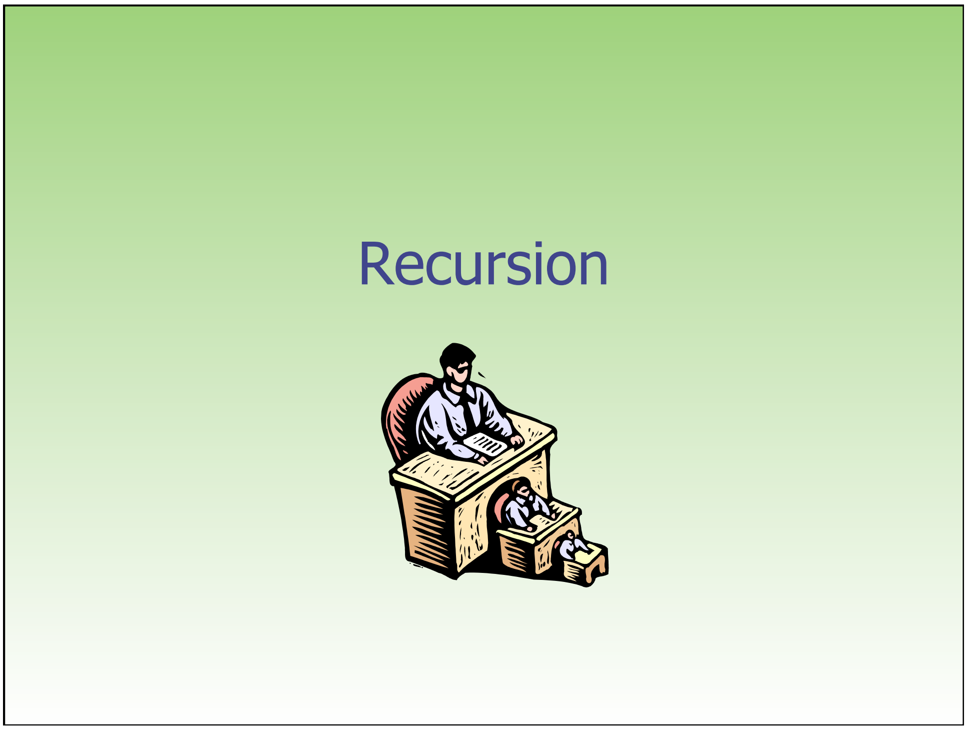
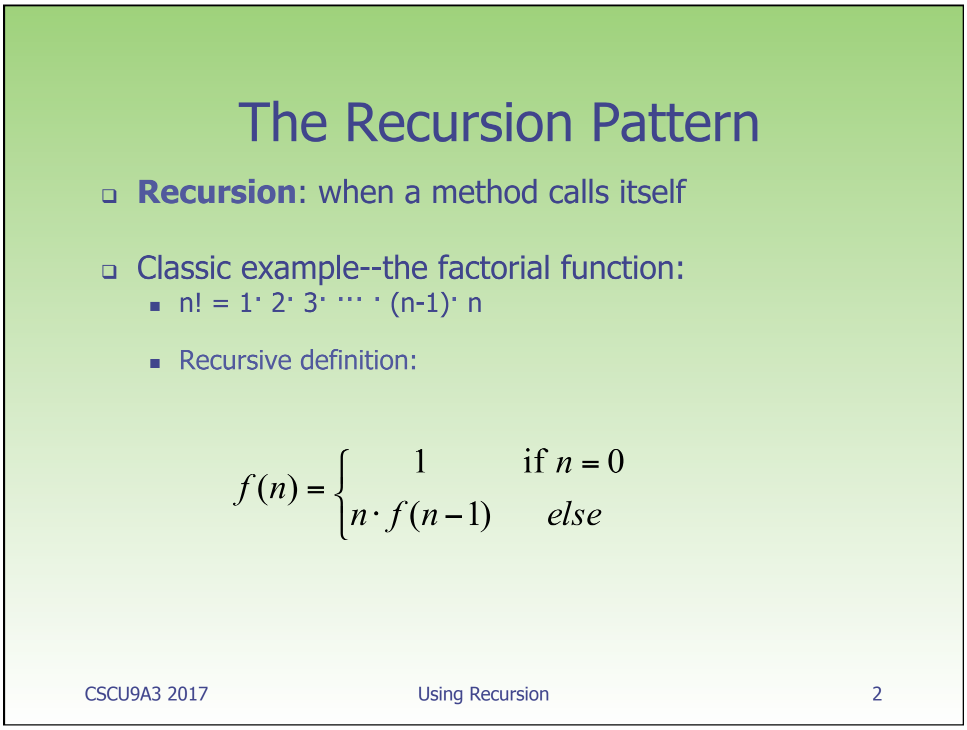


