Recursion in Java

Recursion in Java is a programming technique where a **method within a class calls itself to solve a smaller version of the same problem**. This approach is an integral part of problem-solving in object-oriented programming (OOP), particularly for tasks involving **hierarchical or repetitive structures like trees and graphs**.

How Recursion Works in Java

Recursion utilizes the **call stack of the Java Virtual Machine (JVM)**.

1. **Method Call:** When a recursive method is first called, a new **stack frame** is **pushed onto the call stack**. This **frame stores local variables** and **parameters for that specific call instance**.
2. **Stack Growth:** If the base case is not met, the **method calls itself again**, and **another stack frame is pushed on top of the previous one**. This continues until the **base case is reached**.
3. **Unwinding (Return Phase):** Once the base case is hit, **the results are returned from the most recent call**, and its **stack frame is popped off**. The execution then resumes in the previous stack frame, continuing this process until all frames are resolved and popped off the stack, and the result is returned to the original caller.

Example: Calculating Factorial

A factorial, denoted by **n!** is the **product of all positive integers from 1 up to a given number n** (**n! = n×(n-1) ×…× 1**). It represents the number of ways to arrange **n** distinct objects. Key rules include **0!=1** and **n! = n×(n-1)!**. For example, 5!= 5×4×3×2×1=120.

```
public class FactorialExample {
   // The recursive method
   public static int factorial(int n) {
      // Base Case: stops the recursion when n becomes 1
      if (n == 1) {
         return 1; }
      else {
         // Recursive Case: n * factorial of (n-1)
```

```
        return n * factorial(n - 1);  }}


  public static void main(String[] args) {
      int result = factorial(5); // Calls the recursive method
      System.out.println("Factorial of 5 is: " + result); // Output: 120
   } }
```

Output

Factorial of 5 is: 120

Example: Calculating Factorial using object

```
class GFG {

    // recursive method
    int fact(int n)   {
        int result;

        if (n == 1)
            return 1;
        result = fact(n - 1) * n;
        return result;  }}

// Driver Class
class Recursion {
   public static void main(String[] args)    {
        GFG f = new GFG();

        System.out.println("Factorial of 3 is " + f.fact(3));
        System.out.println("Factorial of 4 is " + f.fact(4));
        System.out.println("Factorial of 5 is "+ f.fact(5)); }}
```

Output

Factorial of 3 is 6
Factorial of 4 is 24
Factorial of 5 is 120

Example

Use recursion to **Summing Integers Numbers (1 to K)**. **S(K)= K+S(K-1) for K>0**

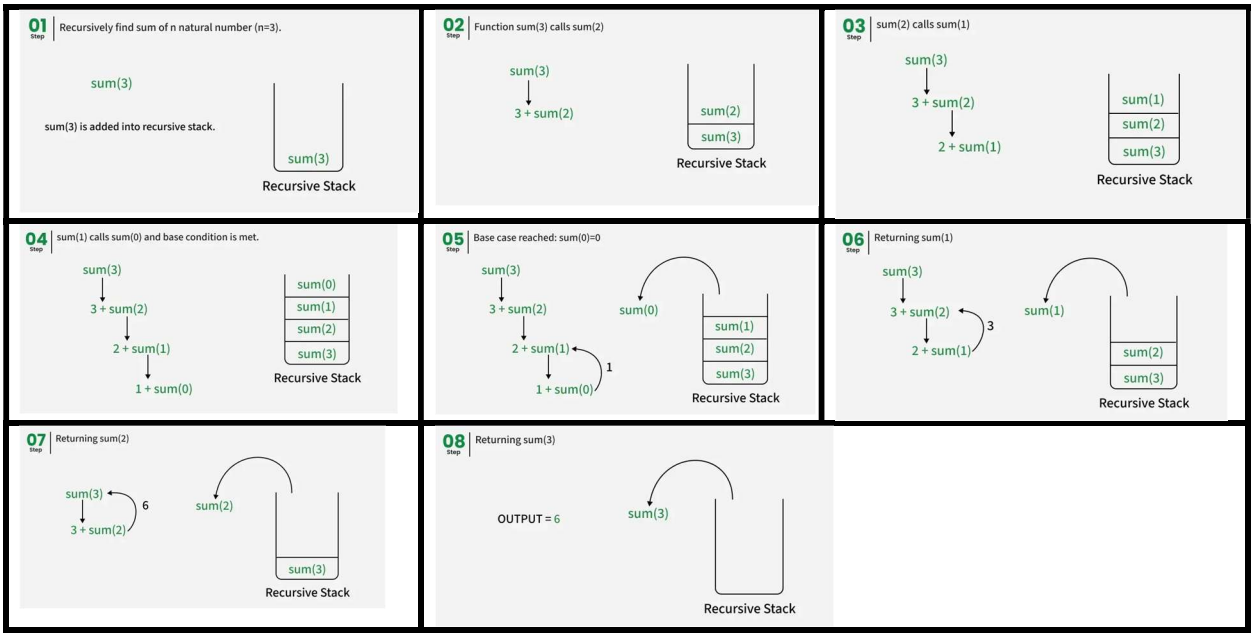
```
public class Main {

  public static int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1); }
    else {
      return 0; }  }

  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result); }}
```

