

CS-259

Data Structures with Java

Inheritance



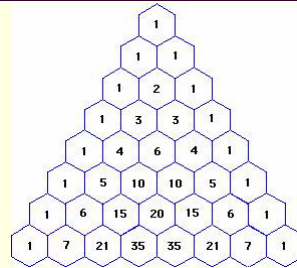
Instructor: **Joel Castellanos**

e-mail: joel@unm.edu

web: <http://cs.unm.edu/~joel/>

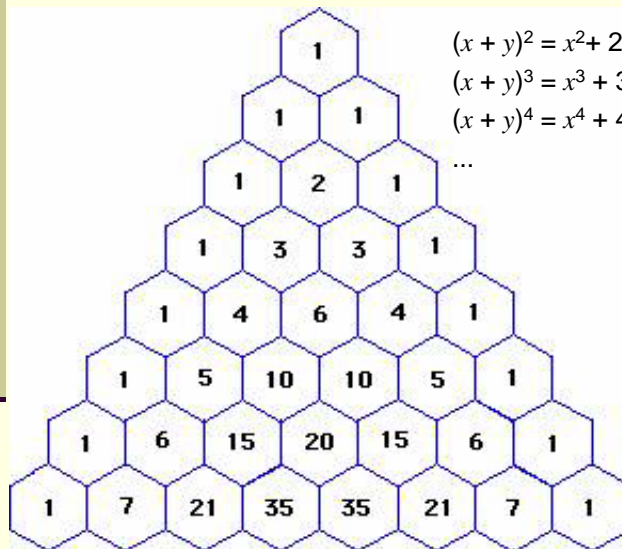
Course website:

<http://cs.unm.edu/~joel/cs259/>



1

Pascal's Triangle (binomial coefficients)



$$(x + y)^2 = x^2 + 2xy + y^2$$

$$(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$$

$$(x + y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4$$

...

-2-

Pascal's Triangle - Brainstorm

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

- 1) Start by simplifying output by leaving off leading spaces.
- 2) Start and end every row with 1.

- 3) Use an array so that calculations can be generalized:
 $\text{row4}[n] = \text{row3}[n-1] + \text{row3}[n]$.
- 4) To calculate row k , only row $k-1$ is needed.
- 5) Could create a dynamically sized array for each new row.
- 6) Each row has one more element than the previous row.

-3-

Pascal's Triangle

```
1
1 1
1 2 1
a) 1 3 3 1
b) 1 4 6 4 1
c) 1 5 10 10 5 1
```

```
b[0] = 1
b[1] = a[0]+a[1]
b[2] = a[1]+a[2]
b[3] = a[2]+a[3]
b[4] = 1
```

```
c[0] = 1
c[1] = b[0]+b[1]
c[2] = b[1]+b[2]
c[3] = b[2]+b[3]
c[4] = b[3]+b[4]
c[5] = 1
```

-4-

Pascal's Triangle: Implementation 1

```
public static void main(String[] args)
{
    int[] lastRow = {1};
    lastRow[0] = 1;
    printIntArray(lastRow);

    for (int row = 1; row<MAX_ROWS; row++)
    {
        int[] nextRow = new int[lastRow.length+1];
        nextRow[0] = 1;
        nextRow[lastRow.length] = 1;
        for (int i=1; i<row; i++)
        {
            nextRow[i] = lastRow[i-1] + lastRow[i];
        }

        lastRow = nextRow;
        printIntArray(lastRow);
    }
}
```

Hardcode the first row.

Create a new array one element larger than the last.

Does not copy values of the array, just changes where lastRow points. Where lastRow had pointed can now be marked for garbage collection.

-5-

Pascal's Triangle: printIntArray

```
public class HelloWorld
{
    public static final int MAX_ROWS = 10;

    public static void printIntArray(int[] a)
    {
        for (int i=0; i<=a.length-1; i++)
        {
            System.out.print(a[i]+"\\t");
        }
        System.out.println();
    }

    public static void main(String[] args)
    {
    }
}
```

-6-

Pascal's Triangle: Implementation 2

```

public static void main(String[] args)
{
    int[] lastRow = new int[MAX_ROWS];
    int[] nextRow = new int[MAX_ROWS];
    lastRow[0] = 1;
    printIntArray(lastRow);

    for (int row = 1; row < MAX_ROWS; row++)
    {
        nextRow[0] = 1;
        nextRow[row] = 1;
        for (int i = 1; i < row; i++)
        {
            nextRow[i] = lastRow[i-1] + lastRow[i];
        }

        int[] tmp = lastRow;
        lastRow = nextRow;
        nextRow = tmp;
        printIntArray(lastRow);
    }
}

```

Array creation occurs outside loop.

Does not create a new array, just creates a new reference.

Does not swap data in each array, just swaps references.

-7-

Pascal's Triangle: Reference Swap

```
int[] tmp = lastRow;
```

tmp

Array 1

Array 2

lastRow

nextRow

```
lastRow = nextRow;
```

tmp

Array 1

Array 2

lastRow

nextRow

```
nextRow = tmp;
```

tmp

Array 1

Array 2

lastRow

nextRow

-8-