

Introduction to Recursion



Colorado State University
Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu),
updated by Marcia Moraes (marcia.moraes@colostate.edu)

Announcements


TODO Reminders:

Readings are due **before** lecture

- Reading 23 (zybooks) – you should have already done that 😊
- Lab 15
- Reading 24 (zyBooks)
- Lab 16
- Practical Project Lecture
- RPA 11

Keep practicing your RPAs in a spaced and mixed manner 😊

NEXT WEEK – Exam 3 Week
Catch up, if you need!



It doesn't matter
what others are
doing. It matters
what you are doing.

<https://www.facebook.com/corpnet/posts/it-doesnt-matter-what-others-are-doing-it-matters-what-you-are-doing/10161039565148812/>

Monday Help Desk –
12-2pm CSB120

Monday Help Session –
3-4pm CSB325

Tuesday Help Desk –
6-8pm Teams

Tuesday Help Session –
10-11am Teams

Recursion

- Simple recursion is a loop
 - A **method** that calls itself!
- How to write it?
 1. Write a base case! (condition!!)
 2. Write method that calls itself
- Starting with Factorial example
 1. Let's take a look at an interactive solution (loop)

```
public static long factorialLoop (int n){  
    long fact = 1;  
    for(int i = n; i > 1; i--)  
        fact *= i;  
    return fact;  
}
```

- Building our first recursion method
 - Factorial
 - $0! = 1$
 - $1! = 1$
 - $n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$
- How do you go about creating a recursive solution?

1. What is the base case?

```
if (n == 1 || n == 0) return 1;
```

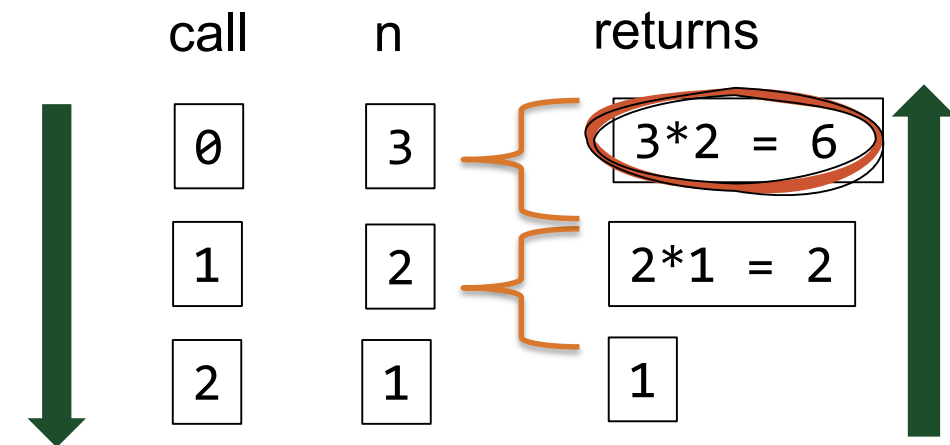
2. How we build the recursive call?

```
return n * factorial(n-1);
```

```
public static long factorial (int n){  
    if(n == 1 || n == 0) return 1;  
    return n * factorial(n-1);  
}
```

Recursion Factorial Walk Through

```
public class Recursion {  
    public static long factorial(int n){  
        if(n == 1 || n == 0 ) return 1;  
        return n * factorial(n-1);  
    }  
    public static void main(String args[]){  
        System.out.println("Factorial of 3: " + factorial(3));  
    }  
}
```



Another way to look at this:

```
factorial(3)  
    factorial(2)  
        factorial(1)  
            return 1  
        return 2*1 = 2  
    return 3*2 = 6
```

Recursion – String Reverse

- Let's start with a String reverse method that uses a loop solution to return a reversed String

```
public static String reverseLoop (String str){  
    String reversed = "";  
    for(int i = str.length(); i > -1; i--)  
        reversed += str.charAt(i);  
    return reversed;  
}
```

- How to write a recursive version of it?

- Write a base case `if (index < 0) return "";`
- Write method that calls itself

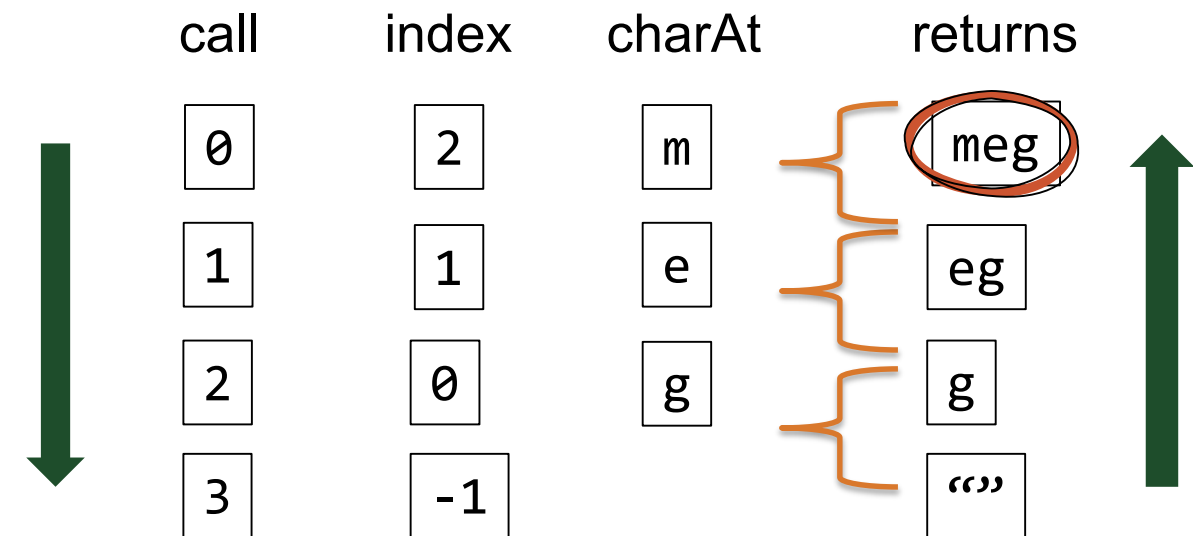
```
return str.charAt(index) + reverseString(str, index-1)
```

```
public static String reverseString(String str) {  
    return reverseString(str, str.length()-1);  
}  
  
public static String reverseString(String str, int index) {  
    if(index < 0) return ""; // Base Case  
    return str.charAt(index) + reverseString(str, index-1);  
} // Recursive Call  
  
public static void main(String[] args) {  
    System.out.println(reverseString("arepO eht fo motnahP"));  
}
```

