Recursion Wrap Up

- Remember Recursion Tips
- Think about base case and recursive case
- Recursion is elegant but rarely efficient
- Can simulate recursion using a stack to push process, and pop records (Reference optional Text*)
- Recursion causes many method calls and thus many activation records on the call stack

*__Data Structures and Abstractions with Java__ by Carrano and Henry

Exception: StackOverflowError

Method calls are placed on the stack.

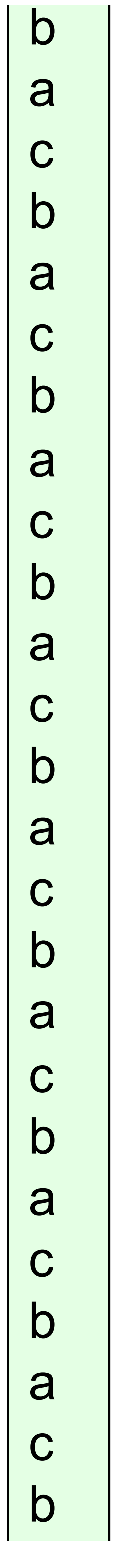
Beware of runaway recursion!

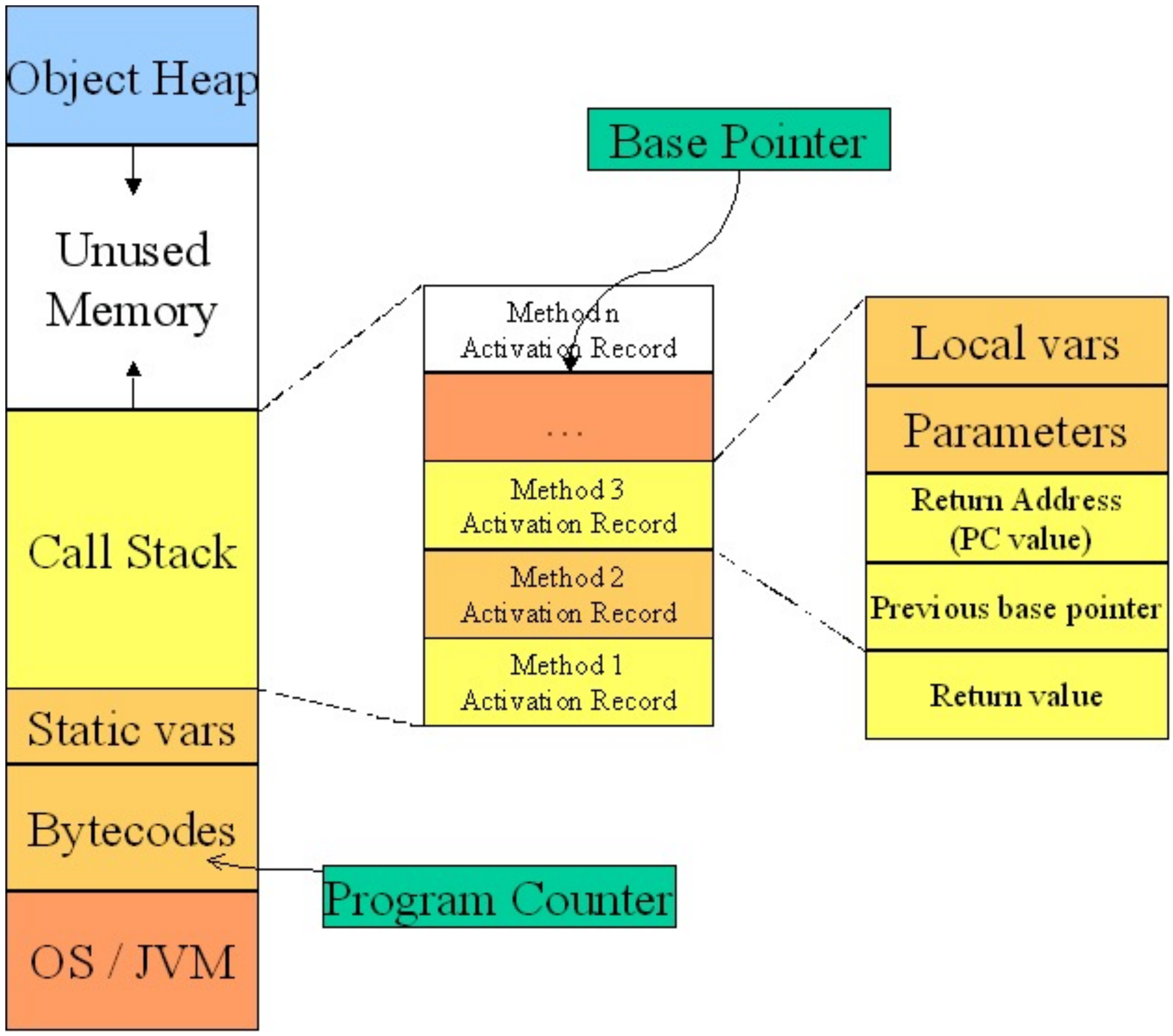
a() calling b(),

b() calling c(),

c() calling a()

The stack, would grow infinitely large, and at some point the VM cuts it off. This results in a "StackOverflowException"




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
Local vars
Parameters
Return Address (PC value)
Previous base pointer
Return value
Program Counter

Visualize Code!

```
public class ClassNameHere {
  public static void main(String[] args) {

    int[] array= new int[] {1,2,3,4,5,6,7};
    displayArray(array, 0,6);

  }

  public static void displayArray(int array[], int first,
                                  int last)
{
    if (first == last)
      System.out.print(array[first] + " ");
    else
    {
      int mid = (first + last) / 2;
      displayArray(array, first, mid);
      displayArray(array, mid + 1, last);
    }
}

}
```

Experiment with Java Visualizer: (https://cscircles.cemc.uwaterloo.ca/java_visualize/#)

Trace displayArrary: https://cscircles.cemc.uwaterloo.ca/java_visualize/#code=public+class+ClassNameHere+%7B%0A+++public+static+void+main(String%5B%5D+args)+%7B%0A++++++%0A++++++int%5B%5D+array%3D+new+int%5B%5D+%7B1,2,3,4,5,6,7%7D%3B%0A++++++displayArray(array,+0,6)%3B%0A+++%