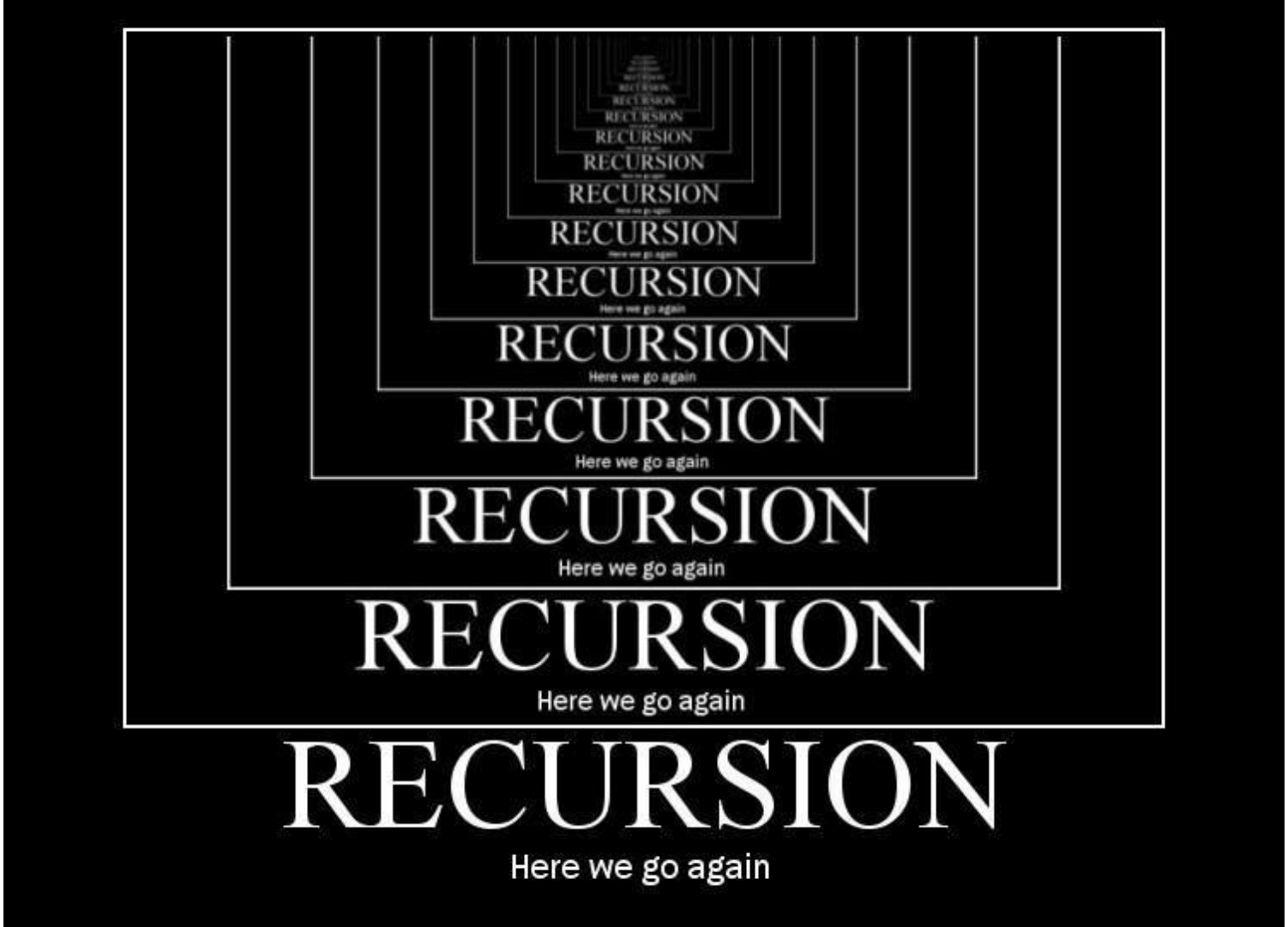
Assignment 4

Recursion, call-by-value, call-by-reference

Data Structures, Fall 2018
TA: Marija Stanojevic


RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again

Introduction

Recursion is useful when problems can be represented by a smaller version of the same problem.

