

6주차: 재귀 호출 (Recursion)

핵심 개념: 함수가 자기 자신을 다시 호출하여 큰 문제를 작은 단위로 쪼개는 기법

필수 구성 요소:

기저 조건 (Base Case): 반드시 재귀가 멈추는 조건이 있어야 함 (없으면 StackOverflowError 발생).

재귀 단계 (Recursive Step): 문제를 해결하기 위해 자신을 호출하되, 매개변수가 점점 기저 조건에 가까워져야 함.

특징:

코드의 간결성: 반복문(for/while)보다 로직을 훨씬 깔끔하게 표현 가능.

메모리 구조: 호출될 때마다 스택 메모리에 쌓이므로 깊이가 너무 깊어지면 위험함.

```
public class C03 {  
  
    // [1] 팩토리얼 계산기: n! = n * (n-1) * (n-2) * ... * 1  
  
    public static int factorial(int n) {  
  
        // 호출 순서를 보여주기 위한 출력  
  
        System.out.println("-> factorial(" + n + ") 호출됨");  
  
        // 1. 기저 조건 (Base Case): 1에 도달하면 멈추고 1을 돌려줌  
        if (n == 1) {  
            return 1;  
        }  
        // 2. 재귀 단계 (Recursive Step): n! = n * (n-1)!  
        else {  
            return n * factorial(n - 1);  
        }  
    }  
}
```

```
if (n <= 1) {  
  
    System.out.println("    [멈춤] 1에 도달하여 값을 반환하기 시작합니다.");  
  
    return 1;  
  
}
```

// 2. 재귀 단계: n과 (n-1)의 팩토리얼 결과값을 곱함

// 결과가 나올 때까지 잠시 기다렸다가(Stack에 쌓임) 계산합니다.

```
int result = n * factorial(n - 1);
```

```
System.out.println("<- factorial(" + n + ") 계산 완료: " + result);
```

```
return result;
```

```
}
```

// [2] 합계 계산기: 1부터 n까지의 합

```
public static int sum(int n) {
```

// 기저 조건: n이 1이면 1을 반환

```
if (n == 1) {
```

```
    return 1;
```

```
}
```

```
// n + (n-1까지의 합)
```

```
return n + sum(n - 1);
```

```
}
```

```
// 메인 메서드: 실제 실행이 일어나는 곳
```

```
public static void main(String[] args) {
```

```
System.out.println("===[팩토리얼 테스트] ===");
```

```
int factResult = factorial(3); // 3 * 2 * 1 = 6
```

```
System.out.println("결과: 3! = " + factResult);
```

```
System.out.println("n===[1부터 N까지 합 테스트] ===");
```

```
int sumResult = sum(10); // 1+2+...+10 = 55
```

```
System.out.println("결과: 1부터 10까지의 합 = " + sumResult);
```

```
}
```

```
}
```