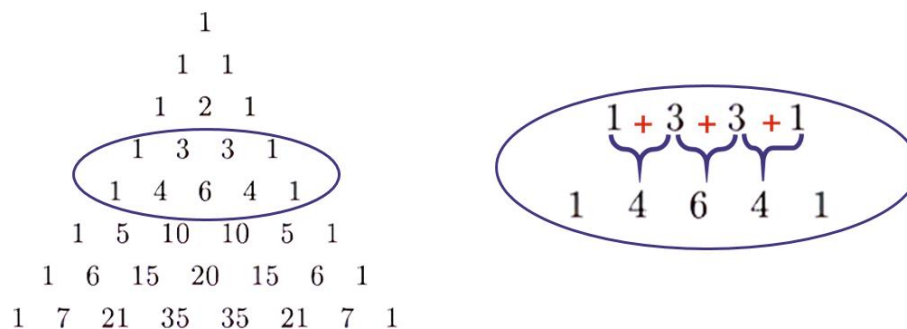
Basic Specifications

The focus for this lab is on jagged arrays. What that essentially means is a matrix that does not have a uniform column size. To setup this structure, declare the matrix by only specifying the rows, for example:

```
int[][] jagged = new int[3][];  
jagged[0] = new int[2];  
jagged[1] = new int[4];  
jagged[2] = new int[3];
```

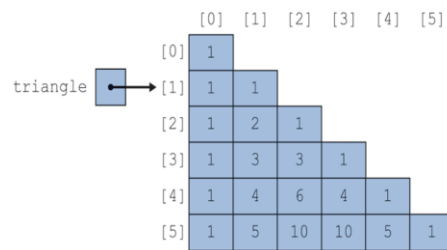


What is Pascal’s Triangle? Pascal’s triangle is a triangular form of growing data, surrounded by 1’s, where any given row can be used to compute the next. This is achieved by adding adjacent pairs of values together:



Comment each method in this file. You are encouraged, but not required, to use standard JavaDoc conventions for this. *Also provide a comment at the top of this file that gives the name of the file, your name, and an explanation for the purpose of this file as part of the whole application.* The methods to provide are the following:

- **pascal(int rows):** |**int[][]**| Returns a jagged matrix that stores Pascal's Triangle with the number of rows specified.
 - Example Structure:



The diagram illustrates the structure of Pascal's Triangle as a jagged array. A blue box labeled 'triangle' has an arrow pointing to the first row of the triangle. The triangle is composed of blue boxes containing numbers, arranged in rows. The columns are indexed from [0] to [5] at the top. The rows are indexed from [0] to [5] on the left. The values in the boxes are: Row [0]: 1; Row [1]: 1, 1; Row [2]: 1, 2, 1; Row [3]: 1, 3, 3, 1; Row [4]: 1, 4, 6, 4, 1; Row [5]: 1, 5, 10, 10, 5, 1.

| | [0] | [1] | [2] | [3] | [4] | [5] |
|-----|-----|-----|-----|-----|-----|-----|
| [0] | 1 | | | | | |
| [1] | 1 | 1 | | | | |
| [2] | 1 | 2 | 1 | | | |
| [3] | 1 | 3 | 3 | 1 | | |
| [4] | 1 | 4 | 6 | 4 | 1 | |
| [5] | 1 | 5 | 10 | 10 | 5 | 1 |

Grading Rubric

In this assignment, you can earn 100 points as follows.

- File Information correctly commented at the top of the file: **5**
- Implementing a dynamic method to output a jagged array filled with Pascal's Triangle: **70** points. If any of these are missing or incorrect, subtract 20 for not having a dynamic function or not having a jagged array.
- **15** points for testing your algorithm in the java file.
- **10** points for writing very clear readable code with **clear comments** in English for each new class, method and function, and static variable. Any other variables should have names suggestive of their meaning, with comments unless the variable is a loop variable (i.e., int i) or a size or length (int n).

Miscellaneous Advice

Check the Google Drive or Schoology to view the slides that go with this lab. printf for bonus...

BONUS

Output the structure as an equilateral triangle: **+5** point