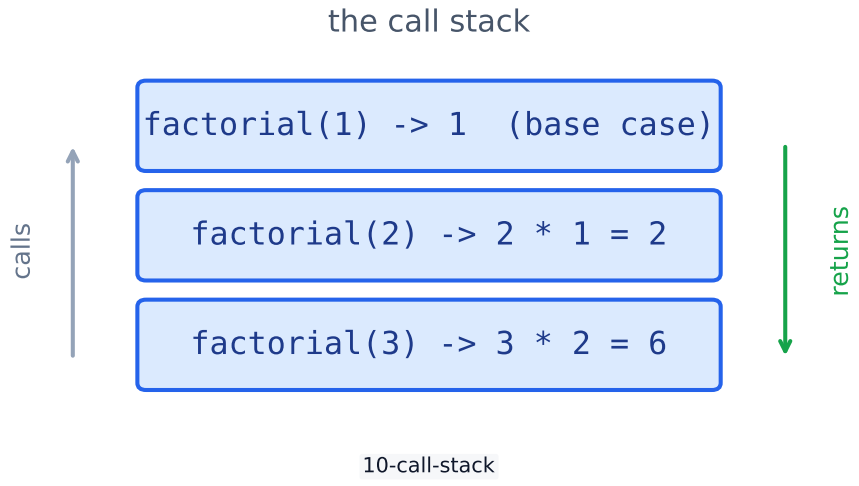
递归

Java Reference

递归 (recursion) 是一个调用自己的方法。每个递归都需要一个基准情形 (base case, 即何时停止), 以及一个朝它靠近的递归调用 (recursive call)。没有基准情形它就永不停止, 并以栈溢出崩溃。



factorial(3) 的调用栈: 每个调用等待下层返回, 再按相反顺序返回

```java
public class Main {
    public static void main(String[] args) {
        System.out.println(factorial(5));   // 120
    }

    public static int factorial(int n) {
        if (n <= 1) return 1;             // 基准情形
        return n * factorial(n - 1);      // 递归调用:n * (n-1)!
    }
}
```

跟踪一下:`factorial(5)` 等待 `factorial(4)`, 后者又等待 `factorial(3)`…… 一直到 `factorial(1)` 返回 `1`。然后结果再逐层相乘返回:`1 → 2 → 6 → 24 → 120`。

递归同样适用于字符串 — 每次调用剥掉一个字符:

```
public class Main {
    public static void main(String[] args) {
        System.out.println(reverse("PYTHON"));   // NOHTYP
    }

    static String reverse(String s) {
        if (s.length() <= 1) return s;                // 基准情形
        return reverse(s.substring(1)) + s.charAt(0);
    }
}
```

归并排序 (merge sort) 是 AP 考试里的递归排序: 把数组分成两半, 分别递归排序, 再把两个有序的一半合并 (merge)。它是 O(n log n)— 在大数组上远快于 $O(n^2)$ 的排序。

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] a = {5, 2, 9, 1, 7, 3};
        mergeSort(a, 0, a.length - 1);
        System.out.println(Arrays.toString(a));   // [1, 2, 3, 5, 7, 9]
    }

    static void mergeSort(int[] a, int lo, int hi) {
        if (lo >= hi) return;                  // 基准情形: 只剩一个元素
        int mid = (lo + hi) / 2;
        mergeSort(a, lo, mid);                 // 排序左半
        mergeSort(a, mid + 1, hi);             // 排序右半
        merge(a, lo, mid, hi);                 // 合并两半
    }

    static void merge(int[] a, int lo, int mid, int hi) {
        int[] tmp = new int[hi - lo + 1];
        int i = lo, j = mid + 1, k = 0;
        while (i <= mid && j <= hi) {
            if (a[i] <= a[j]) tmp[k++] = a[i++];
            else              tmp[k++] = a[j++];
        }
        while (i <= mid) tmp[k++] = a[i++];
        while (j <= hi)  tmp[k++] = a[j++];
        for (k = 0; k < tmp.length; k++) a[lo + k] = tmp[k];
    }
}
```

常见错误

- 递归必须有基准情形, 否则会抛出 StackOverflowError。
- 每次递归调用都要更**靠近**基准情形。
- 手动追踪一个小例子, 检查递归返回的值对不对。
- 归并排序里真正干活的是合并那一步; 递归只负责拆分数组。