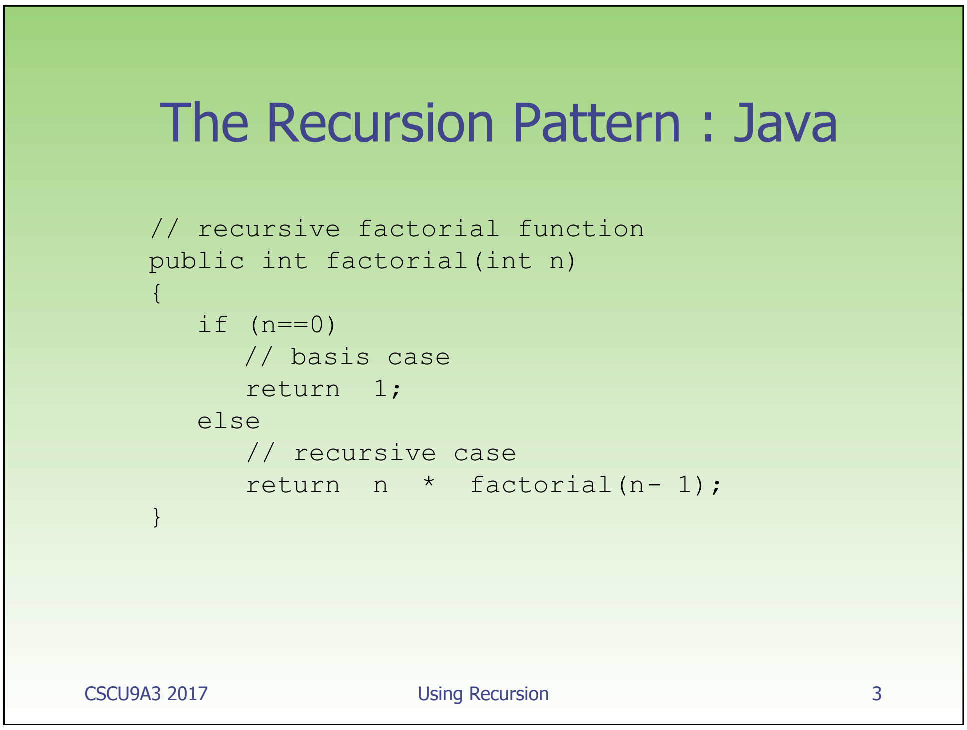
Recursion



The Recursion Pattern

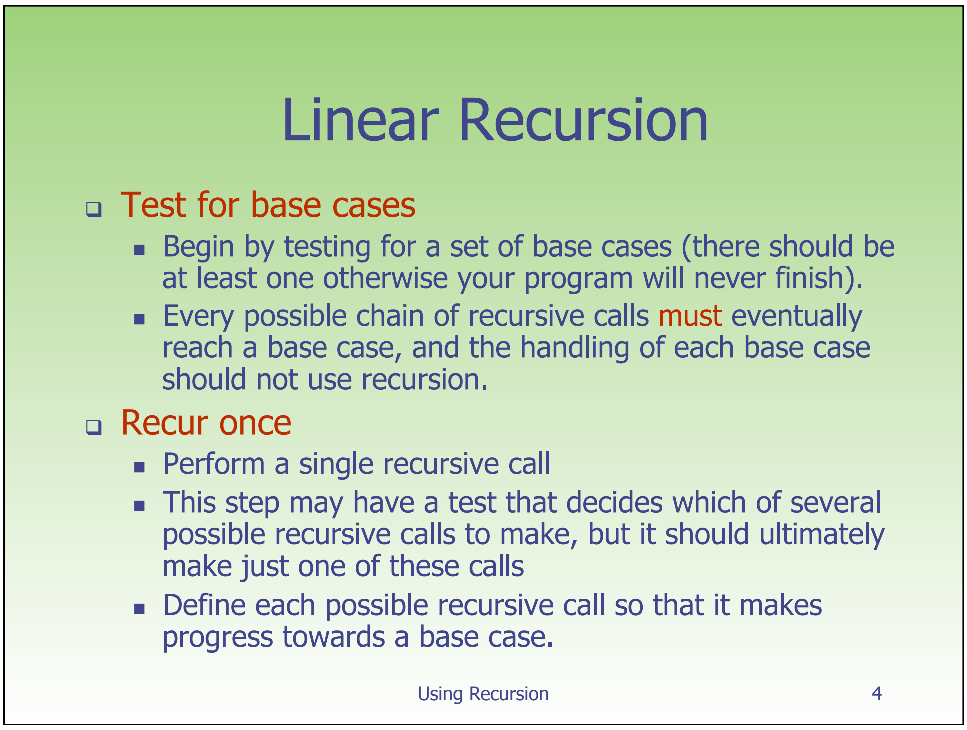
- **Recursion**: when a method calls itself
- Classic example--the factorial function:
  - n! = 1· 2· 3· ··· · (n-1)· n
  - Recursive definition:

