Recursion

- Recursion is a fundamental programming technique that can provide an elegant solution certain kinds of problems.

Recursive Thinking

- A recursive definition is one which uses the word or concept being defined in the definition itself.
- When defining an English word, a recursive definition is often not helpful.
- But in other situations, a recursive definition can be an appropriate way to express a concept.
- Before applying recursion to programming, it is best to practice thinking recursively.

Recursive Definitions

- Consider the following list of numbers: 24, 88, 40, 37.
- Such a list can be defined as follows:
  
  A LIST is a: number
  or a: number comma LIST

- That is, a LIST is defined to be a single number, or a number followed by a comma followed by a LIST.
- The concept of a LIST is used to define itself.
Recursive Definitions

- The recursive part of the LIST definition is used several times, terminating with the non-recursive part:

```plaintext
number comma LIST
24 , 88, 40, 37
  number comma LIST
  88 , 40, 37
  number comma LIST
  40 , 37
  number
  37
```

Infinite Recursion

- All recursive definitions have to have a non-recursive part
- If they didn’t, there would be no way to terminate the recursive path
- Such a definition would cause infinite recursion
- This problem is similar to an infinite loop, but the non-terminating “loop” is part of the definition itself
- The non-recursive part is often called the base case

Recursive Definitions

- N!, for any positive integer N, is defined to be the product of all integers between 1 and N inclusive
- This definition can be expressed recursively as:

```plaintext
1! = 1
N! = N * (N-1)!
```
- A factorial is defined in terms of another factorial
- Eventually, the base case of 1! is reached

Recursive Definitions

5!
5 * 4!
4 * 3!
3 * 2!
2 * 1!
1

Recursive Programming

- A method in Java can invoke itself; if set up that way, it is called a recursive method
- The code of a recursive method must be structured to handle both the base case and the recursive case
- Each call to the method sets up a new execution environment, with new parameters and local variables
- As with any method call, when the method completes, control returns to the method that invoked it (which may be an earlier invocation of itself)
Recursive Programming

- Consider the problem of computing the sum of all the numbers between 1 and any positive integer \( N \).
- This problem can be recursively defined as:

\[
\sum_{i=1}^{N} i = N + \sum_{i=1}^{N-1} i
\]

\[
= N + N-1 + \cdots + \sum_{i=1}^{1}
\]

// This method returns the sum of 1 to num
public int sum (int num) {
    int result;
    if (num == 1) {
        result = 1;
    } else {
        result = num + sum (num-1);
    }
    return result;
}

Note that just because we can use recursion to solve a problem, doesn’t mean we should.

For instance, we usually would not use recursion to solve the sum of 1 to \( N \) problem, because the iterative version is easier to understand.

However, for some problems, recursion provides an elegant solution, often cleaner than an iterative version.

You must carefully decide whether recursion is the correct technique for any problem.

Indirect Recursion

- A method invoking itself is considered to be direct recursion.
- A method could invoke another method, which invokes another, etc., until eventually the original method is invoked again.
- For example, method \( m1 \) could invoke \( m2 \), which invokes \( m3 \), which in turn invokes \( m1 \) again.
- This is called indirect recursion, and requires all the same care as direct recursion.
- It is often more difficult to trace and debug.
Towers of Hanoi

- The Towers of Hanoi is a puzzle made up of three vertical pegs and several disks that slide on the pegs
- The disks are of varying size, initially placed on one peg with the largest disk on the bottom with increasingly smaller ones on top
- The goal is to move all of the disks from one peg to another under the following rules:
  - We can move only one disk at a time
  - We cannot move a larger disk on top of a smaller one

An iterative solution to the Towers of Hanoi is quite complex
A recursive solution is much shorter and more elegant
See solution on Rephactor topic "Example: Towers of Hanoi"