Java Programming

4-3: Recursion

Practice Activities

Lesson Objectives:

- Create linear recursive methods
- Create non-linear recursive methods
- Compare the pros and cons of recursion

Vocabulary:

Identify the vocabulary word for each definition below.

	Is the process of recursively calling a copy of the same method until a base case is reached. In some cases, like the Fibonacci problem, two values may trigger alternate base cases that return values in non-linear recursion. In non-linear recursion, the base case may need to accommodate multiple return values.
	Is the process of calling one and only one copy of the same method within a method.
	Is the process of calling two or more copies of the same method within a method. As a rule, the calls are separated by an operator in an algorithm.
	Is the process of backing into a problem by recursively calling a copy of the method until you arrive at a base case, and then returning the values up the chain to resolve a problem.
	The last case processed by a recursive program, which may also be processed for the last couple values. This is true when resolving a Fibonacci sequence for the first two values of the sequence. They are the last values calculated when recursively resolving the problem.
	Is the alternative to the base case, and run when the base case criteria isn't met. It runs when the program needs to call yet another copy or set of copies of itself to resolve a problem.

Try It/Solve It:

1. Create a class to define and test linear recursion. Name the class Linear. Implement the following:
   a) A public class named "**Linear**".
   b) A pubic static method named "**factorial**" that takes a double and returns a double.
   c) The base case should check whether the input value is less than or equal to 1.
   d) The recursive case should return the input value multiplied against the result of a recursive call to the factorial method with the input value minus 1.

Create a static main method that tests the following:

a) A double variable named "**d**".

b) Assign the value of d to be 5.0

c) Print the factorial value and the original input value.

d) While formatting may differ, this is the expected output:

e) Factorial [120.0] of [5.0]

2. Create a class to define and test linear recursion. Name the class NonLinear. Implement the following:

a) A public class named "NonLinear"

b) A pubic static method named "fibonacci" that takes a double and returns a double.

c) The base case should check whether the input value is less than 2.

d) The recursive case should return the following:

e) return fibonacci(d - 1) + fibonacci(d - 2);

Create a static main method that tests the following:

a) A double variable named "d".

b) Check if the argument list contains any values, and assign the value of argument zero to the local double variable. If the argument list is empty assign a value of 5.

c) Print the fibonacci value and the original input value.

d) While formatting may differ, this is the expected output:

Fibonacci index [0.0] value [0.0]

Fibonacci index [1.0] value [1.0]

Fibonacci index [2.0] value [1.0]

Fibonacci index [3.0] value [2.0]

Fibonacci index [4.0] value [3.0]

3. Trace through the following code using a chart, and then apply backwards thinking to find the factorial of 7.

```
public static double factorial(double d) {

       // Sort elements by title case.
       if (d <= 1) {
             return 1;
       }
       else {
          return d * factorial(d - 1);
       }//endif
}//end method factorial
```