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot f(n-1) & else \end{cases}$$



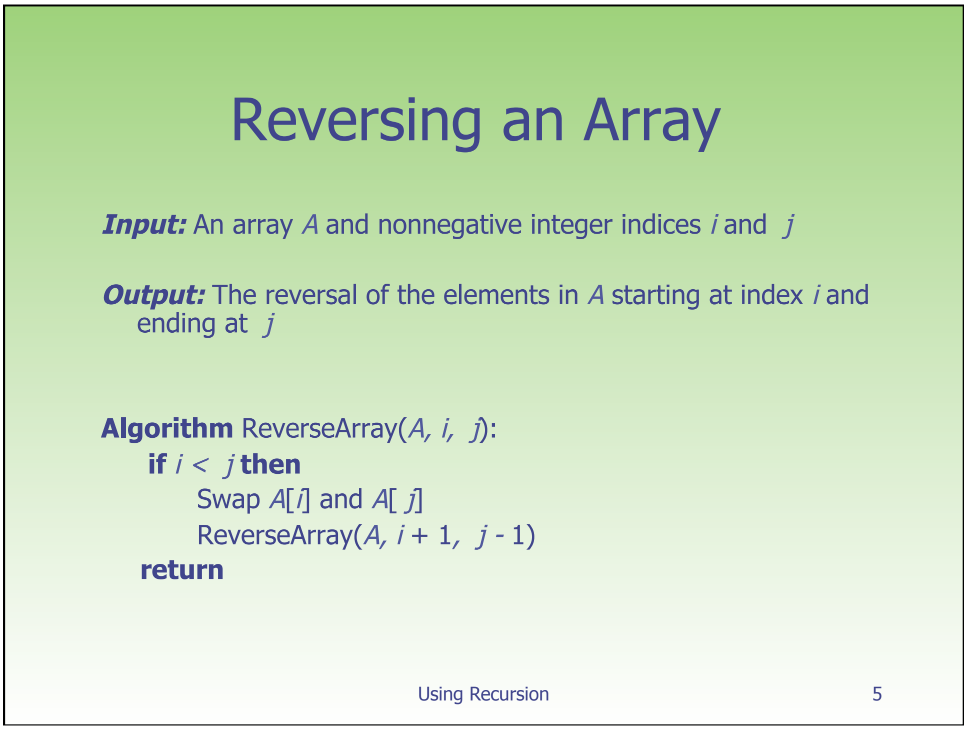
The Recursion Pattern : Java

```
// recursive factorial function
public int factorial(int n)
{
   if (n==0)
      // basis case
      return  1;
   else
      // recursive case
      return  n  *  factorial(n- 1);
}
```