Tracing

10 + sum(9)
10 + (9 + sum(8))
10 + (9 + (8 + sum(7)))
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0=55



Fibonacci Series

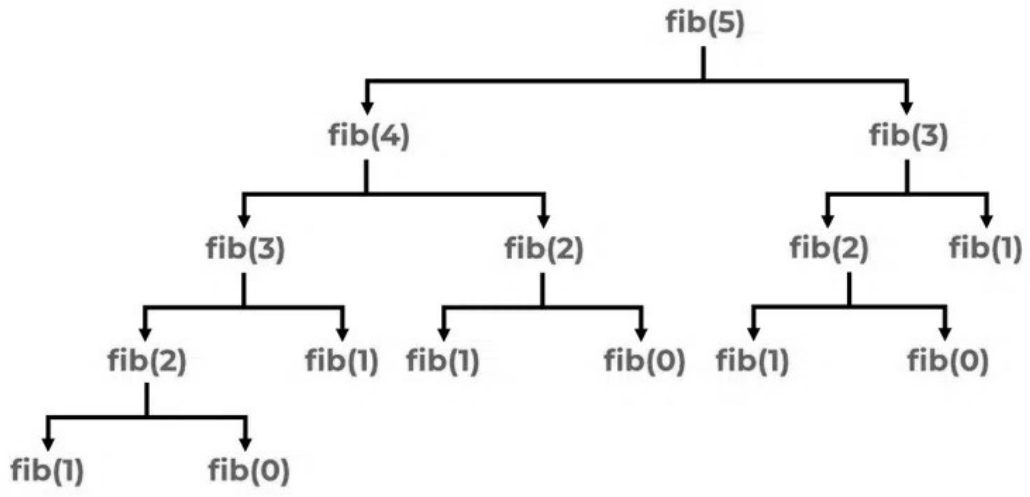
The **Fibonacci sequence is a series of numbers** where each number is the **sum of the two preceding ones**, usually **starting with 0 and 1**, resulting in: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so on.

The Fibonacci numbers may be defined by the recurrence relation

$F_0 = 0, \quad F_1 = 1,$
$F_n = F_{n-1} + F_{n-2} \qquad$ for $n > 1$.

```
public class Fibonacci {

   // Recursive method to find the nth Fibonacci number
   public static int fib(int n) {
      if (n <= 1) {
         return n; }
      else {
         return fib(n - 1) + fib(n - 2); }}
 public static void main(String[] args) {
 int n = 5; // The index of the desired Fibonacci number
 System.out.println("Fibonacci number at position " + n + " is: " + fib(n));}}
```



fib(2)= fib(1)+fib(0)=1+0=1
fib(3) = fib(2) + fib(1) = fib(1) + 0 + 1 = 1+1 = 2
fib(4) = fib(3) + fib(2) = 2+1 = 3
fib(5) = fib(4) + fib(3) = 3 + 2 = 5

Example recursion using object

```
class A {

   void recursive (int i) {
      System.out.println("A.recursive(" + i + ")");
      if (i > 0) {
         recursive (i - 1);   }  }}

public class Demo {

   public static void main(String[] args) {
      A a = new A();
      a.recursive(5);  }}
```

Output

```
A.recursive(5)
A.recursive(4)
A.recursive(3)
A.recursive(2)
A.recursive(1)
A.recursive(0)
```

Example of Reversing a String

Reversing a string can be done recursively by taking the **first character** and **placing it after the recursive reversal of the remaining substring.**

Recursive case: **return** the **reverse of the substring** after the **first character**, plus the **first character.**

- **charAt(0), used to retrieve the first character of a string.**

- **Substring(1), return a new string that starts from the character at index 1 and extends to the end of the original string, meaning the first character is at index 0**

```
public class ReverseString {

   public static String reverse(String str) {

   // Base case: if the string is empty or has only one character, it's already reversed
      if (str.isEmpty() || str.length() == 1) {
         return str;        }
```

```
  // Recursive case
    return reverse(str.substring(1)) + str.charAt(0); }

public static void main(String[] args) {
        String original = "Hello";
        String reversed = reverse(original);
        System.out.println("Original string: " + original);
        System.out.println("Reversed string: " + reversed); }}
```

Tracing

substring=ello
charAt=H
substring=llo
charAt=e
substring=lo
charAt=l
substring=o
charAt=l
Original string: Hello
Reversed string: olleH

Example use recursion to create a countdown function:

```
public class Main {
  static void countdown(int n) {
    if (n > 0) {
      System.out.print(n + " ");
      countdown(n - 1);  }  }
  public static void main(String[] args) {
    countdown(5);  }}
```

Output

5 4 3 2 1

Example Print Numbers from 1 to N

```
public class Printn {

   static void printNumbers(int n) {
           if (n == 0)
                return;
            else {
               System.out.print(n + " ");
               printNumbers(n - 1);   } }

    public static void main(String[] args) {
    printNumbers(4);}}
```

Output

4 3 2 1