MULTI-DIMENSIONAL ARRAYS

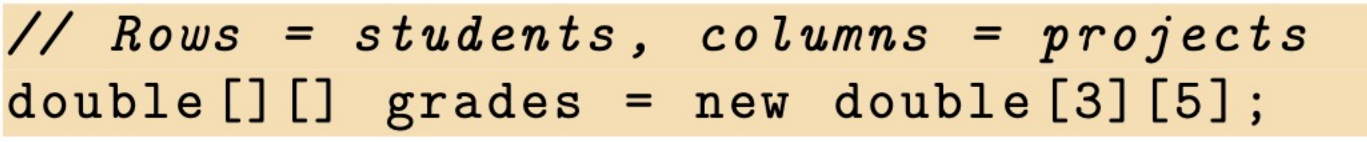
zyBook 6.9

MULTI-DIMENSIONAL DATA

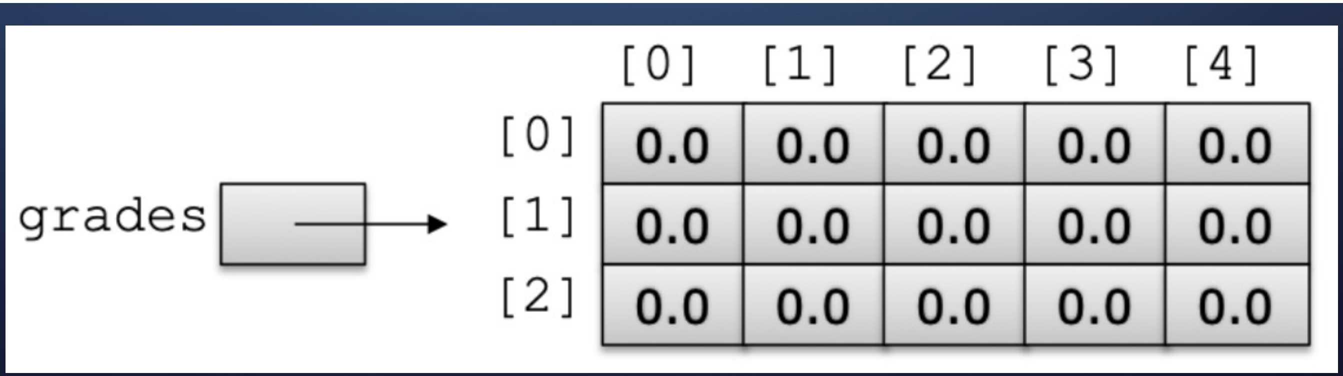
- int → one integer
- int[] → one-dimensional array of integers
- int[][] → a two-dimensional grid of integers
- int[][][] → a three-dimensional collection of integers
- …

RECTANGULAR TWO-DIMENSIONAL ARRAY

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



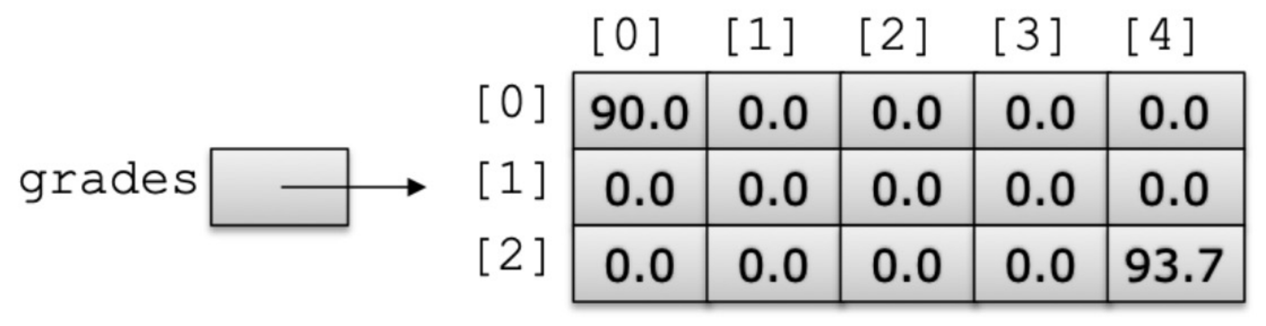
```
// Rows = students, columns = projects
double[][] grades = new double[3][5];
```



