Recursion

- In computer science, some problems are more easily solved by using recursive methods.

World's Simplest Recursion Program

```
public class Recursion
{
   public static void main (String args[])
   {
       count(0);
       System.out.println();
   }

   public static void count (int index)
   {
       System.out.print (index);
       if (index < 2)
             count(index+1);
   }
}
```

This is where the recursion occurs. You can see that the count() method calls itself.

This program

012

First two rules of recursion

- **Base case**: You must always have some base case which can be solved without recursion

- **Making Progress**: For cases that are to be solved recursively, the recursive call must always be a case that makes progress toward the base case.

Factorials

- Computing factorials are a classic problem for examining recursion.
- A factorial is defined as follows:

  n! = n * (n-1) * (n-2) …. * 1;

- For example:

  1! = 1

  2! = 2 * 1 = 2

  3! = 3 * 2 * 1 = 6

  4! = 4 * 3 * 2 * 1 = 24

  5! = 5 * 4 * 3 * 2 * 1 = 120

If you study this table closely, you will start to see a pattern.

Seeing the Pattern

- Seeing the pattern in the factorial example is difficult at first.

- But, once you see the pattern, you can apply this pattern to create a recursive solution to the problem.
- Divide a problem up into:
  - What we know (call this the base case)
  - Making progress towards the base
    - Each step resembles original problem
    - The method launches a new copy of itself (recursion step) to make the progress.

Recursive Solution

```
public class FindFactorialRecursive
{
   public static void main (String args[])
   {
       for (int i = 1; i < 10; i++)
              System.out.println ( i + "! = " +
   findFactorial(i));
   }

   public static int findFactorial (int number)
   {
       if (( number == 1) || (number == 0))
              return 1;
       else
              return (number * findFactorial (number-1));
   }
}
```

Base Case.

Making progress

Recursion pros and cons

- All recursive solutions can be implemented without recursion.

- Recursion is "expensive". The expense of recursion lies in the fact that we have multiple activation frames and the fact that there is overhead involved with calling a method.

- If both of the above statements are true, why would we ever use recursion?

- In many cases, the extra "expense" of recursion is far outweighed by a simpler, clearer algorithm which leads to an implementation that is easier to code.

Recursive Solution

```
public class FindFactorialRecursive
{
   public static void main (String args[])
   {
       for (int i = 1; i < 10; i++)
              System.out.println ( i + "! = " +
   findFactorial(i));
   }

   public static int findFactorial (int number)
   {
       if (( number == 1) || (number == 0))
              return 1;
       else
              return (number * findFactorial (number-1));
   }
}
```

Base Case.

Making progress