Linear Recursion

- Test for base cases
  - Begin by testing for a set of base cases (there should be at least one otherwise your program will never finish).
  - Every possible chain of recursive calls must eventually reach a base case, and the handling of each base case should not use recursion.
- Recur once
  - Perform a single recursive call
  - This step may have a test that decides which of several possible recursive calls to make, but it should ultimately make just one of these calls
  - Define each possible recursive call so that it makes progress towards a base case.

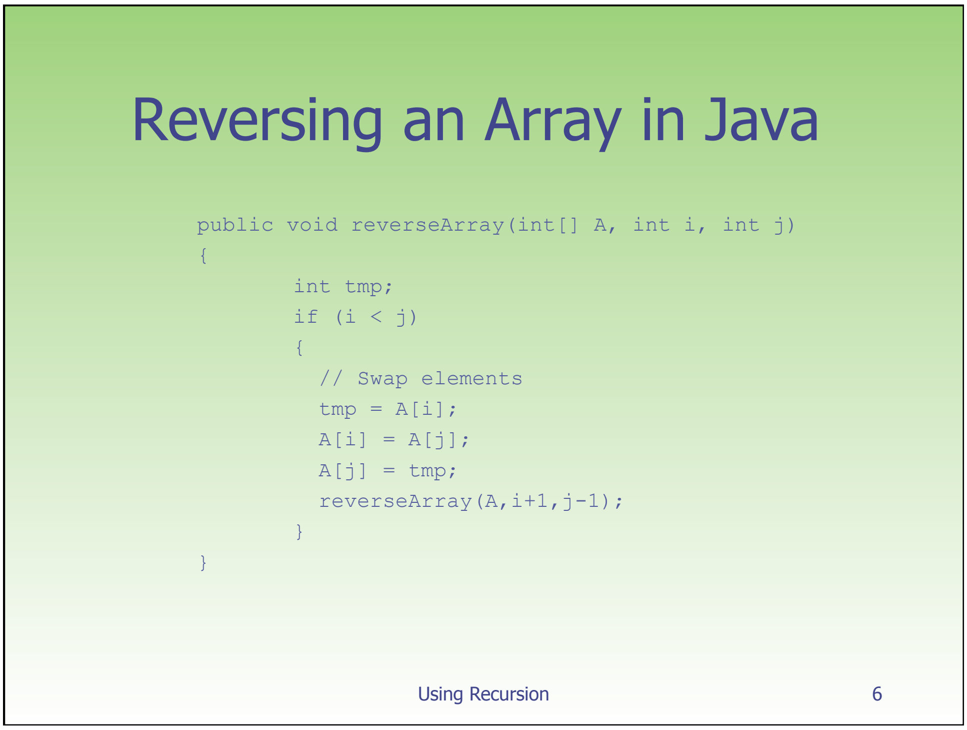


Reversing an Array