Phantom of the Opera

Reverse String Walk Through

```
public static String reverseString(String str) {  
    return reverseString(str, str.length()-1);  
}  
  
public static String reverseString(String str, int index) {  
    if(index < 0) return "";  
    return str.charAt(index) + reverseString(str, index-1);  
}  
  
public static void main(String[] args) {  
    System.out.println(reverseString("gem"));  
}
```



Recursion – Group Practice

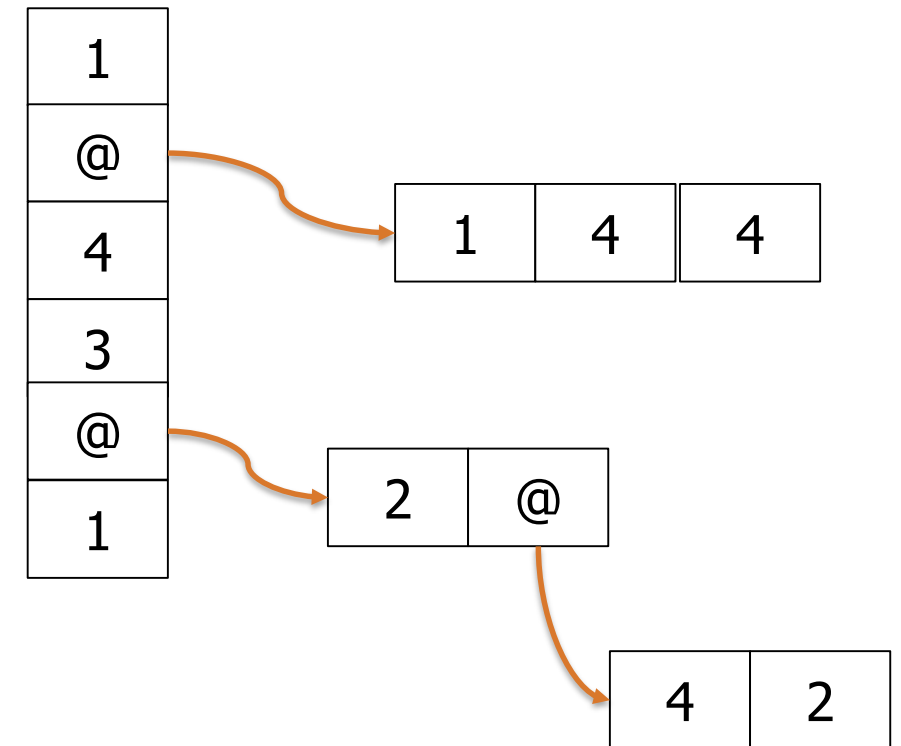
- Consider the method presented below:

```
public static int sum (int [] array){  
    int s = 0;  
    for(int i = 0; i < array.length; i++)  
        s += array[i];  
    return s;  
}
```

- How to write a recursive version of it?
 1. Write a base case
 2. Write method that calls itself
 3. Tip – you can change the number of parameters

Real Example and Sneak Peak Into Inheritance

- Assume you have the following data structure
 - An array of values, but some values can be other arrays!
- How do you represent all the different values?
- **Inheritance (and Polymorphism)**
 - All Objects “inherit” from the class Object
 - Gains properties of Object
 - Which means you can store all objects as Objects
 - But you need to ‘cast’ back to do something useful
- Now – the Real Example
 - How can I sum all the values across the structure to the right?
 - Would a loop work?
 - Why or why not?
- Solution - **Recursion!**



Sneak Peak Solution

```
public class RecursionExample {  
  
    public static int sum(Object[] values) {  
        return sum(values, 0); // overload, for easier initial call  
    }  
    public static int sum(Object[] values, int current) {  
        if(current >= values.length) return 0; // past end of array, return 0  
        if(values[current] instanceof Object[]) // another array!  
            return sum((Object[])values[current], 0) + sum(values, current+1);  
        return (Integer)values[current] + sum(values, current+1); // number plus something  
    }  
    public static void main(String[] args) {  
        Object[] values = new Object[]{1, 2, 3,  
            new Object[]{4, 5, new Object[]{1,1}},  
            10, new Integer[]{2,2}, 1, 10};  
        System.out.println(sum(values));  
    }  
}
```

Add values for the answer...

Worksheet

- Work on your worksheet.
- Turn in to the TAs or myself, this will be your attendance for today's class.