Recursion

- The *process* of a method calling **itself** is known as **recursion**.
    - Concepts like **Recurrence Relations** in mathematics can be implemented using recursion.
    - **Example**: Factorial of a number.

```java
class Factorial {
    public long factorial(int n) {
        if (n == 1) return 1;
        return n * this.factorial(n - 1);
    }

    public static void main(String[] args) {
        int n = 5;
        Factorial f = new Factorial();
        System.out.println("Factorial of " + n + " is: " +
f.factorial(n));
    }
}
```

- During recursion:
    - The JVM uses a **stack** to keep track of active method calls.
    - Each new method call is **pushed** onto the stack.
    - Once the deepest call is reached and begins returning, the stack starts **unwinding** as each method call completes.
    - This ensures **Last-In-First-Out (LIFO)** execution order.

> Recursive calls must always move toward a **base condition** to prevent infinite recursion and `StackOverflowError`.