It consists of two parts:

- exit condition (can be multiple exit conditions)
- solution that is used in all other cases except in exit condition - simplifies problem until it it becomes so simple that it can solved by the exit condition

Every problem that can be solve recursively, can be solved iteratively

Recursion is easier to code, but much more time and memory consuming! WHY?

Example

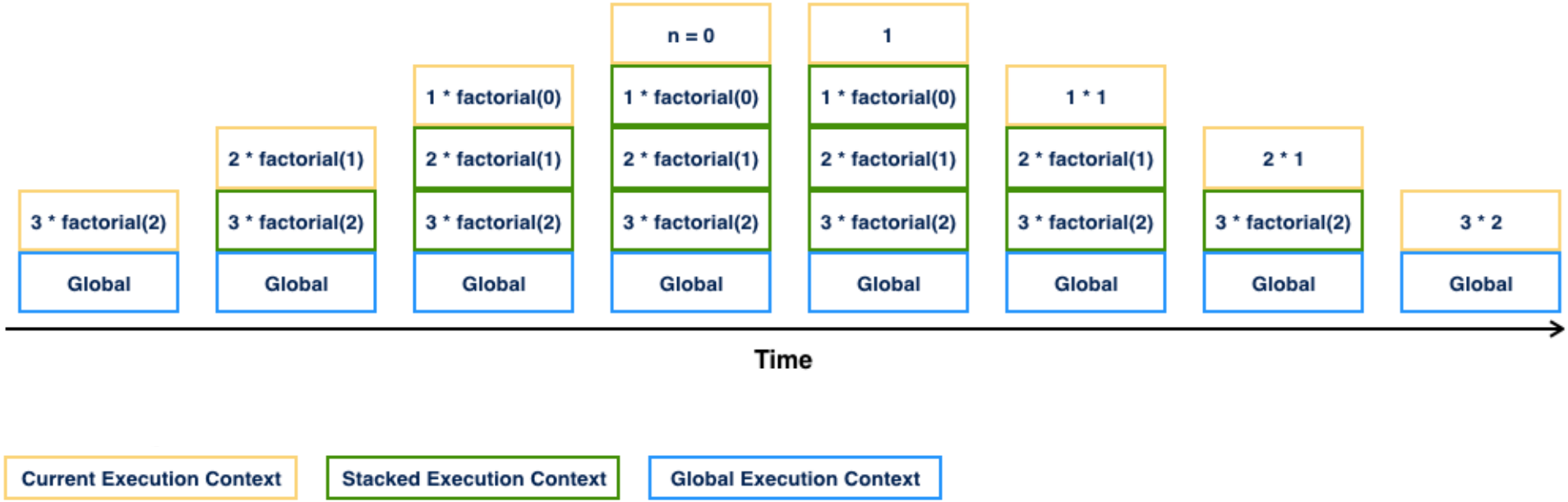
Sum first N numbers:

