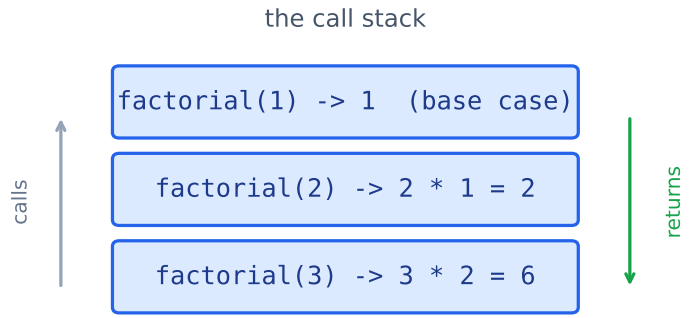
Recursion

Java Reference

Recursion

Recursion 递归 is a method that calls itself. Every recursion needs a base case 基准情形 (when to stop) and a recursive call 递归调用 that moves toward it. Without a base case it never stops and crashes with a *stack overflow*.



10-call-stack

The call stack for factorial(3): each call waits, then returns in reverse order

```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;            // base case
        return n * factorial(n - 1);     // recursive call: n * (n-1)!
    }
}
```

Trace it: `factorial(5)` waits for `factorial(4)`, which waits for `factorial(3)`… down to `factorial(1)` returning `1`. Then the answers multiply back up: `1` → `2` → `6` → `24` → `120`.

Recursion also works on Strings — peel off one character each call:

```java
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;               // base case
        return reverse(s.substring(1)) + s.charAt(0);
    }
}
```

Merge sort 归并排序 is the recursive sort on the AP exam: split the array in half, sort each half recursively, then merge 合并 the two sorted halves. It runs in O(n log n) — far faster than the $O(n^2)$ sorts on big arrays.

```
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;                 // base case: one element
        int mid = (lo + hi) / 2;
        mergeSort(a, lo, mid);                // sort the left half
        mergeSort(a, mid + 1, hi);            // sort the right half
        merge(a, lo, mid, hi);                // merge the two halves
    }

    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];
    }
}
```

Common mistakes

- Recursion needs a base case, or it throws StackOverflowError.
- Each recursive call must move CLOSER to the base case.
- Trace a small example by hand to check the recursion returns the right value.
- In merge sort, the merge step does the real work; the recursion only splits the array.