Input: An array *A* and nonnegative integer indices *i* and *j*

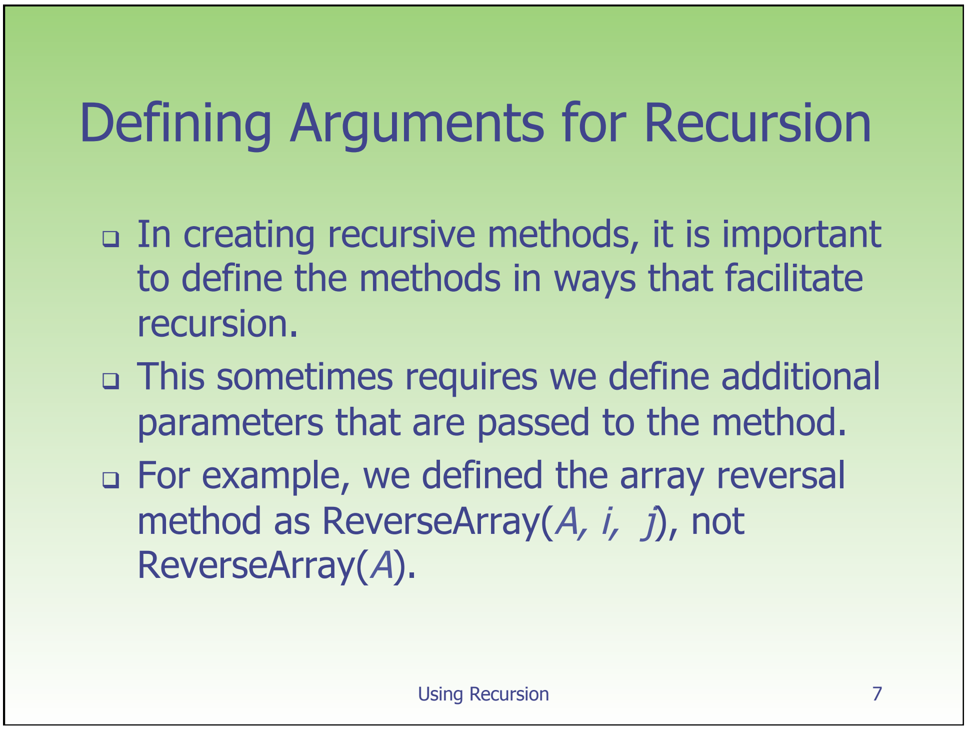
Output: The reversal of the elements in *A* starting at index *i* and ending at *j*

```
Algorithm ReverseArray(A, i, j):
  if i < j then
     Swap A[i] and A[ j]
     ReverseArray(A, i + 1, j - 1)
  return
```



Reversing an Array in Java

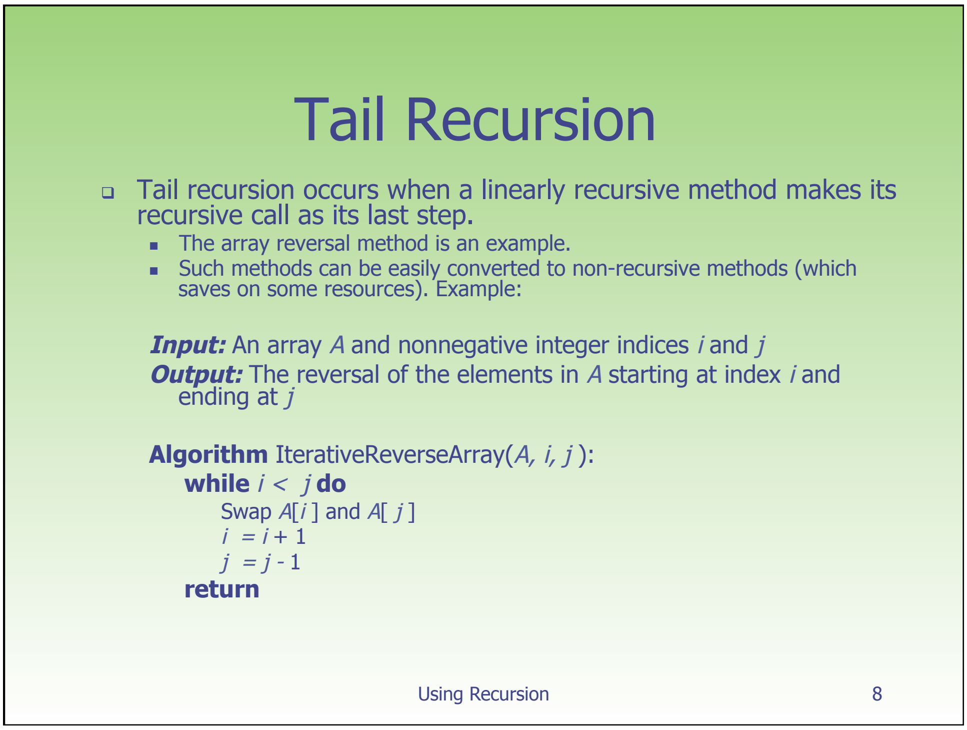
```
public void reverseArray(int[] A, int i, int j)
{
        int tmp;
        if (i < j)
        {
          // Swap elements
          tmp = A[i];
          A[i] = A[j];
          A[j] = tmp;
          reverseArray(A,i+1,j-1);
        }
}
```