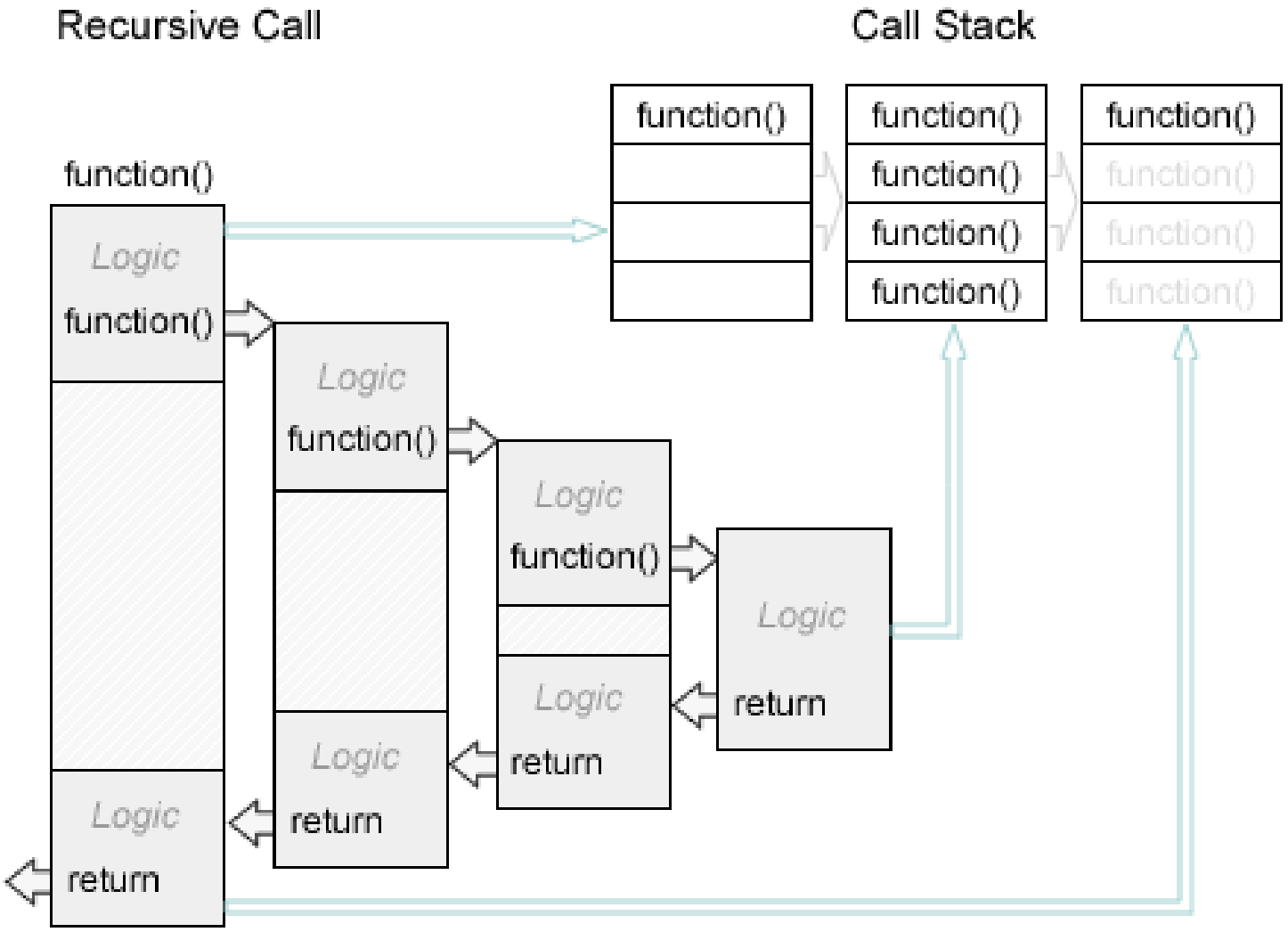
```
private sum(int n) { //recursive
	if (n == 0)
				return 0;
		else
				return n +
sum(n-1);
}
public sumFive() { //non-recursive
	System.out.println(sum(5));
}
```

