More Recursion

Colorado State University
Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu)

Announcements

https://www.brainyquote.com/quotes/katherine_johnson_875699

TODO Reminders:

Readings are due **before** lecture

- Reading 25 (zybooks) – you should have already done that ☺
- Lab 16
- Reading 26 (zyBooks)
- Lab 17
- RPA 12

Keep practicing your RPAs in a spaced and mixed manner ☺

Help Desk

Day	**Time : Room**
Monday	12 PM - 2 PM : CSB 120
Tuesday	6 PM - 8 PM : Teams
Wednesday	3 PM - 5 PM : CSB 120
Thursday	6 PM - 8 PM : Teams
Friday	3 PM - 5 PM : CSB 120

Recall Activity

```
import java.util.Scanner;
public class Recursion {
   public static void recursiveCall(int number, int lowVal, int highVal)
{
      int midVal;
      midVal = (highVal + lowVal) / 2;
      System.out.print(number);
      System.out.print(" ");
      System.out.print(midVal);
      if (number == midVal) {
         System.out.println(" f");
      }
      else {
         if (number < midVal) {
            System.out.println(" l");
            recursiveCall(number, lowVal, midVal);
         }
         else {
            System.out.println(" h");
            recursiveCall(number, midVal + 1, highVal);
         }
      }
   }
```

```
   public static void main(String[] args) {
      Scanner scnr = new Scanner(System.in);
      int number;

      number = scnr.nextInt();
      recursiveCall(number, 0, 10);
   }
}
```

- What does this program do?
- Write down each recursive call and its parameter and return value for number = 3

Recursion Review, Recursion Review…

- Recursion
  - A way to 'repeat' code without loops
  - Methods that call themselves
  - Recursive methods have
    - A base case (condition to stop)
    - recursive call
    - return values (good design)

Example from challenge problems

- another way to reverse string

```
public static String stringManipulator(String str) {
    if (str.isEmpty()) return str;
    return stringManipulator(str.substring(1)) + str.charAt(0);
}
```

