CS272 Lab Assignment #8.

Due: Thursday, 4/19 by 11:30pm.

---

0. Read Chapter 8.

Implement and test 4 recursive methods described below. With the proper use of recursion, none of these methods should require more than a dozen lines of code.

1. Triangle Pattern

Write a recursive method with the following specification and header:

```
// Parameters: m - number of asterisks in the first line
//             n - number of asterisks in the middle 2 lines
// Precondition: m <= n, m > 0, n > 0
// Postcondition: The method has printed a pattern of 2*(n-m+1) lines
// to the standard output. The first line contains m asterisks, the next
// line contains m+1 asterisks, and so on up to a line with n asterisks.
// Then the pattern is repeated backwards, going n back down to m.
/* Example output:
   triangle(3, 5) will print this:
   ***
   ****
   *****
   *****
   ****
   ***
*/
public static void triangle(int m, int n)
```

Hint: Only one of the arguments changes in the recursive call. Which one?

2. Section Numbers

Write a recursive method with the following header:

```
public static void numbers(String prefix, int levels)
```

The method prints output consisting of the String prefix followed by "section numbers" of the form 1.1., 1.2., 1.3., and so on. The levels argument determines how many levels the section numbers have. For example, if levels is 2, then the section numbers have the form x.y and there are 9*9=81 of them. If levels is 3, then section numbers have the form x.y.z and there are 9*9*9 = 729 of them . The digits permitted in each level are always '1' through '9'.

As an example, if prefix is the string "THERBLIG" and levels is 2, then the method would start by printing:

```
THERBLIG1.1.
THERBLIG1.2.
THERBLIG1.3.
```

and end by printing:

```
THERBLIG9.7.
THERBLIG9.8.
THERBLIG9.9.
```

Every line in the output above consists of "THERBLIG" followed by a digit, ".", another digit, and ".". Total there are 81 lines.

As another example, if prefix is the string "Chapter" and levels is 1, then the method would print exactly the following:

```
Chapter1.
Chapter2.
Chapter3.
Chapter4.
Chapter5.
Chapter6.
Chapter7.
Chapter8.
Chapter9.
```

The stopping case of recursion occurs when levels reaches zero (in which case the prefix is printed once by itself followed by nothing else).

The Java String class has many manipulation methods, but you'll need only the ability to make a new string which consists of prefix followed by another character (such as '1') and a period ('.'). If s is the String that you want to create and c is the digit character (such as '1'), then the following statement will correctly form s:

```
s = prefix + c + '.';
```

This new String s can be passed as a parameter to recursive calls of the method.

3. A Fractal Pattern

Examine this pattern of asterisks and blanks, and write a recursive method that can generate patterns such as this:

```
*
* *
  *
* * * *
    *
    * *
      *
* * * * * * * *
        *
        * *
          *
        * * * *
            *
            * *
              *
```

With recursive thinking, the method needs only seven or eight lines of code (including two recursive calls). Your method should have the following specification and header:

```
// Parameters: n and i are integers
// Precondition: n is a power of 2, n >= 1, i>= 0
// Postcondition: A pattern based on the above example has been
// printed. The longest line of the pattern has
// n stars beginning in column i of the output. For example,
// The above pattern is produced by the call pattern(8, 0).
public static void pattern(int n, int i)
```

Hints: You do not need to check the precondition. Think about how the pattern is a fractal. Can you find two smaller versions of the pattern within the large pattern? Here is some code that may be useful within your method:

```
// A loop to print exactly i spaces:
for (k = 0; k < i; k++) System.out.print(" ");
// A loop to print n asterisks, each one followed by a space:
for (k = 0; k < n; k++) System.out.print("* ");
```

4. One Binary Number

Write a recursive method with the following specification and header:

```
// Parameters: n – an integer
// Precondition: n is non-negative.
// Postcondition: The method prints the value of n as a BINARY number.
// If n is zero, then a single zero is printed;
// otherwise no leading zeros are printed in the output.
// The '\n' character is NOT printed at the end of the output.
public static void binaryPrint(int n)
```

EXAMPLES:

```
If n=0 then output is 0
If n=4 then output is 100
If n=27 then output is 11011
```

Your recursive implementation must not use any local variables.

5. Test program For each method write a program that would test it. The program should prompt the user to enter value(s) of parameter(s) and execute the method. You may have four files in your Eclipse project, each of which contains a method and a test program for it.

What to submit:

- Submit your source code (**all** *.java files) electronically on Canvas.