Recursion Try-it

The goals of this lab are:

- To translate a recurrence relation to Java code.
- To understand stack overflow and integer overflow.
- To understand how the placement of statements before and after the recursive call affects the output.

To start, create a project in Eclipse and then complete each one of three parts as explained below.

Part 1. Tracing Recursive methods

WITHOUT using the computer, find the output of the following method when n = 4.

```
public static int mystery(int n){
      if (n == 0)
            return 0;
      else
            return n*n + mystery(n-1);

}
```

30

Part 2. Implementing Recursive Fibonacci method

In this part of the try-it, you will practice writing a recursive method:

1. Create a java class, called `RecursiveMethodsDriver`, that includes a `main` method.
2. In `RecursiveMethodsDriver`, write a static recursive method, called `fibonacci`, that implements the following recurrence relation:

```
fib( 1 ) = 1          (base case)
fib( 2 ) = 1          (base case)
fib( n ) = fib( n-1 ) + fib( n-2 )
```

3. Add statements in `main` to ask the user to enter a number, then call `fibonacci` method with the number as input and display the output of the method on the screen.
4. What is the value returned with the input number is 10? 55

5. What is the value returned when the input number is 40? 102334155
6. What is the value returned when the input number is 47? -1323752223

7. What is the output when n = -1? Threw an exception
8. What happened in step 7 above? Returns stack overflow error

Part 3. Printing Before and After a recursive method

1. Adding the following two method implementations to `RecursiveMethodsDriver`.

```
public static void m1 (int n){
     System.out.println(n);
     if (n > 1){
          m1(n-1);
     }
}
```

```
public static void m2(int n){
     if (n > 1)     {
          m2(n-1);
     }
     System.out.println(n);
}
```

2. Add a statement in main to call the method `m1` with input 5.

3. Add a statement in main to call the method `m2` with input 5.

4. Why are the outputs of `m1` and `m2` different?

Because m1 method is printing the n first, then n value after the first method call. While m2 method implements the method first then prints out the n value.

5. Write a similar recursive method called `m3` which displays the following output when given the input 5. Write a statement in main to call `m3` and make sure it gives the correct output.

```
5
4
3
2
1
1
2
3
4
5
```

6. Write another recursive method called m4 which displays the following output when given the input 5. Write a statement in main to call `m4` and make sure it gives the correct output. Hint: m4 is similar to m3 except that instead of printing the number, n, use a for loop to print `n` asterisks or n exclamation points.

```
*****
****
***
**
*
!
!!
!!!
!!!!
!!!!!
```