

A decorative graphic on the left side of the slide, consisting of a network of white lines and small circles on a dark blue background, resembling a circuit board or a neural network.

MULTI-DIMENSIONAL ARRAYS

zyBook 6.9

@ Dr. Jessica Young Schmidt and NCSU Computer Science Faculty


MULTI-DIMENSIONAL DATA

- `int` → one integer
- `int[]` → **one**-dimensional array of integers
- `int[][]` → a **two**-dimensional grid of integers
- `int[][][]` → a **three**-dimensional collection of integers
- ...

RECTANGULAR TWO-DIMENSIONAL ARRAY

- Project data for three students, where each student has five project grades.
- Convention: [*<rows>*][*<columns>*]
- Example

```
// Rows = students, columns = projects  
double[][] grades = new double[3][5];
```

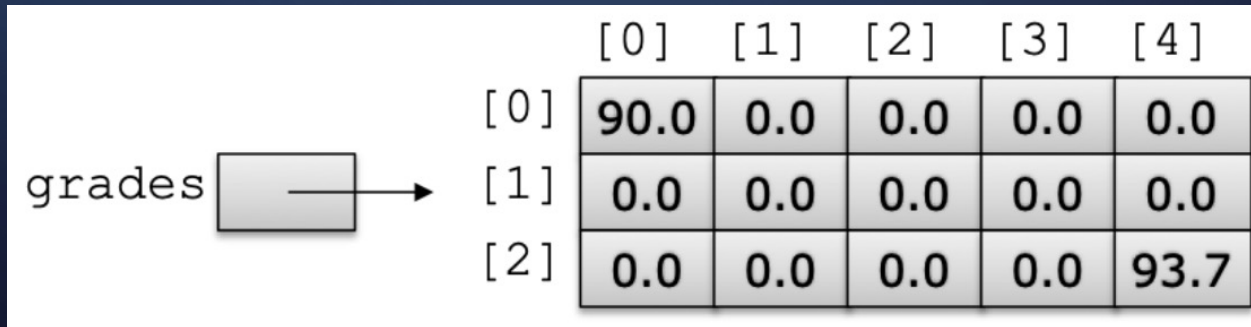


The diagram shows the variable 'grades' with an arrow pointing to a 3x5 array structure. The array is represented as a table with rows indexed [0], [1], and [2], and columns indexed [0], [1], [2], [3], and [4]. Each cell in the table contains the value 0.0.

	[0]	[1]	[2]	[3]	[4]
[0]	0.0	0.0	0.0	0.0	0.0
[1]	0.0	0.0	0.0	0.0	0.0
[2]	0.0	0.0	0.0	0.0	0.0

ACCESSING DATA IN A 2D ARRAY

- `grades` → the entire array
- `grades[1]` → the entire second row // `[0.0, 0.0, 0.0, 0.0]`
- `grades[0][0]` → the first element of the first row // `90.0`
- `grades[2][3]` → the fourth element of the third row // `0.0`
- `grades[2][4]` → the fifth element of the third row // `93.7`



The diagram illustrates the memory structure of a 2D array. On the left, the variable `grades` is shown next to a small gray box representing a pointer. An arrow points from this box to a larger table representing the array data. The table has three rows and five columns. The columns are labeled `[0]`, `[1]`, `[2]`, `[3]`, and `[4]` at the top. The rows are labeled `[0]`, `[1]`, and `[2]` on the left. The values in the table are as follows:

	[0]	[1]	[2]	[3]	[4]
[0]	90.0	0.0	0.0	0.0	0.0
[1]	0.0	0.0	0.0	0.0	0.0
[2]	0.0	0.0	0.0	0.0	93.7

