Programming with Recursion
The Recursion Pattern

- **Recursion**: when a method calls itself
- **Classic example**: the factorial function:
  \[ n! = 1 \cdot 2 \cdot 3 \cdots \cdot (n-1) \cdot n \]
- **Recursive definition**:
  \[
  f(n) = \begin{cases} 
  1 & \text{if } n = 0 \\
  n \cdot f(n-1) & \text{else}
  \end{cases}
  \]
- **As a Java method**:
  ```java
  public static int recursiveFactorial(int n) {
    if (n == 0) return 1; // basis case
    else return n * recursiveFactorial(n-1); // recursive case
  }
  ```
Content of a Recursive Method

- **Base case(s)**
  - Values of the input variables for which we perform no recursive calls are called **base cases** (there should be at least one base case).
  - Every possible chain of recursive calls **must** eventually reach a base case.

- **Recursive calls**
  - Calls to the current method.
  - Each recursive call should be defined so that it makes progress towards a base case.
Visualizing Recursion

- **Recursion trace**
  - A box for each recursive call
  - An arrow from each caller to callee
  - An arrow from each callee to caller showing return value

- **Example**

```
recursiveFactorial(4)
  └── call
      └── recursiveFactorial(3)
          └── call
              └── recursiveFactorial(2)
                  └── call
                      └── recursiveFactorial(1)
                          └── call
                              └── recursiveFactorial(0)

return 1
return 1 * 1 = 1
return 2 * 1 = 2
return 3 * 2 = 6
return 4 * 6 = 24
final answer
```

© 2010 Goodrich, Tamassia
Example: English Ruler

- Print the ticks and numbers like an English ruler:

```

    ---- 0
    ---- 1
    ---- 2

    ---- 0
    ---- 1

    ---- 0
    ---- 1
    ---- 2
    ---- 3

```
Using Recursion

\[ \text{drawTicks}(\text{length}) \]

Input: length of a ‘tick’
Output: ruler with tick of the given length in the middle and smaller rulers on either side

\[ \text{if ( length > 0 ) then} \]
\[ \text{drawTicks}( \text{length} - 1 ) \]
\[ \text{draw tick of the given length} \]
\[ \text{drawTicks}( \text{length} - 1 ) \]
Recursive Drawing Method

- The drawing method is based on the following recursive definition
- An interval with a central tick length $L \geq 1$ consists of:
  - An interval with a central tick length $L - 1$
  - An single tick of length $L$
  - An interval with a central tick length $L - 1$
Java Implementation (1)

// draw ruler
public static void drawRuler(int nInches, int majorLength) {
    drawOneTick(majorLength, 0); // draw tick 0 and its label
    for (int i = 1; i <= nInches; i++) {
        drawTicks(majorLength - 1); // draw ticks for this inch
        drawOneTick(majorLength, i); // draw tick i and its label
    }
}

// draw ticks of given length
public static void drawTicks(int tickLength) {
    if (tickLength > 0) {
        drawTicks(tickLength - 1); // stop when length drops to 0
        drawOneTick(tickLength); // recursively draw left ticks
        drawTicks(tickLength - 1); // draw center tick
        drawTicks(tickLength - 1); // recursively draw right ticks
    }
}
// draw a tick with no label

public static void drawOneTick(int tickLength) {
    drawOneTick(tickLength, -1);
}

// draw one tick

public static void drawOneTick(int tickLength, int tickLabel) {
    for (int i = 0; i < tickLength; i++)
        System.out.print("-");
    if (tickLabel >= 0) System.out.print(" "+ tickLabel);
    System.out.print("\n");
}