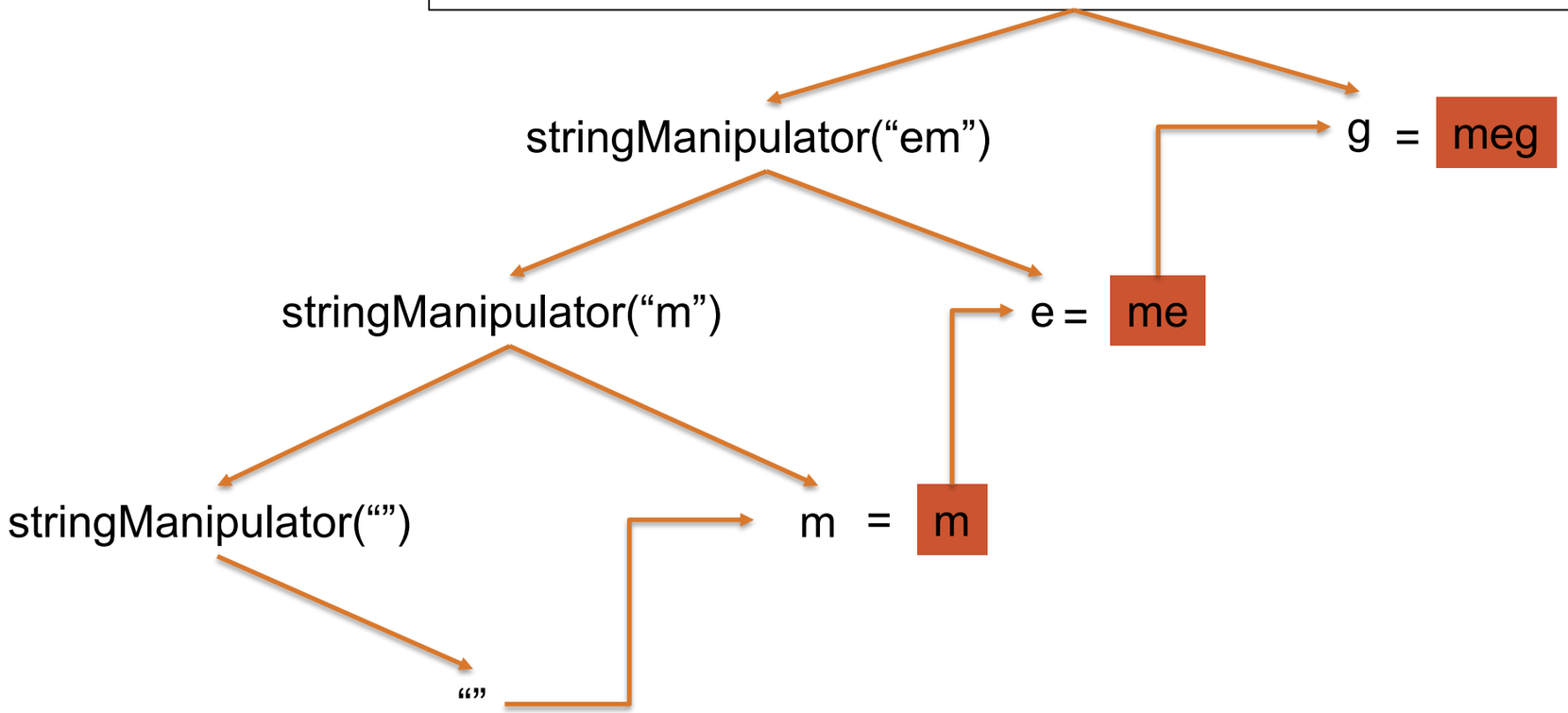
base case

recursive call

Visualizing This Method

```
public static String stringManipulator(String str) {
    if (str.isEmpty()) return str;
    return stringManipulator(str.substring(1)) + str.charAt(0);
}
```

str = gem



The Memory Stack

```
public static int factorial(int n) {
    if(n <= 1) return 1;
    return n * factorial(n-1);
}
```

```
public static void main(String[] args) {
    System.out.println(factorial(3));
}
```

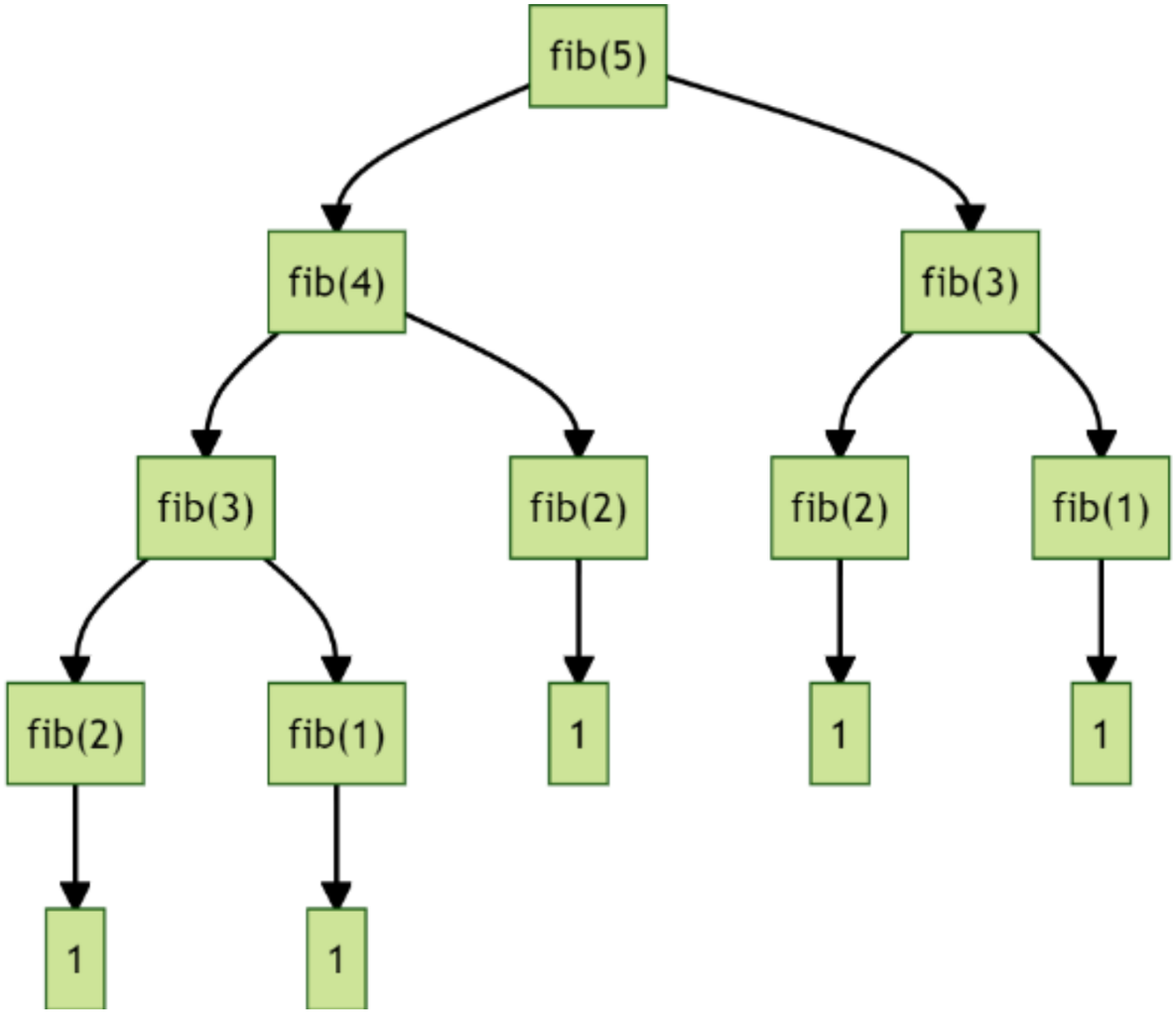
- Math factorials
  - N = 6, factorial is 1 * 2 * 3 * 4 * 5 * 6
- When calling methods
  - method is pushed onto the memory stack
  - removed when done
- This causes the following to happen in memory
  - You will cover this more in CS 250 and CS 220