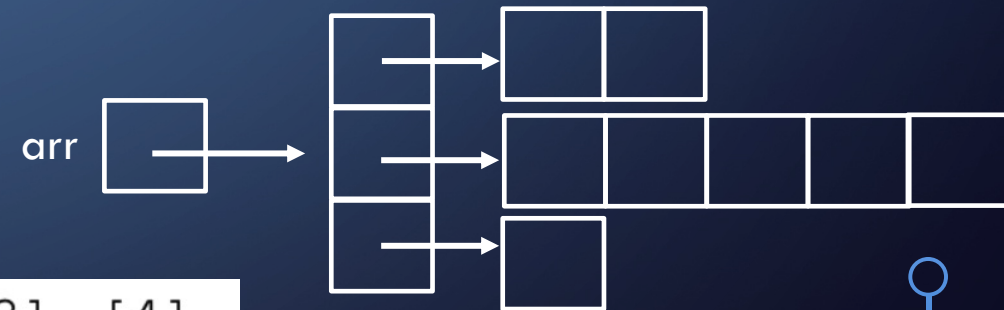
GENERALIZING MULTI-DIMENSIONAL ARRAYS

- Three-dimensional array
 - `int [][][] numbers = new int[4][4][4];`
 - Plane by row by column
 - Plane is array of two-dimensional arrays
 - Row is array of arrays
 - Column is array of integers.
- Multi-dimensional arrays
 - Consistency on what you consider each array of arrays to be
 - Comments to remind you (and others) what each dimension is—program context!

JAGGED ARRAYS

- An array of arrays of varying lengths
- First, construct rows array. Then, construct array for each row.

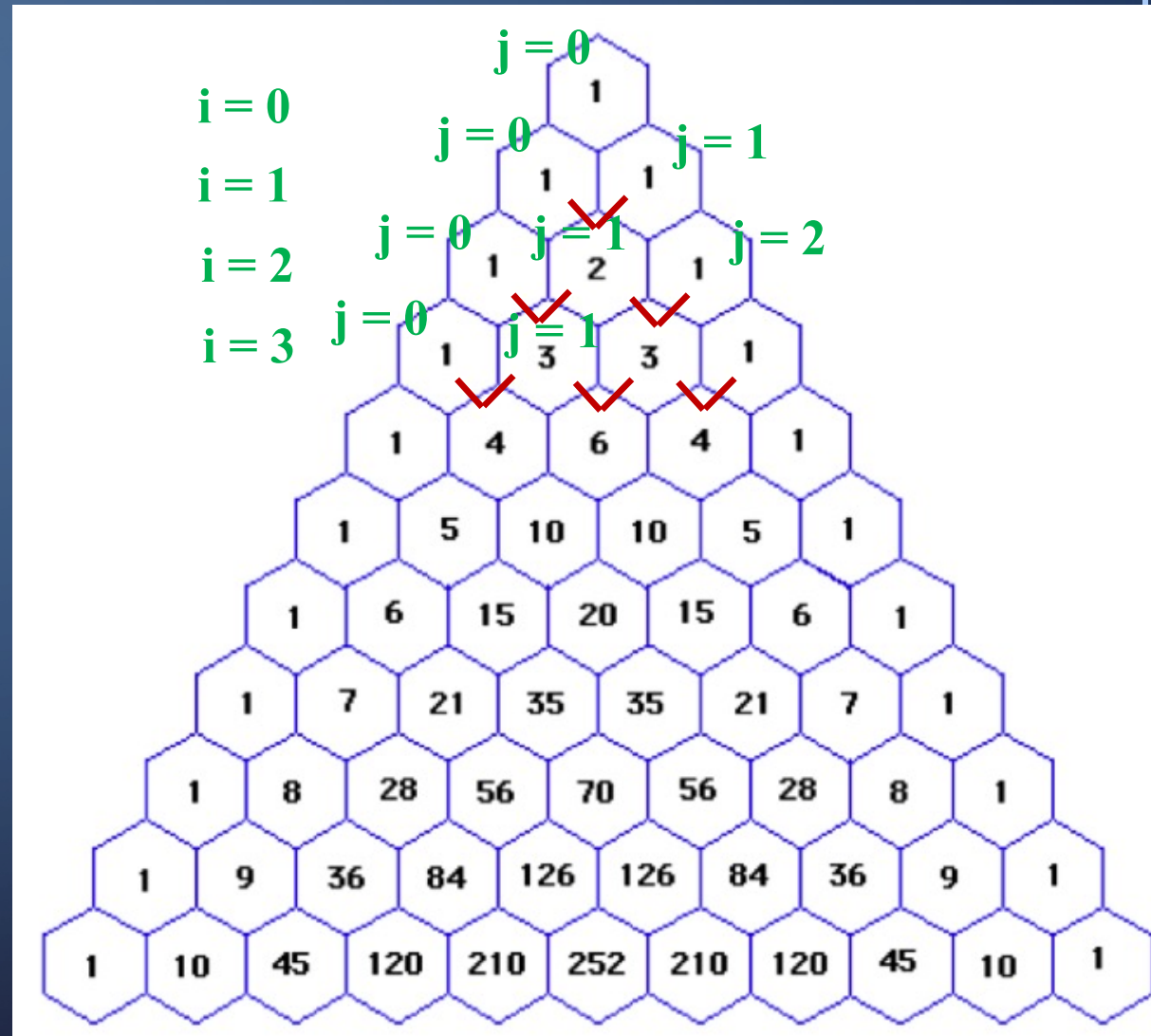
```
int [][] arr = new int [3] [];  
arr [0] = new int [2];  
arr [1] = new int [5];  
arr [2] = new int [1];
```