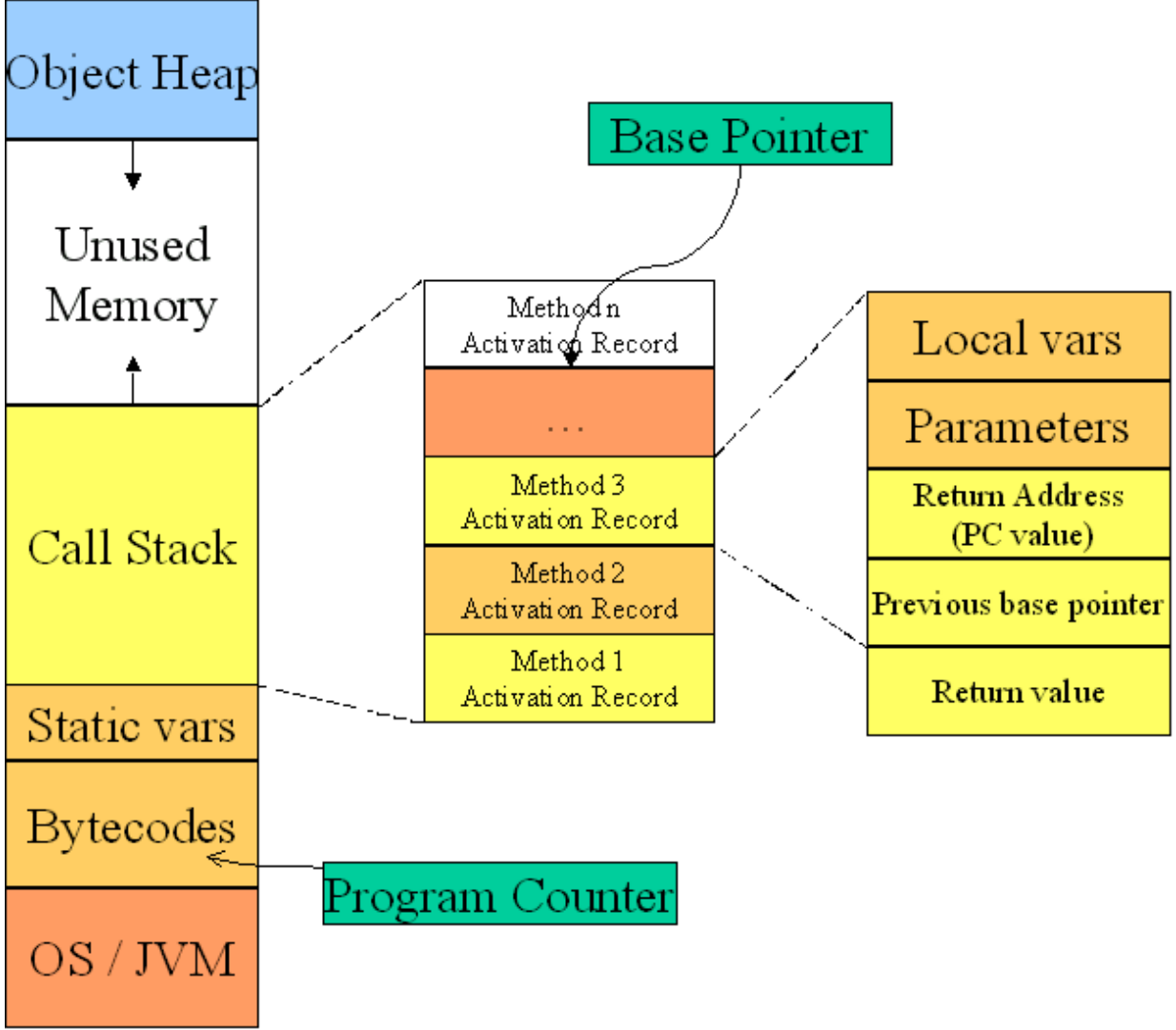
What happens in execution

```
sumFive()  // non-recursive
  sum(5) // recursive, else is true, prints 15
    return 5 + sum(4), else is true, 5 + 10 = 15
      return 4 + sum(3), else is true, 4 + 6 = 10
        return 3 + sum(2), else is true, 3 + 3 = 6
          return 2 + sum(1), else is true, 2 + 1 = 3
            return 1 + sum(0), else is true, 1 + 0 = 1
              return 0; // if condition is true
```

What is happening in memory during recursion?




Recursive Call
Call Stack
function()
Logic
function()
Logic
function()
Logic
function()
Logic
function()
Logic
return
Logic
return
Logic
return
Logic
return
function()
function()
function()
function()
function()
function()
function()
function()
function()


Object Heap
Unused Memory
Call Stack
Static vars
Bytecodes
OS / JVM
Base Pointer
Method n Activation Record
. . .
Method 3 Activation Record
Method 2 Activation Record
Method 1 Activation Record
Program Counter
Local vars
Parameters
Return Address (PC value)
Previous base pointer
Return value

Call-by-value & Call-by-reference

In java call-by-value happens when argument is simple type (int, long, double,...) Otherwise it is call-by-reference.