	[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



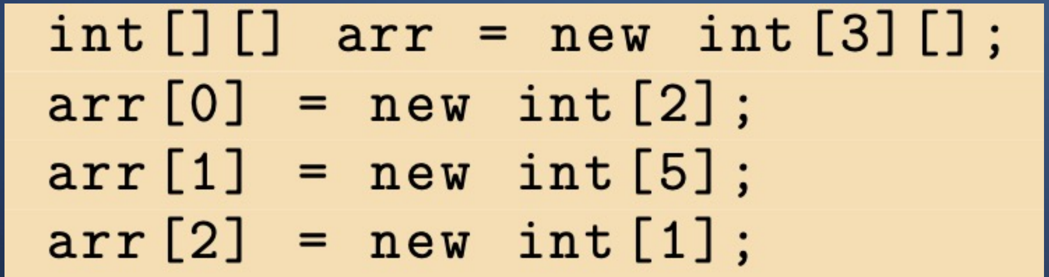
	[0]	[1]	[2]	[3]	[4]
[0]	90.0	0.0	0.0	0.0	0.0
[1]	0.0	0.0	0.0	0.0	0.0
[2]	0.0	0.0	0.0	0.0	93.7

GENERALIZING MULTI-DIMENSIONAL ARRAYS

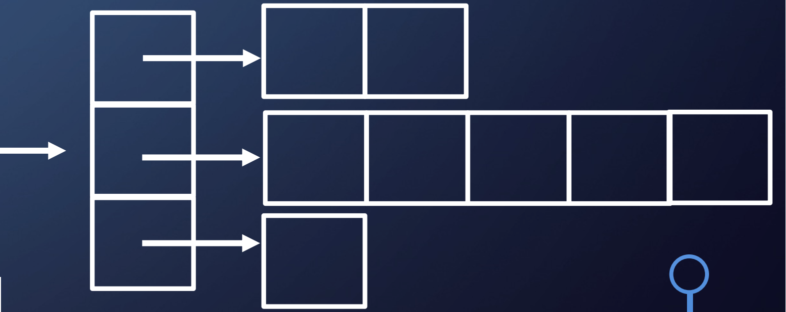
- Three-dimensional array
  - int [][][] numbers = new int[4][4][4];
  - Plane by row by column
  - Plane is array of two-dimensional arrays
  - Row is array of arrays
  - Column is array of integers.
- Multi-dimensional arrays
  - Consistency on what you consider each array of arrays to be
  - Comments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

