Recursion in Java

Josh Wilcox (jw14g24@soton.ac.uk)

February 25, 2025

Contents

1 Recursive Recap

- Breaks a **big** problem into several **small** problem that each require the same process on them
- Recursion can be the easiest conceptual problem for many usecases
- Some languages are optimised for recursion

2 Recursion and the Java Stack

- If a method calls another method in the middle of its code, then a new *stack frame* is pushed onto the call stack.
  - A stack frame in Java typically contains method parameters, local variables, return addresses, and references necessary for the method's execution.
- This is also true if the method calls **itself**. Each recursive call creates a new frame,

3 Tail Recursion

- Tail recursion is a special form of recursion where the recursive call is the *last* operation in the method.
- In many functional programming languages (like Scala, Haskell), tail recursion is optimized through *tail call optimization* (TCO).
  - With TCO, the compiler can transform tail-recursive calls into iterations, reusing the same stack frame.
- **Java does not implement tail call optimization**.
  - Even in tail-recursive form, Java still creates a new stack frame for each recursive call.
  - This means that tail-recursive methods in Java are still at risk of causing `StackOverflowError` for deep recursion.
- Example of tail recursion in Java, which still consumes stack space:

```java
// Tail-recursive factorial implementation
public static int factorialTail(int n, int accumulator) {
    if (n <= 1) {
        return accumulator;
    }
    // This is a tail-recursive call, but Java still creates a new stack frame
    return factorialTail(n - 1, n * accumulator);
}

// Usage
public static int factorial(int n) {
    return factorialTail(n, 1);
}
```

- Despite being in tail position, Java's JVM still allocates a new stack frame for each call to `factorialTail`.
- For large values of `n`, this can still lead to `StackOverflowError`.
- In languages with proper TCO, this would be automatically converted to an iterative process using constant stack space.