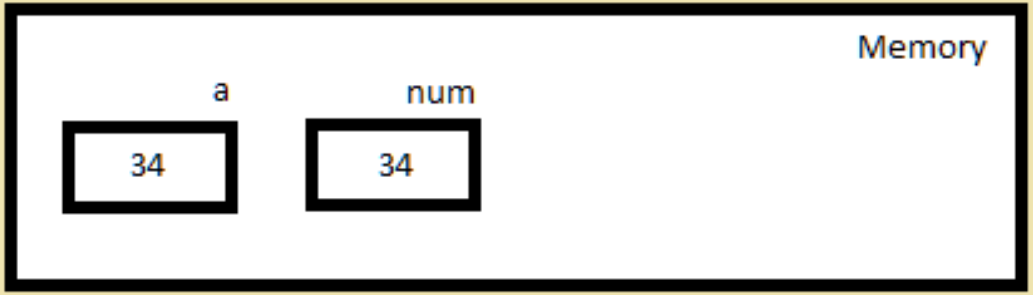
Call-by-value means that value that is given as argument is copied in memory as parameter of function that is called. Because value is copied, if we change it, just local copy is changed. Local copy will be deleted once function ends and it won’t effect original value.

Call-by-reference means that address of object is copied in memory as parameter of function that is called, so that function sees the copy of address. Because address is copied, if we change it just local copy of address changes and it will be deleted once function ends. It won’t effect original address. However, if we change object that address refers to, it will stay changed after we exit function.

Call By Value

public void meth (int num)

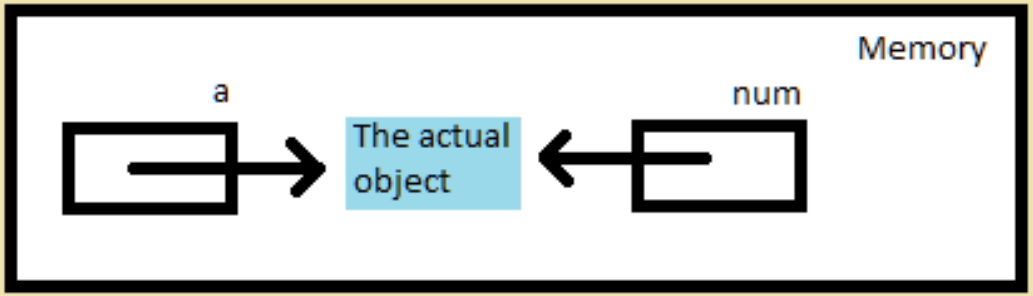
```
int a = 34;
meth ( a )
```



Call by Reference

public void meth (Number num)

```
Number a = new Number ( 34 );
meth ( a ) ;
```



Call-by-value example

```
public void multiplyBy5 (int val) {
        val = val * 5;
        System.out.println(val); //What is printed?
}
public void multiplyBy3 (int val) {
        val = val * 3;
        System.out.println(val); //What is printed?
        return val;
}
int a = 3;
multiplyBy5(a);
System.out.pritnln(a); //What is printed?
multiplyBy3(a);
System.out.pritnln(a); //What is printed?
a = multiplyBy3(a);
System.out.pritnln(a); //What is printed?
```

Call-by-reference example

```
public void multiplyBy5 (Node val) {
        val.data = val.data * 5;
        System.out.println(val); //What is printed?
}
public void multiplyBy3 (Node val) {
        val = new Node(val.data * 3);
        System.out.println(val); //What is printed?
        return val;
}
Node a = new Node(3);
multiplyBy5(a);
System.out.pritnln(a); //What is printed?
multiplyBy3(a);
System.out.pritnln(a); //What is printed?
a = multiplyBy3(a);
```

Call-by-value example

```
public void multiplyBy5 (int val) {
        val = val * 5;
        System.out.println(val); // 15
}
public void multiplyBy3 (int val) {
        val = val * 3;
        System.out.println(val); // 9
        return val;
}
int a = 3;
multiplyBy5(a);
System.out.pritnln(a); // 3
multiplyBy3(a);
System.out.pritnln(a); // 3
a = multiplyBy3(a);
System.out.pritnln(a); // 9
```

Call-by-reference example

```
public void multiplyBy5 (Node val) {
        val.data = val.data * 5;
        System.out.println(val); // 15
}
public void multiplyBy3 (Node val) {
        val = new Node(val.data * 3);
        System.out.println(val); // 45
        return val;
}
Node a = new Node(3);
multiplyBy5(a);
System.out.pritnln(a); // 15
multiplyBy3(a);
System.out.pritnln(a); // 15
a = multiplyBy3(a);
System.out.pritnln(a); // 45
```