

Prompt://Task 1: AI-Generated Logic Without Modularization (Factorial without Functions)

```
class Main{  
    public static void main(String[] args) {  
        int number = 5; // Example number to calculate factorial  
        int factorial = 1;  
        for (int i = 1; i <= number; i++) {  
            factorial *= i;  
        }  
        System.out.println("Factorial of " + number + " is: " + factorial);  
    }  
}
```

Prompt://Task 2: AI Code Optimization & Cleanup (Improving Efficiency)

```
class Main{  
    public static void main(String[] args) {  
        int number = 5; // Example number to calculate factorial  
        long factorial = 1; // Use long to handle larger results  
  
        for (int i = 2; i <= number; i++) { // Start loop from 2  
            factorial *= i;  
        }  
  
        System.out.println("Factorial of " + number + " is: " + factorial);  
    }  
}
```

Prompt://Task 3: Modular Design Using AI Assistance (Factorial with Functions)

```
class Main{  
    public static void main(String[] args) {  
        int number = 5; // Example number to calculate factorial  
        long result = factorial(number);  
        System.out.println("Factorial of " + number + " is: " + result);  
    }  
}
```

```
}

public static long factorial(int n) {
    long fact = 1;
    for (int i = 2; i <= n; i++) {
        fact *= i;
    }
    return fact;
}
}
```

Prompt://Task 4: Comparative Analysis – Procedural vs Modular AI Code (With vsWithout Functions)

// Procedural Approach

```
class Main{
    public static void main(String[] args) {
        int number = 5;
        long factorial = 1;

        for (int i = 2; i <= number; i++) {
            factorial *= i;
        }

        System.out.println("Factorial of " + number + " is: " + factorial);
    }
}
```

Prompt://Task 5: AI-Generated Iterative vs Recursive Thinking (Factorial with Recursion)

```
class Main{

    public static void main(String[] args) {
        int number = 5; // Example number to calculate factorial
        long result = factorial(number);
    }
}
```

```
        System.out.println("Factorial of " + number + " is: " + result);
    }
    public static long factorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
        } else {
            return n * factorial(n - 1);
        }
    }
}
```

output:

```
PS C:\Users\asus> & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.17-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\asus\AppData\Local\Temp\vscodesws_06682\jdt_ws\jdt.ls-java-project\bin' 'Main'
Factorial of 5 is: 120
```

Understanding:

- The file stacks five separate Main classes, each showing a different factorial approach (basic loop, loop starting at 2 with long, modular with helper, procedural loop, recursive). In Java only the last Main will compile/run; the earlier ones are ignored or cause a duplicate-class error if compiled together—pick one class name or split into separate files.
- Task 1: iterative factorial, int result, loop 1..n; fine for very small n but overflows quickly.
- Task 2: iterative factorial using long, starts loop at 2 to skip a no-op multiply by 1; better range, still overflows past 20!.
- Task 3: modular version with a factorial(int n) helper; still iterative, uses long.
- Task 4: another procedural loop identical to Task 2; redundant.
- Task 5: recursive factorial; concise but same overflow limit and adds call-stack depth risk for larger n.