## CS 272

Name Marco Gómez Section________________

## Lab 1 Part 1: Eclipse Tutorial

What time is displayed in the Output window? 23:31:30 GMT

## Lab 1 Part 2: Basics of Debugging

### Objectives:

To practice debugging skills. You need to use debugger on Eclipse. How set breakpoints is explained in Eclipse Tutorial from Part 1.

### Part 1: Understanding the Algorithm

You are going to analyze a program that displays Pascal's triangle. This triangle is a sequence of integers that arises in numerous areas of math and computer science, especially combinatorics and probability. For example, the Pascal triangle of height 4 is:

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
```

Entries in Pascal's triangle are indexed by integers. *n* is the number of the row and *k* is the position from the leftmost member of the row. The indexes in both directions start at zero (0), so in the last row listed above, C(4,0) = 1, C(4,1) = 4 and so on.

The values themselves are computed by the formula $C(n, k) = \frac{n!}{k!(n-k)!}$, which is called a *combination*. *n!* denotes a *factorial*, that is, the product $n*(n-1)*(n-2)*...*2*1$. The combinations can be interpreted as the number of ways to choose k elements from a collection containing n elements. When described, it is customary to say "*n* choose *k*", for instance '4 choose 2 is 6'.

Let's look at this closer: If four objects are numbered 1 through 4, think of how many ways can you combine two of them. This is what "*n* choose *k*" calculated for us.

**1. List all the possible pairings here. (Ex. {1, 2}, {2, 3}, ...) Keep in mind that the order of the numbers in a pairing does not matter – {1,2} is the same pairing as {2,1}. After you list all the pairings, compute C(4, 2) by using the above formula. What is the result? Does it match the number of pairings you listed? (Hint: answer to the last question should be yes)**

{1,2}, {2,3}, {3,4}, {1,3}, {1,4}, {2,4} $\boxed{6}$

$$\frac{4!}{2!(4-2)!} = \frac{24}{2 \cdot 2} \Rightarrow \frac{24}{4} \Rightarrow \boxed{6}$$

## Part 2: The Program Code

Download the files **PascalTriangle.java** and **PascalTriangleTester.java** from the class web page. Compile and run them.

**2. What output do you get when you request a triangle height of 5?**

```
      1
   4 2 1
576 144 24 4 1
```

**3. How many rows should have been generated for a height of 5?**

6

By now, it's obvious that there is a problem. Let's start investigating by setting a breakpoint at the line:

```
skip(spacesToSkip); // space to make a triangle
```

in the PascalTriangle constructor.

**4. Debug the program (height is 5). What is the value of n when the breakpoint is reached?**

0

**5. Run the program until it reaches the breakpoint again. What value do you expect n to be and what is the debugger reporting?**

Expected: 1

Actual: 2

**6. Once again, run the program until it reaches the breakpoint again. What value do you expect n to be and what is the debugger reporting?**

Expected: 2

Actual: 4

**7. The variable n is supposed to take the values 0, 1, 2, 3, 4, 5. Find the problem and fix it. What did you do to fix it?**

Erased the line with "n++"

**8. Run your corrected version again with a height of 5. You should now have six rows of output, but the values are still wrong. What values do you get? How do you know they are wrong?**

```
          1
        1   1
      4   2   1
    36  12  3  1
  576 144 24 2 1
44002880 360 40 5 1
```

It should be symmetric

To determine why the values are wrong, set a breakpoint at the line

```
return comb;
```

in the combination method (you can remove your prior breakpoint). Debug your program until the method is executed with the values n = 3 and k = 1.

**9. What should the value of comb be? (Calculate C(3,1) from part 1 of this lab.) What is the actual value?**

Expected: 3

Actual: 12

**10. Check why the value is computed incorrectly, and fix the computation. What did you do?**

put the facterial(K) *factorial(n-K) in its own Parantheses

**11. After fixing the error, run the test again (without breakpoints). What is the output?**

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

**Submit your corrected PascalTriangle.java using Canvas.**