Understanding the Problem

You should now have a clear understanding of how recursion works - what it is and when it can be used. Before you proceed writing your own recursive code, it is very important to understand the nitty-gritty details of how the code actually works.

Below is a basic example of how the order of execution for a recursive method changes at each successive recursive call.

Example

What do you think is the output of the code below? Take your time to think it over before you execute the code and look at the result.

```
class ExampleClass {

    private static void printNum(int n) {
        // Base case
        if (n == 0) {
            return;
        }
        // Recursive case
        printNum(n-1);
        System.out.print(n + " ");
    }

    public static void main( String args[] ) {
        // Recursive method called here
        printNum(6);
    }
}
```

Run Save Reset

Code Explanation

If we read the method as it is written, we might expect the output to be $6, 5, 4, 3, 2, 1$. Doesn't it seem like the method prints 6 first and then gradually gets decremented?

Sadly, this is not correct. We have to *evaluate the recursive call before we can continue to the end of the method.*

The code snippet below illustrates how the code gets executed.

```
printNum(6)
    printNum(5)
        printNum(4)
            printNum(3)
                printNum(2)
                    printNum(1)
                        printNum(0)
                        System.out.print(1)
                    System.out.print(2)
                System.out.print(3)
            System.out.print(4)
        System.out.print(5)
    System.out.print(6)
```

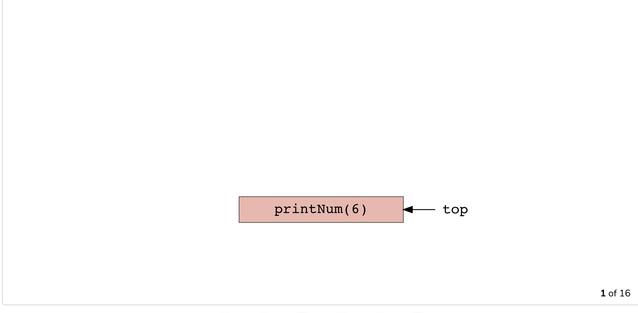
As you can see after running the code, the actual output turned out to be $1, 2, 3, 4, 5, 6$. Interesting, isn't it? This is an important note to recognize: the output from a recursive method will not always turn out as it appears in the order of the code.

How to compute the output

The trick to understanding a recursive code is to act as if you're the compiler and walk through the code line-by-line. Other methods that can help you understand a recursive method and interpret its output correctly are as follows:

Visualizing through a Stack

The concept of a stack is critical in recursion. When you start visualizing your method calls through a stack and how it returns when reaching a base case, the concept of recursive calls and its output becomes easier to comprehend.

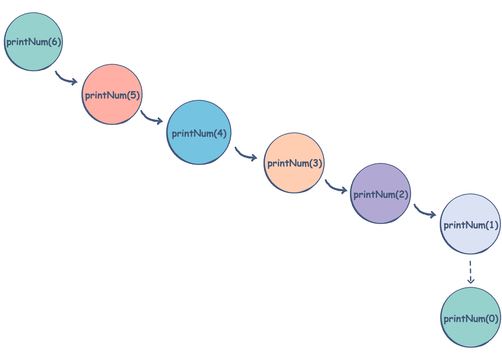


Keeping a Track of your Variables

This is very crucial while writing recursive methods. As the method gets more complicated it becomes harder to store everything in your head and keep a track of all the variables and method calls, especially because, as we saw from the example above, we have to do the recursion out of order.

Drawing a Recursive Tree

Recursive methods usually tend to act like a tree. The parent is the main method call and each recursive call is like a child node.



Practice

Well, practice makes perfect, right? As you proceed with the course and understand how each recursive code is written, you will find it easier to write your own.

Let's move on and learn about a few advantages and disadvantages of recursion in the next lesson.