Multiple Branching

- Most of our cases, the tree only had one branch that continued to branch
- The other branch was always the solution
- What if the tree branched out at both sides?
- Fibonacci problem
  - The sum is equal to the previous two
    - $fib(1) = 1$
    - $fib(2) = 1$
    - $fib(3) = fib(2) + fib(1) = 2$
    - $fib(4) = fib(3) + fib(2) = 2 + 1$
    - $fib(5) = fib(4) + fib(3) = 3 + 2$
    - 1 1 2 3 5 8 13 21 …

Multiple Branching

- Fib(5)



Coding Along – Fibonacci

- Write a method that calculates the fibonacci value based on n
- for example:
  - fib(5) would return 5
  - fib(6) would return 8
  - fib(7) would return 13
    - and so on
    - Both fib(2) and fib(1) return 1, fib(0) or lower returns 0.

What is our case base?

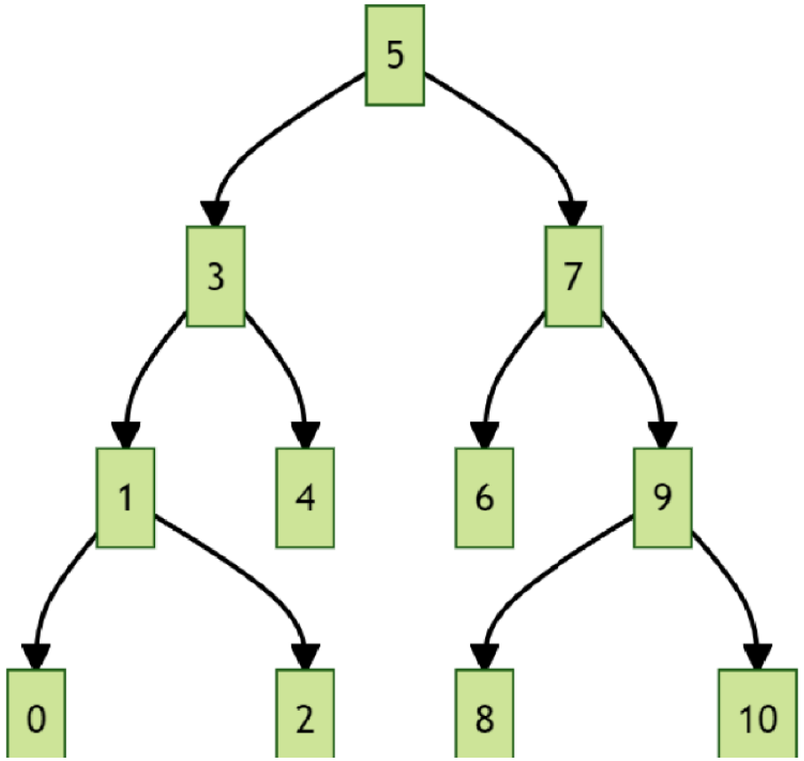
What is our recursive call?

Overview

- Recursion
  - When to use it?
    - When you have a limited paths to follow
    - When you don’t know your loop deep
    - When your data is already set up like a tree
  - you will come across it again – CS165
  - always remember your base case!

Overview

- For example, let's consider the following sequence of numbers
  - [0,1,2,3,4,5,6,7,8,9,10]
  - Numbers are ordered
  - We could structure this number line as a "tree"



How many movements to we need to go from 5 to 1 in this tree structure?

How many movements to we need to go from 5 to 1 in a linear structure as an array?

You will learn about tree structures on CS165 ☺

Practice 1

- How can we change the stringManipulator method to reverse the string backwards, meaning that we start from the end instead of the begin?

```
public static String stringManipulator(String str) {
    if (str.isEmpty()) return str;
    return stringManipulator(str.substring(1)) + str.charAt(0);
}
```

- Remember to think about:
  - base case (condition to stop)
  - recursive call

Practice 2

- Write a recursive method that verifies if a String is a palindrome or not. A word is a **palindrome** if the letters in the word are symmetric.

- Remember to think about:
  - base case (condition to stop)
  - recursive call