Defining Arguments for Recursion

- In creating recursive methods, it is important to define the methods in ways that facilitate recursion.
- This sometimes requires we define additional parameters that are passed to the method.
- For example, we defined the array reversal method as ReverseArray(*A, i, j*), not ReverseArray(*A*).

Using Recursion 7



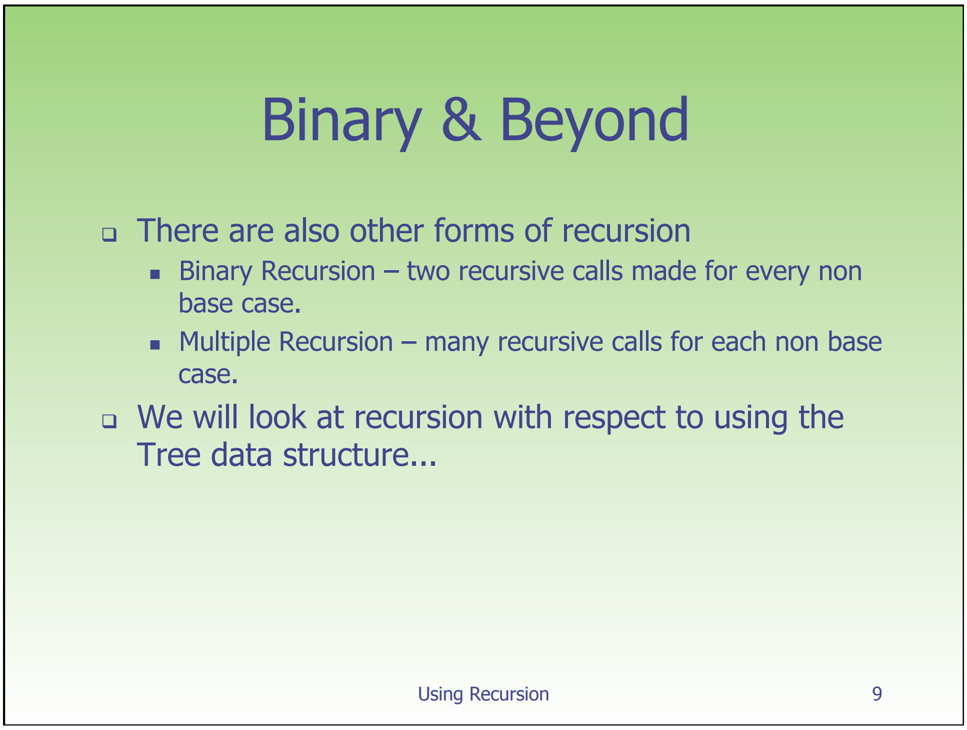
Tail Recursion

- Tail recursion occurs when a linearly recursive method makes its recursive call as its last step.
  - The array reversal method is an example.
  - Such methods can be easily converted to non-recursive methods (which saves on some resources). Example:

Input: An array *A* and nonnegative integer indices *i* and *j*
Output: The reversal of the elements in *A* starting at index *i* and ending at *j*

```
Algorithm IterativeReverseArray(A, i, j ):
    while i < j do
        Swap A[i ] and A[ j ]
        i = i + 1
        j = j - 1
    return
```

Using Recursion 8



Binary & Beyond

- There are also other forms of recursion
  - Binary Recursion – two recursive calls made for every non base case.
  - Multiple Recursion – many recursive calls for each non base case.
- We will look at recursion with respect to using the Tree data structure...