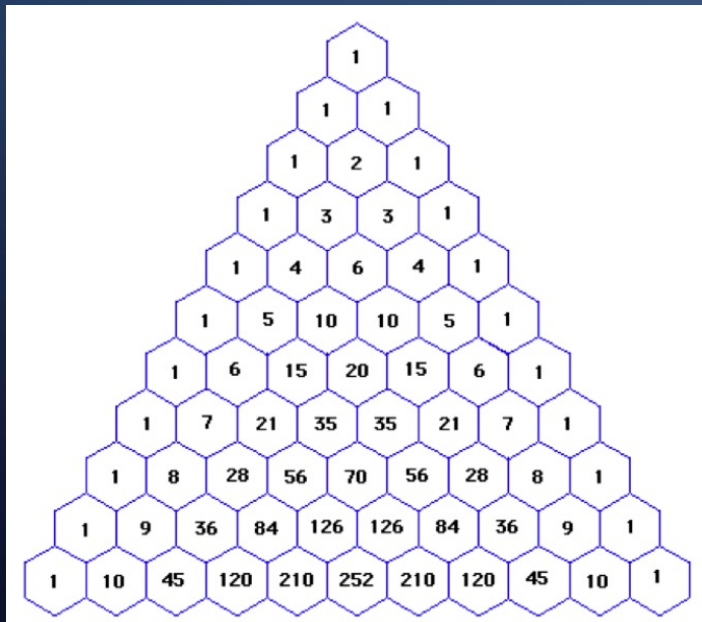
		[0]	[1]	[2]	[3]	[4]
	[0]	0	0			
arr		[1]	0	0	0	0
		[2]	0			

PASCAL'S TRIANGLE EXAMPLE

- Row n calculated from row $n - 1$
- Row 4 calculates middle part of Row 5



```
import java.util.*;
public class PascalTriangle
{
    public static void main(String[] args) {
        int[][] triangle = new int [11][];
        fillInPascalsTriangle(triangle);
        System.out.println(Arrays.deepToString(triangle));
    }
}
```



```
public static void fillInPascalsTriangle(int[][] triangle) {
    for (int i = 0; i < triangle.length ; i++) {
        // Set up the arrays (columns)
        triangle[i] = new int[i + 1];

        // Put in leading and trailing 1s
        triangle[i][0] = 1;
        triangle[i][i] = 1;

        // Fill middle of triangle
        for (int j = 1; j < i; j++) {
            triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
        }
    }
}
```