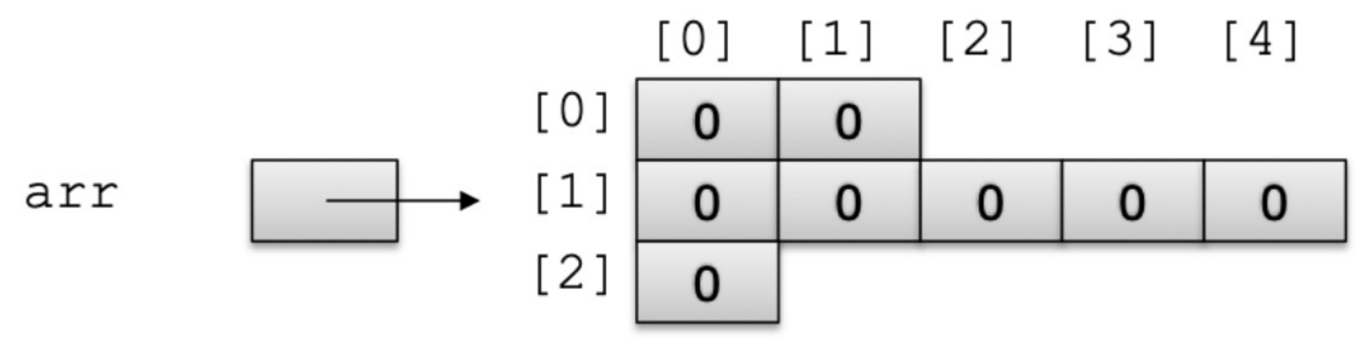
JAGGED ARRAYS

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



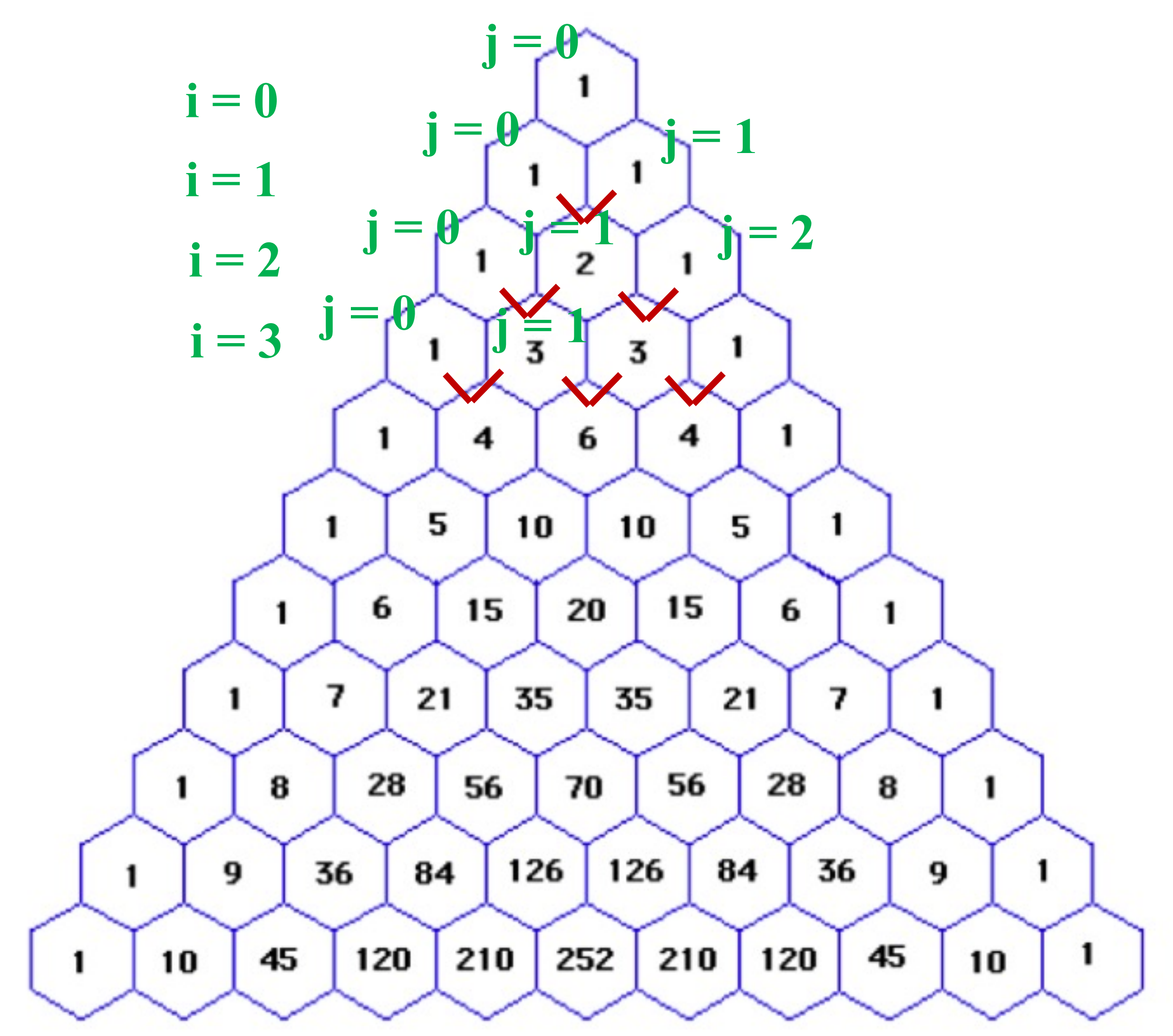
```
int[][] arr = new int[3][];
arr[0] = new int[2];
arr[1] = new int[5];
arr[2] = new int[1];
```





PASCAL'S TRIANGLE EXAMPLE

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



1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

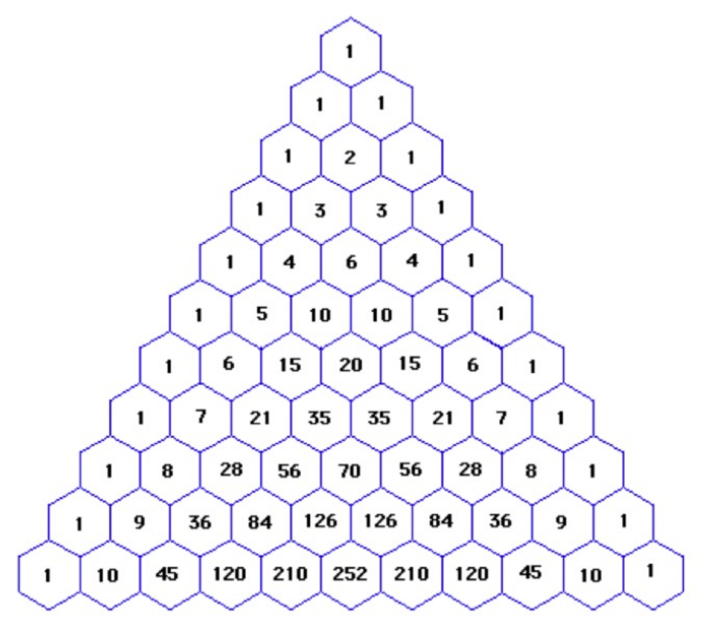
1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

1 10 45 120 210 252 210 120 45 10 1

```
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));
        }
}
```



```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```

```
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];

        }

    }

}
```