classmate
Date
Page

Recursion

Recursion

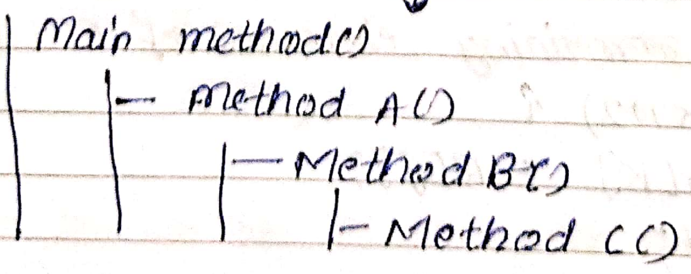
"process where a method calls itself in order to solve a problem. This technique is commonly used to break down a problem into smaller, more manageable sub- problems of the ~~Sme~~ same type.

Each recursive call should move the solution closer to a base case, which stops the recursion.

Base case → Stop repetation of the code. or end.

Note

if the Base case is not provided then Stack Overflow recursion in Java occurs when a recursive method calls itself too many times, exceeding the maximum size of the call stack.



main method() calls Method A(), which calls Method B, which calls Method C. and so on.

Scanned with OKEN Scanner

Call Stack

'memory region that stores information about active method calls, including method names, parameters & return addresses. It's Last-In-First-Out (LIFO) data structures that manages method calls & returns.

Problem statement: printing "Hi" 5 times using recursion in java.

```
public static void printHi(int n) {
        if (n>0) {                              ← base case
              System.out.println("Hi");
              printHi(n-1);             ← Recursive case
        }
}

public static void main(String[] args){
        printHi(5);
}
```

→ printHi method calls itself recursively until n reaches 0.

n=5 in this case.

```
printHi(5)          →  Hi
printHi(5-1) - 4    →  Hi
printHi(4-1) - 3    →  Hi
printHi(3-1) - 2    →  Hi
printHi(2-1) - 1    →  Hi
printHi(1-1) - 0    →  returns /ends. (n is not
                                       greater than
                                       0).
```

Problem Statement 2   "Simple recursive function in Java that calculates the factorial of a number."

5! = 5 x 4 x 3 x 2 x 1 = 120

```
public static int factorial(int n){
        if(n == 0){          ← Base case
                                 Factorial of 0 is 1
            return 1;
        }
        else{   ← Recursive case
            return n * factorial(n-1);
        }
}

public static void main(String[] args){
        int number = 5;
        int result = factorial(number);
        System.out.println("Factorial of "+
          number * " is " + result);
}
```

Scanned with OKEN Scanner

Date ______
Page ______

factorial (5) → 5
Factorial (5-1) = 4 → 4
Factorial (4-1) = 3 → 3
Factorial (3-1) = 2 → 2
factorial (2-1) = 1 → 1
factorial (1-1) = 0 → 1

5×4×3×2×1
= 120.

Explanation:- The factorial method is defined to take an integer 'n'.

- If 'n' is 0, the method returns 1 (the base case).
- otherwise, the method returns 'n * factorial(n-1)', which is a recursive call to itself with 'n-1'.

1. factorial (5) calls factorial (4)
2. factorial (4) calls factorial (3)
3. factorial (3) calls factorial (2)
4. factorial (2) calls factorial (1)
5. factorial (1) calls factorial (0)
6. Factorial (0) returns 1 (base case)

n * factorial(n-1)

backtracking

7. factorial (~~0~~) (1) returns 1 * 1 = 1
8. Factorial (2) returns 2 * 1 = 2
9. factorial (3) returns 3 * 2 = 6
10. factorial (4) returns 4 * 6 = 24
11. factorial (5) returns 5 * 24 = 120